

AUTHORS

1 This file lists all people who have contributed to the SOM VM.
2
3 SOM was originally implemented at the University of Aarhus (Denmark) in
4 2001/2002. The implementation of SOM was done by Jakob Roland Andersen,
Kasper
5 Verdich Lund, Lars Bak, Mads Torgersen, and Ulrik Pagh Schultz. They also
6 wrote the original versions of the SOM Smalltalk libraries, test suites,
and
7 benchmarks, that are (in extended versions) bundled with SOM.
8
9 SOM was used by Michael Haupt in courses on virtual machines at Lancaster
10 University and Technische Universitaet Darmstadt (Germany) in VM courses in
11 2006. During that time, some changes were applied to SOM by Michael Haupt
and
12 Sebastian Kanthak.
13
14 SOM is currently maintained by Michael Haupt at the Hasso-Plattner-
Institut,
15 University of Potsdam, Germany.
16
17 2009-08-14, Michael Haupt
18 michael.haupt@hpi.uni-potsdam.de

```

1 "
2 Copyright (c) 2001-2016 see AUTHORS.md file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22 Ball = (
23
24   | x y xVel yVel |
25
26   bounce = (
27     | xLimit yLimit bounced |
28     xLimit := yLimit := 500.
29     bounced := false.
30
31     x := x + xVel.
32     y := y + yVel.
33     x > xLimit ifTrue: [
34       x := xLimit. xVel := 0 - xVel abs. bounced := true ].
35     x < 0 ifTrue: [
36       x := 0.      xVel := xVel abs.      bounced := true ].
37     y > yLimit ifTrue: [
38       y := yLimit. yVel := 0 - yVel abs. bounced := true ].
39     y < 0 ifTrue: [
40       y := 0.      yVel := yVel abs.      bounced := true ].
41     ^ bounced
42   )
43
44   initialize: random = (
45     x := random next % 500.
46     y := random next % 500.
47     xVel := (random next % 300) - 150.
48     yVel := (random next % 300) - 150.
49   )
50
51   -----
52
53   new: random = ( ^ self new initialize: random )
54

```

55)
56

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Benchmark = (  
23  
24   innerBenchmarkLoop: innerIterations = (  
25     1 to: innerIterations do: [:i |  
26       (self verifyResult: self benchmark) ifFalse: [ ^ false ].  
27     ].  
28     ^ true  
29   )  
30  
31   benchmark = ( self subclassResponsibility )  
32   verifyResult: result = ( self subclassResponsibility )  
33 )  
34
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Bounce = Benchmark (  
23  
24   benchmark = (  
25     | ballCount balls bounces random |  
26  
27     random := SomRandom new.  
28  
29     ballCount := 100.  
30     bounces   := 0.  
31     balls     := Array new: ballCount withAll: [ Ball new: random ].  
32  
33     1 to: 50 do: [ :i |  
34       balls do: [ :ball |  
35         (ball bounce) ifTrue: [ bounces := bounces + 1 ] ] ].  
36  
37     ^ bounces  
38   )  
39  
40   verifyResult: result = (  
41     ^ 1331 = result  
42   )  
43 )  
44
```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 Aircraft = (  
34   | callsign position |  
35  
36   init: aCallsign pos: aPosition = (  
37     callsign := aCallsign.  
38     position := aPosition.  
39   )  
40  
41   callsign = ( ^ callsign )  
42   position = ( ^ position )  
43  
44   ----  
45   new: callsign pos: position = (  
46     ^ self new init: callsign pos: position  
47   )  
48 )  
49 )  
50 )
```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 CD = Benchmark (  
34  
35   benchmark: numAircrafts = (  
36     | numFrames simulator detector actualCollisions |  
37     numFrames := 200.  
38  
39     simulator := Simulator new: numAircrafts.  
40     detector := CollisionDetector new.  
41  
42     actualCollisions := 0.  
43  
44     0 to: numFrames - 1 do: [:i |  
45       | time collisions |  
46       time := i // 10.0.  
47       collisions := detector handleNewFrame: (simulator simulate: time).  
48       actualCollisions := actualCollisions + collisions size ].  
49     ^ actualCollisions  
50   )  
51  
52   innerBenchmarkLoop: innerIterations = (  
53     ^ self verify: (self benchmark: innerIterations) resultFor:  
54     innerIterations  
55   )  
56  
57   verify: actualCollisions resultFor: numAircrafts = (  
58     | numFrames |  
59     numFrames := 200.  
60     simulator := Simulator new: numAircrafts.  
61     detector := CollisionDetector new.  
62     actualCollisions := 0.  
63     0 to: numFrames - 1 do: [:i |  
64       | time collisions |  
65       time := i // 10.0.  
66       collisions := detector handleNewFrame: (simulator simulate: time).  
67       actualCollisions := actualCollisions + collisions size ].  
68     ^ actualCollisions  
69   )  
70 )
```

```

54     numAircrafts = 1000 ifTrue: [ ^ actualCollisions = 14484 ].
55     numAircrafts = 500 ifTrue: [ ^ actualCollisions = 14484 ].
56     numAircrafts = 250 ifTrue: [ ^ actualCollisions = 10830 ].
57     numAircrafts = 200 ifTrue: [ ^ actualCollisions = 8655 ].
58     numAircrafts = 100 ifTrue: [ ^ actualCollisions = 4305 ].
59     numAircrafts = 10 ifTrue: [ ^ actualCollisions = 390 ].
60     numAircrafts = 2 ifTrue: [ ^ actualCollisions = 42 ].
61
62     ('No verification result for ' + numAircrafts + ' found.') println.
63     ('Result is: ' + actualCollisions) println.
64     ^ false
65 )
66
67 ----
68
69 new = (
70     Constants initialize.
71     ^ super new
72 )
73 )
74

```



```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 CallSign = (  
34   | value |  
35  
36   init: val = ( value := val )  
37  
38   value = ( ^ value )  
39  
40   compareTo: other = (  
41     ^ value = other value  
42     ifTrue: [ 0 ]  
43     ifFalse: [  
44       value < other value ifTrue: [ -1 ] ifFalse: [ 1 ] ]  
45   )  
46  
47   ----  
48   new: val = (  
49     ^ self new init: val  
50   )  
51 )  
52 )  
53 )  
54 )  
55 )  
56 )  
57 )  
58 )  
59 )  
60 )  
61 )  
62 )  
63 )  
64 )  
65 )  
66 )  
67 )  
68 )  
69 )  
70 )  
71 )  
72 )  
73 )  
74 )  
75 )  
76 )  
77 )  
78 )  
79 )  
80 )  
81 )  
82 )  
83 )  
84 )  
85 )  
86 )  
87 )  
88 )  
89 )  
90 )  
91 )  
92 )  
93 )  
94 )  
95 )  
96 )  
97 )  
98 )  
99 )  
100 )
```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 Collision = (  
34   | aircraftA aircraftB position |  
35  
36   init: anA b: aB pos: aPos = (  
37     aircraftA := anA.  
38     aircraftB := aB.  
39     position := aPos  
40   )  
41  
42   aircraftA = ( ^ aircraftA )  
43   aircraftB = ( ^ aircraftB )  
44   position = ( ^ position )  
45  
46   ----  
47   a: aircraftA b: aircraftB pos: position = (  
48     ^ self new init: aircraftA b: aircraftB pos: position  
49   )  
50 )
```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are  
met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
IS' AND  
19 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
IMPLIED  
20 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
21 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
22 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
23 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
24 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED  
AND  
25 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
26 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
THIS  
27 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
28 "  
29 CollisionDetector = (  
30   | state |  
31  
32   initialize = (  
33     state := RedBlackTree new  
34   )  
35  
36   handleNewFrame: frame =(  
37     | motions seen toRemove allReduced collisions |  
38     motions := Vector new.  
39     seen := RedBlackTree new.  
40  
41     frame forEach: [:aircraft |  
42       | oldPosition newPosition |  
43       oldPosition := state at: aircraft callsign put: aircraft position.  
44       newPosition := aircraft position.  
45       seen at: aircraft callsign put: true.  
46  
47       oldPosition isNil ifTrue: [  
48         "Treat newly introduced aircraft as if they were stationary"  
49         oldPosition := newPosition ].  
50  
51       motions append: (Motion new: aircraft callsign old: oldPosition  
new: newPosition) ].
```

```

52
53 " Remove aircraft that are no longer present "
54 toRemove := Vector new.
55 state forEach: [:e |
56     (seen at: e key) ifFalse: [ toRemove append: e key ] ].
57
58 toRemove forEach: [:e | state remove: e ].
59
60 allReduced := self reduceCollisionSet: motions.
61 collisions := Vector new.
62 allReduced forEach: [:reduced |
63     1 to: reduced size do: [:i |
64         | motion1 |
65         motion1 := reduced at: i.
66         i + 1 to: reduced size do: [:j |
67             | motion2 collision |
68             motion2 := reduced at: j.
69             collision := motion1 findIntersection: motion2.
70             collision notNil ifTrue: [
71                 collisions append: (Collision a: motion1 callsign b: motion2
callsign pos: collision) ] ] ] ].
72
73     ^ collisions
74 )
75
76 isInVoxel: voxel motion: motion = (
77     | init fin v_s r v_x x0 xv v_y y0 yv low_x high_x low_y high_y |
78     (voxel x > Constants MaxX or: [
79         voxel x < Constants MinX or: [
80             voxel y > Constants MaxY or: [
81                 voxel y < Constants MinY ]]) ifTrue: [ ^ false ].
82
83     init := motion posOne.
84     fin := motion posTwo.
85
86     v_s := Constants GoodVoxelSize.
87     r := Constants ProximityRadius // 2.0.
88
89     v_x := voxel x.
90     x0 := init x.
91     xv := fin x - init x.
92
93     v_y := voxel y.
94     y0 := init y.
95     yv := fin y - init y.
96
97     low_x := (v_x - r - x0) // xv.
98     high_x := (v_x + v_s + r - x0) // xv.
99
100     xv < 0.0 ifTrue: [
101         | tmp |
102         tmp := low_x.
103         low_x := high_x.
104         high_x := tmp ].
105
106     low_y := (v_y - r - y0) // yv.
107     high_y := (v_y + v_s + r - y0) // yv.
108
109     yv < 0.0 ifTrue: [
110         | tmp |
111         tmp := low_y.

```

```

112     low_y := high_y.
113     high_y := tmp ].
114
115     ^ (((xv = 0.0 and: [v_x <= (x0 + r) and: [(x0 - r) <= (v_x + v_s)]))
or: [ "no motion in x"
116         (low_x <= 1.0 and: [1.0 <= high_x]) or: [
117         (low_x <= 0.0 and: [0.0 <= high_x]) or: [
118         (0.0 <= low_x and: [high_x <= 1.0])]]]) and: [
119
120         (yv = 0.0 and: [v_y <= (y0 + r) and: [(y0 - r) <= (v_y + v_s)]))
or: [ "no motion in y"
121         (low_y <= 1.0 and: [1.0 <= high_y]) or: [
122         (low_y <= 0.0 and: [0.0 <= high_y]) or: [
123         (0.0 <= low_y and: [high_y <= 1.0])]]]) and: [
124
125         xv = 0.0 or: [
126         yv = 0.0 or: [ "no motion in x or y or both"
127         (low_y <= high_x and: [high_x <= high_y]) or: [
128         (low_y <= low_x and: [low_x <= high_y]) or: [
129         (low_x <= low_y and: [high_y <= high_x]) ]]]])
130     )
131
132     put: motion and: voxel into: voxelMap = (
133     | array |
134     array := voxelMap at: voxel.
135     array isNil ifTrue: [
136     array := Vector new.
137     voxelMap at: voxel put: array ].
138     array append: motion
139     )
140
141     recurse: voxelMap seen: seen voxel: nextVoxel motion: motion = (
142     (self isInVoxel: nextVoxel motion: motion) ifFalse: [ ^ self ].
143     (seen at: nextVoxel put: true) = true ifTrue: [ ^ self ].
144
145     self put: motion and: nextVoxel into: voxelMap.
146
147     self recurse: voxelMap seen: seen voxel: (nextVoxel minus: Constants
horizontal) motion: motion.
148     self recurse: voxelMap seen: seen voxel: (nextVoxel plus: Constants
horizontal) motion: motion.
149     self recurse: voxelMap seen: seen voxel: (nextVoxel minus: Constants
vertical) motion: motion.
150     self recurse: voxelMap seen: seen voxel: (nextVoxel plus: Constants
vertical) motion: motion.
151     self recurse: voxelMap seen: seen voxel: ((nextVoxel minus: Constants
horizontal) minus: Constants vertical) motion: motion.
152     self recurse: voxelMap seen: seen voxel: ((nextVoxel minus: Constants
horizontal) plus: Constants vertical) motion: motion.
153     self recurse: voxelMap seen: seen voxel: ((nextVoxel plus: Constants
horizontal) minus: Constants vertical) motion: motion.
154     self recurse: voxelMap seen: seen voxel: ((nextVoxel plus: Constants
horizontal) plus: Constants vertical) motion: motion.
155     )
156
157     reduceCollisionSet: motions = (
158     | voxelMap result |
159     voxelMap := RedBlackTree new.
160     motions forEach: [:motion | self draw: motion on: voxelMap ].
161
162     result := Vector new.

```

```

163     voxelMap forEach: [:e |
164         e value size > 1 ifTrue: [ result append: e value ] ].
165     ^ result
166 )
167
168 voxelHash: position = (
169     | xDiv yDiv x y |
170     xDiv := (position x // Constants GoodVoxelSize) asInteger.
171     yDiv := (position y // Constants GoodVoxelSize) asInteger.
172
173     x := Constants GoodVoxelSize * xDiv.
174     y := Constants GoodVoxelSize * yDiv.
175
176     position x < 0 ifTrue: [ x := x - Constants GoodVoxelSize ].
177     position y < 0 ifTrue: [ y := y - Constants GoodVoxelSize ].
178
179     ^ Vector2D x: x y: y
180 )
181
182 draw: motion on: voxelMap = (
183     | seen |
184     seen := RedBlackTree new.
185     self recurse: voxelMap seen: seen voxel: (self voxelHash: motion
posOne) motion: motion
186 )
187
188 ----
189
190 new = ( ^ super new initialize )
191
192 )
193

```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 Constants = (  
34     ----  
35     | horizontal vertical |  
36  
37     initialize = (  
38         horizontal := Vector2D x: self GoodVoxelSize y: 0.0.  
39         vertical    := Vector2D x: 0.0 y: self GoodVoxelSize.  
40     )  
41  
42     MinX = ( ^ 0.0 )  
43     MinY = ( ^ 0.0 )  
44     MaxX = ( ^ 1000.0 )  
45     MaxY = ( ^ 1000.0 )  
46     MinZ = ( ^ 0.0 )  
47     MaxZ = ( ^ 10.0 )  
48  
49     ProximityRadius = ( ^ 1.0 )  
50     GoodVoxelSize   = ( "ProximityRadius *" ^ 2.0 )  
51  
52     horizontal = ( ^ horizontal )  
53     vertical   = ( ^ vertical )  
54 )
```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 InsertResult = (  
34   | isNewEntry newNode oldValue |  
35  
36   init: aBool node: aNode value: val = (  
37     isNewEntry := aBool.  
38     newNode     := aNode.  
39     oldValue    := val.  
40   )  
41  
42   isNewEntry = ( ^ isNewEntry )  
43   newNode    = ( ^ newNode    )  
44   oldValue   = ( ^ oldValue   )  
45  
46   ----  
47   new: isNewEntry node: newNode value: oldValue = (  
48     ^ self new init: isNewEntry node: newNode value: oldValue  
49   )  
50 )  
51 )  
52 )  
53 )  
54 )  
55 )  
56 )  
57 )  
58 )  
59 )  
60 )  
61 )  
62 )  
63 )  
64 )  
65 )  
66 )  
67 )  
68 )  
69 )  
70 )  
71 )  
72 )  
73 )  
74 )  
75 )  
76 )  
77 )  
78 )  
79 )  
80 )  
81 )  
82 )  
83 )  
84 )  
85 )  
86 )  
87 )  
88 )  
89 )  
90 )  
91 )  
92 )  
93 )  
94 )  
95 )  
96 )  
97 )  
98 )  
99 )  
100 )
```



```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are  
met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
IS' AND  
19 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
IMPLIED  
20 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
21 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
22 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
23 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
24 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED  
AND  
25 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
26 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
THIS  
27 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
28 "  
29 Motion = (  
30   | callsign posOne posTwo |  
31  
32   init: aCallsign old: aPosOne new: aPosTwo = (  
33     callsign := aCallsign.  
34     posOne   := aPosOne.  
35     posTwo   := aPosTwo.  
36   )  
37  
38   callsign = ( ^ callsign )  
39   posOne   = ( ^ posOne )  
40   posTwo   = ( ^ posTwo )  
41  
42   delta = (  
43     ^ posTwo minus: posOne  
44   )  
45  
46   findIntersection: other = (  
47     | init1 init2 vec1 vec2 radius a dist |  
48     init1 := posOne.  
49     init2 := other posOne.  
50     vec1  := self delta.  
51     vec2  := other delta.  
52     radius := Constants ProximityRadius.
```

```

53
54     " this test is not geometrical 3-d intersection test, it takes the
fact that the aircraft move
55     into account ; so it is more like a 4d test
56     (it assumes that both of the aircraft have a constant speed over
the tested interval)
57
58     we thus have two points, each of them moving on its line segment
at constant speed ; we are looking
59     for times when the distance between these two points is smaller
than r
60
61     vec1 is vector of aircraft 1
62     vec2 is vector of aircraft 2
63
64     a = (V2 - V1)^T * (V2 - V1)"
65     a := (vec2 minus: vec1) squaredMagnitude.
66
67     a <> 0.0 ifTrue: [
68         | b c discr v1 v2 |
69         " we are first looking for instances of time when the planes are
exactly r from each other
70         at least one plane is moving ; if the planes are moving in
parallel, they do not have constant speed
71
72         if the planes are moving in parallel, then
73         if the faster starts behind the slower, we can have 2, 1, or 0
solutions
74         if the faster plane starts in front of the slower, we can have
0 or 1 solutions
75
76         if the planes are not moving in parallel, then
77
78         point P1 = I1 + vV1
79         point P2 = I2 + vV2
80         - looking for v, such that dist(P1,P2) = || P1 - P2 || = r
81
82         it follows that || P1 - P2 || = sqrt( < P1-P2, P1-P2 > )
83         0 = -r^2 + < P1 - P2, P1 - P2 >
84         from properties of dot product
85         0 = -r^2 + <I1-I2,I1-I2> + v * 2<I1-I2, V1-V2> + v^2 *<V1-
V2,V1-V2>
86         so we calculate a, b, c - and solve the quadratic equation
87         0 = c + bv + av^2
88
89         b = 2 * <I1-I2, V1-V2>"
90         b := 2.0 * ((init1 minus: init2) dot: (vec1 minus: vec2)).
91
92         "c = -r^2 + (I2 - I1)^T * (I2 - I1)"
93         c := ((0.0 - radius) * radius) + ((init2 minus: init1)
squaredMagnitude).
94
95         discr := (b * b) - (4.0 * a * c).
96         discr < 0.0 ifTrue: [ ^ nil ].
97
98         v1 := ((0.0 - b) - discr sqrt) // (2.0 * a).
99         v2 := ((0.0 - b) + discr sqrt) // (2.0 * a).
100
101         (v1 <= v2 and: [(v1 <= 1.0 and: [1.0 <= v2]) or: [
102             (v1 <= 0.0 and: [0.0 <= v2]) or: [
103                 (0.0 <= v1 and: [v2 <= 1.0])]])]) ifTrue: [

```

```

104         "Pick a good 'time' at which to report the collision"
105         | v result1 result2 result |
106         v1 <= 0.0
107         ifTrue: [
108             "The collision started before this frame. Report it at the
start of the frame"
109             v := 0.0 ]
110         ifFalse: [
111             "The collision started during this frame. Report it at that
moment"
112             v := v1 ].
113
114         result1 := init1 plus: (vec1 times: v).
115         result2 := init2 plus: (vec2 times: v).
116
117         result := (result1 plus: result2) times: 0.5.
118
119         (result x >= Constants MinX and: [
120             result x <= Constants MaxX and: [
121                 result y >= Constants MinY and: [
122                     result y <= Constants MaxY and: [
123                         result z >= Constants MinZ and: [
124                             result z <= Constants MaxZ ]]]]] ifTrue: [ ^ result ] ].
125
126         ^ nil ].
127
128     " the planes have the same speeds and are moving in parallel (or
they are not moving at all)
129     they thus have the same distance all the time ; we calculate it
from the initial point
130
131     dist = || i2 - i1 || = sqrt( ( i2 - i1 )^T * ( i2 - i1 ) )"
132     dist := (init2 minus: init1) magnitude.
133     dist <= radius ifTrue: [
134         ^ (init1 plus: init2) times: 0.5 ].
135
136     ^ nil
137 )
138
139 ----
140
141 new: callsign old: posOne new: posTwo = (
142     ^ self new init: callsign old: posOne new: posTwo
143 )
144 )

```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 Node = (  
34   | key value left right parent color |  
35  
36   init: aKey value: aValue = (  
37     key    := aKey.  
38     value := aValue.  
39     color := #red.  
40   )  
41   key    = ( ^ key )  
42   value = ( ^ value )  
43   value: val = ( value := val )  
44  
45   left      = ( ^ left )  
46   left: n    = ( left := n )  
47   right     = ( ^ right )  
48   right: n   = ( right := n )  
49   parent    = ( ^ parent )  
50   parent: n  = ( parent := n )  
51   color     = ( ^ color )  
52   color: sym = ( color := sym )  
53  
54   successor = (  
55     | x y |  
56     x := self.
```

```

55     x right notNil ifTrue: [
56         ^ RedBlackTree treeMinimum: x right ].
57
58     y := x parent.
59     [ y notNil and: [ x == y right ]] whileTrue: [
60         x := y.
61         y := y parent ].
62     ^ y
63 )
64
65 ----
66
67 key: key value: value = (
68     ^ self new init: key value: value
69 )
70 )

```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 RbtEntry = (  
34   | key value |  
35  
36   init: aKey value: val = (  
37     key    := aKey.  
38     value := val.  
39   )  
40  
41   key    = ( ^ key )  
42   value = ( ^ value )  
43  
44   ----  
45   key: key value: value = (  
46     ^ self new init: key value: value  
47   )  
48 )  
49 )  
50 )  
51 )  
52 )  
53 )  
54 )  
55 )  
56 )  
57 )  
58 )  
59 )  
60 )  
61 )  
62 )  
63 )  
64 )  
65 )  
66 )  
67 )  
68 )  
69 )  
70 )  
71 )  
72 )  
73 )  
74 )  
75 )  
76 )  
77 )  
78 )  
79 )  
80 )  
81 )  
82 )  
83 )  
84 )  
85 )  
86 )  
87 )  
88 )  
89 )  
90 )  
91 )  
92 )  
93 )  
94 )  
95 )  
96 )  
97 )  
98 )  
99 )  
100 )
```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are  
met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
IS' AND  
19 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
IMPLIED  
20 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
21 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
22 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
23 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
SERVICES;  
24 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED  
AND  
25 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
26 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
THIS  
27 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
28 "  
29 RedBlackTree = (  
30   | root |  
31  
32   at: key put: value = (  
33     | insertionResult x |  
34     insertionResult := self treeAt: key insert: value.  
35     insertionResult isNewEntry ifFalse: [  
36       ^ insertionResult oldValue ].  
37  
38     x := insertionResult newNode.  
39  
40     [ x ~= root and: [ x parent color = #red ]] whileTrue: [  
41       x parent == x parent parent left  
42       ifTrue: [  
43         | y |  
44         y := x parent parent right.  
45         (y notNil and: [y color = #red])  
46         ifTrue: [  
47           "Case 1"  
48           x parent color: #black.  
49           y color: #black.  
50           x parent parent color: #red.  
51           x := x parent parent ]  
52         ifFalse: [  

```

```

53         x == x parent right ifTrue: [
54             "Case 2"
55             x := x parent.
56             self leftRotate: x ].
57
58         "Case 3"
59         x parent color: #black.
60         x parent parent color: #red.
61         self rightRotate: x parent parent ] ]
62     ifFalse: [
63         "Same as 'then' clause with 'right' and 'left' exchanged"
64         | y |
65         y := x parent parent left.
66         (y notNil and: [ y color = #red ])
67         ifTrue: [
68             "Case 1"
69             x parent color: #black.
70             y color: #black.
71             x parent parent color: #red.
72             x := x parent parent ]
73         ifFalse: [
74             x == x parent left ifTrue: [
75                 "Case 2"
76                 x := x parent.
77                 self rightRotate: x ].
78
79                 "Case 3"
80                 x parent color: #black.
81                 x parent parent color: #red.
82                 self leftRotate: x parent parent ] ] ].
83
84     root color: #black.
85     ^ nil
86 )
87
88 remove: key = (
89     | x y z xParent |
90     z := self findNode: key.
91     z isNil ifTrue: [ ^ nil ].
92
93     "Y is the node to be unlinked from the tree."
94     (z left isNil or: [ z right isNil ])
95     ifTrue: [ y := z ]
96     ifFalse: [ y := z successor ].
97
98     "Y is guaranteed to be non-null at this point."
99     y left notNil
100     ifTrue: [ x := y left ]
101     ifFalse: [ x := y right ].
102
103     "X is the child of y which might potentially replace y in the tree.
104     X might be null at this point."
105     x notNil
106     ifTrue: [
107         x parent: y parent.
108         xParent := x parent ]
109     ifFalse: [
110         xParent := y parent ].
111
112     y parent isNil
113     ifTrue: [ root := x ]

```



```

114     ifFalse: [
115         y == y parent left
116         ifTrue: [ y parent left: x ]
117         ifFalse: [ y parent right: x ] ].
118
119     y ~= z ifTrue: [
120         y color = #black ifTrue: [
121             self remove: x andFixup: xParent ].
122
123         y parent: z parent.
124         y color: z color.
125         y left: z left.
126         y right: z right.
127
128         z left notNil ifTrue: [
129             z left parent: y ].
130         x right notNil ifTrue: [
131             z right parent: y ].
132
133         z parent notNil
134         ifTrue: [
135             z parent left == z
136             ifTrue: [ z parent left: y ]
137             ifFalse: [ z parent right: y ] ]
138         ifFalse: [ root := y ] ]
139     ifFalse: [
140         y color = #black ifTrue: [
141             self remove: x andFixup: xParent ] ].
142
143     ^ z value
144 )
145
146 at: key = (
147     | node |
148     node := self findNode: key.
149     node isNil ifTrue: [ ^ nil ].
150     ^ node value
151 )
152
153 forEach: block = (
154     | current |
155     root isNil ifTrue: [ ^ self ].
156     current := RedBlackTree treeMinimum: root.
157     [ current notNil ] whileTrue: [
158         block value: (RbtEntry key: current key value: current value).
159         current := current successor ].
160 )
161
162 findNode: key = (
163     | current |
164     current := root.
165     [ current notNil ] whileTrue: [
166         | comparisonResult |
167         comparisonResult := key compareTo: current key.
168         comparisonResult = 0 ifTrue: [ ^ current ].
169         comparisonResult < 0
170         ifTrue: [ current := current left ]
171         ifFalse: [ current := current right ] ].
172     ^ nil
173 )
174

```

```

175 treeAt: key insert: value = (
176   | x y z |
177   y := nil.
178   x := root.
179
180   [ x notNil ] whileTrue: [
181     | comparisonResult |
182     y := x.
183     comparisonResult := key compareTo: x key.
184     comparisonResult < 0
185       ifTrue: [ x := x left ]
186       ifFalse: [
187         comparisonResult > 0
188           ifTrue: [ x := x right ]
189           ifFalse: [
190             | oldValue |
191             oldValue := x value.
192             x value: value.
193             ^ InsertResult new: false node: nil value: oldValue ] ] ].
194
195   z := Node key: key value: value.
196   z parent: y.
197   y isNil
198     ifTrue: [ root := z ]
199     ifFalse: [
200       (key compareTo: y key) < 0
201         ifTrue: [ y left: z ]
202         ifFalse: [ y right: z ] ].
203
204   ^ InsertResult new: true node: z value: nil
205 )
206
207 leftRotate: x = (
208   | y |
209   y := x right.
210
211   "Turn y's left subtree into x's right subtree"
212   x right: y left.
213   y left notNil ifTrue: [
214     y left parent: x ].
215
216   "Link x's parent to y"
217   y parent: x parent.
218   x parent isNil
219     ifTrue: [ root := y ]
220     ifFalse: [
221       x == x parent left ifTrue: [ x parent left: y ]
222       ifFalse: [ x parent right: y ] ].
223
224   "Put x on y's left"
225   y left: x.
226   x parent: y.
227   ^ y
228 )
229
230 rightRotate: y = (
231   | x |
232   x := y left.
233
234   "Turn x's right subtree into y's left subtree"
235   y left: x right.

```

```

236     x right notNil ifTrue: [ x right parent: y ].
237
238     "Link y's parent to x"
239     x parent: y parent.
240     y parent isNil
241         ifTrue: [ root := x ]
242         ifFalse: [
243             y == y parent left
244                 ifTrue: [ y parent left: x ]
245                 ifFalse: [ y parent right: x ] ].
246
247     x right: y.
248     y parent: x.
249     ^ x
250 )
251
252 remove: anX andFixup: anXParent = (
253     | x xParent |
254     x := anX.
255     xParent := anXParent.
256
257     x ~= root and: [ x isNil or: [ x color = #black ]] whileTrue: [
258         x == xParent left
259             ifTrue: [
260                 | w |
261                 "Note: the text points out that w cannot be null. The reason
is not obvious from
262                 simply looking at the code; it comes about from the
properties of the red-black
263                 tree."
264                 w := xParent right.
265                 w color = #red ifTrue: [
266                     "Case 1"
267                     w color: #black.
268                     xParent color: #red.
269                     self leftRotate: xParent.
270                     w := xParent right ].
271
272                 ((w left isNil or: [ w left color = #black ]) and: [
273                     w right isNil or: [ w right color = #black ]])
274                 ifTrue: [
275                     "Case 2"
276                     w color: #red.
277                     x := xParent.
278                     xParent := x parent ]
279                 ifFalse: [
280                     (w right isNil or: [ w right color = #black ]) ifTrue: [
281                         "Case 3"
282                         w left color: #black.
283                         w color: #red.
284                         self rightRotate: w.
285                         w := xParent right ].
286
287                     "Case 4"
288                     w color: xParent color.
289                     xParent color: #black.
290                     w right notNil ifTrue: [
291                         w right color: #black ].
292
293                     self leftRotate: xParent.
294                     x := root.

```

```

295         xParent := x parent ] ]
296     ifFalse: [
297         | w |
298         "Same as 'then' clause with 'right' and 'left' exchanged"
299         w := xParent left.
300         w color = #red ifTrue: [
301             "Case 1"
302             w color: #black.
303             xParent color: #red.
304             self rightRotate: xParent.
305             w := xParent left ].
306
307         ((w right isNil or: [ w right color = #black ]) and: [
308             (w left isNil or: [ w left color = #black ])]
309         ifTrue: [
310             "Case 2"
311             w color: #red.
312             x := xParent.
313             xParent := x parent ]
314         ifFalse: [
315             w left isNil or: [ w left color = #black ] ifTrue: [
316                 "Case 3"
317                 w right color: #black.
318                 w color: #red.
319                 self leftRotate: w.
320                 w := xParent left ].
321
322                 "Case 4"
323                 w color: xParent color.
324                 xParent color: #black.
325                 w left notNil ifTrue: [
326                     w left color: #black ].
327
328                 self rightRotate: xParent.
329                 x := root.
330                 xParent = x parent ] ] ].
331
332     x notNil ifTrue: [ x color: #black ]
333 )
334
335 ----
336
337 treeMinimum: x = (
338     | current |
339     current := x.
340     [ current left notNil ] whileTrue: [
341         current := current left ].
342     ^ current
343 )
344 )

```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 Simulator = (  
34   | aircrafts |  
35  
36   init: numAircrafts = (  
37     aircrafts := Vector new.  
38  
39     0 to: numAircrafts - 1 do: [:i |  
40       aircrafts append: (CallSign new: i)]  
41     )  
42  
43   simulate: time = (  
44     | frame |  
45     frame := Vector new.  
46     0 to: aircrafts size - 2 by: 2 do: [:i |  
47       frame append: (Aircraft new: (aircrafts at: i + 1)  
48         pos: (Vector3D x: time  
49           y: (time cos * 2.0) + (i *  
50             3.0)  
51             z: 10.0)).  
52       frame append: (Aircraft new: (aircrafts at: i + 2)  
53         pos: (Vector3D x: time  
54           y: (time sin * 2.0) + (i *  
55             3.0)  
56             z: 10.0)) ].  
57     ^ frame  
58   )  
59 )
```

```
53
54 ----
55
56 new: numAircrafts = (
57   ^ self new init: numAircrafts
58 )
59 )
60
```

```
1 "  
2 Ported from the adapted JavaScript and Java versions.  
3  
4 Copyright (c) 2001-2010, Purdue University. All rights reserved.  
5 Copyright (C) 2015 Apple Inc. All rights reserved.  
6  
7 Redistribution and use in source and binary forms, with or without  
8 modification, are permitted provided that the following conditions are met:  
9 * Redistributions of source code must retain the above copyright  
10 notice, this list of conditions and the following disclaimer.  
11 * Redistributions in binary form must reproduce the above copyright  
12 notice, this list of conditions and the following disclaimer in the  
13 documentation and/or other materials provided with the distribution.  
14 * Neither the name of the Purdue University nor the  
15 names of its contributors may be used to endorse or promote products  
16 derived from this software without specific prior written permission.  
17  
18 THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS  
19 IS' AND  
20 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE  
21 IMPLIED  
22 WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
23 DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY  
24 DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES  
25 (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR  
26 SERVICES;  
27 LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND  
28 ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT  
29 (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF  
30 THIS  
31 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.  
32 "  
33 Vector2D = (  
34   | x y |  
35  
36   x = ( ^ x )  
37   y = ( ^ y )  
38  
39   initX: anX y: aY = (  
40     x := anX.  
41     y := aY  
42   )  
43  
44   plus: other = (  
45     ^ Vector2D x: x + other x  
46     y: y + other y  
47   )  
48  
49   minus: other = (  
50     ^ Vector2D x: x - other x  
51     y: y - other y  
52   )  
53  
54   compareTo: other = (  
55     | result |  
56     result := self compare: x and: other x.  
57     result <> 0 ifTrue: [ ^ result ].  
58     ^ self compare: y and: other y
```

```

55 )
56
57 compare: a and: b = (
58     a = b ifTrue: [ ^ 0 ].
59     a < b ifTrue: [ ^ -1 ].
60     a > b ifTrue: [ ^ 1 ].
61
62     "We say that NaN is smaller than non-NaN."
63     a = a ifTrue: [ ^ 1 ].
64     ^ -1
65 )
66
67 ----
68
69 x: anX y: aY = (
70     ^ self new initX: anX y: aY
71 )
72 )
73

```



```

1  "
2  Ported from the adapted JavaScript and Java versions.
3
4  Copyright (c) 2001-2010, Purdue University. All rights reserved.
5  Copyright (C) 2015 Apple Inc. All rights reserved.
6
7  Redistribution and use in source and binary forms, with or without
8  modification, are permitted provided that the following conditions are met:
9  * Redistributions of source code must retain the above copyright
10  notice, this list of conditions and the following disclaimer.
11  * Redistributions in binary form must reproduce the above copyright
12  notice, this list of conditions and the following disclaimer in the
13  documentation and/or other materials provided with the distribution.
14  * Neither the name of the Purdue University nor the
15  names of its contributors may be used to endorse or promote products
16  derived from this software without specific prior written permission.
17
18  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS
19  IS' AND
20  ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
21  IMPLIED
22  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
23  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
24  DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
25  (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
26  SERVICES;
27  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
28  ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
29  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
30  THIS
31  SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32  "
33  Vector3D = (
34  | x y z |
35
36  x = ( ^ x )
37  y = ( ^ y )
38  z = ( ^ z )
39
40  initX: anX y: aY z: aZ = (
41  x := anX.
42  y := aY.
43  z := aZ
44  )
45
46  plus: other = (
47  ^ Vector3D x: x + other x
48  y: y + other y
49  z: z + other z
50  )
51
52  minus: other = (
53  ^ Vector3D x: x - other x
54  y: y - other y
55  z: z - other z
56  )
57
58  dot: other = (

```

```

55     ^ (x * other x) + (y * other y) + (z * other z)
56 )
57
58 squaredMagnitude = (
59     ^ self dot: self
60 )
61
62 magnitude = (
63     ^ self squaredMagnitude sqrt
64 )
65
66 times: amount = (
67     ^ Vector3D x: x * amount
68                 y: y * amount
69                 z: z * amount
70 )
71
72 ----
73
74 x: x y: y z: z = (
75     ^ self new initX: x y: y z: z
76 )
77 )

```

```
1 "  
2 Copyright (c) 2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 DictEntry = (  
24   | hash key value next |  
25  
26   init: aHash key: aKey value: val next: anEntry = (  
27     hash  := aHash.  
28     key   := aKey.  
29     value := val.  
30     next  := anEntry.  
31   )  
32  
33   hash = ( ^ hash )  
34   key  = ( ^ key  )  
35  
36   value = ( ^ value )  
37   value: val = ( value := val )  
38  
39   next = ( ^ next )  
40   next: e = ( next := e )  
41  
42   match: aHash key: aKey = (  
43     ^ hash = aHash and: [key = aKey]  
44   )  
45  
46   ----  
47  
48   new: hash key: key value: val next: next = (  
49     ^ self new init: hash key: key value: val next: next  
50   )  
51 )  
52
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 DictIdEntry = DictEntry (  
23  
24   match: aHash key: aKey = (  
25     ^ hash = aHash and: [key == aKey]  
26   )  
27  
28   ----  
29  
30   new: hash key: key value: val next: next = (  
31     ^ self new init: hash key: key value: val next: next  
32   )  
33 )  
34
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 Pair = (  
24   | key value |  
25  
26   initialize: aKey and: aValue = (  
27     key    := aKey.  
28     value  := aValue.  
29   )  
30  
31   key    = ( ^ key )  
32   value  = ( ^ value )  
33  
34   key:    aKey    = ( key    := aKey )  
35   value:  aValue  = ( value  := aValue )  
36  
37   ----  
38  
39   withKey: aKey andValue: aValue = (  
40     ^ self new initialize: aKey and: aValue  
41   )  
42 )  
43
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 SomDictionary = (  
24   | buckets size_ |  
25  
26   initialize: size = (  
27     buckets := Array new: size.  
28     size_   := 0  
29   )  
30  
31   hash: key = (  
32     | hash |  
33     key isNil ifTrue: [ ^ 0 ].  
34     hash := key customHash.  
35     ^ hash bitXor: (hash >>> 16)  
36   )  
37  
38   bucketIdx: hash = (  
39     ^ 1 + ((buckets length - 1) & hash)  
40   )  
41  
42   bucket: hash = (  
43     ^ buckets at: (self bucketIdx: hash)  
44   )  
45  
46   at: aKey = (  
47     | hash e |  
48     hash := self hash: aKey.  
49     e    := self bucket: hash.  
50  
51     [ e notNil ] whileTrue: [  
52       (e match: hash key: aKey)  
53       ifTrue: [ ^ e value ].
```

```

54     e := e next ].
55     ^ nil
56 )
57
58 containsKey: aKey = (
59     | hash e |
60     hash := self hash: aKey.
61     e     := self bucket: hash.
62
63     [ e notNil ] whileTrue: [
64         (e match: hash key: aKey)
65         ifTrue: [ ^ true ].
66         e := e next ].
67     ^ false
68 )
69
70 at: aKey put: aVal = (
71     | hash i current |
72     hash := self hash: aKey.
73     i     := self bucketIdx: hash.
74     current := buckets at: i.
75
76     current isNil
77     ifTrue: [
78         buckets at: i put: (self newEntry: aKey value: aVal hash: hash).
79         size_ := size_ + 1 ]
80     ifFalse: [
81         self insertBucketEntry: aKey value: aVal hash: hash head:
current ].
82
83     size_ > buckets length ifTrue: [ self resize ]
84 )
85
86 newEntry: aKey value: value hash: hash = (
87     ^ DictEntry new: hash key: aKey value: value next: nil
88 )
89
90 insertBucketEntry: key value: value hash: hash head: head = (
91     | current |
92     current := head.
93
94     [true] whileTrue: [
95         (current match: hash key: key) ifTrue: [
96             current value: value.
97             ^ self ].
98         current next isNil ifTrue: [
99             size_ := size_ + 1.
100             current next: (self newEntry: key value: value hash: hash).
101             ^ self ].
102         current := current next ]
103 )
104
105 resize = (
106     | oldStorage |
107     oldStorage := buckets.
108     buckets := Array new: oldStorage length * 2.
109     self transferEntries: oldStorage
110 )
111
112 transferEntries: oldStorage = (
113     1 to: oldStorage length do: [:i |

```

```

114         | current |
115         current := oldStorage at: i.
116         current notNil ifTrue: [
117             oldStorage at: i put: nil.
118             current next isNil
119             ifTrue: [
120                 buckets at: 1 + (current hash & (buckets length - 1)) put:
current ]
121             ifFalse: [
122                 self splitBucket: oldStorage bucket: i head: current ] ] ]
123     )
124
125     splitBucket: oldStorage bucket: i head: head = (
126         | loHead loTail hiHead hiTail current |
127         loHead := nil. loTail := nil.
128         hiHead := nil. hiTail := nil.
129         current := head.
130
131         [current notNil] whileTrue: [
132             (current hash & oldStorage length) = 0
133             ifTrue: [
134                 loTail isNil
135                 ifTrue: [ loHead := current ]
136                 ifFalse: [ loTail next: current ].
137                 loTail := current ]
138             ifFalse: [
139                 hiTail isNil
140                 ifTrue: [ hiHead := current ]
141                 ifFalse: [ hiTail next: current ].
142                 hiTail := current ].
143             current := current next ].
144
145         loTail notNil ifTrue: [
146             loTail next: nil.
147             buckets at: i put: loHead ].
148         hiTail notNil ifTrue: [
149             hiTail next: nil.
150             buckets at: i + oldStorage length put: hiHead ]
151     )
152
153     size      = ( ^ size_ )
154     isEmpty   = ( ^ size_ = 0 )
155     removeAll = (
156         buckets := Array new: buckets length.
157         size_ := 0.
158     )
159
160     keys = (
161         | keys |
162         keys := Vector new: size_.
163         buckets do: [:b |
164             | current |
165             current := b.
166             [ current notNil ] whileTrue: [
167                 keys append: current key.
168                 current := current next ] ].
169         ^ keys
170     )
171
172     values = (
173         | values |

```



```

174     values := Vector new: size_.
175     buckets do: [:b |
176         | current |
177         current := b.
178         [ current notNil ] whileTrue: [
179             values append: current value.
180             current := current next ] ].
181     ^ values
182 )
183
184 ----
185
186 new: size = (
187     ^ super new initialize: size
188 )
189
190 new = (
191     ^ self new: 16
192 )
193 )
194

```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 SomIdentityDictionary = SomDictionary (  
24  
25   newEntry: aKey value: value hash: hash = (  
26     ^ DictIdEntry new: hash key: aKey value: value next: nil  
27   )  
28  
29   ----  
30  
31   new: size = (  
32     ^ self new: size  
33   )  
34  
35   new = ( ^ super new: 16 )  
36 )  
37
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 SomIdentitySet = SomSet (  
23   contains: anObject = (  
24     ^ self hasSome: [ :it | it == anObject ]  
25   )  
26   ----  
27  
28   new: size = (  
29     ^ super new initialize: size.  
30   )  
31 )  
32
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 SomSet = (  
24  
25   | items |  
26  
27   initialize: size = (  
28     items := Vector new: size.  
29   )  
30  
31   forEach: block = ( items forEach: block )  
32   hasSome: block = ( ^ items hasSome: block )  
33   getOne:  block = ( ^ items getOne:  block )  
34  
35   add: anObject = (  
36     (self contains: anObject)  
37     ifFalse: [ items append: anObject ]  
38   )  
39  
40   collect: block = ( | coll |  
41     coll := Vector new.  
42     self forEach: [ :e | coll append: (block value: e) ].  
43     ^ coll  
44   )  
45  
46   contains: anObject = (  
47     ^ self hasSome: [ :it | it = anObject ]  
48   )  
49  
50   size = ( ^ items size )  
51   removeAll = ( ^ items removeAll )  
52  
53   ----  
54
```

```
55   new = (  
56       ^ super new initialize: 10.  
57   )  
58 )  
59
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 Vector = (  
24  
25   | first last storage |  
26  
27   initialize: size = (  
28     first := 1.  
29     last  := 1.  
30     storage := Array new: size.  
31   )  
32  
33   at: index = (  
34     index > storage length ifTrue: [ ^ nil ].  
35     ^ storage at: index  
36   )  
37  
38   at: index put: val = (  
39     index > storage length ifTrue: [  
40       | newLength newStorage |  
41       newLength := storage length.  
42       [ newLength < index ] whileTrue: [  
43         newLength := newLength * 2 ].  
44       newStorage := Array new: newLength.  
45       storage doIndexes: [:i | newStorage at: i put: (storage at: i) ].  
46       storage := newStorage ].  
47  
48     storage at: index put: val.  
49     last < (index + 1) ifTrue: [  
50       last := index + 1 ]  
51   )  
52  
53   append: element = (  

```

```

54     (last > storage length) ifTrue: [
55         "Need to expand capacity first"
56         | newStorage |
57         newStorage := Array new: (2 * storage length).
58         storage doIndexes: [ :i | newStorage at: i put: (storage at: i) ].
59         storage := newStorage. ].
60
61     storage at: last put: element.
62     last := last + 1.
63     ^ self
64 )
65
66 isEmpty = ( ^ last = first )
67
68 forEach: block = (
69     first to: last - 1 do: [ :i | block value: (storage at: i) ]
70 )
71
72 hasSome: block = (
73     first to: last - 1 do: [ :i |
74         (block value: (storage at: i))
75         ifTrue: [ ^ true ] ].
76     ^ false
77 )
78
79 getOne: block = (
80     first to: last - 1 do: [ :i |
81         | e |
82         e := storage at: i.
83         (block value: e)
84         ifTrue: [ ^ e ] ].
85     ^ nil
86 )
87
88 removeFirst = (
89     self isEmpty ifTrue: [ ^ nil ].
90     first := first + 1.
91     ^ storage at: first - 1
92 )
93
94 removeAll = (
95     first := 1.
96     last := 1.
97     storage := Array new: storage length
98 )
99
100 remove: object = (
101     | newArray newLast found |
102     newArray := Array new: self capacity.
103     newLast := 1.
104     found := false.
105
106     self forEach: [ :it |
107         it == object
108             ifTrue: [ found := true ]
109             ifFalse: [
110                 newArray at: newLast put: it.
111                 newLast := newLast + 1 ] ].
112
113     storage := newArray.
114     last := newLast.

```

```

115     first := 1.
116     ^ found
117 )
118
119 size      = ( ^ last - first )
120 capacity = ( ^ storage length )
121
122 sort: aBlock = (
123     " Make the argument, aBlock, be the criterion for ordering elements of
124     the receiver.
125     sortBlocks with side effects may not work right "
126     self size > 0 ifTrue: [
127         self sort: first
128             to: last - 1
129             with: aBlock ]
130 )
131
132 sort: i to: j with: sortBlock = (
133     " Sort elements i through j of self to be non-descending according to
134     sortBlock. "
135     | di dij dj tt ij k l n |
136     sortBlock isNil ifTrue: [ ^ self defaultSort: i to: j ].
137
138     "The prefix d means the data at that index."
139     (n := j + 1 - i) <= 1 ifTrue: [ ^ self ]. "Nothing to sort."
140     " Sort di,dj. "
141     di := storage at: i.
142     dj := storage at: j.
143
144     "i.e., should di precede dj?"
145     (sortBlock value: di with: dj) ifFalse: [
146         storage swap: i with: j.
147         tt := di.
148         di := dj.
149         dj := tt
150     ].
151
152     n > 2 ifTrue: [ " More than two elements. "
153         ij := (i + j) / 2. " ij is the midpoint of i and j. "
154         dij := storage at: ij. " Sort di,dij,dj. Make dij be their
median. "
155         (sortBlock value: di with: dij)
156         ifTrue: [ " i.e. should di precede dij? "
157             (sortBlock value: dij with: dj) ifFalse: [ " i.e., should dij
precede dj? "
158                 storage swap: j with: ij.
159                 dij := dj]]
160         ifFalse: [ " i.e. di should come after dij "
161             storage swap: i with: ij.
162             dij := di].
163
164         n > 3 ifTrue: [
165             " More than three elements.
166             Find k>i and l<j such that dk,dij,dl are in reverse order.
167             Swap k and l. Repeat this procedure until k and l pass each
other. "
168             k := i.
169             l := j.
170             [ [l := l - 1. k <= l and: [sortBlock value: dij with: (storage
at: l)]]]
171                 whileTrue. " i.e. while dl succeeds dij "

```



```

172         [k := k + 1.  k <= 1 and: [sortBlock value: (storage at: k)
with: dij]]
173         whileTrue.  " i.e. while dij succeeds dk "
174         k <= 1]
175         whileTrue:
176             [ storage swap: k with: 1 ].
177
178         " Now l<k (either 1 or 2 less), and di through dl are all less
than or equal to dk
179         through dj.  Sort those two segments. "
180         self sort: i to: l with: sortBlock.
181         self sort: k to: j with: sortBlock
182     ]
183 ]
184 )
185
186 -----
187
188 "Allocation"
189 new          = ( ^ self new: 50 )
190 new: initialSize = ( ^ super new initialize: initialSize )
191
192 with: elem = (
193     | newVector |
194     newVector := self new: 1.
195     newVector append: elem.
196     ^ newVector
197 )
198 )
199

```

```
1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java\_benchmarking/index.html
7 "
8 AbstractConstraint = (
9   "I am an abstract class representing a system-maintainable relationship
(or
10   'constraint') between a set of variables. I supply a strength instance
11   variable; concrete subclasses provide a means of storing the constrained
12   variables and other information required to represent a constraint.
13
14   Instance variables:
15       strength          the strength of this constraint <Strength>"
16   | strength |
17
18   initialize: strengthSymbol = (
19     strength := Strength of: strengthSymbol
20   )
21
22   "accessing"
23
24   strength = (
25     "Answer my strength."
26     ^ strength
27   )
28
29   "queries"
30   isInput = (
31     "Normal constraints are not input constraints. An input constraint is
32     one that depends on external state, such as the mouse, the keyboard,
33     a clock, or some arbitrary piece of imperative code."
34     ^ false
35   )
36
37   isSatisfied = (
38     "Answer true if this constraint is satisfied in the current solution."
39     self subclassResponsibility
40   )
41
42   "add/remove"
43   addConstraint: planner = (
44     "Activate this constraint and attempt to satisfy it."
45
46     self addToGraph.
47     planner incrementalAdd: self.
48   )
49
50   addToGraph = (
51     "Add myself to the constraint graph."
52     self subclassResponsibility
53   )
54
55   destroyConstraint: planner = (
```

```

56     "Deactivate this constraint, remove it from the constraint graph,
57     possibly causing other constraints to be satisfied, and destroy it."
58
59     self isSatisfied ifTrue: [planner incrementalRemove: self].
60     self removeFromGraph.
61 )
62
63 removeFromGraph = (
64     "Remove myself from the constraint graph."
65     self subclassResponsibility
66 )
67
68 "planning"
69 chooseMethod: mark = (
70     "Decide if I can be satisfied and record that decision. The output of
71     the chosen method must not have the given mark and must have a
72     walkabout strength less than that of this constraint."
73     self subclassResponsibility
74 )
75
76 execute = (
77     "Enforce this constraint. Assume that it is satisfied."
78     self subclassResponsibility
79 )
80
81 inputsDo: aBlock = (
82     "Assume that I am satisfied. Evaluate the given block on all my
current
83     input variables."
84     self subclassResponsibility
85 )
86
87 inputsKnown: mark = (
88     "Assume that I am satisfied. Answer true if all my current inputs are
89     known. A variable is known if either a) it is 'stay' (i.e. it is a
90     constant at plan execution time), b) it has the given mark
(indicating
91     that it has been computed by a constraint appearing earlier in the
92     plan), or c) it is not determined by any constraint."
93
94     ^ (self inputsHasOne: [:v |
95         (v mark = mark or: [v stay or: [v determinedBy == nil]]) not ]) not
96 )
97
98 markUnsatisfied = (
99     "Record the fact that I am unsatisfied."
100     self subclassResponsibility
101 )
102
103 output = (
104     "Answer my current output variable. Raise an error if I am not
105     currently satisfied."
106     self subclassResponsibility
107 )
108
109 recalculate = (
110     "Calculate the walkabout strength, the stay flag, and, if it is
'stay',
111     the value for the current output of this constraint. Assume this
112     constraint is satisfied."
113     self subclassResponsibility

```

```

114     )
115
116     satisfy: mark propagate: planner = (
117         "Attempt to find a way to enforce this (still unsatisfied) constraint.
118         If successful, record the solution, perhaps modifying the current
119         dataflow graph. Answer the constraint that this constraint
overrides,
120         if there is one, or nil, if there isn't."
121
122         | overridden |
123         self chooseMethod: mark.
124         self isSatisfied
125             ifTrue: "constraint can be satisfied"
126                 ["mark inputs to allow cycle detection in addPropagate"
127                     | out |
128                     self inputsDo: [: in | in mark: mark].
129                     out := self output.
130                     overridden := out determinedBy.
131                     (overridden == nil) ifFalse: [overridden markUnsatisfied].
132                     out determinedBy: self.
133                     (planner addPropagate: self mark: mark) ifFalse:
134                         [self error: 'Cycle encountered adding:\tConstraint
removed.'.
135                             ^nil].
136                     out mark: mark]
137                 ifFalse: "constraint cannot be satisfied"
138                     [overridden := nil.
139                     (strength sameAs: (Strength required)) ifTrue:
140                         [self error: 'Failed to satisfy a required constraint']].
141         ^ overridden
142     )
143 )
144

```

```

1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 BinaryConstraint = AbstractConstraint (
9   "I am an abstract superclass for constraints having two possible output
10  variables.
11
12  Instance variables:
13    v1, v2      possible output variables <Variable>
14    direction    one of:
15                  #forward (v2 is output)
16                  #backward ( v1 is output)
17                  nil (not satisfied)"
18  | v1 v2 direction |
19
20  "initialize-release"
21
22  initializeVar: variable1 var: variable2 strength: strengthSymbol addTo:
planner = (
23    "Initialize myself with the given variables and strength."
24    super initialize: strengthSymbol.
25
26    v1 := variable1.
27    v2 := variable2.
28    direction := nil.
29  )
30
31  "queries"
32  isSatisfied = (
33    "Answer true if this constraint is satisfied in the current solution."
34
35    ^ direction notNil
36  )
37
38  "add/remove"
39  addToGraph = (
40    "Add myself to the constraint graph."
41    v1 addConstraint: self.
42    v2 addConstraint: self.
43    direction := nil
44  )
45
46  removeFromGraph = (
47    "Remove myself from the constraint graph."
48    (v1 == nil) ifFalse: [v1 removeConstraint: self].
49    (v2 == nil) ifFalse: [v2 removeConstraint: self].
50    direction := nil.
51  )
52
53  "planning"
54  chooseMethod: mark = (
55    "Decide if I can be satisfied and which way I should flow based on

```

```

56     the relative strength of the variables I relate, and record that
57     decision."
58
59     (v1 mark = mark) ifTrue:          "forward or nothing"
60         [((v2 mark <> mark) and: [strength stronger: v2 walkStrength])
61             ifTrue: [ ^ direction := #forward ]
62             ifFalse: [ ^ direction := nil ] ].
63
64     (v2 mark = mark) ifTrue:          "backward or nothing"
65         [((v1 mark <> mark) and: [strength stronger: v1 walkStrength])
66             ifTrue: [ ^ direction := #backward ]
67             ifFalse: [ ^ direction := nil ] ].
68
69     "if we get here, neither variable is marked, so we have choice"
70     (v1 walkStrength weaker: v2 walkStrength)
71         ifTrue:
72             [(strength stronger: v1 walkStrength)
73                 ifTrue: [ ^ direction := #backward ]
74                 ifFalse: [ ^ direction := nil ] ]
75         ifFalse:
76             [(strength stronger: v2 walkStrength)
77                 ifTrue: [ ^ direction := #forward ]
78                 ifFalse: [ ^ direction := nil ] ].
79 )
80
81 execute = (
82     "Enforce this constraint. Assume that it is satisfied."
83     self subclassResponsibility
84 )
85
86 inputsDo: aBlock = (
87     "Evaluate the given block on my current input variable."
88     direction = #forward
89         ifTrue: [ aBlock value: v1 ]
90         ifFalse: [ aBlock value: v2 ].
91 )
92
93 inputsHasOne: aBlock = (
94     ^ direction = #forward
95         ifTrue: [ aBlock value: v1 ]
96         ifFalse: [ aBlock value: v2 ]
97 )
98
99 markUnsatisfied = (
100     "Record the fact that I am unsatisfied."
101     direction := nil.
102 )
103
104 output = (
105     "Answer my current output variable."
106     (direction == #forward)
107         ifTrue: [ ^ v2 ]
108         ifFalse: [ ^ v1 ]
109 )
110
111 recalculate = (
112     "Calculate the walkabout strength, the stay flag, and, if it is
113     'stay',
114     the value for the current output of this constraint. Assume this
115     constraint is satisfied."

```

```
116     | in out |
117     (direction = #forward)
118         ifTrue: [ in := v1. out := v2 ]
119         ifFalse: [ in := v2. out := v1 ].
120     out walkStrength: (strength weakest: in walkStrength).
121     out stay: in stay.
122     out stay ifTrue: [self execute].          "stay optimization"
123 )
124 )
125
```

```

1 "
2 This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
3
4 It is modified to use the SOM class library.
5 License details:
6
7 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
8
9 Original comment.
10 "ä ÖY"FVÇF &ÇVR &Væ6†Ö &°
11 " UD„ö)"|ö†â Ö ÆöæW'Â †V f-Ç' ÖöF-f-VB ' ' Ö &-ò vöÆ7|¶ù
(Mario.Wolczko@sun.com)
12 "eTä5D"ôéLanguage implementation benchmark
13 •5BÖdU%4"ôé4.1
14 • $U$U T•4•DU9
15 "4ôädÄ"5E9
16 "D•5E$"%UD"ôéworld
17 •dU%4"ôé"
18 "D DY "" Ö7B ""`
19 SUMMARY
20 This benchmark is an implementation of the DeltaBlue Constraint Solver
21 described in `The DeltaBlue Algorithm: An Incremental Constraint
22 Hierarchy Solver', by Bjorn N. Freeman-Benson and John Maloney,
23 Communications of the ACM, January 1990 (also as University of
24 Washington TR 89-08-06)
25 "
26 DeltaBlue = Benchmark (
27
28     innerBenchmarkLoop: innerIterations = (
29         Planner chainTest: innerIterations.
30         Planner projectionTest: innerIterations.
31         ^ true
32     )
33
34     ----
35
36     new = (
37         Strength initialize.
38         ^ super new
39     )
40 )
41

```



```
1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 EditConstraint = UnaryConstraint (
9   "I am a unary input constraint used to mark a variable that the client
10   wishes to change."
11
12   "queries"
13   isInput = (
14     "I indicate that a variable is to be changed by imperative code."
15     ^ true
16   )
17
18   "execution"
19   execute = (
20     "Edit constraints do nothing."
21   )
22
23   ----
24
25   "instance creation"
26
27   var: aVariable strength: strengthSymbol addTo: planner = (
28     "Install an edit constraint with the given strength on the given
29     variable."
30
31     ^ self new initializeVar: aVariable strength: strengthSymbol addTo:
planner
32   )
33 )
34
```

```
1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 EqualityConstraint = BinaryConstraint (
9   "I constrain two variables to have the same value: `v1 = v2`."
10
11   initializeVar: variable1 var: variable2 strength: strengthSymbol addTo:
planner = (
12     super initializeVar: variable1 var: variable2 strength: strengthSymbol
addTo: planner.
13     self addConstraint: planner.
14   )
15
16   "execution"
17   execute = (
18     "Enforce this constraint. Assume that it is satisfied."
19     direction = #forward
20     ifTrue: [ v2 value: v1 value ]
21     ifFalse: [ v1 value: v2 value ].
22   )
23
24   ----
25
26   "instance creation"
27
28   var: variable1 var: variable2 strength: strengthSymbol addTo: planner = (
29     "Install a constraint with the given strength equating the given
30     variables."
31
32     ^ self new initializeVar: variable1 var: variable2 strength:
strengthSymbol addTo: planner
33   )
34 )
35
```

```
1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java\_benchmarking/index.html
7 "
8 Plan = Vector (
9   "A Plan is an ordered list of constraints to be executed in sequence to
10   resatisfy all currently satisfiable constraints in the face of one or
more
11   changing inputs."
12
13   "execution"
14   execute = (
15     "Execute my constraints in order."
16
17     self forEach: [: c | c execute ]
18   )
19
20   ----
21
22   new = ( ^ self new: 15 )
23 )
24
```

```
1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java\_benchmarking/index.html
7 "
8 Planner = (
9   "This benchmark is an implementation of the DeltaBlue Constraint Solver
10   described in `The DeltaBlue Algorithm: An Incremental Constraint
11   Hierarchy Solver', by Bjorn N. Freeman-Benson and John Maloney,
12   Communications of the ACM, January 1990 (also as University of
13   Washington TR 89-08-06).
14
15   To run the benchmark, execute the expression `Planner
standardBenchmark`."
16   | currentMark |
17
18   initialize = (
19     currentMark := 1
20   )
21
22   "add/remove"
23
24   incrementalAdd: c = (
25     "Attempt to satisfy the given constraint and, if successful,
26     incrementally update the dataflow graph.
27
28     Details: If satisfying the constraint is successful, it may override a
29     weaker constraint on its output. The algorithm attempts to resatisfy
30     that constraint using some other method. This process is repeated
31     until either a) it reaches a variable that was not previously
32     determined by any constraint or b) it reaches a constraint that
33     is too weak to be satisfied using any of its methods. The variables
34     of constraints that have been processed are marked with a unique mark
35     value so that we know where we've been. This allows the algorithm to
36     avoid getting into an infinite loop even if the constraint graph has
37     an inadvertent cycle."
38
39     | mark overridden |
40     mark := self newMark.
41     overridden := c satisfy: mark propagate: self.
42     [overridden == nil] whileFalse:
43       [overridden := overridden satisfy: mark propagate: self]
44   )
45
46   incrementalRemove: c = (
47     "Entry point for retracting a constraint. Remove the given constraint,
48     which should be satisfied, and incrementally update the dataflow
49     graph.
50
51     Details: Retracting the given constraint may allow some currently
52     unsatisfiable downstream constraint be satisfied. We thus collect a
53     list of unsatisfied downstream constraints and attempt to satisfy
54     each one in turn. This list is sorted by constraint strength,
55     strongest first, as a heuristic for avoiding unnecessarily adding
```

```

56     and then overriding weak constraints."
57
58     | out unsatisfied |
59     out := c output.
60     c markUnsatisfied.
61     c removeFromGraph.
62     unsatisfied := self removePropagateFrom: out.
63     unsatisfied forEach: [: u | self incrementalAdd: u]
64 )
65
66 "planning/value propagation"
67 extractPlanFromConstraints: constraints = (
68     "Extract a plan for resatisfaction starting from the outputs of the
69     given constraints, usually a set of input constraints."
70
71     | sources |
72     sources := Vector new.
73     constraints forEach: [: c |
74         (c isInput and: [c isSatisfied]) ifTrue: [sources append: c]].
75     ^ self makePlan: sources
76 )
77
78 makePlan: sources = (
79     "Extract a plan for resatisfaction starting from the given satisfied
80     source constraints, usually a set of input constraints. This method
81     assumes that stay optimization is desired; the plan will contain only
82     constraints whose output variables are not stay. Constraints that do
83     no computation, such as stay and edit constraints, are not included
84     in the plan.
85
86     Details: The outputs of a constraint are marked when it is added to
87     the plan under construction. A constraint may be appended to the plan
88     when all its input variables are known. A variable is known if either
89     a) the variable is marked (indicating that has been computed by a
90     constraint appearing earlier in the plan), b) the variable is 'stay'
91     (i.e. it is a constant at plan execution time), or c) the variable
92     is not determined by any constraint. The last provision is for past
93     states of history variables, which are not stay but which are also
94     not computed by any constraint."
95
96     | mark plan todo c |
97     mark := self newMark.
98     plan := Plan new.
99     todo := sources.
100     [todo isEmpty] whileFalse:
101         [c := todo removeFirst.
102             ((c output mark <> mark) and:      "not in plan already and..."
103             [c inputsKnown: mark]) ifTrue:    "eligible for inclusion"
104             [plan append: c.
105                 c output mark: mark.
106                 self addConstraintsConsuming: c output to: todo]].
107     ^ plan
108 )
109
110 propagateFrom: v = (
111     "The given variable has changed. Propagate new values downstream."
112     | todo c |
113     todo := Vector new.
114     self addConstraintsConsuming: v to: todo.
115     [todo isEmpty] whileFalse:
116         [c := todo removeFirst.

```

```

117         c execute.
118         self addConstraintsConsuming: c output to: todo].
119     )
120
121     "private"
122     addConstraintsConsuming: v to: aCollection = (
123         | determiningC |
124         determiningC := v determinedBy.
125         v constraints forEach: [: c |
126             ((c == determiningC) or: [c isSatisfied not]) ifFalse:
127                 [aCollection append: c]].
128     )
129
130     addPropagate: c mark: mark = (
131         "Recompute the walkabout strengths and stay flags of all variables
132         downstream of the given constraint and recompute the actual values
133         of all variables whose stay flag is true. If a cycle is detected,
134         remove the given constraint and answer false. Otherwise, answer true.
135
136         Details: Cycles are detected when a marked variable is encountered
137         downstream of the given constraint. The sender is assumed to have
138         marked the inputs of the given constraint with the given mark. Thus,
139         encountering a marked node downstream of the output constraint means
140         that there is a path from the constraint's output to one of its
141         inputs."
142
143         | todo d |
144         todo := Vector with: c.
145         [todo isEmpty] whileFalse:
146             [d := todo removeFirst.
147              (d output mark = mark) ifTrue:
148                  [self incrementalRemove: c.
149                   ^ false].
150              d recalculate.
151              self addConstraintsConsuming: d output to: todo].
152         ^ true
153     )
154
155     changeVar: aVariable newValue: newValue = (
156         | editConstraint plan |
157         editConstraint := EditConstraint var: aVariable strength: Strength
158         SymPreferred addTo: self.
159         plan := self extractPlanFromConstraints: (Vector with:
160         editConstraint).
161         10 timesRepeat: [
162             aVariable value: newValue.
163             plan execute ].
164         editConstraint destroyConstraint: self.
165     )
166
167     constraintsConsuming: v do: aBlock = (
168         | determiningC |
169         determiningC := v determinedBy.
170         v constraints forEach: [: c |
171             (c == determiningC or: [c isSatisfied not]) ifFalse:
172                 [aBlock value: c]].
173     )
174
175     newMark = (
176         "Select a previously unused mark value.

```

```

176     Details: We just keep incrementing. If necessary, the counter will
177     turn into a LargePositiveInteger. In that case, it will be a bit
178     slower to compute the next mark but the algorithms will all behave
179     correctly. We reserve the value '0' to mean 'unmarked'. Thus, this
180     generator starts at '1' and will never produce '0' as a mark value."
181
182     ^ currentMark := currentMark + 1
183 )
184
185     removePropagateFrom: out = (
186         "Update the walkabout strengths and stay flags of all variables
187         downstream of the given constraint. Answer a collection of
unsatisfied
188         constraints sorted in order of decreasing strength."
189
190         | unsatisfied todo v |
191         unsatisfied := Vector new.
192
193         out determinedBy: nil.
194         out walkStrength: Strength absoluteWeakest.
195         out stay: true.
196         todo := Vector with: out.
197         [todo isEmpty] whileFalse: [
198             v := todo removeFirst.
199             v constraints forEach: [:c |
200                 c isSatisfied iffFalse: [unsatisfied append: c]].
201             self constraintsConsuming: v do: [:c |
202                 c recalculate.
203                 todo append: c output]].
204
205             unsatisfied sort: [:c1 :c2 | c1 strength stronger: c2 strength].
206         ^ unsatisfied
207     )
208
209     ----
210
211     "instance creation"
212     new = (
213         ^ super new initialize
214     )
215
216     "benchmarks"
217     chainTest: n = (
218         "Do chain-of-equality-constraints performance tests."
219         | vars editConstraint plan planner |
220
221         planner := Planner new.
222         vars := Array new: n+1 withAll: [ Variable new ].
223
224         "thread a chain of equality constraints through the variables"
225         1 to: n do: [:i |
226             | v1 v2 |
227             v1 := vars at: i.
228             v2 := vars at: i + 1.
229             EqualityConstraint var: v1 var: v2 strength: Strength SymRequired
addTo: planner].
230
231             StayConstraint var: vars last strength: Strength SymStrongDefault
addTo: planner.
232             editConstraint := EditConstraint var: vars first strength: Strength
SymPreferred addTo: planner.

```

```

233     plan := planner extractPlanFromConstraints: (Vector with:
editConstraint).
234
235     1 to: 100 do: [ :v |
236         vars first value: v.
237         plan execute.
238         vars last value <> v ifTrue: [self error: 'Chain test failed!!!']].
239
240     editConstraint destroyConstraint: planner
241 )
242
243 projectionTest: n = (
244     "This test constructs a two sets of variables related to each other
by
245     a simple linear transformation (scale and offset)."

```



```

1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 ScaleConstraint = BinaryConstraint (
9   "I relate two variables by the linear scaling relationship:
10   `v2 = (v1 * scale) + offset`. Either v1 or v2 may be changed to maintain
11   this relationship but the scale factor and offset are considered read-
only.
12
13   Instance variables:
14       scale      scale factor input variable <Variable>
15       offset      offset input variable <Variable>"
16   | scale offset |
17
18   "initialize-release"
19   initializeSrc: srcVar scale: scaleVar offset: offsetVar dst: dstVar
strength: strengthSymbol addTo: planner = (
20     "Initialize myself with the given variables and strength."
21
22     super initializeVar: srcVar var: dstVar strength: strengthSymbol
addTo: planner.
23     scale := scaleVar.
24     offset := offsetVar.
25
26     self addConstraint: planner.
27   )
28
29   "add/remove"
30   addToGraph = (
31     "Add myself to the constraint graph."
32     v1 addConstraint: self.
33     v2 addConstraint: self.
34     scale addConstraint: self.
35     offset addConstraint: self.
36     direction := nil.
37   )
38
39   removeFromGraph = (
40     "Remove myself from the constraint graph."
41     v1 == nil ifFalse: [ v1 removeConstraint: self ].
42     v2 == nil ifFalse: [ v2 removeConstraint: self ].
43     scale == nil ifFalse: [ scale removeConstraint: self ].
44     offset == nil ifFalse: [ offset removeConstraint: self ].
45     direction := nil.
46   )
47
48   "planning"
49   execute = (
50     "Enforce this constraint. Assume that it is satisfied."
51     direction = #forward
52       ifTrue: [ v2 value: (v1 value * scale value) + offset value ]
53       ifFalse: [ v1 value: (v2 value - offset value) / scale value ].

```

```

54 )
55
56 inputsDo: aBlock = (
57     "Evaluate the given block on my current input variable."
58     direction = #forward
59         ifTrue: [aBlock value: v1.
60                 aBlock value: scale.
61                 aBlock value: offset]
62         ifFalse: [aBlock value: v2.
63                  aBlock value: scale.
64                  aBlock value: offset].
65 )
66
67 recalculate = (
68     "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
69     the value for the current output of this constraint. Assume this
70     constraint is satisfied."
71     | in out |
72     direction = #forward
73         ifTrue: [in := v1. out := v2]
74         ifFalse: [out := v1. in := v2].
75     out walkStrength: (strength weakest: in walkStrength).
76     out stay: (in stay and: [scale stay and: [offset stay]]).
77     out stay ifTrue: [self execute].      "stay optimization"
78 )
79
80
81 ----
82
83 "instance creation"
84
85 var: src var: scale var: offset var: dst strength: strengthSymbol addTo:
86 planner = (
87     "Install a scale constraint with the given strength on the given
88     variables."
89     ^ self new initializeSrc: src scale: scale offset: offset dst: dst
90     strength: strengthSymbol addTo: planner
91 )
92 )

```

```
1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java\_benchmarking/index.html
7 "
8 StayConstraint = UnaryConstraint (
9   "I mark variables that should, with some level of preference, stay the
same.
10   I have one method with zero inputs and one output, which does nothing.
11   Planners may exploit the fact that, if I am satisfied, my output will
not
12   change during plan execution. This is called 'stay optimization.'"
13
14   "execution"
15
16   execute = (
17     "Stay constraints do nothing."
18   )
19
20   ----
21
22   "instance creation"
23   var: aVariable strength: strengthSymbol addTo: planner = (
24     "Install a stay constraint with the given strength on the given
variable."
25
26     ^ self new initializeVar: aVariable strength: strengthSymbol addTo:
planner
27   )
28 )
29
```

```

1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 Strength = (
9   "Strengths are used to measure the relative importance of constraints.
The
10   hierarchy of available strengths is determined by the class variable
11   StrengthTable (see my class initialization method). Because Strengths
are
12   invariant, references to Strength instances are shared (i.e. all
references
13   to `Strength of: #required` point to a single, shared instance). New
14   strengths may be inserted in the strength hierarchy without disrupting
15   current constraints.
16
17   Instance variables:
18       symbolicValue      symbolic strength name (e.g. #required) <Symbol>
19       arithmeticValue    index of the constraint in the hierarchy, used
for comparisons <Number>"
20   | symbolicValue arithmeticValue |
21
22   initializeWith: symVal = (
23     symbolicValue      := symVal.
24     arithmeticValue := Strength strengthTable at: symVal
25   )
26
27   "comparing"
28   sameAs: aStrength = (
29     "Answer true if I am the same strength as the given Strength."
30     ^ arithmeticValue = aStrength arithmeticValue
31   )
32
33   stronger: aStrength = (
34     "Answer true if I am stronger than the given Strength."
35     ^ arithmeticValue < aStrength arithmeticValue
36   )
37
38   weaker: aStrength = (
39     "Answer true if I am weaker than the given Strength."
40     ^ arithmeticValue > aStrength arithmeticValue
41   )
42
43   "max/min"
44   strongest: aStrength = (
45     "Answer the stronger of myself and aStrength."
46
47     (aStrength stronger: self)
48     ifTrue: [ ^ aStrength ]
49     ifFalse: [ ^ self ].
50   )
51
52   weakest: aStrength = (

```

```

53     "Answer the weaker of myself and aStrength."
54
55     (aStrength weaker: self)
56         ifTrue: [ ^ aStrength ]
57         ifFalse: [ ^ self ].
58 )
59
60 arithmeticValue = (
61     "Answer my arithmetic value. Used for comparisons. Note that
62     STRONGER constraints have SMALLER arithmetic values."
63
64     ^ arithmeticValue
65 )
66
67 ----
68 | AbsoluteStrongest AbsoluteWeakest Required StrengthConstants
StrengthTable
69     SymAbsoluteStrongest
70     SymRequired
71     SymStrongPreferred
72     SymPreferred
73     SymStrongDefault
74     SymDefault
75     SymWeakDefault
76     SymAbsoluteWeakest |
77
78 new: symVal = (
79     ^ self new initializeWith: symVal
80 )
81
82 strengthTable = (
83     ^ StrengthTable
84 )
85
86 "class initialization"
87
88 createStrengthTable = (
89     | table |
90     table := SomIdentityDictionary new.
91     table at: SymAbsoluteStrongest put: -10000.
92     table at: SymRequired put: -800.
93     table at: SymStrongPreferred put: -600.
94     table at: SymPreferred put: -400.
95     table at: SymStrongDefault put: -200.
96     table at: SymDefault put: 0.
97     table at: SymWeakDefault put: 500.
98     table at: SymAbsoluteWeakest put: 10000.
99     ^ table
100 )
101
102 createStrengthConstants = (
103     | constants |
104     constants := SomIdentityDictionary new.
105     StrengthTable keys forEach: [:strengthSymbol |
106         constants
107             at: strengthSymbol
108             put: (self new: strengthSymbol)].
109     ^ constants
110 )
111
112 initialize = (

```

```

113     SymAbsoluteStrongest := Sym new: 0.
114     SymRequired          := Sym new: 1.
115     SymStrongPreferred   := Sym new: 2.
116     SymPreferred         := Sym new: 3.
117     SymStrongDefault     := Sym new: 4.
118     SymDefault           := Sym new: 5.
119     SymWeakDefault       := Sym new: 6.
120     SymAbsoluteWeakest   := Sym new: 7.
121
122     StrengthTable := self createStrengthTable.
123     StrengthConstants := self createStrengthConstants.
124
125     AbsoluteStrongest := Strength of: SymAbsoluteStrongest.
126     AbsoluteWeakest := Strength of: SymAbsoluteWeakest.
127     Required := Strength of: SymRequired.
128 )
129
130 "instance creation"
131 of: aSymbol = (
132     "Answer an instance with the specified strength."
133     ^ StrengthConstants at: aSymbol
134 )
135
136 "constants"
137 absoluteStrongest = (
138     ^ AbsoluteStrongest
139 )
140
141 absoluteWeakest = (
142     ^ AbsoluteWeakest
143 )
144
145 required = (
146     ^ Required
147 )
148
149 SymAbsoluteStrongest = ( ^ SymAbsoluteStrongest )
150 SymRequired          = ( ^ SymRequired )
151 SymStrongPreferred   = ( ^ SymStrongPreferred )
152 SymPreferred         = ( ^ SymPreferred )
153 SymStrongDefault     = ( ^ SymStrongDefault )
154 SymDefault           = ( ^ SymDefault )
155 SymWeakDefault       = ( ^ SymWeakDefault )
156 SymAbsoluteWeakest   = ( ^ SymAbsoluteWeakest )
157 )
158

```

```
1 "  
2 Copyright (c) 2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 Sym = (  
24   | hash |  
25   init: aHash = ( hash := aHash )  
26  
27   customHash = ( ^ hash )  
28   ----  
29  
30   new: aHash = (  
31     ^ self new init: aHash  
32   )  
33 )  
34
```

```

1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 UnaryConstraint = AbstractConstraint (
9   "I am an abstract superclass for constraints having a single possible
output
10   variable.
11
12   Instance variables:
13     output      possible output variable <Variable>
14     satisfied    true if I am currently satisfied <Boolean>"
15   | output satisfied |
16
17   "initialize-release"
18
19   initializeVar: aVariable strength: strengthSymbol addTo: planner = (
20     "Initialize myself with the given variable and strength."
21     super initialize: strengthSymbol.
22     output := aVariable.
23     satisfied := false.
24     self addConstraint: planner.
25   )
26
27   "queries"
28   isSatisfied = (
29     "Answer true if this constraint is satisfied in the current solution."
30     ^ satisfied
31   )
32
33   "add/remove"
34
35   addToGraph = (
36     "Add myself to the constraint graph."
37     output addConstraint: self.
38     satisfied := false.
39   )
40
41   removeFromGraph = (
42     "Remove myself from the constraint graph."
43     output == nil ifFalse: [output removeConstraint: self].
44     satisfied := false.
45   )
46
47   "planning"
48   chooseMethod: mark = (
49     "Decide if I can be satisfied and record that decision."
50
51     satisfied :=
52       output mark <> mark and:
53       [strength stronger: output walkStrength].
54     ^ nil
55   )

```



```

56
57 execute = (
58     "Enforce this constraint. Assume that it is satisfied."
59     self subclassResponsibility
60 )
61
62 inputsDo: aBlock = (
63     "I have no input variables."
64 )
65
66 inputsHasOne: aBlock = (
67     ^ false
68 )
69
70 markUnsatisfied = (
71     "Record the fact that I am unsatisfied."
72     satisfied := false.
73 )
74
75 output = (
76     "Answer my current output variable."
77     ^ output
78 )
79
80 recalculate = (
81     "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
82     the value for the current output of this constraint. Assume this
83     constraint is satisfied."
84
85     output walkStrength: strength.
86     output stay: self isInput not.
87     output stay ifTrue: [self execute].    "stay optimization"
88 )
89 )
90

```

```

1 "This benchmark is derived from Mario Wolczko's Smalltalk version of
DeltaBlue.
2
3 It is modified to use the SOM class library.
4 License details:
5
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 Variable = (
9   "I represent a constrained variable. In addition to my value, I
maintain the
10   structure of the constraint graph, the current dataflow graph, and
various
11   parameters of interest to the DeltaBlue incremental constraint solver.
12
13   Instance variables:
14     value          my value; changed by constraints, read by client
<Object>
15     constraints     normal constraints that reference me <Array of
Constraint>
16     determinedBy    the constraint that currently determines
17                     my value (or nil if there isn't one) <Constraint>
18     walkStrength     my walkabout strength <Strength>
19     stay            true if I am a planning-time constant <Boolean>
20     mark            used by the planner to mark constraints <Number>"
21 | value constraints determinedBy walkStrength stay mark |
22
23 "initialize-release"
24
25 initialize = (
26   value := 0.
27   constraints := Vector new: 2.
28   determinedBy := nil.
29   walkStrength := Strength absoluteWeakest.
30   stay := true.
31   mark := 0.
32 )
33
34 "access"
35 addConstraint: aConstraint = (
36   "Add the given constraint to the set of all constraints that refer
37   to me."
38
39   constraints append: aConstraint.
40 )
41
42 constraints = (
43   "Answer the set of constraints that refer to me."
44   ^ constraints
45 )
46
47 determinedBy = (
48   "Answer the constraint that determines my value in the current
49   dataflow."
50   ^ determinedBy
51 )
52

```

```

53     determinedBy: aConstraint = (
54         "Record that the given constraint determines my value in the current
55         data flow."
56         determinedBy := aConstraint.
57     )
58
59     mark = (
60         "Answer my mark value."
61         ^ mark
62     )
63
64     mark: markValue = (
65         "Set my mark value."
66         mark := markValue.
67     )
68
69     removeConstraint: c = (
70         "Remove all traces of c from this variable."
71         constraints remove: c.
72         determinedBy == c ifTrue: [ determinedBy := nil ].
73     )
74
75     stay = (
76         "Answer my stay flag."
77         ^ stay
78     )
79
80     stay: aBoolean = (
81         "Set my stay flag."
82         stay := aBoolean
83     )
84
85     value = (
86         "Answer my value."
87         ^ value
88     )
89
90     value: anObject = (
91         "Set my value."
92         value := anObject.
93     )
94
95     walkStrength = (
96         "Answer my walkabout strength in the current dataflow."
97         ^ walkStrength
98     )
99
100    walkStrength: aStrength = (
101        "Set my walkabout strength in the current dataflow."
102        walkStrength := aStrength.
103    )
104
105    ----
106
107    "instance creation"
108
109    new = (
110        ^ super new initialize
111    )
112
113    value: aValue = (

```

```
114         | o |
115         o := self new.
116         o value: aValue.
117         ^ o
118     )
119 )
120
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Harness = (  
23  
24   processArguments: args= (  
25     | run i doGC |  
26     doGC := false.  
27     i := 2.  
28  
29     ((args at: i) = '--gc-between-iterations' or: [(args at: i) = '--gc'])  
30     ifTrue: [  
31       doGC := true.  
32       i := i + 1 ].  
33  
34     run := Run new: (args at: i).  
35     run doGC: doGC.  
36  
37     args length > i ifTrue: [  
38       run numIterations: (args at: i + 1) asInteger.  
39       args length > (i + 1) ifTrue: [  
40         run innerIterations: (args at: i + 2) asInteger.  
41       ] ].  
42     ^ run  
43   )  
44  
45   run: args = (  
46     | run |  
47     args length < 2 ifTrue: [  
48       self printUsage.  
49       system exit: 1  
50     ].  
51  
52     run := self processArguments: args.  
53  
54     run runBenchmark.
```

```

55     run printTotal.
56 )
57
58 printUsage = (
59     './som -cp Smalltalk Benchmarks/Harness.som [--gc-between-iterations|-
gc] benchmark [num-iterations [inner-iter]]' println.
60     '' println.
61     ' --gc-between-iterations | --gc - trigger a full GC between
benchmark iteratons' println.
62     ' benchmark - benchmark class name' println.
63     ' num-iterations - number of times to execute benchmark, default: 1'
println.
64     ' inner-iter - number of times the benchmark is executed in an
inner loop, ' println.
65     ' which is measured in total, default: 1' println.
66 )
67 )
68

```

```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8   http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 BasicBlock = (  
18   | inEdges outEdges name |  
19  
20   init: aName = (  
21     inEdges  := Vector new: 2.  
22     outEdges := Vector new: 2.  
23     name     := aName  
24   )  
25  
26   inEdges = ( ^ inEdges )  
27   outEdges = ( ^ outEdges )  
28  
29   numPred = ( ^ inEdges size )  
30  
31   addOutEdge: to = (  
32     outEdges append: to  
33   )  
34  
35   addInEdge: from = (  
36     inEdges append: from  
37   )  
38  
39   customHash = (  
40     ^ name  
41   )  
42  
43   ----  
44  
45   new: name = (  
46     ^ self new init: name  
47   )  
48 )  
49
```

```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8   http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 BasicBlockEdge = (  
18   | from to |  
19  
20   init: cfg from: fromName to: toName = (  
21     from := cfg createNode: fromName.  
22     to   := cfg createNode: toName.  
23  
24     from addOutEdge: to.  
25     to   addInEdge: from.  
26     cfg  addEdge:   self  
27   )  
28  
29   ----  
30  
31   for: cfg from: fromName to: toName = (  
32     ^ self new init: cfg from: fromName to: toName  
33   )  
34 )  
35
```



```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8   http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 ControlFlowGraph = (  
18   | basicBlockMap startNode edgeList |  
19  
20   initialize = (  
21     basicBlockMap := Vector new.  
22     edgeList := Vector new.  
23   )  
24  
25   createNode: name = (  
26     | node |  
27  
28     (basicBlockMap at: name) notNil  
29     ifTrue: [ node := basicBlockMap at: name ]  
30     ifFalse: [  
31       node := BasicBlock new: name.  
32       basicBlockMap at: name put: node ].  
33  
34     self numNodes = 1 ifTrue: [startNode := node].  
35     ^ node  
36   )  
37  
38   addEdge: edge = (  
39     edgeList append: edge  
40   )  
41  
42   numNodes = (  
43     ^ basicBlockMap size  
44   )  
45  
46   startBasicBlock = (  
47     ^ startNode  
48   )  
49  
50   basicBlocks = (  
51     ^ basicBlockMap  
52   )  
53  
54   ----  
55  
56   new = ( ^ super new initialize )  
57 )  
58
```

```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8   http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 Havlak = Benchmark (  
18   innerBenchmarkLoop: innerIterations = (  
19     ^ self verifyResult:  
20       (LoopTesterApp new main: innerIterations loop: 50 p: 10 p: 10 p: 5)  
21       iterations: innerIterations  
22   )  
23  
24   verifyResult: result iterations: innerIterations = (  
25     innerIterations = 15000 ifTrue: [ ^ (result at: 1) = 46602 and:  
26     [(result at: 2) = 5213] ].  
27     innerIterations = 1500 ifTrue: [ ^ (result at: 1) = 6102 and:  
28     [(result at: 2) = 5213] ].  
29     innerIterations = 150 ifTrue: [ ^ (result at: 1) = 2052 and:  
30     [(result at: 2) = 5213] ].  
31     innerIterations = 15 ifTrue: [ ^ (result at: 1) = 1647 and:  
32     [(result at: 2) = 5213] ].  
33     innerIterations = 1 ifTrue: [ ^ (result at: 1) = 1605 and:  
34     [(result at: 2) = 5213] ].  
35     ('No verification result for' + innerIterations + ' found') println.  
36     ('Result is ' + (result at: 1) + ', ' + (result at: 2)) println.  
37     ^ false  
38   )  
39 )  
40 )  
41 )  
42 )  
43 )  
44 )  
45 )  
46 )  
47 )  
48 )  
49 )  
50 )  
51 )  
52 )  
53 )  
54 )  
55 )  
56 )  
57 )  
58 )  
59 )  
60 )  
61 )  
62 )  
63 )  
64 )  
65 )  
66 )  
67 )  
68 )  
69 )  
70 )  
71 )  
72 )  
73 )  
74 )  
75 )  
76 )  
77 )  
78 )  
79 )  
80 )  
81 )  
82 )  
83 )  
84 )  
85 )  
86 )  
87 )  
88 )  
89 )  
90 )  
91 )  
92 )  
93 )  
94 )  
95 )  
96 )  
97 )  
98 )  
99 )  
100 )  
101 )  
102 )  
103 )  
104 )  
105 )  
106 )  
107 )  
108 )  
109 )  
110 )  
111 )  
112 )  
113 )  
114 )  
115 )  
116 )  
117 )  
118 )  
119 )  
120 )  
121 )  
122 )  
123 )  
124 )  
125 )  
126 )  
127 )  
128 )  
129 )  
130 )  
131 )  
132 )  
133 )  
134 )  
135 )  
136 )  
137 )  
138 )  
139 )  
140 )  
141 )  
142 )  
143 )  
144 )  
145 )  
146 )  
147 )  
148 )  
149 )  
150 )  
151 )  
152 )  
153 )  
154 )  
155 )  
156 )  
157 )  
158 )  
159 )  
160 )  
161 )  
162 )  
163 )  
164 )  
165 )  
166 )  
167 )  
168 )  
169 )  
170 )  
171 )  
172 )  
173 )  
174 )  
175 )  
176 )  
177 )  
178 )  
179 )  
180 )  
181 )  
182 )  
183 )  
184 )  
185 )  
186 )  
187 )  
188 )  
189 )  
190 )  
191 )  
192 )  
193 )  
194 )  
195 )  
196 )  
197 )  
198 )  
199 )  
200 )  
201 )  
202 )  
203 )  
204 )  
205 )  
206 )  
207 )  
208 )  
209 )  
210 )  
211 )  
212 )  
213 )  
214 )  
215 )  
216 )  
217 )  
218 )  
219 )  
220 )  
221 )  
222 )  
223 )  
224 )  
225 )  
226 )  
227 )  
228 )  
229 )  
230 )  
231 )  
232 )  
233 )  
234 )  
235 )  
236 )  
237 )  
238 )  
239 )  
240 )  
241 )  
242 )  
243 )  
244 )  
245 )  
246 )  
247 )  
248 )  
249 )  
250 )  
251 )  
252 )  
253 )  
254 )  
255 )  
256 )  
257 )  
258 )  
259 )  
260 )  
261 )  
262 )  
263 )  
264 )  
265 )  
266 )  
267 )  
268 )  
269 )  
270 )  
271 )  
272 )  
273 )  
274 )  
275 )  
276 )  
277 )  
278 )  
279 )  
280 )  
281 )  
282 )  
283 )  
284 )  
285 )  
286 )  
287 )  
288 )  
289 )  
290 )  
291 )  
292 )  
293 )  
294 )  
295 )  
296 )  
297 )  
298 )  
299 )  
300 )  
301 )  
302 )  
303 )  
304 )  
305 )  
306 )  
307 )  
308 )  
309 )  
310 )  
311 )  
312 )  
313 )  
314 )  
315 )  
316 )  
317 )  
318 )  
319 )  
320 )  
321 )  
322 )  
323 )  
324 )  
325 )  
326 )  
327 )  
328 )  
329 )  
330 )  
331 )  
332 )  
333 )  
334 )  
335 )  
336 )  
337 )  
338 )  
339 )  
340 )  
341 )  
342 )  
343 )  
344 )  
345 )  
346 )  
347 )  
348 )  
349 )  
350 )  
351 )  
352 )  
353 )  
354 )  
355 )  
356 )  
357 )  
358 )  
359 )  
360 )  
361 )  
362 )  
363 )  
364 )  
365 )  
366 )  
367 )  
368 )  
369 )  
370 )  
371 )  
372 )  
373 )  
374 )  
375 )  
376 )  
377 )  
378 )  
379 )  
380 )  
381 )  
382 )  
383 )  
384 )  
385 )  
386 )  
387 )  
388 )  
389 )  
390 )  
391 )  
392 )  
393 )  
394 )  
395 )  
396 )  
397 )  
398 )  
399 )  
400 )  
401 )  
402 )  
403 )  
404 )  
405 )  
406 )  
407 )  
408 )  
409 )  
410 )  
411 )  
412 )  
413 )  
414 )  
415 )  
416 )  
417 )  
418 )  
419 )  
420 )  
421 )  
422 )  
423 )  
424 )  
425 )  
426 )  
427 )  
428 )  
429 )  
430 )  
431 )  
432 )  
433 )  
434 )  
435 )  
436 )  
437 )  
438 )  
439 )  
440 )  
441 )  
442 )  
443 )  
444 )  
445 )  
446 )  
447 )  
448 )  
449 )  
450 )  
451 )  
452 )  
453 )  
454 )  
455 )  
456 )  
457 )  
458 )  
459 )  
460 )  
461 )  
462 )  
463 )  
464 )  
465 )  
466 )  
467 )  
468 )  
469 )  
470 )  
471 )  
472 )  
473 )  
474 )  
475 )  
476 )  
477 )  
478 )  
479 )  
480 )  
481 )  
482 )  
483 )  
484 )  
485 )  
486 )  
487 )  
488 )  
489 )  
490 )  
491 )  
492 )  
493 )  
494 )  
495 )  
496 )  
497 )  
498 )  
499 )  
500 )  
501 )  
502 )  
503 )  
504 )  
505 )  
506 )  
507 )  
508 )  
509 )  
510 )  
511 )  
512 )  
513 )  
514 )  
515 )  
516 )  
517 )  
518 )  
519 )  
520 )  
521 )  
522 )  
523 )  
524 )  
525 )  
526 )  
527 )  
528 )  
529 )  
530 )  
531 )  
532 )  
533 )  
534 )  
535 )  
536 )  
537 )  
538 )  
539 )  
540 )  
541 )  
542 )  
543 )  
544 )  
545 )  
546 )  
547 )  
548 )  
549 )  
550 )  
551 )  
552 )  
553 )  
554 )  
555 )  
556 )  
557 )  
558 )  
559 )  
560 )  
561 )  
562 )  
563 )  
564 )  
565 )  
566 )  
567 )  
568 )  
569 )  
570 )  
571 )  
572 )  
573 )  
574 )  
575 )  
576 )  
577 )  
578 )  
579 )  
580 )  
581 )  
582 )  
583 )  
584 )  
585 )  
586 )  
587 )  
588 )  
589 )  
590 )  
591 )  
592 )  
593 )  
594 )  
595 )  
596 )  
597 )  
598 )  
599 )  
600 )  
601 )  
602 )  
603 )  
604 )  
605 )  
606 )  
607 )  
608 )  
609 )  
610 )  
611 )  
612 )  
613 )  
614 )  
615 )  
616 )  
617 )  
618 )  
619 )  
620 )  
621 )  
622 )  
623 )  
624 )  
625 )  
626 )  
627 )  
628 )  
629 )  
630 )  
631 )  
632 )  
633 )  
634 )  
635 )  
636 )  
637 )  
638 )  
639 )  
640 )  
641 )  
642 )  
643 )  
644 )  
645 )  
646 )  
647 )  
648 )  
649 )  
650 )  
651 )  
652 )  
653 )  
654 )  
655 )  
656 )  
657 )  
658 )  
659 )  
660 )  
661 )  
662 )  
663 )  
664 )  
665 )  
666 )  
667 )  
668 )  
669 )  
670 )  
671 )  
672 )  
673 )  
674 )  
675 )  
676 )  
677 )  
678 )  
679 )  
680 )  
681 )  
682 )  
683 )  
684 )  
685 )  
686 )  
687 )  
688 )  
689 )  
690 )  
691 )  
692 )  
693 )  
694 )  
695 )  
696 )  
697 )  
698 )  
699 )  
700 )  
701 )  
702 )  
703 )  
704 )  
705 )  
706 )  
707 )  
708 )  
709 )  
710 )  
711 )  
712 )  
713 )  
714 )  
715 )  
716 )  
717 )  
718 )  
719 )  
720 )  
721 )  
722 )  
723 )  
724 )  
725 )  
726 )  
727 )  
728 )  
729 )  
730 )  
731 )  
732 )  
733 )  
734 )  
735 )  
736 )  
737 )  
738 )  
739 )  
740 )  
741 )  
742 )  
743 )  
744 )  
745 )  
746 )  
747 )  
748 )  
749 )  
750 )  
751 )  
752 )  
753 )  
754 )  
755 )  
756 )  
757 )  
758 )  
759 )  
760 )  
761 )  
762 )  
763 )  
764 )  
765 )  
766 )  
767 )  
768 )  
769 )  
770 )  
771 )  
772 )  
773 )  
774 )  
775 )  
776 )  
777 )  
778 )  
779 )  
780 )  
781 )  
782 )  
783 )  
784 )  
785 )  
786 )  
787 )  
788 )  
789 )  
790 )  
791 )  
792 )  
793 )  
794 )  
795 )  
796 )  
797 )  
798 )  
799 )  
800 )  
801 )  
802 )  
803 )  
804 )  
805 )  
806 )  
807 )  
808 )  
809 )  
810 )  
811 )  
812 )  
813 )  
814 )  
815 )  
816 )  
817 )  
818 )  
819 )  
820 )  
821 )  
822 )  
823 )  
824 )  
825 )  
826 )  
827 )  
828 )  
829 )  
830 )  
831 )  
832 )  
833 )  
834 )  
835 )  
836 )  
837 )  
838 )  
839 )  
840 )  
841 )  
842 )  
843 )  
844 )  
845 )  
846 )  
847 )  
848 )  
849 )  
850 )  
851 )  
852 )  
853 )  
854 )  
855 )  
856 )  
857 )  
858 )  
859 )  
860 )  
861 )  
862 )  
863 )  
864 )  
865 )  
866 )  
867 )  
868 )  
869 )  
870 )  
871 )  
872 )  
873 )  
874 )  
875 )  
876 )  
877 )  
878 )  
879 )  
880 )  
881 )  
882 )  
883 )  
884 )  
885 )  
886 )  
887 )  
888 )  
889 )  
890 )  
891 )  
892 )  
893 )  
894 )  
895 )  
896 )  
897 )  
898 )  
899 )  
900 )  
901 )  
902 )  
903 )  
904 )  
905 )  
906 )  
907 )  
908 )  
909 )  
910 )  
911 )  
912 )  
913 )  
914 )  
915 )  
916 )  
917 )  
918 )  
919 )  
920 )  
921 )  
922 )  
923 )  
924 )  
925 )  
926 )  
927 )  
928 )  
929 )  
930 )  
931 )  
932 )  
933 )  
934 )  
935 )  
936 )  
937 )  
938 )  
939 )  
940 )  
941 )  
942 )  
943 )  
944 )  
945 )  
946 )  
947 )  
948 )  
949 )  
950 )  
951 )  
952 )  
953 )  
954 )  
955 )  
956 )  
957 )  
958 )  
959 )  
960 )  
961 )  
962 )  
963 )  
964 )  
965 )  
966 )  
967 )  
968 )  
969 )  
970 )  
971 )  
972 )  
973 )  
974 )  
975 )  
976 )  
977 )  
978 )  
979 )  
980 )  
981 )  
982 )  
983 )  
984 )  
985 )  
986 )  
987 )  
988 )  
989 )  
990 )  
991 )  
992 )  
993 )  
994 )  
995 )  
996 )  
997 )  
998 )  
999 )  
1000 )  
1001 )  
1002 )  
1003 )  
1004 )  
1005 )  
1006 )  
1007 )  
1008 )  
1009 )  
1010 )  
1011 )  
1012 )  
1013 )  
1014 )  
1015 )  
1016 )  
1017 )  
1018 )  
1019 )  
1020 )  
1021 )  
1022 )  
1023 )  
1024 )  
1025 )  
1026 )  
1027 )  
1028 )  
1029 )  
1030 )  
1031 )  
1032 )  
1033 )  
1034 )  
1035 )  
1036 )  
1037 )  
1038 )  
1039 )  
1040 )  
1041 )  
1042 )  
1043 )  
1044 )  
1045 )  
1046 )  
1047 )  
1048 )  
1049 )  
1050 )  
1051 )  
1052 )  
1053 )  
1054 )  
1055 )  
1056 )  
1057 )  
1058 )  
1059 )  
1060 )  
1061 )  
1062 )  
1063 )  
1064 )  
1065 )  
1066 )  
1067 )  
1068 )  
1069 )  
1070 )  
1071 )  
1072 )  
1073 )  
1074 )  
1075 )  
1076 )  
1077 )  
1078 )  
1079 )  
1080 )  
1081 )  
1082 )  
1083 )  
1084 )  
1085 )  
1086 )  
1087 )  
1088 )  
1089 )  
1090 )  
1091 )  
1092 )  
1093 )  
1094 )  
1095 )  
1096 )  
1097 )  
1098 )  
1099 )  
1100 )  
1101 )  
1102 )  
1103 )  
1104 )  
1105 )  
1106 )  
1107 )  
1108 )  
1109 )  
1110 )  
1111 )  
1112 )  
1113 )  
1114 )  
1115 )  
1116 )  
1117 )  
1118 )  
1119 )  
1120 )  
1121 )  
1122 )  
1123 )  
1124 )  
1125 )  
1126 )  
1127 )  
1128 )  
1129 )  
1130 )  
1131 )  
1132 )  
1133 )  
1134 )  
1135 )  
1136 )  
1137 )  
1138 )  
1139 )  
1140 )  
1141 )  
1142 )  
1143 )  
1144 )  
1145 )  
1146 )  
1147 )  
1148 )  
1149 )  
1150 )  
1151 )  
1152 )  
1153 )  
1154 )  
1155 )  
1156 )  
1157 )  
1158 )  
1159 )  
1160 )  
1161 )  
1162 )  
1163 )  
1164 )  
1165 )  
1166 )  
1167 )  
1168 )  
1169 )  
1170 )  
1171 )  
1172 )  
1173 )  
1174 )  
1175 )  
1176 )  
1177 )  
1178 )  
1179 )  
1180 )  
1181 )  
1182 )  
1183 )  
1184 )  
1185 )  
1186 )  
1187 )  
1188 )  
1189 )  
1190 )  
1191 )  
1192 )  
1193 )  
1194 )  
1195 )  
1196 )  
1197 )  
1198 )  
1199 )  
1200 )  
1201 )  
1202 )  
1203 )  
1204 )  
1205 )  
1206 )  
1207 )  
1208 )  
1209 )  
1210 )  
1211 )  
1212 )  
1213 )  
1214 )  
1215 )  
1216 )  
1217 )  
1218 )  
1219 )  
1220 )  
1221 )  
1222 )  
1223 )  
1224 )  
1225 )  
1226 )  
1227 )  
1228 )  
1229 )  
1230 )  
1231 )  
1232 )  
1233 )  
1234 )  
1235 )  
1236 )  
1237 )  
1238 )  
1239 )  
1240 )  
1241 )  
1242 )  
1243 )  
1244 )  
1245 )  
1246 )  
1247 )  
1248 )  
1249 )  
1250 )  
1251 )  
1252 )  
1253 )  
1254 )  
1255 )  
1256 )  
1257 )  
1258 )  
1259 )  
1260 )  
1261 )  
1262 )  
1263 )  
1264 )  
1265 )  
1266 )  
1267 )  
1268 )  
1269 )  
1270 )  
1271 )  
1272 )  
1273 )  
1274 )  
1275 )  
1276 )  
1277 )  
1278 )  
1279 )  
1280 )  
1281 )  
1282 )  
1283 )  
1284 )  
1285 )  
1286 )  
1287 )  
1288 )  
1289 )  
1290 )  
1291 )  
1292 )  
1293 )  
1294 )  
1295 )  
1296 )  
1297 )  
1298 )  
1299 )  
1300 )  
1301 )  
1302 )  
1303 )  
1304 )  
1305 )  
1306 )  
1307 )  
1308 )  
1309 )  
1310 )  
1311 )  
1312 )  
1313 )  
1314 )  
1315 )  
1316 )  
1317 )  
1318 )  
1319 )  
1320 )  
1321 )  
1322 )  
1323 )  
1324 )  
1325 )  
1326 )  
1327 )  
1328 )  
1329 )  
1330 )  
1331 )  
1332 )  
1333 )  
1334 )  
1335 )  
1336 )  
1337 )  
1338 )  
1339 )  
1340 )  
1341 )  
1342 )  
1343 )  
1344 )  
1345 )  
1346 )  
1347 )  
1348 )  
1349 )  
1350 )  
1351 )  
1352 )  
1353 )  
1354 )  
1355 )  
1356 )  
1357 )  
1358 )  
1359 )  
1360 )  
1361 )  
1362 )  
1363 )  
1364 )  
1365 )  
1366 )  
1367 )  
1368 )  
1369 )  
1370 )  
1371 )  
1372 )  
1373 )  
1374 )  
1375 )  
1376 )  
1377 )  
1378 )  
1379 )  
1380 )  
1381 )  
1382 )  
1383 )  
1384 )  
1385 )  
1386 )  
1387 )  
1388 )  
1389 )  
1390 )  
1391 )  
1392 )  
1393 )  
1394 )  
1395 )  
1396 )  
1397 )  
1398 )  
1399 )  
1400 )  
1401 )  
1402 )  
1403 )  
1404 )  
1405 )  
1406 )  
1407 )  
1408 )  
1409 )  
1410 )  
1411 )  
1412 )  
1413 )  
1414 )  
1415 )  
1416 )  
1417 )  
1418 )  
1419 )  
1420 )  
1421 )  
1422 )  
1423 )  
1424 )  
1425 )  
1426 )  
1427 )  
1428 )  
1429 )  
1430 )  
1431 )  
1432 )  
1433 )  
1434 )  
1435 )  
1436 )  
1437 )  
1438 )  
1439 )  
1440 )  
1441 )  
1442 )  
1443 )  
1444 )  
1445 )  
1446 )  
1447 )  
1448 )  
1449 )  
1450 )  
1451 )  
1452 )  
1453 )  
1454 )  
1455 )  
1456 )  
1457 )  
1458 )  
1459 )  
1460 )  
1461 )  
1462 )  
1463 )  
1464 )  
1465 )  
1466 )  
1467 )  
1468 )  
1469 )  
1470 )  
1471 )  
1472 )  
1473 )  
1474 )  
1475 )  
1476 )  
1477 )  
1478 )  
1479 )  
1480 )  
1481 )  
1482 )  
1483 )  
1484 )  
1485 )  
1486 )  
1487 )  
1488 )  
1489 )  
1490 )  
1491 )  
1492 )  
1493 )  
1494 )  
1495 )  
1496 )  
1497 )  
1498 )  
1499 )  
1500 )  
1501 )  
1502 )  
1503 )  
1504 )  
1505 )  
1506 )  
1507 )  
1508 )  
1509 )  
1510 )  
1511 )  
1512 )  
1513 )  
1514 )  
1515 )  
1516 )  
1517 )  
1518 )  
1519 )  
1520 )  
1521 )  
1522 )  
1523 )  
1524 )  
1525 )  
1526 )  
1527 )  
1528 )  
1529 )  
1530 )  
1531 )  
1532 )  
1533 )  
1534 )  
1535 )  
1536 )  
1537 )  
1538 )  
1539 )  
1540 )  
1541 )  
1542 )  
1543 )  
1544 )  
1545 )  
1546 )  
1547 )  
1548 )  
1549 )  
1550 )  
1551 )  
1552 )  
1553 )  
1554 )  
1555 )  
1556 )  
1557 )  
1558 )  
1559 )  
1560 )  
1561 )  
1562 )  
1563 )  
1564 )  
1565 )  
1566 )  
1567 )  
1568 )  
1569 )  
1570 )  
1571 )  
1572 )  
1573 )  
1574 )  
1575 )  
1576 )  
1577 )  
1578 )  
1579 )  
1580 )  
1581 )  
1582 )  
1583 )  
1584 )  
1585 )  
1586 )  
1587 )  
1588 )  
1589 )  
1590 )  
1591 )  
1592 )  
1593 )  
1594 )  
1595 )  
1596 )  
1597 )  
1598 )  
1599 )  
1600 )  
1601 )  
1602 )  
1603 )  
1604 )  
1605 )  
1606 )  
1607 )  
1608 )  
1609 )  
1610 )  
1611 )  
1612 )  
1613 )  
1614 )  
1615 )  
1616 )  
1617 )  
1618 )  
1619 )  
1620 )  
1621 )  
1622 )  
1623 )  
1624 )  
1625 )  
1626 )  
1627 )  
1628 )  
1629 )  

```

```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8     http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 HavlakLoopFinder = (  
18   | cfg lsg  
19   nonBackPreds backPreds number  
20   maxSize header type last nodes |  
21  
22   init: aCfg lsg: aLsg = (  
23     cfg := aCfg.  
24     lsg := aLsg.  
25  
26     nonBackPreds := Vector new.  
27     backPreds     := Vector new.  
28     number := SomIdentityDictionary new.  
29  
30     maxSize := 0.  
31   )  
32  
33   Unvisited = ( ^ 2147483647 )  
34   MaxNonBackPreds = ( ^ 32 * 1024 )  
35  
36   isAncestor: w v: v = (  
37     ^ (w <= v) && (v <= (last at: w))  
38   )  
39  
40   doDFS: currentNode current: current = (  
41     | lastId outerBlocks |  
42  
43     (nodes at: current) initNode: currentNode dfs: current.  
44     number at: currentNode put: current.  
45  
46     lastId := current.  
47     outerBlocks := currentNode outEdges.  
48  
49     1 to: outerBlocks size do: [:i |  
50       | target |  
51       target := outerBlocks at: i.  
52       (number at: target) = self Unvisited ifTrue: [  
53         lastId := self doDFS: target current: lastId + 1 ] ].  
54  
55     last at: current put: lastId.  
56     ^ lastId  
57   )  
58
```

```

59  initAllNodes = (
60      cfg basicBlocks forEach: [:bb |
61          number at: bb put: self Unvisited ].
62
63      self doDFS: cfg startBasicBlock current: 1
64  )
65
66  identifyEdges: size = (
67      1 to: size do: [:w |
68          | nodeW |
69          header at: w put: 1.
70          type    at: w put: #BBNonHeader.
71
72          nodeW := (nodes at: w) bb.
73          nodeW isNil
74              ifTrue: [ type at: w put: #BBDead ]
75              ifFalse: [ self processEdges: nodeW w: w ] ]
76  )
77
78  processEdges: nodeW w: w = (
79      nodeW numPred > 0 ifTrue: [
80          nodeW inEdges forEach: [:nodeV |
81              | v |
82              v := number at: nodeV.
83              v <> self Unvisited ifTrue: [
84                  (self isAncestor: w v: v)
85                  ifTrue: [ (backPreds at: w) append: v ]
86                  ifFalse: [ (nonBackPreds at: w) add: v ] ] ] ]
87  )
88
89  findLoops = (
90      | size |
91      cfg startBasicBlock isNil ifTrue: [ ^ self ].
92
93      size := cfg numNodes.
94
95      nonBackPreds removeAll.
96      backPreds removeAll.
97      number removeAll.
98
99      size > maxSize ifTrue: [
100          header := Array new: size.
101          type    := Array new: size.
102          last    := Array new: size.
103          nodes   := Array new: size.
104          maxSize := size ].
105
106      1 to: size do: [:i |
107          nonBackPreds append: SomSet new.
108          backPreds append: Vector new.
109          nodes at: i put: UnionFindNode new ].
110
111      self initAllNodes.
112      self identifyEdges: size.
113      header at: 1 put: 1.
114
115      size downTo: 1 do: [:w |
116          | nodePool nodeW |
117          nodePool := Vector new.
118          nodeW := (nodes at: w) bb.
119

```

```

120     nodeW notNil ifTrue: [
121         | workList |
122         self stepD: w nodePool: nodePool.
123
124         workList := Vector new.
125         nodePool forEach: [:niter | workList append: niter ].
126
127         nodePool size <> 0 ifTrue: [
128             type at: w put: #BBReducible.
129         ].
130
131         [ workList isEmpty ] whileFalse: [
132             | x nonBackSize |
133             x := workList removeFirst.
134
135             nonBackSize := (nonBackPreds at: x dfsNumber) size.
136             nonBackSize > self MaxNonBackPreds ifTrue: [ ^ self ].
137             self stepEProcessNonBackPreds: w nodePool: nodePool workList:
workList x: x ].
138
139             (nodePool size > 0 or: [(type at: w) = #BBSelf]) ifTrue: [
140                 | loop |
141                 loop := lsg createNewLoop: nodeW reducible: ((type at: w) ~=
#BBIrreducible).
142                 self setLoopAttribute: w nodePool: nodePool loop: loop ] ]
143         )
144
145     stepEProcessNonBackPreds: w nodePool: nodePool workList: workList x: x
= (
146         (nonBackPreds at: x dfsNumber) forEach: [:iter |
147             | y ydash |
148             y := nodes at: iter.
149             ydash := y findSet.
150
151             (self isAncestor: w v: ydash dfsNumber) not
152             ifTrue: [
153                 type at: w put: #BBIrreducible.
154                 (nonBackPreds at: w) add: ydash dfsNumber ]
155             ifFalse: [
156                 ydash dfsNumber <> w ifTrue: [
157                     (nodePool hasSome: [:e | e == ydash]) ifFalse: [
158                         workList append: ydash.
159                         nodePool append: ydash ] ] ] ]
160         )
161
162     setLoopAttribute: w nodePool: nodePool loop: loop = (
163         (nodes at: w) loop: loop.
164
165         nodePool forEach: [:node |
166             header at: node dfsNumber put: w.
167             node union: (nodes at: w).
168
169             node loop notNil
170             ifTrue: [ node loop parent: loop ]
171             ifFalse: [ loop addNode: node bb ] ]
172         )
173
174     stepD: w nodePool: nodePool = (
175         (backPreds at: w) forEach: [:v |
176             v <> w ifTrue: [ nodePool append: (nodes at: v) findSet ]
177             ifFalse: [ type at: w put: #BBSelf ] ]

```

```
178     )
179
180     ----
181
182     new: cfg lsg: lsg = (
183         ^ self new init: cfg lsg: lsg
184     )
185 )
186
```

```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8   http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 LoopStructureGraph = (  
18   | root loops loopCounter |  
19  
20   initialize = (  
21     root := SimpleLoop basicBlock: nil reducible: false.  
22     loops := Vector new.  
23     loopCounter := 0.  
24  
25     root nestingLevel: 0.  
26     root counter: loopCounter.  
27     loopCounter := loopCounter + 1.  
28     loops append: root  
29   )  
30  
31   createNewLoop: bb reducible: isReducible = (  
32     | loop |  
33     loop := SimpleLoop basicBlock: bb reducible: isReducible.  
34     loop counter: loopCounter.  
35     loopCounter := loopCounter + 1.  
36     loops append: loop.  
37     ^ loop  
38   )  
39  
40   calculateNestingLevel = (  
41     loops forEach: [:liter |  
42       liter isRoot ifFalse: [  
43         liter parent isNil ifTrue: [  
44           liter parent: root ] ] ].  
45  
46     self calculateNestingLevelRec: root depth: 0  
47   )  
48  
49   calculateNestingLevelRec: loop depth: depth = (  
50     loop depthLevel: depth.  
51     loop children forEach: [:liter |  
52       self calculateNestingLevelRec: liter depth: depth + 1.  
53     loop nestingLevel: (loop nestingLevel max: 1 + liter nestingLevel) ]  
54   )  
55  
56   numLoops = (  
57     ^ loops size  
58   )  
59 )
```

```
59
60 ----
61
62 new = (
63     ^ super new initialize
64 )
65 )
66
```



```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8     http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 LoopTesterApp = (  
18   | cfg lsg |  
19  
20   initialize = (  
21     cfg := ControlFlowGraph new.  
22     lsg := LoopStructureGraph new.  
23     cfg createNode: 1.  
24   )  
25  
26   buildDiamond: start = (  
27     | bb0 |  
28     bb0 := start.  
29     BasicBlockEdge for: cfg from: bb0 to: bb0 + 1.  
30     BasicBlockEdge for: cfg from: bb0 to: bb0 + 2.  
31     BasicBlockEdge for: cfg from: bb0 + 1 to: bb0 + 3.  
32     BasicBlockEdge for: cfg from: bb0 + 2 to: bb0 + 3.  
33     ^ bb0 + 3  
34   )  
35  
36   buildConnect: start end: end = (  
37     BasicBlockEdge for: cfg from: start to: end  
38   )  
39  
40   buildStraight: start n: n = (  
41     0 to: n - 1 do: [:i |  
42       self buildConnect: start + i end: start + i + 1 ].  
43     ^ start + n  
44   )  
45  
46   buildBaseLoop: from = (  
47     | header diamond1 d11 diamond2 footer |  
48     header := self buildStraight: from n: 1.  
49     diamond1 := self buildDiamond: header.  
50     d11 := self buildStraight: diamond1 n: 1.  
51     diamond2 := self buildDiamond: d11.  
52     footer := self buildStraight: diamond2 n: 1.  
53  
54     self buildConnect: diamond2 end: d11.  
55     self buildConnect: diamond1 end: header.  
56     self buildConnect: footer end: from.  
57     footer := self buildStraight: footer n: 1.  
58     ^ footer
```

```

59   )
60
61   main: numDummyLoops loop: findLoopIterations p: parLoop p: pparLoops p:
ppparLoops = (
62     self constructSimpleCFG.
63     self addDummyLoops: numDummyLoops.
64     self constructCFG: parLoop p: pparLoops p: ppparLoops.
65
66     self findLoops: lsg.
67     findLoopIterations timesRepeat: [
68       self findLoops: LoopStructureGraph new ].
69
70     lsg calculateNestingLevel.
71     ^ Array with: lsg numLoops with: cfg numNodes
72   )
73
74   constructCFG: parLoops p: pparLoops p: ppparLoops = (
75     | n |
76     n := 3.
77
78     parLoops timesRepeat: [
79       cfg createNode: n + 1.
80       self buildConnect: 2 end: n + 1.
81       n := n + 1.
82
83       pparLoops timesRepeat: [
84         | top bottom |
85         top := n.
86         n := self buildStraight: n n:1.
87         ppparLoops timesRepeat: [ n := self buildBaseLoop: n ].
88         bottom := self buildStraight: n n: 1.
89         self buildConnect: n end: top.
90         n := bottom ].
91
92     self buildConnect: n end: 1 ]
93   )
94
95   addDummyLoops: numDummyLoops = (
96     numDummyLoops timesRepeat: [
97       self findLoops: lsg ]
98   )
99
100  findLoops: loopStructure = (
101    | finder |
102    finder := HavlakLoopFinder new: cfg lsg: loopStructure.
103    finder findLoops
104  )
105
106  constructSimpleCFG = (
107    cfg createNode: 1.
108    self buildBaseLoop: 1.
109    cfg createNode: 2.
110    BasicBlockEdge for: cfg from: 1 to: 3
111  )
112
113  ----
114
115  new = ( ^ super new initialize )
116 )
117

```

```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8   http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 SimpleLoop = (  
18   | counter depthLevel  
19   parent_ isRoot_ nestingLevel_  
20   header isReducible basicBlocks children |  
21  
22   init: aBB reducible: aBool = (  
23     counter      := 0.  
24     depthLevel  := 0.  
25  
26     isRoot_     := false.  
27     nestingLevel_ := 0.  
28     header      := aBB.  
29     isReducible := aBool.  
30     basicBlocks := SomIdentitySet new.  
31     children    := SomIdentitySet new.  
32  
33     aBB notNil ifTrue: [ basicBlocks add: aBB ]  
34   )  
35  
36   counter = ( ^ counter )  
37   counter: val = ( counter := val )  
38  
39   depthLevel = ( ^ depthLevel )  
40   depthLevel: val = ( depthLevel := val )  
41  
42   children = ( ^ children )  
43  
44   addNode: bb = (  
45     basicBlocks add: bb  
46   )  
47  
48   addChildLoop: loop = (  
49     children add: loop  
50   )  
51  
52   parent = ( ^ parent_ )  
53   parent: val = (  
54     parent_ := val.  
55     parent_ addChildLoop: self  
56   )  
57  
58   isRoot      = ( ^ isRoot_ )
```

```

59  setIsRoot = ( isRoot_ := true )
60
61  nestingLevel = ( ^ nestingLevel_ )
62
63  nestingLevel: level = (
64      nestingLevel_ := level.
65      level = 0 ifTrue: [ self setIsRoot ]
66  )
67
68  ----
69
70  basicBlock: bb reducible: isReducible = (
71      ^ self new init: bb reducible: isReducible
72  )
73 )
74

```

```
1 "  
2 Copyright 2011 Google Inc.  
3  
4 Licensed under the Apache License, Version 2.0 (the 'License');  
5 you may not use this file except in compliance with the License.  
6 You may obtain a copy of the License at  
7  
8   http://www.apache.org/licenses/LICENSE-2.0  
9  
10 Unless required by applicable law or agreed to in writing, software  
11 distributed under the License is distributed on an 'AS IS' BASIS,  
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13 See the License for the specific language governing permissions and  
14 limitations under the License.  
15 "  
16  
17 UnionFindNode = (  
18   | parent_ bb_ dfsNumber_ loop |  
19  
20   initialize = (  
21     dfsNumber_ := 0.  
22   )  
23  
24   initNode: bb dfs: dfsNumber = (  
25     parent_ := self.  
26     bb_ := bb.  
27     dfsNumber_ := dfsNumber.  
28   )  
29  
30   loop = ( ^ loop )  
31   loop: aLoop = ( loop := aLoop )  
32  
33   findSet = (  
34     | nodeList node |  
35     nodeList := Vector new.  
36  
37     node := self.  
38  
39     [node ~= node parent] whileTrue: [  
40       ((node parent) ~= (node parent parent)) ifTrue: [  
41         nodeList append: node ].  
42       node := node parent ].  
43  
44     nodeList forEach: [:iter | iter union: parent_ ].  
45     ^ node  
46   )  
47  
48   union: basicBlock = (  
49     parent_ := basicBlock  
50   )  
51  
52   parent    = ( ^ parent_ )  
53   bb        = ( ^ bb_ )  
54   dfsNumber = ( ^ dfsNumber_ )  
55  
56   ----  
57  
58   new = ( ^ super new initialize )
```

59)
60

```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 HashIndexTable = (  
27   | hashTable |  
28  
29   initialize = (  
30     hashTable := Array new: 32 withAll: 0  
31   )  
32  
33   at: name put: index = (  
34     | slot |  
35     slot := self hashSlotFor: name.  
36  
37     index < 255  
38     ifTrue: [ hashTable at: slot put: index + 1 ]  
39     ifFalse: [ hashTable at: slot put: 0 ]  
40   )  
41  
42   at: name = (  
43     | slot |  
44     slot := self hashSlotFor: name.  
45  
46     " subtract 1, 0 stands for empty "  
47     ^ ((hashTable at: slot) & 255) - 1  
48   )  
49  
50   stringHash: s = (  
51     "this is not a proper hash, but sufficient for the benchmark,  
52     and very portable!"
```

```
53      ^ s length * 1402589
54  )
55
56  hashSlotFor: element = (
57      ^ ((self stringHash: element) & (hashTable length - 1)) + 1
58  )
59
60  ----
61
62  new = ( ^ super new initialize )
63 )
64
```



```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 Json = Benchmark (  
27  
28   benchmark = (  
29     ^ (JsonParser with: Json RapBenchmarkMinified) parse.  
30   )  
31  
32   verifyResult: result = (  
33     result isObject ifFalse: [ ^ false ].  
34     (result asObject at: 'head') isObject ifFalse: [ ^ false ].  
35     (result asObject at: 'operations') isArray ifFalse: [ ^ false ].  
36     ^ (result asObject at: 'operations') asArray size = 156  
37   )  
38  
39   ----  
40  
41   new = (  
42     JsonLiteral initialize.  
43     ^ super new  
44   )  
45  
46   RapBenchmarkMinified = (  
47     ^ ' {"head":{"requestCounter":4},"operations":[["destroy","w54"],  
["set","w2",{"activeControl":"w99"}],["set","w21",  
{"customVariant":"variant_navigation"}],["set","w28",  
{"customVariant":"variant_selected"}],["set","w53",{"children":["w95"]}]],  
["create","w95","rwt.widgets.Composite",{"parent":"w53","style":  
["NONE"],"bounds":[0,0,1008,586],"children":
```

```

["w96","w97"],["tabIndex":-1,"clientArea":[0,0,1008,586]]],
["create","w96","rwt.widgets.Label",{ "parent":"w95","style":["NONE"],"bounds":
[10,30,112,26],"tabIndex":-1,"customVariant":"variant_pageHeadline","text":"Ta
bleViewer"}],[ "create","w97","rwt.widgets.Composite",{ "parent":"w95","style":
["NONE"],"bounds":[0,61,1008,525],"children":
["w98","w99","w226","w228"],["tabIndex":-1,"clientArea":[0,0,1008,525]]],
["create","w98","rwt.widgets.Text",{ "parent":"w97","style":
["LEFT","SINGLE","BORDER"],"bounds":[10,10,988,32],"tabIndex":22,"activeKeys":
["#13","#27","#40"]}],["listen","w98",{ "KeyDown":true,"Modify":true}],
["create","w99","rwt.widgets.Grid",{ "parent":"w97","style":["SINGLE","BORDER"]
,"appearance":"table","indentationWidth":0,"treeColumn":-1,"markupEnabled":false
}],["create","w100","rwt.widgets.ScrollBar",{ "parent":"w99","style":
["HORIZONTAL"]}],["create","w101","rwt.widgets.ScrollBar",
{ "parent":"w99","style":["VERTICAL"]}],["set","w99",{ "bounds":
[10,52,988,402],"children":[],"tabIndex":23,"activeKeys":["CTRL+#70","CTRL+#78
","CTRL+#82","CTRL+#89","CTRL+#83","CTRL+#71","CTRL+#69"],"cancelKeys":
["CTRL+#70","CTRL+#78","CTRL+#82","CTRL+#89","CTRL+#83","CTRL+#71","CTRL+#69"]}],
["listen","w99",
{ "MouseDown":true,"MouseUp":true,"MouseDoubleClick":true,"KeyDown":true}],
["set","w99",{ "itemCount":118,"itemHeight":28,"itemMetrics":
[[0,0,50,3,0,3,44],[1,50,50,53,0,53,44],[2,100,140,103,0,103,134],
[3,240,180,243,0,243,174],[4,420,50,423,0,423,44],[5,470,50,473,0,473,44]],"co
lumnCount":6,"headerHeight":35,"headerVisible":true,"linesVisible":true,"focus
Item":"w108","selection":["w108"]}],["listen","w99",
{ "Selection":true,"DefaultSelection":true}],["set","w99",
{ "enableCellToolTip":true}],["listen","w100",{ "Selection":true}],
["set","w101",{ "visibility":true}],["listen","w101",{ "Selection":true}],
["create","w102","rwt.widgets.GridColumn",
{ "parent":"w99","text":"Nr.","width":50,"moveable":true}],["listen","w102",
{ "Selection":true}],["create","w103","rwt.widgets.GridColumn",
{ "parent":"w99","text":"Sym.","index":1,"left":50,"width":50,"moveable":true}],
["listen","w103",{ "Selection":true}],
["create","w104","rwt.widgets.GridColumn",
{ "parent":"w99","text":"Name","index":2,"left":100,"width":140,"moveable":true}],
["listen","w104",{ "Selection":true}],
["create","w105","rwt.widgets.GridColumn",{ "parent":"w99","text":"Series","ind
ex":3,"left":240,"width":180,"moveable":true}],["listen","w105",
{ "Selection":true}],["create","w106","rwt.widgets.GridColumn",
{ "parent":"w99","text":"Group","index":4,"left":420,"width":50,"moveable":true}],
["listen","w106",{ "Selection":true}],
["create","w107","rwt.widgets.GridColumn",{ "parent":"w99","text":"Period","ind
ex":5,"left":470,"width":50,"moveable":true}],["listen","w107",
{ "Selection":true}],["create","w108","rwt.widgets.GridItem",
{ "parent":"w99","index":0,"texts":
["1","H","Hydrogen","Nonmetal","1","1"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w109","rwt.widgets.GridItem",
{ "parent":"w99","index":1,"texts":["2","He","Helium","Noble
gas","18","1"],"cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]}],["create","w110","rwt.widgets.GridItem",
{ "parent":"w99","index":2,"texts":["3","Li","Lithium","Alkali
metal","1","2"],"cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]}],["create","w111","rwt.widgets.GridItem",
{ "parent":"w99","index":3,"texts":["4","Be","Beryllium","Alkaline earth
metal","2","2"],"cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]}],["create","w112","rwt.widgets.GridItem",
{ "parent":"w99","index":4,"texts":
["5","B","Boron","Metalloid","13","2"],"cellBackgrounds":[null,null,null,
[156,159,153,255],null,null]}],["create","w113","rwt.widgets.GridItem",
{ "parent":"w99","index":5,"texts":
["6","C","Carbon","Nonmetal","14","2"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w114","rwt.widgets.GridItem",

```

```

{"parent":"w99","index":6,"texts":
[["7","N","Nitrogen","Nonmetal","15","2"],["cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]]],[["create","w115","rwt.widgets.GridItem",
{"parent":"w99","index":7,"texts":
[["8","O","Oxygen","Nonmetal","16","2"],["cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]]],[["create","w116","rwt.widgets.GridItem",
{"parent":"w99","index":8,"texts":
[["9","F","Fluorine","Halogen","17","2"],["cellBackgrounds":[null,null,null,
[252,233,79,255],null,null]]],[["create","w117","rwt.widgets.GridItem",
{"parent":"w99","index":9,"texts":["10","Ne","Neon","Noble
gas","18","2"],["cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]]],[["create","w118","rwt.widgets.GridItem",
{"parent":"w99","index":10,"texts":["11","Na","Sodium","Alkali
metal","1","3"],["cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]]],[["create","w119","rwt.widgets.GridItem",
{"parent":"w99","index":11,"texts":["12","Mg","Magnesium","Alkaline earth
metal","2","3"],["cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]]],[["create","w120","rwt.widgets.GridItem",
{"parent":"w99","index":12,"texts":["13","Al","Aluminium","Poor
metal","13","3"],["cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]],[["create","w121","rwt.widgets.GridItem",
{"parent":"w99","index":13,"texts":
[["14","Si","Silicon","Metalloid","14","3"],["cellBackgrounds":[null,null,null,
[156,159,153,255],null,null]]],[["create","w122","rwt.widgets.GridItem",
{"parent":"w99","index":14,"texts":
[["15","P","Phosphorus","Nonmetal","15","3"],["cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]]],[["create","w123","rwt.widgets.GridItem",
{"parent":"w99","index":15,"texts":
[["16","S","Sulfur","Nonmetal","16","3"],["cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]]],[["create","w124","rwt.widgets.GridItem",
{"parent":"w99","index":16,"texts":
[["17","Cl","Chlorine","Halogen","17","3"],["cellBackgrounds":[null,null,null,
[252,233,79,255],null,null]]],[["create","w125","rwt.widgets.GridItem",
{"parent":"w99","index":17,"texts":["18","Ar","Argon","Noble
gas","18","3"],["cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]]],[["create","w126","rwt.widgets.GridItem",
{"parent":"w99","index":18,"texts":["19","K","Potassium","Alkali
metal","1","4"],["cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]]],[["create","w127","rwt.widgets.GridItem",
{"parent":"w99","index":19,"texts":["20","Ca","Calcium","Alkaline earth
metal","2","4"],["cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]]],[["create","w128","rwt.widgets.GridItem",
{"parent":"w99","index":20,"texts":["21","Sc","Scandium","Transition
metal","3","4"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]],[["create","w129","rwt.widgets.GridItem",
{"parent":"w99","index":21,"texts":["22","Ti","Titanium","Transition
metal","4","4"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]],[["create","w130","rwt.widgets.GridItem",
{"parent":"w99","index":22,"texts":["23","V","Vanadium","Transition
metal","5","4"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]],[["create","w131","rwt.widgets.GridItem",
{"parent":"w99","index":23,"texts":["24","Cr","Chromium","Transition
metal","6","4"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]],[["create","w132","rwt.widgets.GridItem",
{"parent":"w99","index":24,"texts":["25","Mn","Manganese","Transition
metal","7","4"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]],[["create","w133","rwt.widgets.GridItem",
{"parent":"w99","index":25,"texts":["26","Fe","Iron","Transition
metal","8","4"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]],[["create","w134","rwt.widgets.GridItem",
{"parent":"w99","index":26,"texts":["27","Co","Cobalt","Transition

```

```

metal", "9", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w135", "rwt.widgets.GridItem",
{"parent": "w99", "index": 27, "texts": ["28", "Ni", "Nickel", "Transition
metal", "10", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w136", "rwt.widgets.GridItem",
{"parent": "w99", "index": 28, "texts": ["29", "Cu", "Copper", "Transition
metal", "11", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w137", "rwt.widgets.GridItem",
{"parent": "w99", "index": 29, "texts": ["30", "Zn", "Zinc", "Transition
metal", "12", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w138", "rwt.widgets.GridItem",
{"parent": "w99", "index": 30, "texts": ["31", "Ga", "Gallium", "Poor
metal", "13", "4"], "cellBackgrounds": [null, null, null,
[238, 238, 236, 255], null, null]]], [{"create", "w139", "rwt.widgets.GridItem",
{"parent": "w99", "index": 31, "texts":
["32", "Ge", "Germanium", "Metalloid", "14", "4"], "cellBackgrounds":
[null, null, null, [156, 159, 153, 255], null, null]]],
[{"create", "w140", "rwt.widgets.GridItem", {"parent": "w99", "index": 32, "texts":
["33", "As", "Arsenic", "Metalloid", "15", "4"], "cellBackgrounds": [null, null, null,
[156, 159, 153, 255], null, null]]], [{"create", "w141", "rwt.widgets.GridItem",
{"parent": "w99", "index": 33, "texts":
["34", "Se", "Selenium", "Nonmetal", "16", "4"], "cellBackgrounds": [null, null, null,
[138, 226, 52, 255], null, null]]], [{"create", "w142", "rwt.widgets.GridItem",
{"parent": "w99", "index": 34, "texts":
["35", "Br", "Bromine", "Halogen", "17", "4"], "cellBackgrounds": [null, null, null,
[252, 233, 79, 255], null, null]]], [{"create", "w143", "rwt.widgets.GridItem",
{"parent": "w99", "index": 35, "texts": ["36", "Kr", "Krypton", "Noble
gas", "18", "4"], "cellBackgrounds": [null, null, null,
[114, 159, 207, 255], null, null]]], [{"create", "w144", "rwt.widgets.GridItem",
{"parent": "w99", "index": 36, "texts": ["37", "Rb", "Rubidium", "Alkali
metal", "1", "5"], "cellBackgrounds": [null, null, null,
[239, 41, 41, 255], null, null]]], [{"create", "w145", "rwt.widgets.GridItem",
{"parent": "w99", "index": 37, "texts": ["38", "Sr", "Strontium", "Alkaline earth
metal", "2", "5"], "cellBackgrounds": [null, null, null,
[233, 185, 110, 255], null, null]]], [{"create", "w146", "rwt.widgets.GridItem",
{"parent": "w99", "index": 38, "texts": ["39", "Y", "Yttrium", "Transition
metal", "3", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w147", "rwt.widgets.GridItem",
{"parent": "w99", "index": 39, "texts": ["40", "Zr", "Zirconium", "Transition
metal", "4", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w148", "rwt.widgets.GridItem",
{"parent": "w99", "index": 40, "texts": ["41", "Nb", "Niobium", "Transition
metal", "5", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w149", "rwt.widgets.GridItem",
{"parent": "w99", "index": 41, "texts": ["42", "Mo", "Molybdenum", "Transition
metal", "6", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w150", "rwt.widgets.GridItem",
{"parent": "w99", "index": 42, "texts": ["43", "Tc", "Technetium", "Transition
metal", "7", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w151", "rwt.widgets.GridItem",
{"parent": "w99", "index": 43, "texts": ["44", "Ru", "Ruthenium", "Transition
metal", "8", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w152", "rwt.widgets.GridItem",
{"parent": "w99", "index": 44, "texts": ["45", "Rh", "Rhodium", "Transition
metal", "9", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w153", "rwt.widgets.GridItem",
{"parent": "w99", "index": 45, "texts": ["46", "Pd", "Palladium", "Transition
metal", "10", "5"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]], [{"create", "w154", "rwt.widgets.GridItem",
{"parent": "w99", "index": 46, "texts": ["47", "Ag", "Silver", "Transition
metal", "11", "5"], "cellBackgrounds": [null, null, null,

```

```

[252,175,62,255],null,null]]],[ "create","w155","rwt.widgets.GridItem",
{"parent":"w99","index":47,"texts":["48","Cd","Cadmium","Transition
metal","12","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]],[ "create","w156","rwt.widgets.GridItem",
{"parent":"w99","index":48,"texts":["49","In","Indium","Poor
metal","13","5"],"cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]],[ "create","w157","rwt.widgets.GridItem",
{"parent":"w99","index":49,"texts":["50","Sn","Tin","Poor
metal","14","5"],"cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]],[ "create","w158","rwt.widgets.GridItem",
{"parent":"w99","index":50,"texts":
["51","Sb","Antimony","Metalloid","15","5"],"cellBackgrounds":[null,null,null,
[156,159,153,255],null,null]]],[ "create","w159","rwt.widgets.GridItem",
{"parent":"w99","index":51,"texts":
["52","Te","Tellurium","Metalloid","16","5"],"cellBackgrounds":
[null,null,null,[156,159,153,255],null,null]]],[
["create","w160","rwt.widgets.GridItem",{ "parent":"w99","index":52,"texts":
["53","I","Iodine","Halogen","17","5"],"cellBackgrounds":[null,null,null,
[252,233,79,255],null,null]]],[ "create","w161","rwt.widgets.GridItem",
{"parent":"w99","index":53,"texts":["54","Xe","Xenon","Noble
gas","18","5"],"cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]]],[ "create","w162","rwt.widgets.GridItem",
{"parent":"w99","index":54,"texts":["55","Cs","Caesium","Alkali
metal","1","6"],"cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]]],[ "create","w163","rwt.widgets.GridItem",
{"parent":"w99","index":55,"texts":["56","Ba","Barium","Alkaline earth
metal","2","6"],"cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]]],[ "create","w164","rwt.widgets.GridItem",
{"parent":"w99","index":56,"texts":
["57","La","Lanthanum","Lanthanide","3","6"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]],[
["create","w165","rwt.widgets.GridItem",{ "parent":"w99","index":57,"texts":
["58","Ce","Cerium","Lanthanide","3","6"],"cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[ "create","w166","rwt.widgets.GridItem",
{"parent":"w99","index":58,"texts":
["59","Pr","Praseodymium","Lanthanide","3","6"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]],[
["create","w167","rwt.widgets.GridItem",{ "parent":"w99","index":59,"texts":
["60","Nd","Neodymium","Lanthanide","3","6"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]],[
["create","w168","rwt.widgets.GridItem",{ "parent":"w99","index":60,"texts":
["61","Pm","Promethium","Lanthanide","3","6"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]],[
["create","w169","rwt.widgets.GridItem",{ "parent":"w99","index":61,"texts":
["62","Sm","Samarium","Lanthanide","3","6"],"cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[ "create","w170","rwt.widgets.GridItem",
{"parent":"w99","index":62,"texts":
["63","Eu","Europium","Lanthanide","3","6"],"cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[ "create","w171","rwt.widgets.GridItem",
{"parent":"w99","index":63,"texts":
["64","Gd","Gadolinium","Lanthanide","3","6"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]],[
["create","w172","rwt.widgets.GridItem",{ "parent":"w99","index":64,"texts":
["65","Tb","Terbium","Lanthanide","3","6"],"cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[ "create","w173","rwt.widgets.GridItem",
{"parent":"w99","index":65,"texts":
["66","Dy","Dysprosium","Lanthanide","3","6"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]],[
["create","w174","rwt.widgets.GridItem",{ "parent":"w99","index":66,"texts":
["67","Ho","Holmium","Lanthanide","3","6"],"cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[ "create","w175","rwt.widgets.GridItem",

```

```

{"parent":"w99","index":67,"texts":
[["68","Er","Erbium","Lanthanide","3","6"],["cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[{"create","w176","rwt.widgets.GridItem",
{"parent":"w99","index":68,"texts":
[["69","Tm","Thulium","Lanthanide","3","6"],["cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[{"create","w177","rwt.widgets.GridItem",
{"parent":"w99","index":69,"texts":
[["70","Yb","Ytterbium","Lanthanide","3","6"],["cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]}],
[{"create","w178","rwt.widgets.GridItem",{ "parent":"w99","index":70,"texts":
[["71","Lu","Lutetium","Lanthanide","3","6"],["cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]],[{"create","w179","rwt.widgets.GridItem",
{"parent":"w99","index":71,"texts":["72","Hf","Hafnium","Transition
metal","4","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w180","rwt.widgets.GridItem",
{"parent":"w99","index":72,"texts":["73","Ta","Tantalum","Transition
metal","5","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w181","rwt.widgets.GridItem",
{"parent":"w99","index":73,"texts":["74","W","Tungsten","Transition
metal","6","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w182","rwt.widgets.GridItem",
{"parent":"w99","index":74,"texts":["75","Re","Rhenium","Transition
metal","7","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w183","rwt.widgets.GridItem",
{"parent":"w99","index":75,"texts":["76","Os","Osmium","Transition
metal","8","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w184","rwt.widgets.GridItem",
{"parent":"w99","index":76,"texts":["77","Ir","Iridium","Transition
metal","9","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w185","rwt.widgets.GridItem",
{"parent":"w99","index":77,"texts":["78","Pt","Platinum","Transition
metal","10","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w186","rwt.widgets.GridItem",
{"parent":"w99","index":78,"texts":["79","Au","Gold","Transition
metal","11","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w187","rwt.widgets.GridItem",
{"parent":"w99","index":79,"texts":["80","Hg","Mercury","Transition
metal","12","6"],["cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]}],["create","w188","rwt.widgets.GridItem",
{"parent":"w99","index":80,"texts":["81","Tl","Thallium","Poor
metal","13","6"],["cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]}],["create","w189","rwt.widgets.GridItem",
{"parent":"w99","index":81,"texts":["82","Pb","Lead","Poor
metal","14","6"],["cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]}],["create","w190","rwt.widgets.GridItem",
{"parent":"w99","index":82,"texts":["83","Bi","Bismuth","Poor
metal","15","6"],["cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]}],["create","w191","rwt.widgets.GridItem",
{"parent":"w99","index":83,"texts":
[["84","Po","Polonium","Metalloid","16","6"],["cellBackgrounds":[null,null,null,
[156,159,153,255],null,null]]}],["create","w192","rwt.widgets.GridItem",
{"parent":"w99","index":84,"texts":
[["85","At","Astatine","Halogen","17","6"],["cellBackgrounds":[null,null,null,
[252,233,79,255],null,null]]}],["create","w193","rwt.widgets.GridItem",
{"parent":"w99","index":85,"texts":["86","Rn","Radon","Noble
gas","18","6"],["cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]]}],["create","w194","rwt.widgets.GridItem",
{"parent":"w99","index":86,"texts":["87","Fr","Francium","Alkali
metal","1","7"],["cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]]}],["create","w195","rwt.widgets.GridItem",
{"parent":"w99","index":87,"texts":["88","Ra","Radium","Alkaline earth

```

```

metal", "2", "7"], "cellBackgrounds": [null, null, null,
[233, 185, 110, 255], null, null]]}, {"create", "w196", "rwt.widgets.GridItem",
{"parent": "w99", "index": 88, "texts":
["89", "Ac", "Actinium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w197", "rwt.widgets.GridItem",
{"parent": "w99", "index": 89, "texts":
["90", "Th", "Thorium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w198", "rwt.widgets.GridItem",
{"parent": "w99", "index": 90, "texts":
["91", "Pa", "Protactinium", "Actinide", "3", "7"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]]},
{"create", "w199", "rwt.widgets.GridItem", {"parent": "w99", "index": 91, "texts":
["92", "U", "Uranium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w200", "rwt.widgets.GridItem",
{"parent": "w99", "index": 92, "texts":
["93", "Np", "Neptunium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w201", "rwt.widgets.GridItem",
{"parent": "w99", "index": 93, "texts":
["94", "Pu", "Plutonium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w202", "rwt.widgets.GridItem",
{"parent": "w99", "index": 94, "texts":
["95", "Am", "Americium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w203", "rwt.widgets.GridItem",
{"parent": "w99", "index": 95, "texts":
["96", "Cm", "Curium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w204", "rwt.widgets.GridItem",
{"parent": "w99", "index": 96, "texts":
["97", "Bk", "Berkelium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w205", "rwt.widgets.GridItem",
{"parent": "w99", "index": 97, "texts":
["98", "Cf", "Californium", "Actinide", "3", "7"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]]},
{"create", "w206", "rwt.widgets.GridItem", {"parent": "w99", "index": 98, "texts":
["99", "Es", "Einsteinium", "Actinide", "3", "7"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]]},
{"create", "w207", "rwt.widgets.GridItem", {"parent": "w99", "index": 99, "texts":
["100", "Fm", "Fermium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w208", "rwt.widgets.GridItem",
{"parent": "w99", "index": 100, "texts":
["101", "Md", "Mendelevium", "Actinide", "3", "7"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]]},
{"create", "w209", "rwt.widgets.GridItem", {"parent": "w99", "index": 101, "texts":
["102", "No", "Nobelium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w210", "rwt.widgets.GridItem",
{"parent": "w99", "index": 102, "texts":
["103", "Lr", "Lawrencium", "Actinide", "3", "7"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]]},
{"create", "w211", "rwt.widgets.GridItem", {"parent": "w99", "index": 103, "texts":
["104", "Rf", "Rutherfordium", "Transition metal", "4", "7"], "cellBackgrounds":
[null, null, null, [252, 175, 62, 255], null, null]]},
{"create", "w212", "rwt.widgets.GridItem", {"parent": "w99", "index": 104, "texts":
["105", "Db", "Dubnium", "Transition metal", "5", "7"], "cellBackgrounds":
[null, null, null, [252, 175, 62, 255], null, null]]},
{"create", "w213", "rwt.widgets.GridItem", {"parent": "w99", "index": 105, "texts":
["106", "Sg", "Seaborgium", "Transition metal", "6", "7"], "cellBackgrounds":
[null, null, null, [252, 175, 62, 255], null, null]]},
{"create", "w214", "rwt.widgets.GridItem", {"parent": "w99", "index": 106, "texts":
["107", "Bh", "Bohrium", "Transition metal", "7", "7"], "cellBackgrounds":
[null, null, null, [252, 175, 62, 255], null, null]]},
{"create", "w215", "rwt.widgets.GridItem", {"parent": "w99", "index": 107, "texts":
["108", "Hs", "Hassium", "Transition metal", "8", "7"], "cellBackgrounds":

```

```

[null,null,null,[252,175,62,255],null,null]]],
["create","w216","rwt.widgets.GridItem",{ "parent":"w99","index":108,"texts":
["109","Mt","Meitnerium","Transition metal","9","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]],
["create","w217","rwt.widgets.GridItem",{ "parent":"w99","index":109,"texts":
["110","Ds","Darmstadtium","Transition metal","10","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]],
["create","w218","rwt.widgets.GridItem",{ "parent":"w99","index":110,"texts":
["111","Rg","Roentgenium","Transition metal","11","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]],
["create","w219","rwt.widgets.GridItem",{ "parent":"w99","index":111,"texts":
["112","Uub","Ununbium","Transition metal","12","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]],
["create","w220","rwt.widgets.GridItem",{ "parent":"w99","index":112,"texts":
["113","Uut","Ununtrium","Poor metal","13","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]],
["create","w221","rwt.widgets.GridItem",{ "parent":"w99","index":113,"texts":
["114","Uuq","Ununquadium","Poor metal","14","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]],
["create","w222","rwt.widgets.GridItem",{ "parent":"w99","index":114,"texts":
["115","Uup","Ununpentium","Poor metal","15","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]],
["create","w223","rwt.widgets.GridItem",{ "parent":"w99","index":115,"texts":
["116","Uuh","Ununhexium","Poor metal","16","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]],
["create","w224","rwt.widgets.GridItem",{ "parent":"w99","index":116,"texts":
["117","Uus","Ununseptium","Halogen","17","7"],"cellBackgrounds":
[null,null,null,[252,233,79,255],null,null]]],
["create","w225","rwt.widgets.GridItem",{ "parent":"w99","index":117,"texts":
["118","Uuo","Ununoctium","Noble gas","18","7"],"cellBackgrounds":
[null,null,null,[114,159,207,255],null,null]]],
["create","w226","rwt.widgets.Composite",{ "parent":"w97","style":
["BORDER"],"bounds":[10,464,988,25],"children":
["w227"],"tabIndex":-1,"clientArea":[0,0,986,23]]},
["create","w227","rwt.widgets.Label",{ "parent":"w226","style":
["NONE"],"bounds":[10,10,966,3],"tabIndex":-1,"text":"Hydrogen (H)"}],
["create","w228","rwt.widgets.Label",{ "parent":"w97","style":
["WRAP"],"bounds":[10,499,988,16],"tabIndex":-1,"foreground":
[150,150,150,255],"font":[["Verdana","Lucida Sans","Arial","Helvetica","sans-
serif"],10,false,false],"text":"Shortcuts: [CTRL+F] - Filter | Sort by:
[CTRL+R] - Number, [CTRL+Y] - Symbol, [CTRL+N] - Name, [CTRL+S] - Series,
[CTRL+G] - Group, [CTRL+E] - Period"}],["set","w1",{ "focusControl":"w99"}],
["call","rwt.client.BrowserNavigation","addToHistory",{ "entries":
[["tableviewer","TableViewer"]]]}]'
48 )
49 )

```



```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 JsonArray = JsonValue (  
27   | values |  
28  
29   initialize = ( values := Vector new )  
30  
31   add: value = (  
32     value ifNil: [ self error: 'value is null' ].  
33     values append: value  
34   )  
35  
36   size = (  
37     ^ values size  
38   )  
39  
40   at: index = (  
41     ^ values at: index  
42   )  
43  
44   isArray = (  
45     ^ true  
46   )  
47  
48   asArray = (  
49     ^ self  
50   )  
51  
52 ----
```

```
53
54   new = ( ^ super new initialize )
55 )
56
```

```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 JsonLiteral = JsonValue (  
27   | value isNull isTrue isFalse |  
28  
29   initializeWith: val = (  
30     value      := val.  
31     isNull     := 'null'  = val.  
32     isTrue     := 'true'  = val.  
33     isFalse    := 'false' = val.  
34   )  
35  
36   asString      = ( ^ value )  
37   isNull        = ( ^ isNull )  
38   isTrue        = ( ^ isTrue )  
39   isFalse       = ( ^ isFalse )  
40   isBoolean     = ( ^ isTrue || isFalse )  
41  
42   ----  
43  
44   | NULL TRUE FALSE |  
45  
46   initialize = (  
47     NULL := self new initializeWith: 'null'.  
48     TRUE  := self new initializeWith: 'true'.  
49     FALSE := self new initializeWith: 'false'.  
50   )  
51  
52   NULL = ( ^ NULL )
```

```
53  TRUE  = ( ^ TRUE )
54  FALSE = ( ^ FALSE )
55 )
56
```

```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 JsonNumber = JsonValue (  
27   | string |  
28  
29   initializeWith: str = ( string := str )  
30  
31   asString = ( ^ string )  
32   isNumber = ( ^ true )  
33  
34   ----  
35  
36   new: string = (  
37     string ifNil: [ self error: 'string is null' ].  
38     ^ self new initializeWith: string  
39   )  
40 )  
41
```

```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 JsonObject = JsonValue (  
27   | names values table |  
28  
29   initialize = (  
30     names := Vector new.  
31     values := Vector new.  
32     table := HashIndexTable new  
33   )  
34  
35   add: name with: aJsonValue = (  
36     name ifNil: [ self error: 'name is null' ].  
37     aJsonValue ifNil: [ self error: 'aJsonValue is null' ].  
38  
39     table at: name put: names size + 1. "+ 1 for 1-based indexing"  
40     names append: name.  
41     values append: aJsonValue.  
42   )  
43  
44   at: name = (  
45     | idx |  
46     name ifNil: [ self error: 'name is null' ].  
47     idx := self indexOf: name.  
48     idx = 0  
49     ifTrue: [ ^ nil ]  
50     ifFalse: [ ^ values at: idx ]  
51   )  
52
```

```

53  size = (
54      ^ names size
55  )
56
57  isEmpty = (
58      ^ names isEmpty
59  )
60
61  isObject = ( ^ true )
62  asObject = ( ^ self )
63
64  indexOf: name = (
65      | idx |
66      idx := table at: name.
67      idx <> 0 && (name = (names at: idx)) ifTrue: [ ^ idx ].
68      ^ self error: 'not implement'
69  )
70
71  ----
72
73  new = ( ^ super new initialize )
74
75  readFrom: string = (
76      ^ (JsonValue readFrom: string) asObject
77  )
78 )
79

```

```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included  
in all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 JsonParser = (  
27   | input index line column current captureBuffer captureStart  
exceptionBlock |  
28  
29   initializeWith: string = (  
30     input := string.  
31     index := 0.  
32     line := 1.  
33     column := 0.  
34     current := nil.  
35     captureBuffer := ''.  
36     captureStart := -1.  
37   )  
38  
39   parse = (  
40     | result |  
41     exceptionBlock := [:ex | ^ ex ].  
42     self read.  
43     self skipWhiteSpace.  
44     result := self readValue.  
45     self skipWhiteSpace.  
46     self isEndOfText ifFalse: [ self error: 'Unexpected character'].  
47     ^ result  
48   )  
49  
50   readValue = (  

```



```

51     current = 'n' ifTrue: [ ^ self readNull ].
52     current = 't' ifTrue: [ ^ self readTrue ].
53     current = 'f' ifTrue: [ ^ self readFalse ].
54     current = '"' ifTrue: [ ^ self readString ].
55     current = '[' ifTrue: [ ^ self readArray ].
56     current = '{' ifTrue: [ ^ self readObject ].
57
58     "Is this really the best way to write this?, or better #or:?,
59     but with all the nesting, it's just ugly."
60     current = '-' ifTrue: [ ^ self readNumber ].
61     current = '0' ifTrue: [ ^ self readNumber ].
62     current = '1' ifTrue: [ ^ self readNumber ].
63     current = '2' ifTrue: [ ^ self readNumber ].
64     current = '3' ifTrue: [ ^ self readNumber ].
65     current = '4' ifTrue: [ ^ self readNumber ].
66     current = '5' ifTrue: [ ^ self readNumber ].
67     current = '6' ifTrue: [ ^ self readNumber ].
68     current = '7' ifTrue: [ ^ self readNumber ].
69     current = '8' ifTrue: [ ^ self readNumber ].
70     current = '9' ifTrue: [ ^ self readNumber ].
71
72     "else"
73     self expected: 'value'
74 )
75
76 readArrayElement: array = (
77     self skipWhiteSpace.
78     array add: self readValue.
79     self skipWhiteSpace.
80 )
81
82 readArray = (
83     | array |
84     self read.
85     array := JSONArray new.
86
87     "Array might be empty"
88     self skipWhiteSpace.
89     (self readChar: ']') ifTrue: [
90         ^ array
91     ].
92
93     self readArrayElement: array.
94     [self readChar: ','] whileTrue: [
95         self readArrayElement: array.
96     ].
97
98     (self readChar: ']') ifFalse: [
99         self expected: '"', ' or '"]
100 ].
101 ^ array
102 )
103
104 readObjectKeyValuePair: object = (
105     | name |
106     self skipWhiteSpace.
107     name := self readName.
108     self skipWhiteSpace.
109
110     (self readChar: ':') ifFalse: [ self expected: ':' ].
111

```

```

112     self skipWhiteSpace.
113
114     object add: name with: self readValue.
115
116     self skipWhiteSpace.
117 )
118
119 readObject = (
120     | object |
121     self read.
122     object := JsonObject new.
123     self skipWhiteSpace.
124
125     (self readChar: '}') ifTrue: [
126         ^ object
127     ].
128
129     self readObjectKeyValuePair: object.
130     [self readChar: ','] whileTrue: [
131         self readObjectKeyValuePair: object.
132     ].
133
134     (self readChar: '}') ifFalse: [
135         self expected: '"', ' ' or '}'
136     ].
137
138     ^ object
139 )
140
141 readName = (
142     current = '' ifFalse: [ self expected: 'name' ].
143     ^ self readStringInternal
144 )
145
146 readNull = (
147     self read.
148     self readRequiredChar: 'u'.
149     self readRequiredChar: 'l'.
150     self readRequiredChar: 'l'.
151     ^ JsonLiteral NULL
152 )
153
154 readTrue = (
155     self read.
156     self readRequiredChar: 'r'.
157     self readRequiredChar: 'u'.
158     self readRequiredChar: 'e'.
159     ^ JsonLiteral TRUE
160 )
161
162 readFalse = (
163     self read.
164     self readRequiredChar: 'a'.
165     self readRequiredChar: 'l'.
166     self readRequiredChar: 's'.
167     self readRequiredChar: 'e'.
168     ^ JsonLiteral FALSE
169 )
170
171 readRequiredChar: ch = (
172     (self readChar: ch) ifFalse: [

```

```

173         self expected: 'character: ' + ch
174     ]
175 )
176
177 readString = (
178     ^ JsonString new: self readStringInternal
179 )
180
181 readStringInternal = (
182     | string |
183     self read.
184     self startCapture.
185
186     [current = '"] whileFalse: [
187         current = '\\' ifTrue: [
188             self pauseCapture.
189             self readEscape.
190             self startCapture.
191         ] ifFalse: [
192             "if (current < 0x20) { throw expected('valid string
character'); }"
193             "we currently don't have a way to get the ordinal value for a
character"
194             "} else {"
195             self read.
196         ]
197     ].
198     string := self endCapture.
199     self read.
200     ^ string
201 )
202
203 readEscapeChar = (
204     current = '"' ifTrue: [ ^ '"' ].
205     current = '/' ifTrue: [ ^ '/' ].
206     current = '\\' ifTrue: [ ^ '\\' ].
207
208     current = 'b' ifTrue: [ ^ '\b' ].
209     current = 'f' ifTrue: [ ^ '\f' ].
210     current = 'n' ifTrue: [ ^ '\n' ].
211     current = 'r' ifTrue: [ ^ '\r' ].
212     current = 't' ifTrue: [ ^ '\t' ].
213
214     "TODO: SOM doesn't have a way to create unicode characters."
215     self expected: 'valid escape sequence. note, some are not supported'
216 )
217
218 readEscape = (
219     self read.
220     captureBuffer := captureBuffer concatenate: self readEscapeChar.
221     self read
222 )
223
224 readNumber = (
225     | firstDigit |
226     self startCapture.
227     self readChar: '-'.
228     firstDigit := current.
229
230     self readDigit ifFalse: [ self expected: 'digit' ].
231     firstDigit <> '0' ifTrue: [ [self readDigit] whileTrue: []].

```

```

232
233     self readFraction.
234     self readExponent.
235     ^ JsonNumber new: self endCapture
236 )
237
238 readFraction = (
239     (self readChar: '.') ifFalse: [ ^ false ].
240
241     self readDigit ifFalse: [ self expected: 'digit' ].
242
243     [self readDigit] whileTrue: [].
244
245     ^ true
246 )
247
248 readExponent = (
249     ((self readChar: 'e') not and: [
250         (self readChar: 'E') not]) ifTrue: [ ^ false ].
251
252     (self readChar: '+') ifFalse: [ self readChar: '-' ].
253
254     self readDigit ifFalse: [ self expected: 'digit' ].
255
256     [self readDigit] whileTrue: [].
257
258     ^ true
259 )
260
261 readChar: ch = (
262     current = ch ifFalse: [ ^ false ].
263     self read.
264     ^ true
265 )
266
267 readDigit = (
268     self isDigit ifFalse: [ ^ false ].
269     self read.
270     ^ true
271 )
272
273 skipWhiteSpace = (
274     [ self isWhiteSpace ]
275     whileTrue:
276         [ self read ].
277 )
278
279 read = (
280     current = '\n' ifTrue: [
281         line := line + 1.
282         column := 0.
283     ].
284
285     index := index + 1.
286     column := column + 1.
287
288     input ifNil: [ self error:'input nil' ].
289     index <= input length
290         ifTrue: [ current := input charAt: index ]
291         ifFalse: [ current := nil ]
292 )

```

```

293
294 startCapture = (
295     captureStart := index
296 )
297
298 pauseCapture = (
299     captureBuffer := captureBuffer concatenate: (
300         input substringFrom: captureStart to: index - 1).
301     captureStart := -1
302 )
303
304 endCapture = (
305     | captured |
306     '' = captureBuffer
307     ifTrue: [ captured := input substringFrom: captureStart to: index
- 1 ]
308     ifFalse: [
309         self pauseCapture.
310         captured := captureBuffer.
311         captureBuffer := '' ].
312     captureStart := -1.
313
314     ^ captured
315 )
316
317 expected: expected = (
318     self isEndOfText ifTrue: [
319         self error: 'Unexpected end of input, expected ' + expected asString
320     ].
321     self error: 'Expected ' + expected
322 )
323
324 error: message = (
325     "('error:' + message + ':' + line + ':' + column) println."
326     exceptionBlock value: (ParseException with: message at: index
327                             line: line      column: column )
328 )
329
330 isWhiteSpace = (
331     current = ' ' ifTrue: [^ true].
332     current = '\t' ifTrue: [^ true].
333     current = '\n' ifTrue: [^ true].
334     current = '\r' ifTrue: [^ true].
335     ^ false
336 )
337
338 isDigit = (
339     current = '0' ifTrue: [^ true].
340     current = '1' ifTrue: [^ true].
341     current = '2' ifTrue: [^ true].
342     current = '3' ifTrue: [^ true].
343     current = '4' ifTrue: [^ true].
344     current = '5' ifTrue: [^ true].
345     current = '6' ifTrue: [^ true].
346     current = '7' ifTrue: [^ true].
347     current = '8' ifTrue: [^ true].
348     current = '9' ifTrue: [^ true].
349     ^ false
350 )
351
352 isEndOfText = (

```

```
353     ^ current isNil
354   )
355
356   ----
357
358   with: aJsonString = (
359     ^ self new initializeWith: aJsonString
360   )
361 )
362
```

```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 JsonString = JsonValue (  
27   | string |  
28  
29   initializeWith: str = ( string := str )  
30  
31   isString = (  
32     ^ true  
33   )  
34  
35   ----  
36  
37   new: str = ( ^ self new initializeWith: str )  
38 )  
39
```

```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 JsonValue = (  
27  
28     isObject  = ( ^ false )  
29     isArray   = ( ^ false )  
30     isNumber  = ( ^ false )  
31     isString  = ( ^ false )  
32     isBoolean = ( ^ false )  
33  
34     isTrue    = ( ^ false )  
35     isFalse   = ( ^ false )  
36     isNull    = ( ^ false )  
37  
38     asObject = (  
39         self error: 'Unsupported operation, not an object: ' + self asString  
40     )  
41  
42     asArray = (  
43         self error: 'Unsupported operation, not an array: ' + self asString  
44     )  
45 )  
46
```



```
1 "  
2 This benchmark is based on the minimal-json Java library maintained at:  
3 https://github.com/ralfstx/minimal-json  
4  
5 Original copyright information:  
6  
7 Copyright (c) 2013, 2014 EclipseSource  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
all  
17 copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE  
25 SOFTWARE."  
26 ParseException = (  
27   | offset line column msg |  
28  
29   initializeWith: message at: anOffset line: aLine column: aColumn = (  
30     msg      := message.  
31     offset   := anOffset.  
32     line     := aLine.  
33     column   := aColumn.  
34   )  
35  
36   message = ( ^ msg )  
37  
38   offset = ( ^ offset )  
39   line   = ( ^ line   )  
40   column = ( ^ column )  
41  
42   asString = ( ^ msg + ':' + line + ':' + column )  
43  
44   ----  
45  
46   with: aMessageString at: offset line: line column: column = (  
47     ^ self new initializeWith: aMessageString  
48       at: offset  
49       line: line  
50       column: column  
51   )  
52 )
```



```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 List = Benchmark (  
23  
24   benchmark = ( | result |  
25     result := self  
26       tailWithX: (self makeList: 15)  
27       withY:      (self makeList: 10)  
28       withZ:      (self makeList: 6).  
29   ^ result length  
30 )  
31  
32   verifyResult: result = (  
33     ^ 10 = result  
34 )  
35  
36   makeList: length = (  
37     (length = 0)  
38     ifTrue: [ ^ nil ]  
39     ifFalse: [  
40       | e |  
41       e := ListElement new: length.  
42       e next: (self makeList: (length - 1)).  
43       ^ e ]  
44 )  
45  
46   isShorter: x than: y = (  
47     | xTail yTail |  
48  
49     xTail := x. yTail := y.  
50     [ yTail isNil ]  
51     whileFalse: [  
52       xTail isNil ifTrue: [ ^ true ].  
53       xTail := xTail next.  
54       yTail := yTail next ]
```

```
55
56     ^ false
57 )
58
59 tailWithX: x withY: y withZ: z = (
60     (self isShorter: y than: x)
61     ifTrue: [
62         ^ (self
63             tailWithX: (self tailWithX: x next withY: y withZ: z)
64             withY: (self tailWithX: y next withY: z withZ: x)
65             withZ: (self tailWithX: z next withY: x withZ: y)) ]
66     ifFalse: [ ^ z ].
67 )
68 )
69
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 ListElement = (  
23   | val next |  
24  
25   length = ( next isNil ifTrue: [ ^ 1 ] ifFalse: [ ^ (1 + next length) ] )  
26  
27   val      = ( ^ val )  
28   val: n = ( val := n )  
29  
30   next      = ( ^ next )  
31   next: element = ( next := element )  
32  
33   initialize: n = (  
34     val := n.  
35     next := nil.  
36   )  
37  
38   -----  
39  
40   new: n = ( ^ self new initialize: n )  
41 )  
42  
43
```

Examples/AreWeFastYet/Mandelbrot.som

```
1 " This version is a transcription of the Ruby implementation mandelbrot.rb
2 found with JRuby
3 https://raw.githubusercontent.com/jruby/
jruby/3e43676ee6dc3c13e70fe4a52cce685128c23b8e/bench/truffle/mandelbrot.rb
4
5 Since then it has been modified in a number of ways by Stefan Marr.
6
7 The original copyright statement reads as follows:
8
9 # Copyright © 2004-2013 Brent Fulgham
10 #
11 # All rights reserved.
12 #
13 # Redistribution and use in source and binary forms, with or without
14 # modification, are permitted provided that the following conditions are
met:
15 #
16 # * Redistributions of source code must retain the above copyright
notice,
17 # this list of conditions and the following disclaimer.
18 #
19 # * Redistributions in binary form must reproduce the above copyright
notice,
20 # this list of conditions and the following disclaimer in the
documentation
21 # and/or other materials provided with the distribution.
22 #
23 # * Neither the name of 'The Computer Language Benchmarks Game' nor the
name
24 # of 'The Computer Language Shootout Benchmarks' nor the names of its
25 # contributors may be used to endorse or promote products derived
from this
26 # software without specific prior written permission.
27 #
28 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS
IS'
29 # AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
30 # IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE
31 # DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE
32 # FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
33 # DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR
34 # SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER
35 # CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY,
36 # OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE
37 # OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
38 #
39 # The Computer Language Benchmarks Game
40 # http://benchmarksgame.alioth.debian.org
41 #
42 # contributed by Karl von Laudermann
```

```

43 # modified by Jeremy Echols
44 # modified by Detlef Reichl
45 # modified by Joseph LaFata
46 # modified by Peter Zotov
47 #
48 # http://benchmarksgame.alioth.debian.org/u64q/program.php?
test=mandelbrot&lang=yarv&id=3
49 "
50 Mandelbrot = Benchmark (
51
52     innerBenchmarkLoop: innerIterations = (
53         ^ self verify: (self mandelbrot: innerIterations) inner:
innerIterations
54     )
55
56     verify: result inner: innerIterations = (
57         innerIterations = 500 ifTrue: [ ^ result = 191 ].
58         innerIterations = 750 ifTrue: [ ^ result = 50 ].
59         innerIterations = 1    ifTrue: [ ^ result = 128 ].
60
61         ('No verification result for ' + innerIterations + ' found') println.
62         ('Result is: ' + result asString) println.
63         ^ false
64     )
65
66     mandelbrot: size = (
67         | sum byteAcc bitNum y |
68         sum      := 0.
69         byteAcc  := 0.
70         bitNum   := 0.
71
72         y := 0.
73
74         [y < size] whileTrue: [
75             | ci x |
76             ci := (2.0 * y // size) - 1.0.
77             x  := 0.
78
79             [x < size] whileTrue: [
80                 | zr zr zi zi zi cr escape z notDone |
81                 zr zr := 0.0.
82                 zi zi := 0.0.
83                 cr   := (2.0 * x // size) - 1.5.
84
85                 z := 0.
86                 notDone := true.
87                 escape := 0.
88                 [notDone and: [z < 50]] whileTrue: [
89                     | zr |
90                     zr := zr zr - zi zi + cr.
91                     zi := 2.0 * zr * zi + ci.
92
93                     "preserve recalculation"
94                     zr zr := zr * zr.
95                     zi zi := zi * zi.
96
97                     (zr zr + zi zi > 4.0) ifTrue: [
98                         notDone := false.
99                         escape  := 1.
100                 ].
101                 z := z + 1.

```

```

102         ].
103
104         byteAcc := (byteAcc << 1) + escape.
105         bitNum  := bitNum + 1.
106
107         " Code is very similar for these cases, but using separate
blocks
108         ensures we skip the shifting when it's unnecessary,
109         which is most cases. "
110         bitNum = 8
111         ifTrue: [
112             sum := sum bitXor: byteAcc.
113             byteAcc := 0.
114             bitNum  := 0. ]
115         ifFalse: [
116             (x = (size - 1)) ifTrue: [
117                 byteAcc := byteAcc << (8 - bitNum).
118                 sum := sum bitXor: byteAcc.
119                 byteAcc := 0.
120                 bitNum  := 0. ]].
121         x := x + 1.
122     ].
123     y := y + 1.
124 ].
125
126 ^ sum
127 )
128 )
129

```



```

1 " The Computer Language Benchmarks Game
2 http://benchmarksgame.alioth.debian.org/
3
4 contributed by Mark C. Lewis
5 modified slightly by Chad Whipkey
6
7 Based on nbody.java ported to SOM by Stefan Marr.
8 See LICENSE.md file.
9 "
10 Body = (
11   | x y z vx vy vz mass |
12
13   initX: anX y: aY z: aZ vx: aVX vy: aVY vz: aVZ mass: aMass = (
14     x := anX.
15     y := aY.
16     z := aZ.
17     vx := aVX * Body DaysPerYear.
18     vy := aVY * Body DaysPerYear.
19     vz := aVZ * Body DaysPerYear.
20     mass := aMass * Body SolarMass.
21   )
22
23   x = ( ^ x )
24   y = ( ^ y )
25   z = ( ^ z )
26
27   vx = ( ^ vx )
28   vy = ( ^ vy )
29   vz = ( ^ vz )
30
31   mass = ( ^ mass )
32
33   x: val = ( x := val )
34   y: val = ( y := val )
35   z: val = ( z := val )
36
37   vx: val = ( vx := val )
38   vy: val = ( vy := val )
39   vz: val = ( vz := val )
40
41   mass: val = ( mass := val )
42
43   offsetMomentumX: px y: py z: pz = (
44     vx := 0.0 - (px // Body SolarMass).
45     vy := 0.0 - (py // Body SolarMass).
46     vz := 0.0 - (pz // Body SolarMass).
47   )
48
49   print = (
50     'x: ' print. x println.
51     'y: ' print. y println.
52     'z: ' print. z println.
53
54     'vx: ' print. vx println.
55     'vy: ' print. vy println.
56     'vz: ' print. vz println.
57
58     'mass: ' print. mass println.

```

```

59 )
60
61 ----
62
63 | solarMass |
64
65 Pi          = ( ^ 3.141592653589793 )
66 SolarMass   = ( ^ solarMass )
67 DaysPerYear = ( ^ 365.24 )
68
69 initialize = (
70     solarMass := 4 * self Pi * self Pi.
71 )
72
73 jupiter = (
74     ^ super new
75     initX:    4.8414314424647209
76     y:        -1.16032004402742839
77     z:        -0.103622044471123109
78     vx:       0.00166007664274403694
79     vy:       0.00769901118419740425
80     vz:       -0.0000690460016972063023
81     mass:     0.000954791938424326609
82 )
83
84 saturn = (
85     ^ super new
86     initX:    8.34336671824457987
87     y:        4.12479856412430479
88     z:        -0.403523417114321381
89     vx:       -0.00276742510726862411
90     vy:       0.00499852801234917238
91     vz:       0.0000230417297573763929
92     mass:     0.000285885980666130812
93 )
94
95 uranus = (
96     ^ super new
97     initX:    12.894369562139131
98     y:        -15.1111514016986312
99     z:        -0.223307578892655734
100     vx:      0.00296460137564761618
101     vy:      0.0023784717395948095
102     vz:      -0.0000296589568540237556
103     mass:    0.0000436624404335156298
104 )
105
106 neptune = (
107     ^ super new
108     initX:    15.3796971148509165
109     y:        -25.9193146099879641
110     z:        0.179258772950371181
111     vx:       0.00268067772490389322
112     vy:       0.00162824170038242295
113     vz:       -0.000095159225451971587
114     mass:     0.0000515138902046611451
115 )
116
117 sun = (
118     ^ super new initX: 0.0 y: 0.0 z: 0.0 vx: 0.0 vy: 0.0 vz: 0.0 mass: 1.0
119 )

```

120)

```
1 " The Computer Language Benchmarks Game
2 http://shootout.alioth.debian.org/
3
4 contributed by Mark C. Lewis
5 modified slightly by Chad Whipkey
6
7 Based on nbody.java ported to SOM by Stefan Marr.
8 See LICENSE.md file.
9 "
10 NBody = Benchmark (
11   innerBenchmarkLoop: innerIterations = (
12     | system |
13     system := NBodySystem new.
14
15     1 to: innerIterations do: [:i |
16       system advance: 0.01.
17     ].
18
19     ^ self verify: system energy for: innerIterations.
20   )
21
22   verify: result for: innerIterations = (
23     innerIterations = 250000 ifTrue: [ ^ result = -0.1690859889909308 ].
24     innerIterations =      1 ifTrue: [ ^ result = -0.16907495402506745 ].
25
26     ('No verification result for ' + innerIterations asString + ' found')
27     println.
28     ('Result is: ' + result asString) println.
29     ^ false
30   )
31   ----
32
33   new = (
34     Body initialize.
35     ^ super new
36   )
37 )
38
```

```

1 " The Computer Language Benchmarks Game
2 http://shootout.alioth.debian.org/
3
4 contributed by Mark C. Lewis
5 modified slightly by Chad Whipkey
6
7 Based on nbody.java ported to SOM by Stefan Marr.
8 See LICENSE.md file.
9 "
10 NBodySystem = (
11   | bodies |
12
13   initialize = (
14     bodies := self createBodies
15   )
16
17   createBodies = (
18     | px py pz bodies |
19
20     bodies := Array new: 5.
21     bodies at: 1 put: Body sun.
22     bodies at: 2 put: Body jupiter.
23     bodies at: 3 put: Body saturn.
24     bodies at: 4 put: Body uranus.
25     bodies at: 5 put: Body neptune.
26
27     "bodies do: [:b | b print. '' println ]."
28
29     px := 0.0. py := 0.0. pz := 0.0.
30
31     bodies do: [:b |
32       px := px + (b vx * b mass).
33       py := py + (b vy * b mass).
34       pz := pz + (b vz * b mass).
35     ].
36
37     (bodies at: 1) offsetMomentumX: px y: py z: pz.
38
39     "bodies do: [:b | b print. '' println ]."
40     ^ bodies
41   )
42
43   advance: dt = (
44     1 to: bodies length do: [:i |
45       | iBody |
46       iBody := bodies at: i.
47
48       i + 1 to: bodies length do: [:j |
49         | dx dy dz jBody dSquared distance mag |
50         jBody := bodies at: j.
51         dx := iBody x - jBody x.
52         dy := iBody y - jBody y.
53         dz := iBody z - jBody z.
54
55         dSquared := (dx * dx) + (dy * dy) + (dz * dz).
56         distance := dSquared sqrt.
57         mag := dt // (dSquared * distance).
58

```

```

59         iBody vx: iBody vx - (dx * jBody mass * mag).
60         iBody vy: iBody vy - (dy * jBody mass * mag).
61         iBody vz: iBody vz - (dz * jBody mass * mag).
62
63         jBody vx: jBody vx + (dx * iBody mass * mag).
64         jBody vy: jBody vy + (dy * iBody mass * mag).
65         jBody vz: jBody vz + (dz * iBody mass * mag).
66     ].
67 ].
68
69     bodies do: [:body |
70         body x: body x + (dt * body vx).
71         body y: body y + (dt * body vy).
72         body z: body z + (dt * body vz).
73     ].
74 )
75
76 energy = (
77     | e |
78     e := 0.0.
79
80     1 to: bodies length do: [:i |
81         | iBody |
82         iBody := bodies at: i.
83
84         e := e + (0.5 * iBody mass *
85             ((iBody vx * iBody vx) +
86             (iBody vy * iBody vy) +
87             (iBody vz * iBody vz))).
88
89         i + 1 to: bodies length do: [:j |
90             | jBody dx dy dz distance |
91             jBody := bodies at: j.
92
93             dx := iBody x - jBody x.
94             dy := iBody y - jBody y.
95             dz := iBody z - jBody z.
96
97             distance := ((dx*dx) + (dy*dy) + (dz*dz)) sqrt.
98             e := e - ((iBody mass * jBody mass) // distance).
99         ].
100     ].
101     ^ e
102 )
103
104 ----
105
106 new = (
107     ^ super new initialize
108 )
109 )
110

```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Permute = Benchmark (  
23  
24   | count v |  
25  
26   benchmark = (  
27     count := 0.  
28     v      := Array new: 6 withAll: 0.  
29     self permute: 6.  
30     ^ count  
31   )  
32  
33   verifyResult: result = (  
34     ^ 8660 = result  
35   )  
36  
37   permute: n = (  
38     count := count + 1.  
39     (n <> 0)  
40     ifTrue: [  
41       self permute: n - 1.  
42       n downTo: 1 do: [ :i |  
43         self swap: n with: i.  
44         self permute: n - 1.  
45         self swap: n with: i ] ]  
46   )  
47  
48   swap: i with: j = (  
49     | tmp |  
50     tmp := v at: i.  
51     v at: i put: (v at: j).  
52     v at: j put: tmp  
53   )  
54 )
```



```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Queens = Benchmark (  
23  
24   | freeMaxs freeRows freeMins queenRows |  
25  
26   benchmark = (  
27     | result |  
28     result := true.  
29     1 to: 10 do: [ :i | result := result and: self queens ].  
30     ^ result  
31   )  
32  
33   verifyResult: result = (  
34     ^ result  
35   )  
36  
37   queens = (  
38     freeRows := Array new: 8 withAll: true.  
39     freeMaxs := Array new: 16 withAll: true.  
40     freeMins := Array new: 16 withAll: true.  
41     queenRows := Array new: 8 withAll: -1.  
42     ^ self placeQueen: 1  
43   )  
44  
45   placeQueen: c = (  
46     1 to: 8 do: [ :r |  
47       (self row: r column: c)  
48         ifTrue: [  
49           queenRows at: r put: c.  
50           self row: r column: c put: false.  
51           (c = 8) ifTrue: [ ^ true ].  
52           (self placeQueen: c + 1) ifTrue: [ ^ true ].  
53           self row: r column: c put: true ] ].  
54     ^ false
```

```
55 )
56
57 row: r column: c = (
58   ^ (freeRows at: r) && (freeMaxs at: c + r) && (freeMins at: c - r + 8)
59 )
60
61 row: r column: c put: v = (
62   freeRows at: r      put: v.
63   freeMaxs at: c + r   put: v.
64   freeMins at: c - r + 8 put: v.
65 )
66 )
67
```

```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java_benchmarking/index.html  
7 "  
8 DeviceTaskDataRecord = RObject (  
9   | pending |  
10  
11   pending = ( ^ pending )  
12   pending: packet = ( pending := packet )  
13  
14   create = ( pending := RObject NoWork )  
15   ----  
16   create = (  
17     ^ super new create  
18   )  
19 )  
20
```

```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java\_benchmarking/index.html  
7 "  
8 HandlerTaskDataRecord = RObject (  
9   | workIn deviceIn |  
10  
11   deviceIn = ( ^ deviceIn )  
12  
13   deviceIn: aPacket = ( deviceIn := aPacket )  
14  
15   deviceInAdd: packet = (  
16     deviceIn := self append: packet head: deviceIn  
17   )  
18  
19   workIn = ( ^ workIn )  
20  
21   workIn: aWorkQueue = ( workIn := aWorkQueue )  
22  
23   workInAdd: packet = (  
24     workIn := self append: packet head: workIn  
25   )  
26  
27   create = (  
28     workIn := deviceIn := RObject NoWork  
29   )  
30   asString = (  
31     ^ 'HandlerTaskDataRecord(' + workIn asString + ', ' + deviceIn  
asString + ')' )  
32   )  
33   ----  
34   create = ( ^ super new create )  
35 )  
36
```

```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java_benchmarking/index.html  
7 "  
8 IdleTaskDataRecord = RObject (  
9   | control count |  
10  
11   control = ( ^ control )  
12  
13   control: aNumber = (  
14     control := aNumber  
15   )  
16  
17   count = ( ^ count )  
18  
19   count: aCount = (  
20     count := aCount  
21   )  
22  
23   create = (  
24     control := 1.  
25     count := 10000  
26   )  
27  
28   ----  
29  
30   create = ( ^ super new create )  
31 )  
32
```

```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java_benchmarking/index.html  
7 "  
8 Packet = RObject (  
9   | link identity kind datum data |  
10  
11   data = ( ^ data )  
12  
13   datum = ( ^ datum )  
14   datum: someData = ( datum := someData )  
15  
16   identity = ( ^ identity )  
17   identity: anIdentity = ( identity := anIdentity )  
18  
19   kind = ( ^ kind )  
20   link = ( ^ link )  
21  
22   link: aWorkQueue = ( link := aWorkQueue )  
23   link: aLink identity: anIdentity kind: aKind = (  
24     link := aLink.  
25     kind := aKind.  
26     identity := anIdentity.  
27     datum := 1.  
28     data := Array new: 4 withAll: 0  
29   )  
30  
31   asString = (  
32     ^ 'Packet(' +  
33       link asString + ', ' +  
34       identity asString + ', ' +  
35       kind asString + ', ' +  
36       datum asString + ', ' +  
37       data asString +  
38       ' )'  
39   )  
40  
41   ----  
42  
43   create: link identity: identity kind: kind = (  
44     ^ super new  
45       link: link  
46       identity: identity  
47       kind: kind  
48   )  
49 )  
50
```

```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java_benchmarking/index.html  
7 "  
8 RBObject = Object (  
9  
10  append: packet head: queueHead = (  
11    | mouse link |  
12    packet link: RBObject NoWork.  
13    RBObject NoWork == queueHead ifTrue: [ ^ packet ].  
14    mouse := queueHead.  
15    [RBObject NoWork == (link := mouse link)]  
16      whileFalse: [mouse := link].  
17    mouse link: packet.  
18    ^ queueHead  
19  )  
20  
21  ----  
22  
23  NoTask   = ( ^ nil )  
24  Idler    = ( ^ 1   )  
25  NoWork   = ( ^ nil )  
26  Worker   = ( ^ 2   )  
27  WorkPacketKind = ( ^ 2 )  
28  HandlerA = ( ^ 3   )  
29  HandlerB = ( ^ 4   )  
30  DeviceA  = ( ^ 5   )  
31  DeviceB  = ( ^ 6   )  
32  DevicePacketKind = ( ^ 1 )  
33  
34 )  
35
```

```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java_benchmarking/index.html  
7 "  
8 Richards = Benchmark (  
9   benchmark = (  
10     ^ Scheduler new start.  
11   )  
12  
13   verifyResult: result = (  
14     ^ result  
15   )  
16 )  
17
```



```

1 "
2 This benchmark is derived from richards.st, which is
3 part of Mario Wolczko's DeltaBlue and Richards collection.
4
5 License details:
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 Scheduler = RObject (
9   |taskList currentTask currentTaskIdentity taskTable tracing layout
queuePacketCount holdCount|
10
11   initialize = (
12     taskList      := RObject NoTask.
13     currentTask   := RObject NoTask.
14     currentTaskIdentity := 0.
15     taskTable     := Array new: 6 withAll: RObject NoTask.
16     layout        := 0.
17     queuePacketCount := 0.
18     holdCount     := 0.
19   )
20
21   tracing = (
22     ^ false
23   )
24
25   createDevice: identity priority: priority work: work state: state = (
26     | data |
27     data := DeviceTaskDataRecord create.
28     self
29       createTask: identity
30       priority: priority
31       work: work
32       state: state
33       function:
34         [:work :word |
35           | data functionWork |
36           data := word.
37           functionWork := work.
38           RObject NoWork == functionWork ifTrue: [
39             RObject NoWork == (functionWork := data pending)
40             ifTrue: [ self wait ]
41             ifFalse: [
42               data pending: RObject NoWork.
43               self queuePacket: functionWork]]
44             ifFalse: [
45               data pending: functionWork.
46               self tracing ifTrue: [
47                 self trace: functionWork datum].
48               self holdSelf]]
49     data: data
50   )
51
52   createHandler: identity priority: priority work: work state: state = (
53     | data |
54     data := HandlerTaskDataRecord create.
55     self createTask: identity
56       priority: priority

```

```

57         work:          work
58         state:         state
59         function:      [:work :word |
60             | data workPacket |
61             data := word.
62             RObject NoWork == work ifFalse: [
63                 RObject WorkPacketKind == work kind
64                 ifTrue: [ data workInAdd: work ]
65                 ifFalse: [ data deviceInAdd: work ]].
66
67             RObject NoWork == (workPacket := data workIn)
68             ifTrue: [ self wait ]
69             ifFalse: [
70                 | count |
71                 count := workPacket datum.
72                 count > 4
73                 ifTrue: [
74                     data workIn: workPacket link.
75                     self queuePacket: workPacket]
76                 ifFalse: [
77                     | devicePacket |
78                     RObject NoWork == (devicePacket := data deviceIn)
79                     ifTrue: [ self wait ]
80                     ifFalse: [
81                         data deviceIn: devicePacket link.
82                         devicePacket datum: (workPacket data at: count).
83                         workPacket datum: count + 1.
84                         self queuePacket: devicePacket]]]]
85         data: data
86     )
87
88     createIdler: identity priority: priority work: work state: state = (
89         | data |
90         data := IdleTaskDataRecord create.
91         self createTask: identity
92             priority: priority
93             work: work
94             state: state
95             function: [:work :word |
96                 | data |
97                 data := word.
98                 data count: data count - 1.
99                 0 = data count
100                 ifTrue: [ self holdSelf ]
101                 ifFalse: [
102                     0 = (data control & 1 )
103                     ifTrue: [
104                         data control: data control / 2.
105                         self release: RObject DeviceA]
106                     ifFalse: [
107                         data control: (data control / 2 bitXor: 53256).
108                         self release: RObject DeviceB]]]
109         data: data
110     )
111
112     createPacket: link identity: identity kind: kind = (
113         ^ Packet create: link
114             identity: identity
115             kind: kind
116     )
117

```

```

118   createTask: identity priority: priority work: work state: state
function: aBlock data: data = (
119   | t |
120   t := TaskControlBlock    link:          taskList
121                           create:         identity
122                           priority:       priority
123                           initialWorkQueue: work
124                           initialState:   state
125                           function:      aBlock
126                           privateData:   data.
127   taskList := t.
128   taskTable at: identity put: t
129 )
130
131   createWorker: identity priority: priority work: work state: state = (
132   | data |
133   data := WorkerTaskDataRecord create.
134   self createTask: identity
135               priority: priority
136               work:      work
137               state:     state
138               function: [:work :word |
139               | data |
140               data := word.
141               RObject NoWork == work
142               ifTrue: [ self wait ]
143               ifFalse: [
144                   data destination: (RObject HandlerA = data destination
145                                   ifTrue: [RObject HandlerB]
146                                   ifFalse: [RObject HandlerA]).
147                   work identity: data destination.
148                   work datum: 1.
149                   1 to: 4 do: [:i |
150                       data count: data count + 1.
151                       data count > 26 ifTrue: [data count: 1].
152                       "work data at: i put: $A asInteger + data count - 1]."
153                       work data at: i put: 65 + data count - 1].
154                   self queuePacket: work]]
155   data: data
156 )
157
158   start = (
159   | workQ |
160   self
161       createIdler: RObject Idler
162       priority: 0
163       work: RObject NoWork
164       state: TaskState running.
165   workQ := self
166               createPacket: RObject NoWork
167               identity: RObject Worker
168               kind: RObject WorkPacketKind.
169   workQ := self
170               createPacket: workQ
171               identity: RObject Worker
172               kind: RObject WorkPacketKind.
173   self
174       createWorker: RObject Worker
175       priority: 1000
176       work: workQ
177       state: TaskState waitingWithPacket.

```

```

178     workQ := self
179         createPacket: RObject NoWork
180         identity: RObject DeviceA
181         kind: RObject DevicePacketKind.
182     workQ := self
183         createPacket: workQ
184         identity: RObject DeviceA
185         kind: RObject DevicePacketKind.
186     workQ := self
187         createPacket: workQ
188         identity: RObject DeviceA
189         kind: RObject DevicePacketKind.
190     self
191         createHandler: RObject HandlerA
192         priority: 2000
193         work: workQ
194         state: TaskState waitingWithPacket.
195     workQ := self
196         createPacket: RObject NoWork
197         identity: RObject DeviceB
198         kind: RObject DevicePacketKind.
199     workQ := self
200         createPacket: workQ
201         identity: RObject DeviceB
202         kind: RObject DevicePacketKind.
203     workQ := self
204         createPacket: workQ
205         identity: RObject DeviceB
206         kind: RObject DevicePacketKind.
207     self
208         createHandler: RObject HandlerB
209         priority: 3000
210         work: workQ
211         state: TaskState waitingWithPacket.
212     self
213         createDevice: RObject DeviceA
214         priority: 4000
215         work: RObject NoWork
216         state: TaskState waiting.
217     self
218         createDevice: RObject DeviceB
219         priority: 5000
220         work: RObject NoWork
221         state: TaskState waiting.
222
223     self schedule.
224
225     ^ queuePacketCount = 23246 and: holdCount = 9297
226 )
227
228 findTask: identity = (
229     | t |
230     t := taskTable at: identity.
231     RObject NoTask == t ifTrue: [self error: 'findTask failed'].
232     ^ t
233 )
234
235 holdSelf = (
236     holdCount := holdCount + 1.
237     currentTask taskHolding: true.
238     ^ currentTask link

```

```

239 )
240
241 queuePacket: packet = (
242     | t |
243     t := self findTask: packet identity.
244     RObject NoTask == t ifTrue: [ ^ RObject NoTask ].
245     queuePacketCount := queuePacketCount + 1.
246     packet link: RObject NoWork.
247     packet identity: currentTaskIdentity.
248     ^ t addInput: packet checkPriority: currentTask
249 )
250
251 release: identity = (
252     | t |
253     t := self findTask: identity.
254     RObject NoTask == t ifTrue: [ ^ RObject NoTask ].
255     t taskHolding: false.
256     t priority > currentTask priority
257         ifTrue: [ ^ t ]
258         ifFalse: [ ^ currentTask ]
259 )
260
261 trace: id = (
262     layout := layout - 1.
263     0 >= layout ifTrue: [
264         '' println.
265         layout := 50 ].
266     id print
267 )
268
269 wait = (
270     currentTask taskWaiting: true.
271     ^currentTask
272 )
273
274 schedule = (
275     currentTask := taskList.
276     [ RObject NoTask == currentTask ] whileFalse: [
277         currentTask isTaskHoldingOrWaiting
278             ifTrue: [ currentTask := currentTask link ]
279             ifFalse: [
280                 currentTaskIdentity := currentTask identity.
281                 self tracing ifTrue: [ self trace: currentTaskIdentity ].
282                 currentTask := currentTask runTask ] ]
283 )
284
285 ----
286
287 new = ( ^ super new initialize )
288 )
289

```

```

1 "
2 This benchmark is derived from richards.st, which is
3 part of Mario Wolczko's DeltaBlue and Richards collection.
4
5 License details:
6 http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
7 "
8 TaskControlBlock = TaskState (
9   | link identity priority input function handle |
10
11   identity = ( ^ identity )
12   link      = ( ^ link )
13   priority  = ( ^ priority )
14
15   link: aLink identity: anIdentity priority: aPriority initialWorkQueue:
anInitialWorkQueue initialState: anInitialState function: aBlock privateData:
aPrivateData = (
16     link      := aLink.
17     identity  := anIdentity.
18     function  := aBlock.
19     priority  := aPriority.
20     input     := anInitialWorkQueue.
21     handle    := aPrivateData.
22     self packetPending: anInitialState isPacketPending.
23     self taskWaiting: anInitialState isTaskWaiting.
24     self taskHolding: anInitialState isTaskHolding.
25   )
26
27   addInput: packet checkPriority: oldTask = (
28     RObject NoWork == input
29     ifTrue: [
30       input := packet.
31       self packetPending: true.
32       priority > oldTask priority ifTrue: [ ^ self ] ]
33     ifFalse: [
34       input := self append: packet head: input ].
35     ^ oldTask
36   )
37
38   runTask = (
39     | message |
40     self isWaitingWithPacket
41     ifTrue: [
42       message := input.
43       input := message link.
44       RObject NoWork == input
45       ifTrue: [self running]
46       ifFalse: [self packetPending]]
47     ifFalse: [message := RObject NoWork].
48     ^ function value: message with: handle
49   )
50
51 ----
52
53   link: link create: identity priority: priority initialWorkQueue:
initialWorkQueue initialState: initialState function: aBlock privateData:
privateData = (

```

```
54         ^super new
55             link: link
56             identity: identity
57             priority: priority
58             initialWorkQueue: initialWorkQueue
59             initialState: initialState
60             function: aBlock
61             privateData: privateData
62     )
63 )
64
```

```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java_benchmarking/index.html  
7 "  
8 TaskState = RObject (  
9   | packetPending taskWaiting taskHolding |  
10  
11   isPacketPending = ( ^ packetPending )  
12  
13   isTaskHolding = ( ^ taskHolding )  
14  
15   isTaskWaiting = ( ^ taskWaiting )  
16  
17   taskHolding: aBoolean  = ( taskHolding  := aBoolean )  
18   taskWaiting: aBoolean  = ( taskWaiting  := aBoolean )  
19   packetPending: aBoolean = ( packetPending := aBoolean )  
20  
21   packetPending = (  
22     packetPending := true.  
23     taskWaiting := false.  
24     taskHolding := false  
25   )  
26  
27   running = (  
28     packetPending := taskWaiting := taskHolding := false  
29   )  
30  
31   waiting = (  
32     packetPending := taskHolding := false.  
33     taskWaiting := true  
34   )  
35  
36   waitingWithPacket = (  
37     taskHolding := false.  
38     taskWaiting := packetPending := true  
39   )  
40  
41   isTaskHoldingOrWaiting = ( ^ taskHolding or: [packetPending not and:  
[taskWaiting]] )  
42  
43   isWaitingWithPacket = ( ^ packetPending and: [taskWaiting and:  
[taskHolding not]] )  
44  
45   ----  
46   running = ( ^ super new running )  
47  
48   waiting = ( ^ super new waiting )  
49  
50   waitingWithPacket = ( ^ super new waitingWithPacket )  
51 )  
52
```



```
1 "  
2 This benchmark is derived from richards.st, which is  
3 part of Mario Wolczko's DeltaBlue and Richards collection.  
4  
5 License details:  
6   http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/  
mario/java_benchmarking/index.html  
7 "  
8 WorkerTaskDataRecord = RObject (  
9   |destination count|  
10  count = ( ^ count )  
11  
12  count: aCount =( count := aCount )  
13  
14  destination = ( ^ destination )  
15  
16  destination: aHandler = ( destination := aHandler )  
17  
18  create = (  
19    destination := RObject HandlerA.  
20    count := 0  
21  )  
22  ----  
23  create = ( ^ super new create )  
24 )  
25
```

```
1 "  
2 Copyright (c) 2011-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Run = (  
23   | total numIterations innerIterations benchmarkSuite name doGC |  
24  
25   initialize: aName = (  
26     name           := aName.  
27     benchmarkSuite := self loadBenchmarkSuite: aName.  
28     total          := 0.  
29     numIterations   := 1.  
30     innerIterations := 1.  
31   )  
32  
33   loadBenchmarkSuite: className = (  
34     | sym cls |  
35     sym := className asSymbol.  
36     cls := system load: sym.  
37     cls ifNil: [  
38       self error: 'Failed loading benchmark: ', className ].  
39     ^ cls  
40   )  
41  
42   name: aString = ( name := aString )  
43   benchmarkSuite: aSuite = ( benchmarkSuite := aSuite )  
44   numIterations: anInt = ( numIterations := anInt )  
45   innerIterations: anInt = ( innerIterations := anInt )  
46   doGC: aBool          = ( doGC := aBool )  
47  
48   runBenchmark = (  
49     ('Starting ' + name + ' benchmark ... ') println.  
50  
51     self doRuns: benchmarkSuite new.  
52     self reportBenchmark.  
53
```

```

54     '' println
55 )
56
57 measure: bench = (
58     | startTime endTime runTime endGcStats startGcStats startCompTime
endCompTime |
59     startGcStats := system gcStats.
60     startCompTime := system totalCompilationTime.
61     startTime := system ticks.
62
63     (bench innerBenchmarkLoop: innerIterations) ifFalse: [
64         self error: 'Benchmark failed with incorrect result'. ].
65
66     endTime := system ticks.
67     endGcStats := system gcStats.
68     endCompTime := system totalCompilationTime.
69
70     runTime := endTime - startTime.
71     self printResult: runTime startGc: startGcStats endGc: endGcStats
compileTime: endCompTime - startCompTime.
72
73     total := total + runTime.
74
75     doGC ifTrue: [
76         system fullGC ]
77 )
78
79 doRuns: bench = (
80     1 to: numIterations do: [:i |
81         self measure: bench
82     ]
83 )
84
85 reportBenchmark = (
86     (name + ': iterations=' + numIterations +
87     ' average: ' + (total / numIterations) + 'us total: ' + total +
88     'us\n') println.
89 )
90
91 printResult: runTime startGc: startGc endGc: endGc compileTime:
compileTime = (
92     name print. ': GC count: ' print. (((endGc) at: 1) - ((startGc)
at: 1)) print. 'n' println.
93     name print. ': GC time: ' print. (((endGc) at: 2) - ((startGc)
at: 2)) print. 'ms' println.
94     name print. ': Allocated: ' print. (((endGc) at: 3) - ((startGc)
at: 3)) print. 'bytes' println.
95     name print. ': Compile time: ' print. compileTime print. 'ms' println.
96
97     (name + ': iterations=1 runtime: ' + runTime + 'us') println
98 )
99
100 printTotal = (
101     ('Total Runtime: ' + total + 'us') println.
102 )
103
104 ----
105
106 new: name = (
107     ^ self new initialize: name
108 )

```

109)
110

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Sieve = Benchmark (  
23  
24   benchmark = (  
25     | flags |  
26     flags := Array new: 5000 withAll: true.  
27     ^ self sieve: flags size: 5000.  
28   )  
29  
30   verifyResult: result = (  
31     ^ 669 = result  
32   )  
33  
34   sieve: flags size: size = (  
35     | primeCount |  
36     primeCount := 0.  
37  
38     2 to: size do: [ :i |  
39       (flags at: i - 1)  
40         ifTrue: [  
41           | k |  
42           primeCount := primeCount + 1.  
43           k := i + i.  
44           [ k <= size ]  
45             whileTrue: [  
46               flags at: k - 1 put: false.  
47               k := k + i ]. ] ].  
48     ^ primeCount  
49   )  
50 )  
51
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 SomRandom = (  
23   | seed |  
24  
25   initialize = ( seed := 74755 )  
26  
27   next = (  
28     seed := ((seed * 1309) + 13849) & 65535.  
29     ^ seed  
30   )  
31  
32   ----  
33  
34   new = ( ^ super new initialize )  
35 )  
36
```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 Storage = Benchmark (  
23  
24   | count |  
25  
26   initialize = (  
27     count := 0.  
28   )  
29  
30   benchmark = (  
31     | random |  
32     random := SomRandom new.  
33     count := 0.  
34     self buildTreeDepth: 7 with: random.  
35     ^ count  
36   )  
37  
38   verifyResult: result = (  
39     ^ 5461 = result  
40   )  
41  
42   buildTreeDepth: depth with: random = (  
43     count := count + 1.  
44     ^ (depth = 1)  
45       ifTrue: [ Array new: random next % 10 + 1 ]  
46       ifFalse: [  
47         Array new: 4 withAll: [ self buildTreeDepth: depth - 1 with:  
random ] ]  
48   )  
49  
50   ----  
51  
52   new = ( ^ super new initialize )  
53 )
```



```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21  
22 Mmm... Hanoi...  
23 "  
24 Towers = Benchmark (  
25  
26   | piles movesdone |  
27  
28   initialize = (  
29     piles      := nil.  
30     movesdone := 0.  
31   )  
32  
33   pushDisk: disk onPile: pile = (  
34     | top |  
35  
36     top := piles at: pile.  
37     top notNil && [ disk size >= top size ]  
38       ifTrue: [ self error: 'Cannot put a big disk on a smaller one' ].  
39  
40     disk next: top.  
41     piles at: pile put: disk.  
42   )  
43  
44   popDiskFrom: pile = (  
45     | top |  
46  
47     top := piles at: pile.  
48     top isNil  
49       ifTrue: [  
50         self error: 'Attempting to remove a disk from an empty pile' ].  
51  
52     piles at: pile put: top next.  
53     top next: nil.  
54     ^ top
```

```

55 )
56
57 moveTopDiskFrom: fromPile to: toPile = (
58     self pushDisk: (self popDiskFrom: fromPile) onPile: toPile.
59     movesdone := movesdone + 1.
60 )
61
62 buildTowerAt: pile disks: disks = (
63     disks downTo: 0 do: [ :i |
64         self pushDisk: (TowersDisk new: i) onPile: pile ]
65 )
66
67 move: disks disksFrom: fromPile to: toPile = (
68     disks = 1
69     ifTrue: [ self moveTopDiskFrom: fromPile to: toPile ]
70     ifFalse: [ | otherPile |
71         otherPile := 6 - fromPile - toPile.
72         self move: disks - 1 disksFrom: fromPile to: otherPile.
73         self moveTopDiskFrom: fromPile to: toPile.
74         self move: disks - 1 disksFrom: otherPile to: toPile. ]
75 )
76
77 benchmark = (
78     piles := Array new: 3.
79     self buildTowerAt: 1 disks: 13.
80     movesdone := 0.
81     self move: 13 disksFrom: 1 to: 2.
82     ^ movesdone
83 )
84
85 verifyResult: result = (
86     ^ 8191 = result
87 )
88
89 ----
90
91 new = ( ^ super new initialize )
92 )
93

```

```
1 "  
2 Copyright (c) 2001-2016 see AUTHORS.md file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 TowersDisk = (  
23  
24   | size next |  
25  
26   initialize: anInt = (  
27     size := anInt  
28   )  
29  
30   size          = ( ^ size          )  
31   next          = ( ^ next          )  
32   next: value = ( next := value )  
33  
34   -----  
35  
36   new: size = ( ^ super new initialize: size )  
37 )  
38
```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 All = BenchmarkHarness (  
24     | summedAverage |  
25  
26     all = (  
27         ^ Fibonacci, Dispatch, Bounce, Loop, Permute, Queens, List,  
Recurse,  
28         Storage, Sieve, BubbleSort, QuickSort, Sum, Towers, TreeSort,  
29         IntegerLoop, FieldLoop  
30     )  
31  
32     run: params = (  
33         params length < 2  
34         ifTrue: [ self exec: 100 ]  
35         ifFalse: [ self exec: (params at: 2) asInteger ]  
36     )  
37  
38     printUsage = (  
39         './som.sh -cp Smalltalk Examples/Benchmarks/All.som [number-of-  
iterations]' println.  
40         '' println.  
41         ' number-of-iterations - the number of time each benchmark is  
executed, default: 1' println.  
42     )  
43  
44     initialize = (  
45         super initialize.  
46         summedAverage := 0.  
47     )  
48  
49     exec: iterations = (  
50         'Start execution of all benchmarks. Iterations: ' print.  
51         iterations println.
```

```

52
53     self all do: [:cls |
54         self initialize.
55         self benchmarkClass: cls.
56         self printAll: false.
57         self maxRuntime: 3. "seconds"
58         self numIterations: iterations.
59         self warmUp: 10.
60
61         self runBenchmark.
62     ].
63     self printTotal.
64 )
65
66     reportBenchmark: bench result: total = (
67         '' println.
68         'Benchmark: ' print.
69         bench name println.
70
71         ('    Iterations: ' + numIterations + ' (elapsed time ' + (total //
1000) round
72         + ' ms')) println.
73         ('    AVERAGE: ' + ((total // numIterations) // 1000) round + '
ms') println.
74
75         summedAverage := summedAverage + (total // numIterations).
76     )
77
78     printTotal = (
79         ('Summed Average Runtime: ' + (summedAverage // 1000) round
asString + ' ms') println.
80     )
81
82 )
83

```

```
1 "  
2  
3 $Id: Ball.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Ball = (  
27  
28   | x y xVel yVel |  
29  
30   bounce = (  
31     | xLimit yLimit bounced |  
32     xLimit := yLimit := 500.  
33     bounced := false.  
34  
35     x := x + xVel.  
36     y := y + yVel.  
37     (x > xLimit)  
38       ifTrue: [ x := xLimit. xVel := 0 - xVel abs. bounced := true ].  
39     (x < 0)  
40       ifTrue: [ x := 0.      xVel := xVel abs.      bounced := true ].  
41     (y > yLimit)  
42       ifTrue: [ y := yLimit. yVel := 0 - yVel abs. bounced := true ].  
43     (y < 0)  
44       ifTrue: [ y := 0.      yVel := yVel abs.      bounced := true ].  
45     ^bounced  
46   )  
47  
48   initialize = (  
49     x := Random next % 500.  
50     y := Random next % 500.  
51     xVel := (Random next % 300) - 150.  
52     yVel := (Random next % 300) - 150.  
53   )  
54
```

```
55  -----
56
57  new = ( ^super new initialize )
58
59 )
60
```

```
1 "  
2  
3  
4 Copyright (c) 2001-2013 see AUTHORS file  
5  
6 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
7 of this software and associated documentation files (the 'Software'), to  
deal  
8 in the Software without restriction, including without limitation the  
rights  
9 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
10 copies of the Software, and to permit persons to whom the Software is  
11 furnished to do so, subject to the following conditions:  
12  
13 The above copyright notice and this permission notice shall be included in  
14 all copies or substantial portions of the Software.  
15  
16 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
17 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
18 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
19 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
20 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
21 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
22 THE SOFTWARE.  
23 "  
24  
25 Benchmark = (  
26  
27   run = (  
28     | harness |  
29     harness := All new.  
30     harness initialize.  
31     harness benchmarkClass: self class.  
32     harness printAll: false.  
33     harness numIterations: 100.  
34  
35     harness runBenchmark.  
36     harness printTotal.  
37   )  
38  
39   oneTimeSetup = ()  
40  
41   innerBenchmarkLoop: innerIterations = (  
42     | i |  
43     i := 0.  
44     [ i < innerIterations ] whileTrue: [  
45       (self verifyResult: self benchmark) ifFalse: [ ^ false ].  
46       i := i + 1.  
47     ].  
48     ^ true  
49   )  
50  
51   benchmark = ( self subclassResponsibility )  
52   verifyResult: result = ( self subclassResponsibility )  
53   name = ( ^self class name asString )  
54
```



```
55     assert: expected equals: value = (  
56         expected = value ifFalse: [  
57             self error: 'Expected value (' + expected asString + ') differs  
from actual (' + value asString + ') benchmark result.'  
58         ].  
59         ^ true  
60     )  
61 )  
62
```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 BenchmarkHarness = (  
24  
25     "The BenchmarkHarness can be invoked on the command line and should be  
26     passed a list of benchmarks to run (class names) as arguments. It will  
27     run each of them and output single as well as a total."  
28  
29     | total benchmarkClass numIterations innerIterations printAll doGC |  
30  
31     benchmarkClass: class = ( benchmarkClass := class )  
32     printAll: aBool = ( printAll := aBool )  
33  
34     numIterations: anInt = (numIterations := anInt)  
35  
36     total = ( ^ total )  
37  
38     run: args = (  
39         args length < 2 ifTrue: [ ^ self printUsage ].  
40  
41         self initialize.  
42         self processArguments: args.  
43  
44         self runBenchmark.  
45         self printTotal.  
46     )  
47  
48     initialize = (  
49         total           := 0.  
50         numIterations   := 1.  
51         innerIterations := 1.  
52         printAll        := true.  
53         doGC            := false.
```

```

54     )
55
56     printUsage = (
57         './som.sh -cp Smalltalk Examples/Benchmarks/BenchmarkHarness.som [--
gc-between-iterations|-gc] benchmark [num-iterations [inner-iter]]' println.
58         '' println.
59         '  --gc-between-iterations | --gc - trigger a full GC between
benchmark iterations' println.
60         '  benchmark          - benchmark class name (e.g., Queens, Fibonacci,
Dispatch)' println.
61         '  num-iterations - number of times to execute benchmark, default:
1' println.
62         '  inner-iter      - number of times the benchmark is executed in an
inner loop, ' println.
63         '                                which is measured in total, default: 1' println.
64     )
65
66     processArguments: args = (
67         | v arg |
68         v := Vector new: args length.
69         v appendAll: args.
70
71         "First argument is the BenchmarkHarness"
72         v removeFirst.
73
74         arg := v removeFirst.
75         (arg = '--gc-between-iterations' or: [arg = '--gc'])
76         ifTrue: [
77             doGC := true.
78             arg := v removeFirst ].
79
80         self loadBenchmarkClass: arg.
81         v size > 0 ifTrue: [
82             arg := v removeFirst.
83             numIterations := arg asInteger.
84             v size > 0 ifTrue: [
85                 arg := v removeFirst.
86                 innerIterations := arg asInteger ] ]
87     )
88
89     loadBenchmarkClass: className = (
90         | sym cls |
91         sym := className asSymbol.
92         cls := system load: sym.
93         cls ifNil: [
94             self error: 'Failed loading benchmark: ' + className ].
95         benchmarkClass := cls.
96     )
97
98     runBenchmark = (
99         | bench result |
100        bench := benchmarkClass new.
101        bench oneTimeSetup.
102
103        ('Starting ' + bench name + ' benchmark.') println.
104        result := self doRuns: bench.
105        total := total + result.
106        self reportBenchmark: bench result: result.
107
108        '' println
109    )

```

```

110
111     doRuns: bench = (
112         | i total |
113         i := 0.
114         total := 0.
115
116         [ i < numIterations ] whileTrue: [
117             | startTime endTime runTime endGcStats startGcStats
startCompTime endCompTime |
118             startGcStats := system gcStats.
119             startCompTime := system totalCompilationTime.
120             startTime := system ticks.
121
122             (bench innerBenchmarkLoop: innerIterations) ifFalse: [
123                 self error: 'Benchmark failed with incorrect result'. ].
124
125             endTime := system ticks.
126             endGcStats := system gcStats.
127             endCompTime := system totalCompilationTime.
128
129             runTime := endTime - startTime.
130             printAll ifTrue: [
131                 self print: bench run: runTime startGc: startGcStats endGc:
endGcStats compileTime: endCompTime - startCompTime ].
132
133             total := total + runTime.
134             i := i + 1.
135
136             doGC ifTrue: [
137                 system fullGC ] ].
138
139         ^ total
140     )
141
142     reportBenchmark: bench result: result = (
143         bench name print.
144         ': iterations=' print.
145         numIterations print.
146         ' average: ' print.
147         (result / numIterations) print.
148         'us' print.
149         ' total: ' print.
150         result print.
151         'us' println.
152     )
153
154     print: bench run: runTime startGc: startGc endGc: endGc compileTime:
compileTime = (
155         bench name print. ': GC count:      ' print. (((endGc) at: 1) -
((startGc) at: 1)) print. 'n' println.
156         bench name print. ': GC time:      ' print. (((endGc) at: 2) -
((startGc) at: 2)) print. 'ms' println.
157         bench name print. ': Allocated:    ' print. (((endGc) at: 3) -
((startGc) at: 3)) print. 'bytes' println.
158         bench name print. ': Compile time: ' print. compileTime print. 'ms'
println.
159
160         bench name print. ': iterations=1 runtime: ' print. runTime
print. 'us' println
161     )
162

```

```
163     printTotal = (  
164         ('Total Runtime: ' + total asString + 'us') println.  
165     )  
166 )  
167
```

```
1 "  
2  
3 $Id: Bounce.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Bounce = Benchmark (  
27  
28   benchmark = (  
29     | ballCount balls bounces |  
30  
31     Random initialize.  
32  
33     ballCount := 100.  
34     bounces   := 0.  
35     balls     := Array new: ballCount withAll: [ Ball new ].  
36  
37     1 to: 50 do: [ :i |  
38       balls do: [ :ball |  
39         (ball bounce) ifTrue: [ bounces := bounces + 1 ] ] ].  
40  
41     ^ bounces  
42   )  
43  
44   verifyResult: result = (  
45     ^ self assert: 1331 equals: result  
46   )  
47  
48 )  
49
```

```
1 "  
2  
3 $Id: BubbleSort.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 BubbleSort = Sort (  
27  
28   sort: array = (  
29     array length downTo: 1 do: [ :i |  
30       1 to: i - 1 do: [ :j |  
31         | current next |  
32         current := array at: j.  
33         next     := array at: j + 1.  
34         (current > next)  
35         ifTrue: [  
36           array at: j put: next.  
37           array at: j + 1 put: current ] ] ].  
38   ^ array  
39 )  
40  
41   dataSize = ( ^130 )  
42  
43 )  
44
```

```

1 AbstractConstraint = (
2   "I am an abstract class representing a system-maintainable
relationship (or
3   'constraint') between a set of variables. I supply a strength instance
4   variable; concrete subclasses provide a means of storing the
constrained
5   variables and other information required to represent a constraint.
6
7   Instance variables:
8       strength          the strength of this constraint <Strength>"
9   | strength |
10
11   "accessing"
12
13   strength = (
14       "Answer my strength."
15       ^strength
16   )
17
18   strength: strengthSymbol = (
19       "Set my strength."
20       strength := Strength of: strengthSymbol.
21   )
22
23   "queries"
24   isInput = (
25       "Normal constraints are not input constraints. An input
constraint is
26       one that depends on external state, such as the mouse, the
keyboard,
27       a clock, or some arbitrary piece of imperative code."
28       ^false
29   )
30
31   isSatisfied = (
32       "Answer true if this constraint is satisfied in the current
solution."
33       self subclassResponsibility
34   )
35
36   "add/remove"
37   addConstraint = (
38       "Activate this constraint and attempt to satisfy it."
39
40       self addToGraph.
41       Planner current incrementalAdd: self.
42   )
43
44   addToGraph = (
45       "Add myself to the constraint graph."
46       self subclassResponsibility
47   )
48
49   destroyConstraint = (
50       "Deactivate this constraint, remove it from the constraint graph,
51       possibly causing other constraints to be satisfied, and destroy
it."
52

```



```

53         (self isSatisfied) ifTrue: [Planner current incrementalRemove:
self].
54         self removeFromGraph.
55         "self release."
56     )
57
58     removeFromGraph = (
59         "Remove myself from the constraint graph."
60         self subclassResponsibility
61     )
62
63     "planning"
64     chooseMethod: mark = (
65         "Decide if I can be satisfied and record that decision. The
output of
66         the chosen method must not have the given mark and must have a
67         walkabout strength less than that of this constraint."
68         self subclassResponsibility
69     )
70
71     execute = (
72         "Enforce this constraint. Assume that it is satisfied."
73         self subclassResponsibility
74     )
75
76     inputsDo: aBlock = (
77         "Assume that I am satisfied. Evaluate the given block on all my
current
78         input variables."
79         self subclassResponsibility
80     )
81
82     inputsKnown: mark = (
83         "Assume that I am satisfied. Answer true if all my current inputs
are
84         known. A variable is known if either a) it is 'stay' (i.e. it is
a
85         constant at plan execution time), b) it has the given mark
(indicating
86         that it has been computed by a constraint appearing earlier in
the
87         plan), or c) it is not determined by any constraint."
88
89         self inputsDo:
90             [: v |
91                 ((v mark = mark) or: [(v stay) or: [v determinedBy == nil]])
ifFalse:
92                     [^false]].
93         ^true
94     )
95
96     markUnsatisfied = (
97         "Record the fact that I am unsatisfied."
98         self subclassResponsibility
99     )
100
101     output = (
102         "Answer my current output variable. Raise an error if I am not
103         currently satisfied."
104         self subclassResponsibility
105     )

```

```

106
107     recalculate = (
108         "Calculate the walkabout strength, the stay flag, and, if it is
109         'stay',
110         the value for the current output of this constraint. Assume this
111         constraint is satisfied."
112         self subclassResponsibility
113     )
114
115     satisfy: mark = (
116         "Attempt to find a way to enforce this (still unsatisfied)
117         constraint.
118         If successful, record the solution, perhaps modifying the current
119         dataflow graph. Answer the constraint that this constraint
120         overrides,
121         if there is one, or nil, if there isn't."
122         | overridden |
123         self chooseMethod: mark.
124         (self isSatisfied)
125         ifTrue: "constraint can be satisfied"
126         ["mark inputs to allow cycle detection in addPropagate"
127         | out |
128         self inputsDo: [: in | in mark: mark].
129         out := self output.
130         overridden := out determinedBy.
131         (overridden == nil) ifFalse: [overridden
132         markUnsatisfied].
133         out determinedBy: self.
134         (Planner current addPropagate: self mark: mark) ifFalse:
135         [self notify:
136         ('Cycle encountered adding:\n    ',
137         self printString,
138         '\nConstraint removed.') withCRs.
139         ^nil].
140         out mark: mark]
141         ifFalse: "constraint cannot be satisfied"
142         [overridden := nil.
143         (strength sameAs: (Strength required)) ifTrue:
144         [self notify: 'Failed to satisfy a required
145         constraint']].
146         ^overridden
147     )
148
149     "printing"
150     longPrintOn: aStream = (
151         | bindings |
152         aStream nextPut: '('.
153         self shortPrintOn: aStream.
154         aStream space.
155         aStream nextPutAll: strength printString.
156         (self isSatisfied)
157         ifTrue:
158         [aStream cr. aStream space. aStream space. aStream space.
159         self inputsDo:
160         [: in | aStream nextPutAll: 'v', in asOop
161         printString, ' '].
162         aStream nextPutAll: '-> '.
163         aStream nextPutAll: 'v', self output asOop printString]
164         ifFalse:

```

```
161         [aStream space. aStream nextPutAll: 'UNSATISFIED'].
162     aStream nextPut: ')'.
163     aStream cr.
164 )
165
166 printOn: aStream = (
167     self shortPrintOn: aStream
168 )
169
170 shortPrintOn: aStream = (
171     aStream nextPutAll: self class name, '(', self asOop printString,
172     ')'.
173 )
174 )
```

```

1 BinaryConstraint = AbstractConstraint (
2   "I am an abstract superclass for constraints having two possible
output
3   variables.
4
5   Instance variables:
6       v1, v2      possible output variables <Variable>
7       direction   one of:
8                   #forward (v2 is output)
9                   #backward ( v1 is output)
10                  nil (not satisfied)"
11   | v1 v2 direction |
12
13   "initialize-release"
14
15   var: variable1 var: variable2 strength: strengthSymbol = (
16       "Initialize myself with the given variables and strength."
17
18       strength := Strength of: strengthSymbol.
19       v1 := variable1.
20       v2 := variable2.
21       direction := nil.
22       self addConstraint.
23   )
24
25   "queries"
26
27   isSatisfied = (
28       "Answer true if this constraint is satisfied in the current
solution."
29
30       ^direction notNil
31   )
32
33   "add/remove"
34
35   addToGraph = (
36       "Add myself to the constraint graph."
37
38       v1 addConstraint: self.
39       v2 addConstraint: self.
40       direction := nil
41   )
42
43   removeFromGraph = (
44       "Remove myself from the constraint graph."
45
46       (v1 == nil) ifFalse: [v1 removeConstraint: self].
47       (v2 == nil) ifFalse: [v2 removeConstraint: self].
48       direction := nil.
49   )
50
51   "planning"
52
53   chooseMethod: mark = (
54       "Decide if I can be satisfied and which way I should flow based on
55       the relative strength of the variables I relate, and record that
56       decision."

```

```

57
58     (v1 mark == mark) ifTrue:          "forward or nothing"
59         [((v2 mark ~= mark) and: [strength stronger: v2 walkStrength])
60             ifTrue: [^direction := #forward]
61             ifFalse: [^direction := nil]].
62
63     (v2 mark == mark) ifTrue:          "backward or nothing"
64         [((v1 mark ~= mark) and: [strength stronger: v1 walkStrength])
65             ifTrue: [^direction := #backward]
66             ifFalse: [^direction := nil]].
67
68     "if we get here, neither variable is marked, so we have choice"
69     (v1 walkStrength weaker: v2 walkStrength)
70         ifTrue:
71             [(strength stronger: v1 walkStrength)
72                 ifTrue: [^direction := #backward]
73                 ifFalse: [^direction := nil]]
74         ifFalse:
75             [(strength stronger: v2 walkStrength)
76                 ifTrue: [^direction := #forward]
77                 ifFalse: [^direction := nil]].
78     )
79
80     execute = (
81         "Enforce this constraint. Assume that it is satisfied."
82
83         self subclassResponsibility
84     )
85
86     inputsDo: aBlock = (
87         "Evaluate the given block on my current input variable."
88
89         (direction == #forward)
90             ifTrue: [aBlock value: v1]
91             ifFalse: [aBlock value: v2].
92     )
93
94     markUnsatisfied = (
95         "Record the fact that I am unsatisfied."
96
97         direction := nil.
98     )
99
100    output = (
101        "Answer my current output variable."
102
103        (direction == #forward)
104            ifTrue: [^v2]
105            ifFalse: [^v1]
106    )
107
108    recalculate = (
109        "Calculate the walkabout strength, the stay flag, and, if it is
110        'stay',
111            the value for the current output of this constraint. Assume
112            this
113            constraint is satisfied."
114
115        | in out |
116        (direction == #forward)
117            ifTrue: [in := v1. out := v2]

```

```
116         ifFalse: [in := v2. out := v1].
117     out walkStrength: (strength weakest: in walkStrength).
118     out stay: (in stay).
119     (out stay) ifTrue: [self execute].          "stay optimization"
120 )
121 )
```

```
1 DeltaBlue = Benchmark (
2
3   oneTimeSetup = ( Strength initialize )
4
5   innerBenchmarkLoop: innerIterations = (
6     Planner chainTest: innerIterations.
7     Planner projectionTest: innerIterations.
8     ^ true
9   )
10 )
```

```
1 EditConstraint = UnaryConstraint (  
2     "I am a unary input constraint used to mark a variable that the client  
3     wishes to change."  
4  
5     "queries"  
6  
7     isInput = (  
8         "I indicate that a variable is to be changed by imperative code."  
9  
10        ^true  
11    )  
12  
13    "execution"  
14  
15    execute = (  
16        "Edit constraints do nothing."  
17    )  
18  
19    ----  
20  
21    "instance creation"  
22  
23    var: aVariable strength: strengthSymbol = (  
24        "Install an edit constraint with the given strength on the given  
25        variable."  
26  
27        ^(self new) var: aVariable strength: strengthSymbol  
28    )  
29 )
```



```

1 EqualityConstraint = BinaryConstraint (
2   "I constrain two variables to have the same value: `v1 = v2`."
3
4   "execution"
5
6   execute = (
7     "Enforce this constraint. Assume that it is satisfied."
8
9     (direction == #forward)
10      ifTrue: [v2 value: v1 value]
11      ifFalse: [v1 value: v2 value].
12   )
13
14   ----
15
16   "instance creation"
17
18   var: variable1 var: variable2 strength: strengthSymbol = (
19     "Install a constraint with the given strength equating the given
20     variables."
21
22     ^(self new) var: variable1 var: variable2 strength: strengthSymbol
23   )
24 )

```

```

1 OrderedCollection = Vector (
2   add: elem = (
3     ^ self append: elem
4   )
5
6   addLast: elem = (
7     ^ self append: elem
8   )
9
10  remove: obj ifAbsent: aBlock = (
11    (self remove: obj) ifFalse: aBlock
12  )
13
14  " private "
15  insert: anObject before: spot = (
16    "spot is an index in the range firstIndex .. lastIndex,
17     such an index is not known from outside the collection.
18     Never use this method in your code, it is meant for private use
by
19     OrderedCollection only.
20     The methods for use are:
21     #add:before:    to insert an object before another object
22     #add:beforeIndex:  to insert an object before a given
position. "
23    | delta spotIndex |
24    spotIndex := spot.
25    delta := spotIndex - first.
26    first = 1 ifTrue: [
27      self makeRoomAtFirst.
28      spotIndex := first + delta].
29    first := first - 1.
30    storage
31      replaceFrom: first
32      to: spotIndex - 2
33      with: storage
34      startingAt: first + 1.
35    storage at: spotIndex - 1 put: anObject.
36    ^ anObject
37  )
38
39  makeRoomAtFirst = (
40    "Make some empty slots at the front of the array.
41     If we have more than 50% free space, then just move the elements,
42     so that the first 50% of the slots are free, otherwise add new
free
43     slots to the front by growing. Precondition: firstIndex = 1"
44
45    | tally newFirstIndex newLastIndex |
46    tally := self size.
47    tally * 2 >= storage length ifTrue: [ ^self growAtFirst ].
48    tally = 0 ifTrue: [ ^self resetTo: storage length + 1 ].
49    newFirstIndex := storage length // 2 + 1.
50    newLastIndex := newFirstIndex - first + last - 1.
51    0 to: tally - 1 do: [ :offset |
52      storage at: newLastIndex - offset put: (storage at: last -
offset - 1) ].
53    storage from: first to: newFirstIndex - 1 put: nil.
54    first := newFirstIndex.

```

```

55         last := newLastIndex + 1
56     )
57
58     resetTo: index = (
59         first := index.
60         last := first
61     )
62
63     sort: i to: j with: sortBlock = (
64         "Sort elements i through j of self to be nondescending according
to
65         sortBlock."
66
67         | di dij dj tt ij k l n |
68         sortBlock ifNil: [^self defaultSort: i to: j].
69         "The prefix d means the data at that index."
70         (n := j + 1 - i) <= 1 ifTrue: [^self]. "Nothing to sort."
71         "Sort di,dj."
72         di := storage at: i.
73         dj := storage at: j.
74         (sortBlock value: di with: dj) "i.e., should di precede dj?"
75         ifFalse:
76             [storage swap: i with: j.
77             tt := di.
78             di := dj.
79             dj := tt].
80         n > 2
81         ifTrue: "More than two elements."
82             [ij := (i + j) / 2. "ij is the midpoint of i and j."
83             dij := storage at: ij. "Sort di,dij,dj. Make dij be
their median."
84             (sortBlock value: di with: dij) "i.e. should di precede
dij?"
85             ifTrue:
86                 [(sortBlock value: dij with: dj) "i.e., should dij
precede dj?"
87                 ifFalse:
88                     [storage swap: j with: ij.
89                     dij := dj]]
90                 ifFalse: "i.e. di should come after dij"
91                     [storage swap: i with: ij.
92                     dij := di].
93             n > 3
94             ifTrue: "More than three elements."
95                 ["Find k>i and l<j such that dk,dij,dl are in reverse
order.
96                 Swap k and l. Repeat this procedure until k and l
pass each other."
97                 k := i.
98                 l := j.
99                 [[l := l - 1. k <= l and: [sortBlock value: dij
with: (storage at: l)]]
100                 whileTrue. "i.e. while dl succeeds dij"
101                 [k := k + 1. k <= l and: [sortBlock value:
(storage at: k) with: dij]]
102                 whileTrue. "i.e. while dij succeeds dk"
103                 k <= l]
104                 whileTrue:
105                     [storage swap: k with: l].
106                 "Now l<k (either 1 or 2 less), and di through dl are all less
than or equal to dk

```

```

107         through dj.  Sort those two segments."
108         self sort: i to: l with: sortBlock.
109         self sort: k to: j with: sortBlock]] )
110
111     sort: aBlock = (
112         "Make the argument, aBlock, be the criterion for ordering
elements of the
113         receiver."
114
115         "sortBlocks with side effects may not work right"
116         self size > 0 ifTrue: [
117             self sort: first
118                 to: last - 1
119                 with: aBlock
120         ]
121     )
122
123
124
125     ----
126
127     with: anElement = (
128         | col |
129         col := self new: 10.
130         col append: anElement.
131         ^ col
132     )
133 )

```

```
1 Plan = OrderedCollection (  
2     "A Plan is an ordered list of constraints to be executed in sequence to  
3     resatisfy all currently satisfiable constraints in the face of one or  
more  
4     changing inputs."  
5  
6     "execution"  
7     execute = (  
8         "Execute my constraints in order."  
9  
10         self do: [: c | c execute]  
11     )  
12 )
```

```

1 Planner = (
2   "This benchmark is an implementation of the DeltaBlue Constraint
Solver
3   described in `The DeltaBlue Algorithm: An Incremental Constraint
4   Hierarchy Solver'', by Bjorn N. Freeman-Benson and John Maloney,
5   Communications of the ACM, January 1990 (also as University of
6   Washington TR 89-08-06).
7
8   To run the benchmark, execute the expression `Planner
standardBenchmark`."
9   | currentMark |
10
11   "initialize"
12
13   initialize = (
14     "Planner initialize"
15
16     currentMark := 1
17   )
18
19   "add/remove"
20
21   incrementalAdd: c = (
22     "Attempt to satisfy the given constraint and, if successful,
23     incrementally update the dataflow graph.
24
25     Details: If satisfying the constraint is successful, it may
override a
26     weaker constraint on its output. The algorithm attempts to
resatisfy
27     that constraint using some other method. This process is repeated
28     until either a) it reaches a variable that was not previously
29     determined by any constraint or b) it reaches a constraint that
30     is too weak to be satisfied using any of its methods. The
variables
31     of constraints that have been processed are marked with a unique
mark
32     value so that we know where we've been. This allows the
algorithm to
33     avoid getting into an infinite loop even if the constraint graph
has
34     an inadvertent cycle."
35
36     | mark overridden |
37     mark := self newMark.
38     overridden := c satisfy: mark.
39     [overridden == nil] whileFalse:
40       [overridden := overridden satisfy: mark]
41   )
42
43   incrementalRemove: c = (
44     "Entry point for retracting a constraint. Remove the given
constraint,
45     which should be satisfied, and incrementally update the dataflow
46     graph.
47
48     Details: Retracting the given constraint may allow some currently
49     unsatisfiable downstream constraint be satisfied. We thus

```

```

collect a
50     list of unsatisfied downstream constraints and attempt to satisfy
51     each one in turn. This list is sorted by constraint strength,
52     strongest first, as a heuristic for avoiding unnecessarily adding
53     and then overriding weak constraints."
54
55     | out unsatisfied |
56     out := c output.
57     c markUnsatisfied.
58     c removeFromGraph.
59     unsatisfied := self removePropagateFrom: out.
60     unsatisfied do: [: u | self incrementalAdd: u]
61 )
62
63 "planning/value propagation"
64
65 extractPlanFromConstraints: constraints = (
66     "Extract a plan for resatisfaction starting from the outputs of
the
67     given constraints, usually a set of input constraints."
68
69     | sources |
70     sources := OrderedCollection new.
71     constraints do:
72         [: c | ((c isInput) and: [c isSatisfied]) ifTrue: [sources
add: c]].
73     ^self makePlan: sources
74 )
75
76 extractPlanFromVariables: variables = (
77     "Extract a plan from the dataflow graph having the given
variables. It
78     is assumed that the given set of variables is complete, or at
least
79     that it contains all the input variables."
80
81     | sources |
82     sources := OrderedCollection new.
83     variables do:
84         [: v |
85             (v constraints) do:
86                 [: c | ((c isInput) and: [c isSatisfied]) ifTrue:
[sources add: c]].
87     ^self makePlan: sources
88 )
89
90 makePlan: sources = (
91     "Extract a plan for resatisfaction starting from the given
satisfied
92     source constraints, usually a set of input constraints. This
method
93     assumes that stay optimization is desired; the plan will contain
only
94     constraints whose output variables are not stay. Constraints
that do
95     no computation, such as stay and edit constraints, are not
included
96     in the plan.
97
98     Details: The outputs of a constraint are marked when it is added
to

```

```

99         the plan under construction. A constraint may be appended to the
plan
100        when all its input variables are known. A variable is known if
either
101        a) the variable is marked (indicating that has been computed by a
102        constraint appearing earlier in the plan), b) the variable is
'stay'
103        (i.e. it is a constant at plan execution time), or c) the
variable
104        is not determined by any constraint. The last provision is for
past
105        states of history variables, which are not stay but which are
also
106        not computed by any constraint."
107
108        | mark plan todo c |
109        mark := self newMark.
110        plan := Plan new.
111        todo := sources.
112        [todo isEmpty] whileFalse:
113            [c := todo removeFirst.
114             ((c output mark ~= mark) and:          "not in plan already
and..."
115              [c inputsKnown: mark]) ifTrue:      "eligible for inclusion"
116              [plan addLast: c.
117               c output mark: mark.
118               self addConstraintsConsuming: c output to: todo]].
119        ^plan
120    )
121
122    propagateFrom: v = (
123        "The given variable has changed. Propagate new values downstream."
124
125        | todo c |
126        todo := OrderedCollection new.
127        self addConstraintsConsuming: v to: todo.
128        [todo isEmpty] whileFalse:
129            [c := todo removeFirst.
130             c execute.
131             self addConstraintsConsuming: c output to: todo].
132    )
133
134    "private"
135
136    addConstraintsConsuming: v to: aCollection = (
137        | determiningC |
138        determiningC := v determinedBy.
139        v constraints do:
140            [: c |
141                ((c == determiningC) or: [c isSatisfied not]) ifFalse:
142                [aCollection add: c]].
143    )
144
145    addPropagate: c mark: mark = (
146        "Recompute the walkabout strengths and stay flags of all variables
147        downstream of the given constraint and recompute the actual
values
148        of all variables whose stay flag is true. If a cycle is detected,
149        remove the given constraint and answer false. Otherwise, answer
true.
150

```



```

151      Details: Cycles are detected when a marked variable is
encountered
152      downstream of the given constraint. The sender is assumed to have
153      marked the inputs of the given constraint with the given mark.
Thus,
154      encountering a marked node downstream of the output constraint
means
155      that there is a path from the constraint's output to one of its
156      inputs."
157
158      | todo d |
159      todo := OrderedCollection with: c.
160      [todo isEmpty] whileFalse:
161          [d := todo removeFirst.
162           (d output mark = mark) ifTrue:
163               [self incrementalRemove: c.
164                ^false].
165           d recalculate.
166           self addConstraintsConsuming: d output to: todo].
167      ^true
168  )
169
170  changeVar: aVariable newValue: newValue = (
171      | editConstraint plan |
172      editConstraint := EditConstraint var: aVariable strength:
#preferred.
173      plan := self extractPlanFromConstraints: (Array with:
editConstraint).
174      10 timesRepeat: [
175          aVariable value: newValue.
176          plan execute].
177      editConstraint destroyConstraint.
178  )
179
180  constraintsConsuming: v do: aBlock = (
181
182      | determiningC |
183      determiningC := v determinedBy.
184      v constraints do:
185          [: c |
186              ((c == determiningC) or: [c isSatisfied not]) ifFalse:
187                  [aBlock value: c]].
188  )
189
190  newMark = (
191      "Select a previously unused mark value.
192
193      Details: We just keep incrementing. If necessary, the counter
will
194      turn into a LargePositiveInteger. In that case, it will be a bit
195      slower to compute the next mark but the algorithms will all
behave
196      correctly. We reserve the value '0' to mean 'unmarked'. Thus,
this
197      generator starts at '1' and will never produce '0' as a mark
value."
198
199      ^currentMark := currentMark + 1
200  )
201
202  removePropagateFrom: out = (

```

```

203      "Update the walkabout strengths and stay flags of all variables
204      downstream of the given constraint. Answer a collection of
unsatisfied
205      constraints sorted in order of decreasing strength."
206
207      | unsatisfied todo v nextC u2 |
208      unsatisfied := OrderedCollection new.
209
210      out determinedBy: nil.
211      out walkStrength: Strength absoluteWeakest.
212      out stay: true.
213      todo := OrderedCollection with: out.
214      [todo isEmpty] whileFalse: [
215          v := todo removeFirst.
216          v constraints do: [:c |
217              (c isSatisfied) ifFalse: [unsatisfied add: c]].
218          self constraintsConsuming: v do:
219              [:c |
220                  c recalculate.
221                  todo add: c output]].
222
223      unsatisfied sort: [:c1 :c2 | c1 strength stronger: c2 strength].
224      ^unsatisfied
225  )
226
227  run = (
228      Planner standardBenchmark
229  )
230
231  ----
232
233  | currentPlanner |
234
235  "instance creation"
236  new = (
237      ^currentPlanner := super new initialize
238  )
239
240  "benchmarks"
241  chainTest: n = (
242      "Do chain-of-equality-constraints performance tests."
243      | vars editConstraint plan planner |
244
245      planner := Planner new.
246      vars := Array new: n+1 withAll: [ Variable new ].
247
248      "thread a chain of equality constraints through the variables"
249      1 to: n do:
250          [ :i | | v1 v2 |
251              v1 := vars at: i.
252              v2 := vars at: i + 1.
253              EqualityConstraint var: v1 var: v2 strength: #required].
254
255      StayConstraint var: vars last strength: #strongDefault.
256      editConstraint := EditConstraint var: (vars first) strength:
#preferred.
257      plan := planner extractPlanFromConstraints: (Array with:
editConstraint).
258      1 to: 100 do: [ :v |
259          vars first value: v.
260          plan execute.

```

```

261         vars last value ~= v ifTrue: [self error: 'Chain test
failed!!!']].
262     editConstraint destroyConstraint
263 )
264
265     projectionTest: n = (
266         "This test constructs a two sets of variables related to each
other by
267         a simple linear transformation (scale and offset).".
268
269         | scale offset src dst planner dests |
270         planner := Planner new.
271         dests := OrderedCollection new.
272         scale := Variable value: 10.
273         offset := Variable value: 1000.
274         1 to: n do:
275             [ :i |
276                 src := Variable value: i.
277                 dst := Variable value: i.
278                 dests add: dst.
279                 StayConstraint var: src strength: #default.
280                 ScaleConstraint var: src var: scale var: offset var: dst
strength: #required].
281
282         planner changeVar: src newValue: 17.
283         dst value ~= 1170 ifTrue: [self error: 'Projection test 1
failed!!!'].
284
285         planner changeVar: dst newValue: 1050.
286         src value ~= 5 ifTrue: [self error: 'Projection test 2 failed!!!'].
287
288         planner changeVar: scale newValue: 5.
289         1 to: n - 1 do: [ :i |
290             (dests at: i) value ~= (i*5 + 1000)
291             ifTrue: [self error: 'Projection test 3 failed!!!']].
292
293         planner changeVar: offset newValue: 2000.
294         1 to: n - 1 do: [ :i |
295             (dests at: i) value ~= (i*5 + 2000)
296             ifTrue: [self error: 'Projection test 4 failed!!!']]
297     )
298
299     report: string times: count run: aBlock = (
300         "Report the time required to execute the given block."
301
302         | startTime endTime |
303         startTime := system ticks.
304         count timesRepeat: aBlock.
305         endTime := system ticks.
306
307         (string + ' ' + ((endTime - startTime) / count / 1000) asString +
' milliseconds') println.
308     )
309
310     standardBenchmark = (
311         "This the combined benchmark."
312         "Planner standardBenchmark"
313
314         Strength initialize.
315
316         self report: 'Chain and projection tests' times: 100 run: [

```

```
317         self chainTest: 100.  
318         self projectionTest: 100  
319     ]  
320 )  
321  
322 "accessing"  
323 current = (  
324     ^currentPlanner  
325 )  
326 )
```

```

1 ScaleConstraint = BinaryConstraint (
2     "I relate two variables by the linear scaling relationship:
3     `v2 = (v1 * scale) + offset`. Either v1 or v2 may be changed to
maintain
4     this relationship but the scale factor and offset are considered read-
only.
5
6     Instance variables:
7         scale      scale factor input variable <Variable>
8         offset      offset input variable <Variable>"
9     | scale offset |
10
11     "initialize-release"
12
13     src: srcVar scale: scaleVar offset: offsetVar dst: dstVar strength:
strengthSymbol = (
14         "Initialize myself with the given variables and strength."
15
16         strength := Strength of: strengthSymbol.
17         v1 := srcVar.
18         v2 := dstVar.
19         scale := scaleVar.
20         offset := offsetVar.
21         direction := nil.
22         self addConstraint.
23     )
24
25     "add/remove"
26
27     addToGraph = (
28         "Add myself to the constraint graph."
29
30         v1 addConstraint: self.
31         v2 addConstraint: self.
32         scale addConstraint: self.
33         offset addConstraint: self.
34         direction := nil.
35     )
36
37     removeFromGraph = (
38         "Remove myself from the constraint graph."
39
40         (v1 == nil) ifFalse: [v1 removeConstraint: self].
41         (v2 == nil) ifFalse: [v2 removeConstraint: self].
42         (scale == nil) ifFalse: [scale removeConstraint: self].
43         (offset == nil) ifFalse: [offset removeConstraint: self].
44         direction := nil.
45     )
46
47     "planning"
48
49     execute = (
50         "Enforce this constraint. Assume that it is satisfied."
51
52         (direction == #forward)
53             ifTrue: [v2 value: (v1 value * scale value) + offset value]
54             ifFalse: [v1 value: (v2 value - offset value) / scale value].
55     )

```

```

56
57     inputsDo: aBlock = (
58         "Evaluate the given block on my current input variable."
59
60         (direction == #forward)
61             ifTrue: [aBlock value: v1.
62                     aBlock value: scale.
63                     aBlock value: offset]
64             ifFalse: [aBlock value: v2.
65                      aBlock value: scale.
66                      aBlock value: offset].
67     )
68
69     recalculate = (
70         "Calculate the walkabout strength, the stay flag, and, if it is
71         'stay',
72         the value for the current output of this constraint. Assume this
73         constraint is satisfied."
74         | in out |
75         (direction == #forward)
76             ifTrue: [in := v1. out := v2]
77             ifFalse: [out := v1. in := v2].
78         out walkStrength: (strength weakest: in walkStrength).
79         out stay: ((in stay) and: [(scale stay) and: [offset stay]]).
80         (out stay) ifTrue: [self execute].      "stay optimization"
81     )
82
83     ----
84
85     "instance creation"
86
87     var: src var: scale var: offset var: dst strength: strengthSymbol = (
88         "Install a scale constraint with the given strength on the given
89         variables."
90
91         ^(self new) src: src scale: scale offset: offset dst: dst
92         strength: strengthSymbol
93     )

```

```

1 SortedCollection = OrderedCollection (
2
3   | sortBlock |
4
5   indexForInserting: newObject = (
6     | index low high |
7     low := first.
8     high := last - 1.
9
10    sortBlock isNil
11      ifTrue: [[index := high + low / 2. low > high]
12        whileFalse:
13          [((storage at: index) <= newObject)
14            ifTrue: [low := index + 1]
15            ifFalse: [high := index - 1]]]
16      ifFalse: [[index := high + low / 2. low > high]
17        whileFalse:
18          [(sortBlock value: (storage at: index) with:
newObject)
19            ifTrue: [low := index + 1]
20            ifFalse: [high := index - 1]]].
21    ^low )
22
23   sort: i to: j = (
24     "Sort elements i through j of self to be nondescending according
to
25     sortBlock."
26
27     | di dij dj tt ij k l n |
28     sortBlock ifNil: [^self defaultSort: i to: j].
29     "The prefix d means the data at that index."
30     (n := j + 1 - i) <= 1 ifTrue: [^self]. "Nothing to sort."
31     "Sort di,dj."
32     di := storage at: i.
33     dj := storage at: j.
34     (sortBlock value: di with: dj) "i.e., should di precede dj?"
35     ifFalse:
36       [storage swap: i with: j.
37         tt := di.
38         di := dj.
39         dj := tt].
40     n > 2
41     ifTrue: "More than two elements."
42     [ij := (i + j) / 2. "ij is the midpoint of i and j."
43     dij := storage at: ij. "Sort di,dij,dj. Make dij be
their median."
44     (sortBlock value: di with: dij) "i.e. should di precede
dij?"
45     ifTrue:
46       [(sortBlock value: dij with: dj) "i.e., should dij
precede dj?"
47       ifFalse:
48         [storage swap: j with: ij.
49         dij := dj]]
50     ifFalse: "i.e. di should come after dij"
51     [storage swap: i with: ij.
52     dij := di].
53     n > 3

```

```

54             ifTrue: "More than three elements."
55             ["Find k>i and l<j such that dk,dij,dl are in reverse
order.
56             Swap k and l. Repeat this procedure until k and l
pass each other."
57             k := i.
58             l := j.
59             [[l := l - 1. k <= l and: [sortBlock value: dij
with: (storage at: l)]]
60             whileTrue. "i.e. while dl succeeds dij"
61             [k := k + 1. k <= l and: [sortBlock value:
(storage at: k) with: dij]]
62             whileTrue. "i.e. while dij succeeds dk"
63             k <= l]
64             whileTrue:
65             [storage swap: k with: l].
66             "Now l<k (either 1 or 2 less), and di through dl are all less
than or equal to dk
67             through dj. Sort those two segments."
68             self sort: i to: l.
69             self sort: k to: j]] )
70
71     addAll: aCollection = (
72     aCollection size > (self size / 3)
73     ifTrue:
74     [aCollection do: [:each | self addLast: each].
75     self reSort]
76     ifFalse: [aCollection do: [:each | self add: each]].
77     ^ aCollection )
78
79     reSort = (
80     self sort: first
81     to: last - 1
82     )
83
84     sortBlock: aBlock = (
85     "Make the argument, aBlock, be the criterion for ordering
elements of the
86     receiver."
87
88     sortBlock := aBlock.
89     "sortBlocks with side effects may not work right"
90     self size > 0 ifTrue: [self reSort] )
91
92     copyEmpty = (
93     "Answer a copy of the receiver without any of the receiver's
elements."
94
95     ^self species sortBlock: sortBlock )
96
97     addFirst: newObject = (
98     self shouldNotImplement )
99
100    insert: anObject before: spot = (
101    self shouldNotImplement )
102
103    defaultSort: i to: j = (
104    "Sort elements i through j of self to be nondescending according
to
105    sortBlock." "Assume the default sort block ([:x :y | x <= y])."
106

```



```

107 | di dij dj tt ij k l n |
108 "The prefix d means the data at that index."
109 (n := j + 1 - i) <= 1 ifTrue: [^self]. "Nothing to sort."
110 "Sort di,dj."
111 di := storage at: i.
112 dj := storage at: j.
113 (di <= dj) "i.e., should di precede dj?"
114 ifFalse:
115     [storage swap: i with: j.
116      tt := di.
117      di := dj.
118      dj := tt].
119 n > 2
120 ifTrue: "More than two elements."
121     [ij := (i + j) / 2. "ij is the midpoint of i and j."
122      dij := storage at: ij. "Sort di,dij,dj. Make dij be
their median."
123      (di <= dij) "i.e. should di precede dij?"
124      ifTrue:
125          [(dij <= dj) "i.e., should dij precede dj?"
126           ifFalse:
127               [storage swap: j with: ij.
128                dij := dij]]
129          ifFalse: "i.e. di should come after dij"
130              [storage swap: i with: ij.
131               dij := di].
132 n > 3
133 ifTrue: "More than three elements."
134     ["Find k>i and l<j such that dk,dij,dl are in reverse
order.
135     Swap k and l. Repeat this procedure until k and l
pass each other."
136     k := i.
137     l := j.
138     [[l := l - 1. k <= l and: [dij <= (storage at: l)]]
139      whileTrue. "i.e. while dl succeeds dij"
140      [k := k + 1. k <= l and: [(storage at: k) <= dij]]
141      whileTrue. "i.e. while dij succeeds dk"
142      k <= l]
143      whileTrue:
144          [storage swap: k with: l].
145     "Now l<k (either 1 or 2 less), and di through dl are all less
than or equal to dk
146     through dj. Sort those two segments."
147     self defaultSort: i to: l.
148     self defaultSort: k to: j]] )
149
150 should: a precede: b = (
151     ^ sortBlock ifNil: [a <= b] ifNotNil: [sortBlock value: a with:
b] )
152
153 median = (
154     "Return the middle element, or as close as we can get."
155     ^ self at: self size + 1 / 2 )
156
157 at: anInteger put: anObject = (
158     self shouldNotImplement )
159
160
161 add: newObject = (
162     ^ super insert: newObject before: (self indexOfInserting:

```

```

newObject) )
163
164     = aSortedCollection = (
165         "Answer true if my and aSortedCollection's species are the same,
166         and if our blocks are the same, and if our elements are the same."
167
168         self species = aSortedCollection species ifFalse: [^ false].
169         sortBlock = aSortedCollection sortBlock
170             ifTrue: [^ super = aSortedCollection]
171             ifFalse: [^ false] )
172
173     collect: aBlock = (
174         "Evaluate aBlock with each of my elements as the argument.
Collect the
175         resulting values into an OrderedCollection. Answer the new
collection.
176         Override the superclass in order to produce an OrderedCollection
instead
177         of a SortedCollection."
178
179         | newCollection |
180         newCollection := OrderedCollection new: self size.
181         self do: [:each | newCollection addLast: (aBlock value: each)].
182         ^ newCollection )
183
184     sort: aSortBlock = (
185         "Sort this storage using aSortBlock. The block should take two
arguments
186         and return true if the first element should precede the second
one."
187
188         super sort: aSortBlock.
189         sortBlock := aSortBlock )
190
191     join: aCollection = (
192         | result |
193         "Curiously addAllLast: does not trigger a reSort, so we must do
it here."
194         result := super join: aCollection.
195         result reSort.
196         ^ result
197     )
198
199     sortTopologically = (
200         "Plenty of room for increased efficiency in this one."
201
202         | remaining result pick |
203         remaining := self asOrderedCollection.
204         result := OrderedCollection new.
205         [remaining isEmpty] whileFalse: [
206             pick := remaining select: [:item |
207                 remaining allSatisfy: [:anotherItem |
208                     item == anotherItem or: [self should: item precede:
anotherItem]]].
209             pick isEmpty ifTrue: [self error: 'bad topological ordering'].
210             result addAll: pick.
211             remaining removeAll: pick].
212         ^self copySameFrom: result )
213
214     sortBlock = (
215         "Answer the blockContext which is the criterion for sorting

```

```
elements of
216         the receiver."
217
218         ^sortBlock )
219
220     ----
221
222     sortBlock: aBlock = (
223         "Answer an instance of me such that its elements are sorted
according to
224         the criterion specified in aBlock."
225
226         ^(super new: 10) sortBlock: aBlock )
227
228 )
229
```

```

1 StayConstraint = UnaryConstraint (
2     "I mark variables that should, with some level of preference, stay the
same.
3     I have one method with zero inputs and one output, which does nothing.
4     Planners may exploit the fact that, if I am satisfied, my output will
not
5     change during plan execution. This is called 'stay optimization.'"
6
7
8     "execution"
9
10    execute = (
11        "Stay constraints do nothing."
12    )
13
14    ----
15
16    "instance creation"
17
18    var: aVariable strength: strengthSymbol = (
19        "Install a stay constraint with the given strength on the given
variable."
20
21        ^(self new) var: aVariable strength: strengthSymbol
22    )
23 )

```

```

1 Strength = (
2   "Strengths are used to measure the relative importance of
constraints. The
3   hierarchy of available strengths is determined by the class variable
4   StrengthTable (see my class initialization method). Because
Strengths are
5   invariant, references to Strength instances are shared (i.e. all
references
6   to `Strength of: #required` point to a single, shared instance). New
7   strengths may be inserted in the strength hierarchy without
disrupting
8   current constraints.
9
10  Instance variables:
11    symbolicValue      symbolic strength name (e.g. #required)
<Symbol>
12    arithmeticValue    index of the constraint in the hierarchy,
used for comparisons <Number>"
13    | symbolicValue arithmeticValue |
14
15    "comparing"
16
17    sameAs: aStrength = (
18      "Answer true if I am the same strength as the given Strength."
19
20      ^arithmeticValue = aStrength arithmeticValue
21    )
22
23    stronger: aStrength = (
24      "Answer true if I am stronger than the given Strength."
25      ^arithmeticValue < aStrength arithmeticValue
26    )
27
28    weaker: aStrength = (
29      "Answer true if I am weaker than the given Strength."
30      ^arithmeticValue > aStrength arithmeticValue
31    )
32
33    "max/min"
34
35    strongest: aStrength = (
36      "Answer the stronger of myself and aStrength."
37
38      (aStrength stronger: self)
39      ifTrue: [^aStrength]
40      ifFalse: [^self].
41    )
42
43    weakest: aStrength = (
44      "Answer the weaker of myself and aStrength."
45
46      (aStrength weaker: self)
47      ifTrue: [^aStrength]
48      ifFalse: [^self].
49    )
50
51    "printing"
52

```

```

53     printOn: aStream = (
54         "Append a string which represents my strength onto aStream."
55
56         aStream nextPutAll: '%', symbolicValue, '%'.
57     )
58
59     "private"
60
61     arithmeticValue = (
62         "Answer my arithmetic value. Used for comparisons. Note that
63         STRONGER constraints have SMALLER arithmetic values."
64
65         ^arithmeticValue
66     )
67
68     initializeWith: symVal = (
69         "Record my symbolic value and reset my arithmetic value."
70
71         symbolicValue := symVal.
72         arithmeticValue := Strength strengthTable at: symbolicValue.
73     )
74
75     ----
76     | AbsoluteStrongest AbsoluteWeakest Required StrengthConstants
StrengthTable |
77
78     strengthTable = (
79         ^ StrengthTable
80     )
81
82     "class initialization"
83
84     initialize = (
85         "Initialize the symbolic strength table. Fix the internally caches
86         values of all existing instances."
87         "Strength initialize"
88
89         StrengthTable := Dictionary new.
90         StrengthTable at: #absoluteStrongest put: -10000.
91         StrengthTable at: #required put: -800.
92         StrengthTable at: #strongPreferred put: -600.
93         StrengthTable at: #preferred put: -400.
94         StrengthTable at: #strongDefault put: -200.
95         StrengthTable at: #default put: 0.
96         StrengthTable at: #weakDefault put: 500.
97         StrengthTable at: #absoluteWeakest put: 10000.
98
99         StrengthConstants := Dictionary new.
100        StrengthTable keys do:
101            [: strengthSymbol |
102                StrengthConstants
103                    at: strengthSymbol
104                    put: ((super new) initializeWith: strengthSymbol)].
105
106        AbsoluteStrongest := Strength of: #absoluteStrongest.
107        AbsoluteWeakest := Strength of: #absoluteWeakest.
108        Required := Strength of: #required.
109    )
110
111    "instance creation"
112    of: aSymbol = (

```

```
113         "Answer an instance with the specified strength."
114         ^StrengthConstants at: aSymbol
115     )
116     "constants"
117     absoluteStrongest = (
118         ^AbsoluteStrongest
119     )
120 )
121
122     absoluteWeakest = (
123         ^AbsoluteWeakest
124     )
125
126     required = (
127         ^Required
128     )
129 )
```

```

1 UnaryConstraint = AbstractConstraint (
2   "I am an abstract superclass for constraints having a single possible
output
3   variable.
4
5   Instance variables:
6       output      possible output variable <Variable>
7       satisfied    true if I am currently satisfied <Boolean>"
8   | output satisfied |
9
10  "initialize-release"
11
12  var: aVariable strength: strengthSymbol = (
13      "Initialize myself with the given variable and strength."
14
15      strength := Strength of: strengthSymbol.
16      output := aVariable.
17      satisfied := false.
18      self addConstraint.
19  )
20
21  "queries"
22  isSatisfied = (
23      "Answer true if this constraint is satisfied in the current
solution."
24
25      ^satisfied
26  )
27
28  "add/remove"
29
30  addToGraph = (
31      "Add myself to the constraint graph."
32
33      output addConstraint: self.
34      satisfied := false.
35  )
36
37  removeFromGraph = (
38      "Remove myself from the constraint graph."
39
40      (output == nil) ifFalse: [output removeConstraint: self].
41      satisfied := false.
42  )
43
44  "planning"
45
46  chooseMethod: mark = (
47      "Decide if I can be satisfied and record that decision."
48
49      satisfied :=
50          (output mark ~= mark) and:
51          [strength stronger: output walkStrength].
52  )
53
54  execute = (
55      "Enforce this constraint. Assume that it is satisfied."
56      self subclassResponsibility

```



```

57     )
58
59     inputsDo: aBlock = (
60         "I have no input variables."
61     )
62
63     markUnsatisfied = (
64         "Record the fact that I am unsatisfied."
65         satisfied := false.
66     )
67
68     output = (
69         "Answer my current output variable."
70         ^ output
71     )
72
73     recalculate = (
74         "Calculate the walkabout strength, the stay flag, and, if it is
'stay',
75         the value for the current output of this constraint. Assume this
76         constraint is satisfied."
77
78         output walkStrength: strength.
79         output stay: (self isInput not).
80         (output stay) ifTrue: [self execute].    "stay optimization"
81     )
82 )

```

```

1 Variable = (
2     "I represent a constrained variable. In addition to my value, I
maintain the
3     structure of the constraint graph, the current dataflow graph, and
various
4     parameters of interest to the DeltaBlue incremental constraint solver.
5
6     Instance variables:
7         value          my value; changed by constraints, read by client
<Object>
8         constraints     normal constraints that reference me <Array of
Constraint>
9         determinedBy    the constraint that currently determines
10                        my value (or nil if there isn't one) <Constraint>
11         walkStrength    my walkabout strength <Strength>
12         stay            true if I am a planning-time constant <Boolean>
13         mark            used by the planner to mark constraints <Number>"
14     | value constraints determinedBy walkStrength stay mark |
15
16     "initialize-release"
17
18     initialize = (
19         value := 0.
20         constraints := OrderedCollection new: 2.
21         determinedBy := nil.
22         walkStrength := Strength absoluteWeakest.
23         stay := true.
24         mark := 0.
25     )
26
27     "access"
28     addConstraint: aConstraint = (
29         "Add the given constraint to the set of all constraints that refer
30         to me."
31
32         constraints add: aConstraint.
33     )
34
35     constraints = (
36         "Answer the set of constraints that refer to me."
37
38         ^constraints
39     )
40
41     determinedBy = (
42         "Answer the constraint that determines my value in the current
43         dataflow."
44
45         ^determinedBy
46     )
47
48     determinedBy: aConstraint = (
49         "Record that the given constraint determines my value in the
current
50         data flow."
51
52         determinedBy := aConstraint.
53     )

```

```

54
55     mark = (
56         "Answer my mark value."
57
58         ^mark
59     )
60
61     mark: markValue = (
62         "Set my mark value."
63
64         mark := markValue.
65     )
66
67     removeConstraint: c = (
68         "Remove all traces of c from this variable."
69
70         constraints remove: c ifAbsent: [].
71         (determinedBy == c) ifTrue: [determinedBy := nil].
72     )
73
74     stay = (
75         "Answer my stay flag."
76
77         ^stay
78     )
79
80     stay: aBoolean = (
81         "Set my stay flag."
82
83         stay := aBoolean
84     )
85
86     value = (
87         "Answer my value."
88
89         ^value
90     )
91
92     value: anObject = (
93         "Set my value."
94
95         value := anObject.
96     )
97
98     walkStrength = (
99         "Answer my walkabout strength in the current dataflow."
100
101         ^walkStrength
102     )
103
104     walkStrength: aStrength = (
105         "Set my walkabout strength in the current dataflow."
106
107         walkStrength := aStrength.
108     )
109
110     "printing"
111     longPrintOn: aStream = (
112
113         self shortPrintOn: aStream.
114         aStream nextPutAll: ' Constraints: '.

```

```

115         (constraints isEmpty)
116         ifTrue: [aStream cr. aStream tab. aStream nextPutAll: 'none']
117         ifFalse:
118             [constraints do:
119                 [: c | aStream cr. aStream tab. c shortPrintOn:
aStream]].
120         (determinedBy isNil) ifFalse:
121             [aStream cr. aStream nextPutAll: '    Determined by: '.
122                 aStream cr. aStream tab. determinedBy shortPrintOn: aStream].
123         aStream cr.
124     )
125
126     printOn: aStream = (
127         self shortPrintOn: aStream
128     )
129
130     shortPrintOn: aStream = (
131
132         aStream nextPutAll: 'V(', self asOop printString, ', '.
133         aStream nextPutAll: walkStrength printString, ', '.
134         (stay isNil) ifFalse:
135             [aStream nextPutAll: (stay ifTrue: ['stay, '] ifFalse:
['changing, '])].
136         aStream nextPutAll: value printString.
137         aStream nextPutAll: ')'.
138         aStream cr.
139     )
140
141     ----
142
143     "instance creation"
144
145     new = (
146         ^super new initialize
147     )
148
149     value: aValue = (
150         | o |
151         o := super new.
152         o initialize.
153         ^ o value: aValue
154     )
155 )

```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 "Pfannkuchen are somehow known in Benchmarking. i.e., there is a Lisp  
story"  
24  
25 " Original written for VisualWorks  
26 * The Computer Language Benchmarks Game  
27   http://shootout.alioth.debian.org/  
28   contributed by Paolo Bonzini *"  
29  
30 Fannkuch = Benchmark (  
31  
32   | timesRotated perm atEnd |  
33  
34   pfannkuchen: anArray = (  
35     | first complement a b k |  
36     k := 0.  
37     [ (first := anArray at: 1) = 1 ] whileFalse: [  
38       k := k + 1.  
39       complement := first + 1.  
40  
41       1 to: first / 2 do: [ :i |  
42         a := anArray at: i.  
43         b := anArray at: complement - i.  
44         anArray at: i put: b.  
45         anArray at: complement - i put: a.  
46       ]  
47     ].  
48     ^k  
49   )  
50  
51   initialize: size = (  
52     perm := (1 to: size).
```

```

53         timesRotated := Array new: size withAll: 0.
54         atEnd := false
55     )
56
57     makeNext = (
58         | temp remainder |
59         "Generate the next permutation. *"
60         2 to: perm length do: [ :r |
61             "Rotate the first r items to the left. *"
62             temp := perm at: 1.
63             1 to: r - 1 do: [ :i | perm at: i put: (perm at: i + 1) ].
64             perm at: r put: temp.
65
66             timesRotated at: r put: ((timesRotated at: r) + 1) % r.
67             remainder := timesRotated at: r.
68             remainder = 0 ifFalse: [ ^self ].
69
70             "After r rotations, the first r items are in their original
positions.
71             Go on rotating the first r+1 items. *"
72             ].
73
74             "We are past the final permutation. *"
75             atEnd := true.
76         )
77
78     maxPfannkuchen = (
79         | max permutation |
80         max := 0.
81         [self atEnd] whileFalse: [
82             permutation := self next.
83             max := max max: (self pfannkuchen: permutation)].
84         ^max
85     )
86
87     atEnd = (
88         ^atEnd
89     )
90
91     next = (
92         | result |
93         result := perm copy.
94         self makeNext.
95         ^result
96     )
97
98     innerBenchmarkLoop: innerIterations = (
99         | result |
100         self initialize: innerIterations.
101         result := self maxPfannkuchen.
102         ^ result = (Fannkuch expectedResult: innerIterations)
103     )
104
105     ----
106     | results |
107
108     expectedResult: problemSize = (
109         results isNil ifTrue: [
110             results := Array new: 12.
111             results at: 1 put: 0.
112

```

```
113         results at: 2 put: 1.
114         results at: 3 put: 2.
115         results at: 4 put: 4.
116         results at: 5 put: 7.
117         results at: 6 put: 10.
118         results at: 7 put: 16.
119         results at: 8 put: 22.
120         results at: 9 put: 30.
121         results at: 10 put: 38.
122         results at: 11 put: 51.
123         results at: 12 put: 65.
124     ].
125     ^ results at: problemSize
126 )
127 )
128
```

```
1 " Ported GCBench from PyPy Project: https://bitbucket.org/pypy/pypy/
src/02ea09544fc9/pypy/translator/goal/gcbench.py
2   Original comment was:
3
4 # Ported from a Java benchmark whose history is :
5 # This is adapted from a benchmark written by John Ellis and Pete Kovac
6 # of Post Communications.
7 # It was modified by Hans Boehm of Silicon Graphics.
8 #
9 #   This is no substitute for real applications.  No actual
application
10 #   is likely to behave in exactly this way.  However, this benchmark
was
11 #   designed to be more representative of real applications than other
12 #   Java GC benchmarks of which we are aware.
13 #   It attempts to model those properties of allocation requests that
14 #   are important to current GC techniques.
15 #   It is designed to be used either to obtain a single overall
performance
16 #   number, or to give a more detailed estimate of how collector
17 #   performance varies with object lifetimes.  It prints the time
18 #   required to allocate and collect balanced binary trees of various
19 #   sizes.  Smaller trees result in shorter object lifetimes.  Each
cycle
20 #   allocates roughly the same amount of memory.
21 #   Two data structures are kept around during the entire process, so
22 #   that the measured performance is representative of applications
23 #   that maintain some live in-memory data.  One of these is a tree
24 #   containing many pointers.  The other is a large array containing
25 #   double precision floating point numbers.  Both should be of
comparable
26 #   size.
27 #
28 #   The results are only really meaningful together with a
specification
29 #   of how much memory was used.  It is possible to trade memory for
30 #   better time performance.  This benchmark should be run in a 32 MB
31 #   heap, though we don't currently know how to enforce that
uniformly.
32 #
33 #   Unlike the original Ellis and Kovac benchmark, we do not attempt
34 #   measure pause times.  This facility should eventually be added
back
35 #   in.  There are several reasons for omitting it for now.  The
original
36 #   implementation depended on assumptions about the thread scheduler
37 #   that don't hold uniformly.  The results really measure both the
38 #   scheduler and GC.  Pause time measurements tend to not fit well
with
39 #   current benchmark suites.  As far as we know, none of the current
40 #   commercial Java implementations seriously attempt to minimize GC
pause
41 #   times.
42 #
43 #   Known deficiencies:
44 #       - No way to check on memory use
45 #       - No cyclic data structures
46 #       - No attempt to measure variation with object size
```



```

47 #           - Results are sensitive to locking cost, but we dont
48 #           check for proper locking
49 "
50 GCBench = (
51     |kStretchTreeDepth kLongLivedTreeDepth kArraySize kMaxTreeDepth
52     kMinTreeDepth cur_depth|
53
54     run = (
55         |temp_tree long_lived_tree array depths cur_depth t_start
t_finish|
56         kStretchTreeDepth := 18.
57         kLongLivedTreeDepth := 16.
58         kArraySize := 500000.
59         kMaxTreeDepth := 16.
60         kMinTreeDepth := 4.
61         'Garbage Collector Test' println.
62         ('Stretching memory with a binary tree of depth ' +
kStretchTreeDepth) println.
63         t_start := system time.
64         temp_tree := self make_tree: kStretchTreeDepth.
65         temp_tree := nil.
66
67         ('Creating a long-lived binary tree of depth ' +
kLongLivedTreeDepth) println.
68         long_lived_tree := Node create.
69         self populate: kLongLivedTreeDepth tree: long_lived_tree.
70
71         ('Creating a long-lived array of ' + kArraySize + ' doubles')
println.
72         array := Array new: kArraySize withAll: [(0//1)].
73         1 to: (kArraySize/2) do: [:value | array at: value put: (1 //
value)].
74         depths := Array new: ((kMaxTreeDepth - kMinTreeDepth)/ 2 + 1).
75         cur_depth := kMinTreeDepth.
76         depths doIndexes: [:value |
77             depths at: value put: cur_depth.
78             cur_depth := cur_depth + 2.
79         ].
80         self time_constructions: depths.
81
82         ((long_lived_tree == nil))
83         ifTrue: ["test failed" println.].
84         t_finish := system time.
85         ('Completed in ' + (t_finish - t_start) + ' ms.') println.
86     )
87
88     make_tree: depth = (
89         depth <= 0
90         ifTrue:
91             [^(Node create)]
92         ifFalse:
93             [^(Node create: (self make_tree: (depth - 1)) with: (self
94                 make_tree: (depth - 1)))]
95     )
96
97     time_constructions: depths = (
98         depths do: [:value | self time_construction: value].
99     )
100
101     time_construction: depth = ( |niters t_start t_finish temp_tree|
102         niters := self num_iters: depth.

```

```

103      ('Creating ' + niters + ' trees of depth ' + depth) println.
104      t_start := system time.
105      0 to: (niters-1) do: [:i |
106          temp_tree := Node create.
107          self populate: depth tree: temp_tree.
108          temp_tree := nil.].
109      t_finish := system time.
110      ('      Top down constrution took ' + (t_finish - t_start) + '
ms.') println.
111      t_start := system time.
112      0 to: (niters-1) do: [:i |
113          temp_tree := self make_tree: depth.
114          temp_tree := nil.].
115      t_finish := system time.
116      ('      Bottom up constrution took ' + (t_finish - t_start) + '
ms.') println.
117
118      )
119
120      num_iters: i = (
121          ^ (2 * (self tree_size: kStretchTreeDepth) / (self tree_size: i)).
122      )
123
124      populate: depth tree: node = (
125          depth <= 0
126              ifFalse: [
127                  depth := depth - 1.
128                  node left: Node create.
129                  node right: Node create.
130                  self populate: depth tree: (node left).
131                  self populate: depth tree: (node right).
132              ]
133      )
134
135      tree_size: i = ( |val|
136          val := 2.
137          i timesRepeat: [val := val * 2].
138          ^ (val - 1).
139      )
140 )
141

```

```
1 Node = (  
2   |left right|  
3  
4   left = (^left)  
5   left: val = (  
6     left := val.  
7   )  
8   right = (^right)  
9   right: val = (  
10    right := val.  
11  )  
12  ----  
13  create = (  
14    ^(Node create: nil with: nil).  
15  )  
16  create: l with: r = (  
17    |n|  
18    n := Node new.  
19    n left: l.  
20    n right: r.  
21    ^n.  
22  )  
23 )  
24
```

```
1 Edge = (  
2   | dest weight |  
3   dest  = ( ^ dest )  
4   weight = ( ^ weight )  
5  
6   initializeWith: destination and: w = (  
7     dest  := destination.  
8     weight := w.  
9   )  
10  
11 ----  
12  
13   newWith: dest and: weight = (  
14     ^ self new initializeWith: dest and: weight  
15   )  
16  
17 )  
18
```

```

1 GraphSearch = Benchmark (
2   | graphNodes graphMask updatingGraphMask graphVisited cost graphEdges k
firstCost |
3
4   initializeGraph: noOfNodes = (
5     | source graph totalEdges |
6     graphNodes      := Array new: noOfNodes.
7     graphMask        := Array new: noOfNodes withAll: false.
8     updatingGraphMask := Array new: noOfNodes withAll: false.
9     graphVisited      := Array new: noOfNodes withAll: false.
10    cost              := Array new: noOfNodes withAll: -1.
11
12    source := 1.
13
14    graph := Array new: noOfNodes withAll: [Vector new].
15
16    graph doIndexes: [:i |
17      | noOfEdges |
18      noOfEdges := (JenkinsRandom random
19        rem: GraphSearch MaxInitEdges - GraphSearch
20        MinEdges + 1) abs
21        + GraphSearch MinEdges.
22      1 to: noOfEdges do: [:j |
23        | nodeId weight |
24        nodeId := (JenkinsRandom random rem: noOfNodes) abs + 1.
25        weight := (JenkinsRandom random rem: (
26          GraphSearch MaxWeight - GraphSearch MinWeight + 1))
27          abs
28          + GraphSearch MinWeight.
29        (graph at: i) append: (Edge newWith: nodeId and: weight).
30        (graph at: nodeId) append: (Edge newWith: i and: weight).
31      ]
32    ].
33    totalEdges := 0.
34    graph doIndexes: [:i |
35      | noOfEdges |
36      noOfEdges := (graph at: i) size.
37      graphNodes at: i put: (Node newWith: totalEdges + 1 and: noOfEdges).
38      totalEdges := totalEdges + noOfEdges
39    ].
40    graphMask at: source put: true.
41    graphVisited at: source put: true.
42
43    graphEdges := Array new: totalEdges withAll: 0.
44
45    k := 1.
46    graph do: [:i |
47      i do: [:j |
48        graphEdges at: k put: j dest.
49        k := k + 1.
50      ]
51    ].
52    cost at: source put: 0.
53  )
54
55  innerBenchmarkLoop: innerIterations = (

```

```

56     | noOfNodes |
57     JenkinsRandom seed: 49734321.
58
59     noOfNodes := GraphSearch ExpectedNoOfNodes / 900000 * innerIterations
* innerIterations * innerIterations.
60
61     self initializeGraph: noOfNodes.
62     self breadthFirstSearch: noOfNodes.
63     ^ self verify: cost inner: innerIterations
64 )
65
66 breadthFirstSearch: noOfNodes = (
67     | stop |
68     stop := true.
69     [stop] whileTrue: [
70         stop := false.
71
72         1 to: noOfNodes do: [:tid |
73             (graphMask at: tid) ifTrue: [
74                 graphMask at: tid put: false.
75                 (graphNodes at: tid) starting
76                     to: ((graphNodes at: tid) noOfEdges + (graphNodes at: tid)
starting) - 1
77                     do: [:i |
78                         | id |
79                         id := graphEdges at: i.
80                         (graphVisited at: id) ifFalse: [
81                             cost at: id put: (cost at: tid) + 1.
82                             updatingGraphMask at: id put: true.
83                         ]
84                     ]
85                 ]
86         ].
87
88         1 to: noOfNodes do: [:tid |
89             (updatingGraphMask at: tid) ifTrue: [
90                 graphMask at: tid put: true.
91                 graphVisited at: tid put: true.
92                 stop := true.
93                 updatingGraphMask at: tid put: false
94             ]
95         ]
96     ]
97 )
98
99     verify: result inner: innerIterations = (
100     | totalCost |
101     cost length = (GraphSearch ExpectedNoOfNodes / 900000 *
innerIterations * innerIterations * innerIterations)
102     ifFalse: [ ^ false ].
103
104     totalCost := 0.
105     cost do: [:c | totalCost := totalCost + c].
106
107     cost length = GraphSearch ExpectedNoOfNodes
108     ifTrue: [
109         totalCost = GraphSearch ExpectedTotalCost ifFalse: [
110             self error: 'ERROR: the total cost obtained for ' + cost length
asString
111                 + ' nodes is ' + totalCost asString + ' while the expected
cost is '

```

```

112             + GraphSearch ExpectedTotalCost
113         ]]
114     ifFalse: [
115         firstCost == nil
116         ifTrue: [
117             firstCost := totalCost.
118             ^ true ]
119         ifFalse: [
120             ^ firstCost = totalCost
121         ]
122     ].
123
124     ^ true
125 )
126
127 ----
128
129 MinEdges = ( ^ 2 )
130 MaxInitEdges = ( ^ 4 )
131 MinWeight = ( ^ 1 )
132 MaxWeight = ( ^ 1 )
133
134 ExpectedNoOfNodes = ( ^ 3000000 )
135 ExpectedTotalCost = ( ^ 26321966 )
136
137 )

```

```
1 Node = (  
2   | starting noOfEdges |  
3  
4   starting = ( ^ starting )  
5   noOfEdges = ( ^ noOfEdges )  
6  
7   initializeWith: start and: edges = (  
8     starting := start.  
9     noOfEdges := edges.  
10  )  
11  
12  ----  
13  
14  newWith: starting and: noOfEdges = (  
15    ^ self new initializeWith: starting and: noOfEdges  
16  )  
17 )  
18
```



```

1 JenkinsRandom = (
2   ----
3   | seed |
4
5   seed: val = ( seed := val )
6
7   "Robert Jenkins' 32 bit integer hash function."
8   random = (
9     seed := ((seed      + 2127912214 "0x7ed55d16")      + (seed
as32BitUnsignedValue << 12) as32BitSignedValue) as32BitSignedValue.
10    seed := ((seed bitXor: 3345072700 "0xc761c23c") bitXor: (seed
as32BitUnsignedValue >>> 19)) as32BitSignedValue.
11    seed := ((seed      + 374761393 "0x165667B1")      + (seed
as32BitUnsignedValue << 5) as32BitSignedValue) as32BitSignedValue.
12    seed := ((seed      + 3550635116 "0xd3a2646c") bitXor: (seed
as32BitUnsignedValue << 9) as32BitSignedValue) as32BitSignedValue.
13    seed := ((seed      + 4251993797 "0xfd7046c5")      + (seed
as32BitUnsignedValue << 3) as32BitSignedValue) as32BitSignedValue.
14    seed := ((seed bitXor: 3042594569 "0xb55a4f09") bitXor: (seed
as32BitUnsignedValue >>> 16)) as32BitSignedValue.
15    ^ seed
16  )
17 )
18

```

```

1 HashIndexTable = (
2   | hashTable |
3
4   initialize = (
5     hashTable := Array new: 32 withAll: 0
6   )
7
8   at: name put: index = (
9     | slot |
10    slot := self hashSlotFor: name.
11
12    index < 255
13      ifTrue: [ hashTable at: slot put: index + 1 ]
14      ifFalse: [ hashTable at: slot put: 0 ]
15  )
16
17  remove: index = (
18    hashTable doIndexes: [:i |
19      index + 1 = (hashTable at: i)
20        ifTrue: [ hashTable at: i put: 0 ]
21        ifFalse: [
22          (hashTable at: i) > index + 1 ifTrue: [
23            hashTable at: i put: (hashTable at: i) - 1]]]
24  )
25
26  at: name = (
27    | slot |
28    slot := self hashSlotFor: name.
29
30    " subtract 1, 0 stands for empty "
31    ^ ((hashTable at: slot) & 255) - 1
32  )
33
34  hashSlotFor: element = (
35    ^ (element hashCode & 31) + 1
36  )
37
38  ----
39
40  new = ( ^ super new initialize )
41 )
42

```

```

1  Json = Benchmark (
2
3      oneTimeSetup = (
4          JsonLiteral initialize.
5      )
6
7      benchmark = (
8          ^ (JsonParser with: Json RapBenchmarkMinified) parse.
9      )
10
11      verifyResult: result = (
12          result class = ParseException ifTrue: [ result println ].
13          result isObject ifFalse: [ ^ false ].
14          (result asObject at: 'head') isObject ifFalse: [ ^ false ].
15          (result asObject at: 'operations') isArray ifFalse: [ ^ false ].
16          ^ (result asObject at: 'operations') asArray size = 156
17      )
18
19      ----
20
21      RapBenchmarkMinified = (
22          ^ ' {"head":{"requestCounter":4},"operations":[["destroy","w54"],
["set","w2",{"activeControl":"w99"}],["set","w21",
{"customVariant":"variant_navigation"}],["set","w28",
{"customVariant":"variant_selected"}],["set","w53",{"children":["w95"]}]],
["create","w95","rwt.widgets.Composite",{"parent":"w53","style":
["NONE"],"bounds":[0,0,1008,586],"children":
["w96","w97"],"tabIndex":-1,"clientArea":[0,0,1008,586]}]],
["create","w96","rwt.widgets.Label",{"parent":"w95","style":["NONE"],"bounds":
[10,30,112,26],"tabIndex":-1,"customVariant":"variant_pageHeadline","text":"Ta
bleViewer"}],["create","w97","rwt.widgets.Composite",{"parent":"w95","style":
["NONE"],"bounds":[0,61,1008,525],"children":
["w98","w99","w226","w228"],"tabIndex":-1,"clientArea":[0,0,1008,525]}]],
["create","w98","rwt.widgets.Text",{"parent":"w97","style":
["LEFT","SINGLE","BORDER"],"bounds":[10,10,988,32],"tabIndex":22,"activeKeys":
["#13","#27","#40"]}],["listen","w98",{"KeyDown":true,"Modify":true}],
["create","w99","rwt.widgets.Grid",{"parent":"w97","style":["SINGLE","BORDER"]
,"appearance":"table","indentationWidth":0,"treeColumn":-1,"markupEnabled":false
}],["create","w100","rwt.widgets.ScrollBar",{"parent":"w99","style":
["HORIZONTAL"]}],["create","w101","rwt.widgets.ScrollBar",
{"parent":"w99","style":["VERTICAL"]}],["set","w99",{"bounds":
[10,52,988,402],"children":[],"tabIndex":23,"activeKeys":["CTRL+#70","CTRL+#78
","CTRL+#82","CTRL+#89","CTRL+#83","CTRL+#71","CTRL+#69"],"cancelKeys":
["CTRL+#70","CTRL+#78","CTRL+#82","CTRL+#89","CTRL+#83","CTRL+#71","CTRL+#69"]}],
["listen","w99",
{"MouseDown":true,"MouseUp":true,"MouseDoubleClick":true,"KeyDown":true}],
["set","w99",{"itemCount":118,"itemHeight":28,"itemMetrics":
[[0,0,50,3,0,3,44],[1,50,50,53,0,53,44],[2,100,140,103,0,103,134],
[3,240,180,243,0,243,174],[4,420,50,423,0,423,44],[5,470,50,473,0,473,44]],"co
lumnCount":6,"headerHeight":35,"headerVisible":true,"linesVisible":true,"focus
Item":"w108","selection":["w108"]}],["listen","w99",
{"Selection":true,"DefaultSelection":true}],["set","w99",
{"enableCellToolTip":true}],["listen","w100",{"Selection":true}],
["set","w101",{"visibility":true}],["listen","w101",{"Selection":true}],
["create","w102","rwt.widgets.GridColumn",
{"parent":"w99","text":"Nr.", "width":50,"moveable":true}],["listen","w102",
{"Selection":true}],["create","w103","rwt.widgets.GridColumn",
{"parent":"w99","text":"Sym.", "index":1,"left":50,"width":50,"moveable":true}],

```

```

["listen","w103",{"Selection":true}],
["create","w104","rwt.widgets.GridColumn",
{"parent":"w99","text":"Name","index":2,"left":100,"width":140,"moveable":true}],
["listen","w104",{"Selection":true}],
["create","w105","rwt.widgets.GridColumn",{"parent":"w99","text":"Series","index":3,"left":240,"width":180,"moveable":true}],["listen","w105",
{"Selection":true}],["create","w106","rwt.widgets.GridColumn",
{"parent":"w99","text":"Group","index":4,"left":420,"width":50,"moveable":true}],
["listen","w106",{"Selection":true}],
["create","w107","rwt.widgets.GridColumn",{"parent":"w99","text":"Period","index":5,"left":470,"width":50,"moveable":true}],["listen","w107",
{"Selection":true}],["create","w108","rwt.widgets.GridItem",
{"parent":"w99","index":0,"texts":
["1","H","Hydrogen","Nonmetal","1","1"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w109","rwt.widgets.GridItem",
{"parent":"w99","index":1,"texts":["2","He","Helium","Noble
gas","18","1"],"cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]}],["create","w110","rwt.widgets.GridItem",
{"parent":"w99","index":2,"texts":["3","Li","Lithium","Alkali
metal","1","2"],"cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]}],["create","w111","rwt.widgets.GridItem",
{"parent":"w99","index":3,"texts":["4","Be","Beryllium","Alkaline earth
metal","2","2"],"cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]}],["create","w112","rwt.widgets.GridItem",
{"parent":"w99","index":4,"texts":
["5","B","Boron","Metalloid","13","2"],"cellBackgrounds":[null,null,null,
[156,159,153,255],null,null]}],["create","w113","rwt.widgets.GridItem",
{"parent":"w99","index":5,"texts":
["6","C","Carbon","Nonmetal","14","2"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w114","rwt.widgets.GridItem",
{"parent":"w99","index":6,"texts":
["7","N","Nitrogen","Nonmetal","15","2"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w115","rwt.widgets.GridItem",
{"parent":"w99","index":7,"texts":
["8","O","Oxygen","Nonmetal","16","2"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w116","rwt.widgets.GridItem",
{"parent":"w99","index":8,"texts":
["9","F","Fluorine","Halogen","17","2"],"cellBackgrounds":[null,null,null,
[252,233,79,255],null,null]}],["create","w117","rwt.widgets.GridItem",
{"parent":"w99","index":9,"texts":["10","Ne","Neon","Noble
gas","18","2"],"cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]}],["create","w118","rwt.widgets.GridItem",
{"parent":"w99","index":10,"texts":["11","Na","Sodium","Alkali
metal","1","3"],"cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]}],["create","w119","rwt.widgets.GridItem",
{"parent":"w99","index":11,"texts":["12","Mg","Magnesium","Alkaline earth
metal","2","3"],"cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]}],["create","w120","rwt.widgets.GridItem",
{"parent":"w99","index":12,"texts":["13","Al","Aluminium","Poor
metal","13","3"],"cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]}],["create","w121","rwt.widgets.GridItem",
{"parent":"w99","index":13,"texts":
["14","Si","Silicon","Metalloid","14","3"],"cellBackgrounds":[null,null,null,
[156,159,153,255],null,null]}],["create","w122","rwt.widgets.GridItem",
{"parent":"w99","index":14,"texts":
["15","P","Phosphorus","Nonmetal","15","3"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w123","rwt.widgets.GridItem",
{"parent":"w99","index":15,"texts":
["16","S","Sulfur","Nonmetal","16","3"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}],["create","w124","rwt.widgets.GridItem",
{"parent":"w99","index":16,"texts":

```

```

["17", "Cl", "Chlorine", "Halogen", "17", "3"], "cellBackgrounds": [null, null, null,
[252, 233, 79, 255], null, null]]], [{"create", "w125", "rwt.widgets.GridItem",
{"parent": "w99", "index": 17, "texts": ["18", "Ar", "Argon", "Noble
gas", "18", "3"], "cellBackgrounds": [null, null, null,
[114, 159, 207, 255], null, null]]}], [{"create", "w126", "rwt.widgets.GridItem",
{"parent": "w99", "index": 18, "texts": ["19", "K", "Potassium", "Alkali
metal", "1", "4"], "cellBackgrounds": [null, null, null,
[239, 41, 41, 255], null, null]]}], [{"create", "w127", "rwt.widgets.GridItem",
{"parent": "w99", "index": 19, "texts": ["20", "Ca", "Calcium", "Alkaline earth
metal", "2", "4"], "cellBackgrounds": [null, null, null,
[233, 185, 110, 255], null, null]]}], [{"create", "w128", "rwt.widgets.GridItem",
{"parent": "w99", "index": 20, "texts": ["21", "Sc", "Scandium", "Transition
metal", "3", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w129", "rwt.widgets.GridItem",
{"parent": "w99", "index": 21, "texts": ["22", "Ti", "Titanium", "Transition
metal", "4", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w130", "rwt.widgets.GridItem",
{"parent": "w99", "index": 22, "texts": ["23", "V", "Vanadium", "Transition
metal", "5", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w131", "rwt.widgets.GridItem",
{"parent": "w99", "index": 23, "texts": ["24", "Cr", "Chromium", "Transition
metal", "6", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w132", "rwt.widgets.GridItem",
{"parent": "w99", "index": 24, "texts": ["25", "Mn", "Manganese", "Transition
metal", "7", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w133", "rwt.widgets.GridItem",
{"parent": "w99", "index": 25, "texts": ["26", "Fe", "Iron", "Transition
metal", "8", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w134", "rwt.widgets.GridItem",
{"parent": "w99", "index": 26, "texts": ["27", "Co", "Cobalt", "Transition
metal", "9", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w135", "rwt.widgets.GridItem",
{"parent": "w99", "index": 27, "texts": ["28", "Ni", "Nickel", "Transition
metal", "10", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w136", "rwt.widgets.GridItem",
{"parent": "w99", "index": 28, "texts": ["29", "Cu", "Copper", "Transition
metal", "11", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w137", "rwt.widgets.GridItem",
{"parent": "w99", "index": 29, "texts": ["30", "Zn", "Zinc", "Transition
metal", "12", "4"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}], [{"create", "w138", "rwt.widgets.GridItem",
{"parent": "w99", "index": 30, "texts": ["31", "Ga", "Gallium", "Poor
metal", "13", "4"], "cellBackgrounds": [null, null, null,
[238, 238, 236, 255], null, null]]}], [{"create", "w139", "rwt.widgets.GridItem",
{"parent": "w99", "index": 31, "texts":
["32", "Ge", "Germanium", "Metalloid", "14", "4"], "cellBackgrounds":
[null, null, null, [156, 159, 153, 255], null, null]]}],
[["create", "w140", "rwt.widgets.GridItem", {"parent": "w99", "index": 32, "texts":
["33", "As", "Arsenic", "Metalloid", "15", "4"], "cellBackgrounds": [null, null, null,
[156, 159, 153, 255], null, null]]}], [{"create", "w141", "rwt.widgets.GridItem",
{"parent": "w99", "index": 33, "texts":
["34", "Se", "Selenium", "Nonmetal", "16", "4"], "cellBackgrounds": [null, null, null,
[138, 226, 52, 255], null, null]]}], [{"create", "w142", "rwt.widgets.GridItem",
{"parent": "w99", "index": 34, "texts":
["35", "Br", "Bromine", "Halogen", "17", "4"], "cellBackgrounds": [null, null, null,
[252, 233, 79, 255], null, null]]}], [{"create", "w143", "rwt.widgets.GridItem",
{"parent": "w99", "index": 35, "texts": ["36", "Kr", "Krypton", "Noble
gas", "18", "4"], "cellBackgrounds": [null, null, null,
[114, 159, 207, 255], null, null]]}], [{"create", "w144", "rwt.widgets.GridItem",
{"parent": "w99", "index": 36, "texts": ["37", "Rb", "Rubidium", "Alkali
metal", "1", "5"], "cellBackgrounds": [null, null, null,

```

```

[239,41,41,255],null,null]]},["create","w145","rwt.widgets.GridItem",
{"parent":"w99","index":37,"texts":["38","Sr","Strontium","Alkaline earth
metal","2","5"],"cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]]},["create","w146","rwt.widgets.GridItem",
{"parent":"w99","index":38,"texts":["39","Y","Yttrium","Transition
metal","3","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w147","rwt.widgets.GridItem",
{"parent":"w99","index":39,"texts":["40","Zr","Zirconium","Transition
metal","4","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w148","rwt.widgets.GridItem",
{"parent":"w99","index":40,"texts":["41","Nb","Niobium","Transition
metal","5","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w149","rwt.widgets.GridItem",
{"parent":"w99","index":41,"texts":["42","Mo","Molybdenum","Transition
metal","6","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w150","rwt.widgets.GridItem",
{"parent":"w99","index":42,"texts":["43","Tc","Technetium","Transition
metal","7","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w151","rwt.widgets.GridItem",
{"parent":"w99","index":43,"texts":["44","Ru","Ruthenium","Transition
metal","8","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w152","rwt.widgets.GridItem",
{"parent":"w99","index":44,"texts":["45","Rh","Rhodium","Transition
metal","9","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w153","rwt.widgets.GridItem",
{"parent":"w99","index":45,"texts":["46","Pd","Palladium","Transition
metal","10","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w154","rwt.widgets.GridItem",
{"parent":"w99","index":46,"texts":["47","Ag","Silver","Transition
metal","11","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w155","rwt.widgets.GridItem",
{"parent":"w99","index":47,"texts":["48","Cd","Cadmium","Transition
metal","12","5"],"cellBackgrounds":[null,null,null,
[252,175,62,255],null,null]]},["create","w156","rwt.widgets.GridItem",
{"parent":"w99","index":48,"texts":["49","In","Indium","Poor
metal","13","5"],"cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]},["create","w157","rwt.widgets.GridItem",
{"parent":"w99","index":49,"texts":["50","Sn","Tin","Poor
metal","14","5"],"cellBackgrounds":[null,null,null,
[238,238,236,255],null,null]]},["create","w158","rwt.widgets.GridItem",
{"parent":"w99","index":50,"texts":
["51","Sb","Antimony","Metalloid","15","5"],"cellBackgrounds":[null,null,null,
[156,159,153,255],null,null]]},["create","w159","rwt.widgets.GridItem",
{"parent":"w99","index":51,"texts":
["52","Te","Tellurium","Metalloid","16","5"],"cellBackgrounds":
[null,null,null,[156,159,153,255],null,null]]},
["create","w160","rwt.widgets.GridItem",{ "parent":"w99","index":52,"texts":
["53","I","Iodine","Halogen","17","5"],"cellBackgrounds":[null,null,null,
[252,233,79,255],null,null]]},["create","w161","rwt.widgets.GridItem",
{"parent":"w99","index":53,"texts":["54","Xe","Xenon","Noble
gas","18","5"],"cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]]},["create","w162","rwt.widgets.GridItem",
{"parent":"w99","index":54,"texts":["55","Cs","Caesium","Alkali
metal","1","6"],"cellBackgrounds":[null,null,null,
[239,41,41,255],null,null]]},["create","w163","rwt.widgets.GridItem",
{"parent":"w99","index":55,"texts":["56","Ba","Barium","Alkaline earth
metal","2","6"],"cellBackgrounds":[null,null,null,
[233,185,110,255],null,null]]},["create","w164","rwt.widgets.GridItem",
{"parent":"w99","index":56,"texts":
["57","La","Lanthanum","Lanthanide","3","6"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]},

```

```

["create", "w165", "rwt.widgets.GridItem", {"parent": "w99", "index": 57, "texts":
["58", "Ce", "Cerium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w166", "rwt.widgets.GridItem",
{"parent": "w99", "index": 58, "texts":
["59", "Pr", "Praseodymium", "Lanthanide", "3", "6"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]}],
["create", "w167", "rwt.widgets.GridItem", {"parent": "w99", "index": 59, "texts":
["60", "Nd", "Neodymium", "Lanthanide", "3", "6"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]}],
["create", "w168", "rwt.widgets.GridItem", {"parent": "w99", "index": 60, "texts":
["61", "Pm", "Promethium", "Lanthanide", "3", "6"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]}],
["create", "w169", "rwt.widgets.GridItem", {"parent": "w99", "index": 61, "texts":
["62", "Sm", "Samarium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w170", "rwt.widgets.GridItem",
{"parent": "w99", "index": 62, "texts":
["63", "Eu", "Europium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w171", "rwt.widgets.GridItem",
{"parent": "w99", "index": 63, "texts":
["64", "Gd", "Gadolinium", "Lanthanide", "3", "6"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]}],
["create", "w172", "rwt.widgets.GridItem", {"parent": "w99", "index": 64, "texts":
["65", "Tb", "Terbium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w173", "rwt.widgets.GridItem",
{"parent": "w99", "index": 65, "texts":
["66", "Dy", "Dysprosium", "Lanthanide", "3", "6"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]}],
["create", "w174", "rwt.widgets.GridItem", {"parent": "w99", "index": 66, "texts":
["67", "Ho", "Holmium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w175", "rwt.widgets.GridItem",
{"parent": "w99", "index": 67, "texts":
["68", "Er", "Erbium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w176", "rwt.widgets.GridItem",
{"parent": "w99", "index": 68, "texts":
["69", "Tm", "Thulium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w177", "rwt.widgets.GridItem",
{"parent": "w99", "index": 69, "texts":
["70", "Yb", "Ytterbium", "Lanthanide", "3", "6"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]}],
["create", "w178", "rwt.widgets.GridItem", {"parent": "w99", "index": 70, "texts":
["71", "Lu", "Lutetium", "Lanthanide", "3", "6"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]}], [{"create", "w179", "rwt.widgets.GridItem",
{"parent": "w99", "index": 71, "texts": ["72", "Hf", "Hafnium", "Transition
metal", "4", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]}], [{"create", "w180", "rwt.widgets.GridItem",
{"parent": "w99", "index": 72, "texts": ["73", "Ta", "Tantalum", "Transition
metal", "5", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]}], [{"create", "w181", "rwt.widgets.GridItem",
{"parent": "w99", "index": 73, "texts": ["74", "W", "Tungsten", "Transition
metal", "6", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]}], [{"create", "w182", "rwt.widgets.GridItem",
{"parent": "w99", "index": 74, "texts": ["75", "Re", "Rhenium", "Transition
metal", "7", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]}], [{"create", "w183", "rwt.widgets.GridItem",
{"parent": "w99", "index": 75, "texts": ["76", "Os", "Osmium", "Transition
metal", "8", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]}], [{"create", "w184", "rwt.widgets.GridItem",
{"parent": "w99", "index": 76, "texts": ["77", "Ir", "Iridium", "Transition
metal", "9", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]}], [{"create", "w185", "rwt.widgets.GridItem",
{"parent": "w99", "index": 77, "texts": ["78", "Pt", "Platinum", "Transition

```

```

metal", "10", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]], [{"create", "w186", "rwt.widgets.GridItem",
{"parent": "w99", "index": 78, "texts": ["79", "Au", "Gold", "Transition
metal", "11", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}, {"create", "w187", "rwt.widgets.GridItem",
{"parent": "w99", "index": 79, "texts": ["80", "Hg", "Mercury", "Transition
metal", "12", "6"], "cellBackgrounds": [null, null, null,
[252, 175, 62, 255], null, null]]}, {"create", "w188", "rwt.widgets.GridItem",
{"parent": "w99", "index": 80, "texts": ["81", "Tl", "Thallium", "Poor
metal", "13", "6"], "cellBackgrounds": [null, null, null,
[238, 238, 236, 255], null, null]]}, {"create", "w189", "rwt.widgets.GridItem",
{"parent": "w99", "index": 81, "texts": ["82", "Pb", "Lead", "Poor
metal", "14", "6"], "cellBackgrounds": [null, null, null,
[238, 238, 236, 255], null, null]]}, {"create", "w190", "rwt.widgets.GridItem",
{"parent": "w99", "index": 82, "texts": ["83", "Bi", "Bismuth", "Poor
metal", "15", "6"], "cellBackgrounds": [null, null, null,
[238, 238, 236, 255], null, null]]}, {"create", "w191", "rwt.widgets.GridItem",
{"parent": "w99", "index": 83, "texts":
["84", "Po", "Polonium", "Metalloid", "16", "6"], "cellBackgrounds": [null, null, null,
[156, 159, 153, 255], null, null]]}, {"create", "w192", "rwt.widgets.GridItem",
{"parent": "w99", "index": 84, "texts":
["85", "At", "Astatine", "Halogen", "17", "6"], "cellBackgrounds": [null, null, null,
[252, 233, 79, 255], null, null]]}, {"create", "w193", "rwt.widgets.GridItem",
{"parent": "w99", "index": 85, "texts": ["86", "Rn", "Radon", "Noble
gas", "18", "6"], "cellBackgrounds": [null, null, null,
[114, 159, 207, 255], null, null]]}, {"create", "w194", "rwt.widgets.GridItem",
{"parent": "w99", "index": 86, "texts": ["87", "Fr", "Francium", "Alkali
metal", "1", "7"], "cellBackgrounds": [null, null, null,
[239, 41, 41, 255], null, null]]}, {"create", "w195", "rwt.widgets.GridItem",
{"parent": "w99", "index": 87, "texts": ["88", "Ra", "Radium", "Alkaline earth
metal", "2", "7"], "cellBackgrounds": [null, null, null,
[233, 185, 110, 255], null, null]]}, {"create", "w196", "rwt.widgets.GridItem",
{"parent": "w99", "index": 88, "texts":
["89", "Ac", "Actinium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w197", "rwt.widgets.GridItem",
{"parent": "w99", "index": 89, "texts":
["90", "Th", "Thorium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w198", "rwt.widgets.GridItem",
{"parent": "w99", "index": 90, "texts":
["91", "Pa", "Protactinium", "Actinide", "3", "7"], "cellBackgrounds":
[null, null, null, [173, 127, 168, 255], null, null]]}, {"create", "w199", "rwt.widgets.GridItem", {"parent": "w99", "index": 91, "texts":
["92", "U", "Uranium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w200", "rwt.widgets.GridItem",
{"parent": "w99", "index": 92, "texts":
["93", "Np", "Neptunium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w201", "rwt.widgets.GridItem",
{"parent": "w99", "index": 93, "texts":
["94", "Pu", "Plutonium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w202", "rwt.widgets.GridItem",
{"parent": "w99", "index": 94, "texts":
["95", "Am", "Americium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w203", "rwt.widgets.GridItem",
{"parent": "w99", "index": 95, "texts":
["96", "Cm", "Curium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w204", "rwt.widgets.GridItem",
{"parent": "w99", "index": 96, "texts":
["97", "Bk", "Berkelium", "Actinide", "3", "7"], "cellBackgrounds": [null, null, null,
[173, 127, 168, 255], null, null]]}, {"create", "w205", "rwt.widgets.GridItem",
{"parent": "w99", "index": 97, "texts":
["98", "Cf", "Californium", "Actinide", "3", "7"], "cellBackgrounds":

```



```

[null,null,null,[173,127,168,255],null,null]]},
["create","w206","rwt.widgets.GridItem",{ "parent":"w99","index":98,"texts":
["99","Es","Einsteinium","Actinide","3","7"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]},
["create","w207","rwt.widgets.GridItem",{ "parent":"w99","index":99,"texts":
["100","Fm","Fermium","Actinide","3","7"],"cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]},["create","w208","rwt.widgets.GridItem",
{"parent":"w99","index":100,"texts":
["101","Md","Mendelevium","Actinide","3","7"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]},
["create","w209","rwt.widgets.GridItem",{ "parent":"w99","index":101,"texts":
["102","No","Nobelium","Actinide","3","7"],"cellBackgrounds":[null,null,null,
[173,127,168,255],null,null]]},["create","w210","rwt.widgets.GridItem",
{"parent":"w99","index":102,"texts":
["103","Lr","Lawrencium","Actinide","3","7"],"cellBackgrounds":
[null,null,null,[173,127,168,255],null,null]]},
["create","w211","rwt.widgets.GridItem",{ "parent":"w99","index":103,"texts":
["104","Rf","Rutherfordium","Transition metal","4","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w212","rwt.widgets.GridItem",{ "parent":"w99","index":104,"texts":
["105","Db","Dubnium","Transition metal","5","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w213","rwt.widgets.GridItem",{ "parent":"w99","index":105,"texts":
["106","Sg","Seaborgium","Transition metal","6","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w214","rwt.widgets.GridItem",{ "parent":"w99","index":106,"texts":
["107","Bh","Bohrium","Transition metal","7","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w215","rwt.widgets.GridItem",{ "parent":"w99","index":107,"texts":
["108","Hs","Hassium","Transition metal","8","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w216","rwt.widgets.GridItem",{ "parent":"w99","index":108,"texts":
["109","Mt","Meitnerium","Transition metal","9","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w217","rwt.widgets.GridItem",{ "parent":"w99","index":109,"texts":
["110","Ds","Darmstadtium","Transition metal","10","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w218","rwt.widgets.GridItem",{ "parent":"w99","index":110,"texts":
["111","Rg","Roentgenium","Transition metal","11","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w219","rwt.widgets.GridItem",{ "parent":"w99","index":111,"texts":
["112","Uub","Ununbium","Transition metal","12","7"],"cellBackgrounds":
[null,null,null,[252,175,62,255],null,null]]},
["create","w220","rwt.widgets.GridItem",{ "parent":"w99","index":112,"texts":
["113","Uut","Ununtrium","Poor metal","13","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]},
["create","w221","rwt.widgets.GridItem",{ "parent":"w99","index":113,"texts":
["114","Uuq","Ununquadium","Poor metal","14","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]},
["create","w222","rwt.widgets.GridItem",{ "parent":"w99","index":114,"texts":
["115","Uup","Ununpentium","Poor metal","15","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]},
["create","w223","rwt.widgets.GridItem",{ "parent":"w99","index":115,"texts":
["116","Uuh","Ununhexium","Poor metal","16","7"],"cellBackgrounds":
[null,null,null,[238,238,236,255],null,null]]},
["create","w224","rwt.widgets.GridItem",{ "parent":"w99","index":116,"texts":
["117","Uus","Ununseptium","Halogen","17","7"],"cellBackgrounds":
[null,null,null,[252,233,79,255],null,null]]},
["create","w225","rwt.widgets.GridItem",{ "parent":"w99","index":117,"texts":
["118","Uuo","Ununoctium","Noble gas","18","7"],"cellBackgrounds":
[null,null,null,[114,159,207,255],null,null]]},

```

```

["create","w226","rwt.widgets.Composite",{ "parent":"w97","style":
["BORDER"],"bounds":[10,464,988,25],"children":
["w227"],"tabIndex":-1,"clientArea":[0,0,986,23]}],
["create","w227","rwt.widgets.Label",{ "parent":"w226","style":
["NONE"],"bounds":[10,10,966,3],"tabIndex":-1,"text":"Hydrogen (H)"}],
["create","w228","rwt.widgets.Label",{ "parent":"w97","style":
["WRAP"],"bounds":[10,499,988,16],"tabIndex":-1,"foreground":
[150,150,150,255],"font":[["Verdana","Lucida Sans","Arial","Helvetica","sans-
serif"],10,false,false],"text":"Shortcuts: [CTRL+F] - Filter | Sort by:
[CTRL+R] - Number, [CTRL+Y] - Symbol, [CTRL+N] - Name, [CTRL+S] - Series,
[CTRL+G] - Group, [CTRL+E] - Period"}],["set","w1",{ "focusControl":"w99"}],
["call","rwt.client.BrowserNavigation","addToHistory",{ "entries":
[["tableviewer","TableViewer"]]}]]}'
23      )
24 )

```

```
1 JsonArray = JsonValue (
2   | values |
3
4   initialize = ( values := Vector new )
5
6   addInteger: value = (
7     values append: (JsonValue integer: value)
8   )
9
10  addDouble: value = (
11    values append: (JsonValue double: value)
12  )
13
14  addBoolean: value = (
15    values append: (JsonValue boolean: value)
16  )
17
18  addString: value = (
19    values append: (JsonValue string: value)
20  )
21
22  add: value = (
23    value ifNil: [ self error: 'value is null' ].
24    values append: value
25  )
26
27  at: index putInteger: value = (
28    values at: index put: (JsonValue integer: value)
29  )
30
31  at: index putDouble: value = (
32    values at: index put: (JsonValue double: value)
33  )
34
35  at: index putBoolean: value = (
36    values at: index put: (JsonValue boolean: value)
37  )
38
39  at: index putString: value = (
40    values at: index put: (JsonValue string: value)
41  )
42
43  at: index put: value = (
44    value ifNil: [ self error: 'value is null' ].
45    values at: index put: value
46  )
47
48  removeAt: index = (
49    values removeAt: index
50  )
51
52  size = (
53    ^ values size
54  )
55
56  isEmpty = (
57    ^ values isEmpty
58  )
```

```

59
60   at: index = (
61     ^ values at: index
62   )
63
64   values = (
65     ^ values
66   )
67
68   isArray = (
69     ^ true
70   )
71
72   asArray = (
73     ^ self
74   )
75
76   hashCode = (
77     ^ values hashCode
78   )
79
80   = other = (
81     self == other ifTrue: [ ^ true ].
82     other == nil ifTrue: [ ^ false ].
83     self class == other class ifFalse: [ ^ false ].
84
85     ^ values = other values
86   )
87
88   ----
89
90   new = ( ^ super new initialize )
91
92   readFrom: string = (
93     ^ (JsonValue readFrom: string) asArray
94   )
95 )
96

```

```

1 JsonLiteral = JsonValue (
2   | value isNull isTrue isFalse |
3
4   initializeWith: val = (
5     value    := val.
6     isNull   := 'null'  = val.
7     isTrue   := 'true'  = val.
8     isFalse  := 'false' = val.
9   )
10
11  asString  = ( ^ value )
12  hashCode = ( ^ value hashCode )
13  isNull    = ( ^ isNull )
14  isTrue    = ( ^ isTrue )
15  isFalse   = ( ^ isFalse )
16  isBoolean = ( ^ isTrue || isFalse )
17
18  asBoolean = (
19    isNull ifTrue: [ ^ super asBoolean ]
20    ifFalse: [ ^ isTrue ]
21  )
22
23  = other = (
24    self == other ifTrue: [ ^ true ].
25    other == nil ifTrue: [ ^ false ].
26    self class == other class ifFalse: [ ^ false ].
27
28    ^ value = other asString
29  )
30
31  ----
32
33  | NULL TRUE FALSE |
34
35  initialize = (
36    NULL := self new initializeWith: 'null'.
37    TRUE  := self new initializeWith: 'true'.
38    FALSE := self new initializeWith: 'false'.
39  )
40
41  NULL = ( ^ NULL )
42  TRUE  = ( ^ TRUE )
43  FALSE = ( ^ FALSE )
44 )
45

```

```

1 JsonNumber = JsonValue (
2   | string |
3
4   initializeWith: str = ( string := str )
5
6   asString = ( ^ string )
7   isNumber = ( ^ true )
8
9   asInteger = (
10    ^ Integer fromString: string
11  )
12
13  asDouble = (
14    ^ Double fromString: string
15  )
16
17  hashCode = (
18    ^ string hashCode
19  )
20
21  = other = (
22    self == other ifTrue: [ ^ true ].
23    other == nil ifTrue: [ ^ false ].
24
25    self class == other class ifFalse: [ ^ false ].
26    ^ string = other asString
27  )
28
29  ----
30
31  new: string = (
32    string ifNil: [ self error: 'string is null' ].
33    ^ self new initializeWith: string
34  )
35 )
36

```

```

1 JsonObject = JsonValue (
2   | names values table |
3
4   initialize = (
5     names  := Vector new.
6     values := Vector new.
7     table  := HashIndexTable new
8   )
9
10  add: name withNumber: value = (
11    self add: name with: (JsonValue number: value)
12  )
13
14  add: name withBoolean: value = (
15    self add: name with: (JsonValue boolean: value)
16  )
17
18  add: name withString: value = (
19    self add: name with: (JsonValue string: value)
20  )
21
22  add: name with: aJsonValue = (
23    name ifNil: [ self error: 'name is null' ].
24    aJsonValue ifNil: [ self error: 'aJsonValue is null' ].
25
26    names append: name.
27    values append: aJsonValue.
28    table at: name put: names size.
29  )
30
31  at: name putNumber: value = (
32    self at: name put: (JsonValue number: value)
33  )
34
35  at: name putBoolean: value = (
36    self at: name put: (JsonValue boolean: value)
37  )
38
39  at: name putString: value = (
40    self at: name put: (JsonValue string: value)
41  )
42
43  at: name put: aJsonValue = (
44    | idx |
45    name ifNil: [ self error: 'name is null' ].
46    aJsonValue ifNil: [ self error: 'aJsonValue is null' ].
47
48    idx := self indexOf: name.
49    idx <> -1
50    ifTrue: [values at: idx put: aJsonValue]
51    ifFalse: [
52      names append: name.
53      values append: aJsonValue.
54      table at: name put: names size.
55    ].
56  )
57
58  remove: name = (

```

```

59     | idx |
60     name ifNil: [ self error: 'name is null' ].
61     idx := self indexOf: name.
62     idx = -1 ifFalse: [
63         table remove: name.
64         names remove: idx.
65         values remove: idx.
66     ].
67 )
68
69 at: name = (
70     | idx |
71     name ifNil: [ self error: 'name is null' ].
72     idx := self indexOf: name.
73     idx = -1
74         ifTrue: [ ^ nil ]
75         ifFalse: [ ^ values at: idx ]
76 )
77
78 at: name asIntegerWith: default = (
79     | value |
80     value := self at: name.
81     value
82         ifNil: [ ^ default ]
83         ifNotNil: [ ^ value asInteger ]
84 )
85
86 at: name asDoubleWith: default = (
87     | value |
88     value := self at: name.
89     value
90         ifNil: [ ^ default ]
91         ifNotNil: [ ^ value asDouble ]
92 )
93
94 at: name asBooleanWith: default = (
95     | value |
96     value := self at: name.
97     value
98         ifNil: [ ^ default ]
99         ifNotNil: [ ^ value asBoolean ]
100 )
101
102 at: name asStringWith: default = (
103     | value |
104     value := self at: name.
105     value
106         ifNil: [ ^ default ]
107         ifNotNil: [ ^ value asString ]
108 )
109
110 size = (
111     ^ names size
112 )
113
114 isEmpty = (
115     ^ names isEmpty
116 )
117
118 names = (
119     ^ names

```



```

120 )
121
122 "TODO:
123 @Override
124 void write( final JsonWriter writer ) throws IOException {
125     writer.writeObjectOpen();
126     Iterator<String> namesIterator = names.iterator();
127     Iterator<JsonValue> valuesIterator = values.iterator();
128     boolean first = true;
129     while( namesIterator.hasNext() ) {
130         if( !first ) {
131             writer.writeObjectSeparator();
132         }
133         writer.writeMemberName( namesIterator.next() );
134         writer.writeMemberSeparator();
135         valuesIterator.next().write( writer );
136         first = false;
137     }
138     writer.writeObjectClose();
139 }"
140
141 isObject = ( ^ true )
142 asObject = ( ^ self )
143
144 hashCode = (
145     | result |
146     result := 1.
147     result := 31 * result + names hashCode.
148     result := 31 * result + values hashCode.
149     ^ result
150 )
151
152 = other = (
153     self == other ifTrue: [ ^ true ].
154     self == nil ifTrue: [ ^ false ].
155     self class == other class ifFalse: [ ^ false ].
156
157     ^ names = other names && values = other values
158 )
159
160 indexOf: name = (
161     | idx |
162     idx := table at: name.
163     idx <> -1 && (name = (names at: idx)) ifTrue: [ ^ idx ].
164     ^ names lastIndexOf: name
165 )
166
167 updateHashIndex = (
168     names doIndexes: [: i |
169         table add: (names at: i) put: i
170     ]
171 )
172
173 ----
174
175 new = ( ^ super new initialize )
176
177 readFrom: string = (
178     ^ (JsonValue readFrom: string) asObject
179 )
180 )

```



```

1 " This is a rough port of minijson from Java.
2  TODO: add details (com.eclipsesource.json) "
3
4 JsonParser = (
5   | input index line column current captureBuffer captureStart
exceptionBlock |
6
7   initializeWith: string = (
8     input := string.
9     index := 0.
10    line  := 1.
11    column := 0.
12    current := nil.
13    captureBuffer := ''.
14  )
15
16  parse = (
17    | result |
18    exceptionBlock := [:ex | ^ ex ].
19    self read.
20    self skipWhiteSpace.
21    result := self readValue.
22    self skipWhiteSpace.
23    self isEndOfText ifFalse: [ self error: 'Unexpected character' ].
24    ^ result
25  )
26
27  readValue = (
28    current = 'n' ifTrue: [ ^ self readNull ].
29    current = 't' ifTrue: [ ^ self readTrue ].
30    current = 'f' ifTrue: [ ^ self readFalse ].
31    current = '"' ifTrue: [ ^ self readString ].
32    current = '[' ifTrue: [ ^ self readArray ].
33    current = '{' ifTrue: [ ^ self readObject ].
34
35    "Is this really the best way to write this?, or better #or:?,
36    but with all the nesting, it's just ugly."
37    current = '-' ifTrue: [ ^ self readNumber ].
38    current = '0' ifTrue: [ ^ self readNumber ].
39    current = '1' ifTrue: [ ^ self readNumber ].
40    current = '2' ifTrue: [ ^ self readNumber ].
41    current = '3' ifTrue: [ ^ self readNumber ].
42    current = '4' ifTrue: [ ^ self readNumber ].
43    current = '5' ifTrue: [ ^ self readNumber ].
44    current = '6' ifTrue: [ ^ self readNumber ].
45    current = '7' ifTrue: [ ^ self readNumber ].
46    current = '8' ifTrue: [ ^ self readNumber ].
47    current = '9' ifTrue: [ ^ self readNumber ].
48
49    "else"
50    self expected: 'value'
51  )
52
53  readArrayElement: array = (
54    self skipWhiteSpace.
55    array add: self readValue.
56    self skipWhiteSpace.
57  )

```

```

58
59 readArray = (
60     | array |
61     self read.
62     array := JsonArray new.
63
64     "Array might be empty"
65     self skipWhiteSpace.
66     (self readChar: ']') ifTrue: [
67         ^ array
68     ].
69
70     self readArrayElement: array.
71     [self readChar: ','] whileTrue: [
72         self readArrayElement: array.
73     ].
74
75     (self readChar: ']') ifFalse: [
76         self expected: '"', ' ' or ']'
77     ].
78     ^ array
79 )
80
81 readObjectKeyValuePair: object = (
82     | name |
83     self skipWhiteSpace.
84     name := self readName.
85     self skipWhiteSpace.
86
87     (self readChar: ':') ifFalse: [ self expected: ':' ].
88
89     self skipWhiteSpace.
90
91     object add: name with: self readValue.
92
93     self skipWhiteSpace.
94 )
95
96 readObject = (
97     | object |
98     self read.
99     object := JsonObject new.
100     self skipWhiteSpace.
101
102     (self readChar: '}') ifTrue: [
103         ^ object
104     ].
105
106     self readObjectKeyValuePair: object.
107     [self readChar: ','] whileTrue: [
108         self readObjectKeyValuePair: object.
109     ].
110
111     (self readChar: '}') ifFalse: [
112         self expected: '"', ' ' or '}'
113     ].
114
115     ^ object
116 )
117
118 readName = (

```

```

119     current = '' ifFalse: [ self expected: 'name' ].
120     ^ self readStringInternal
121 )
122
123 readNull = (
124     self read.
125     self readRequiredChar: 'u'.
126     self readRequiredChar: 'l'.
127     self readRequiredChar: 'l'.
128     ^ JsonLiteral NULL
129 )
130
131 readTrue = (
132     self read.
133     self readRequiredChar: 'r'.
134     self readRequiredChar: 'u'.
135     self readRequiredChar: 'e'.
136     ^ JsonLiteral TRUE
137 )
138
139 readFalse = (
140     self read.
141     self readRequiredChar: 'a'.
142     self readRequiredChar: 'l'.
143     self readRequiredChar: 's'.
144     self readRequiredChar: 'e'.
145     ^ JsonLiteral FALSE
146 )
147
148 readRequiredChar: ch = (
149     (self readChar: ch) ifFalse: [
150         self expected: 'character: ' + ch
151     ]
152 )
153
154 readString = (
155     ^ JsonString new: self readStringInternal
156 )
157
158 readStringInternal = (
159     | string |
160     self read.
161     self startCapture.
162
163     [current = ''] whileFalse: [
164         current = '\\' ifTrue: [
165             self pauseCapture.
166             self readEscape.
167             self startCapture.
168         ] ifFalse: [
169             "if (current < 0x20) { throw expected('valid string
character'); }"
170             "we currently don't have a way to get the ordinal value for a
character"
171             "} else {"
172             self read.
173         ]
174     ].
175     string := self endCapture.
176     self read.
177     ^ string

```

```

178 )
179
180 readEscapeChar = (
181     current = '"' ifTrue: [ ^ '"' ].
182     current = '/' ifTrue: [ ^ '/' ].
183     current = '\\' ifTrue: [ ^ '\\' ].
184
185     current = 'b' ifTrue: [ ^ '\b' ].
186     current = 'f' ifTrue: [ ^ '\f' ].
187     current = 'n' ifTrue: [ ^ '\n' ].
188     current = 'r' ifTrue: [ ^ '\r' ].
189     current = 't' ifTrue: [ ^ '\t' ].
190
191     "TODO: SOM doesn't have a way to create unicode characters."
192     self expected: 'valid escape sequence. note, some are not supported'
193 )
194
195 readEscape = (
196     self read.
197     captureBuffer := captureBuffer concatenate: self readEscapeChar.
198     self read
199 )
200
201 readNumber = (
202     | firstDigit |
203     self startCapture.
204     self readChar: '-'.
205     firstDigit := current.
206
207     self readDigit ifFalse: [ self expected: 'digit' ].
208     firstDigit <> '0' ifTrue: [ [self readDigit] whileTrue: []].
209
210     self readFraction.
211     self readExponent.
212     ^ JsonNumber new: self endCapture
213 )
214
215 readFraction = (
216     (self readChar: '.') ifFalse: [ ^ false ].
217
218     self readDigit ifFalse: [ self expected: 'digit' ].
219
220     [self readDigit] whileTrue: [].
221
222     ^ true
223 )
224
225 readExponent = (
226     ((self readChar: 'e') not and: [
227         (self readChar: 'E') not]) ifTrue: [ ^ false ].
228
229
230     (self readChar: '+') ifFalse: [ self readChar: '-' ].
231
232     self readDigit ifFalse: [ self expected: 'digit' ].
233
234     [self readDigit] whileTrue: [].
235
236     ^ true
237 )
238

```

```

239 readChar: ch = (
240     current = ch ifFalse: [ ^ false ].
241     self read.
242     ^ true
243 )
244
245 readDigit = (
246     self isDigit ifFalse: [ ^ false ].
247     self read.
248     ^ true
249 )
250
251 skipWhiteSpace = (
252     [ self isWhiteSpace ]
253     whileTrue:
254         [ self read ].
255 )
256
257 read = ("TODO: this is probably broken"
258     current = '\n' ifTrue: [
259         line := line + 1.
260         column := 0.
261     ].
262     index := index + 1.
263     column := column + 1.
264
265     input ifNil: [ self error:'input nil'].
266     index <= input length
267         ifTrue: [ current := input charAt: index ]
268         ifFalse: [ current := nil ]
269 )
270
271
272 startCapture = (
273     captureStart := index
274 )
275
276 pauseCapture = (
277     captureBuffer := captureBuffer concatenate: (
278         input substringFrom: captureStart to: index - 1).
279     captureStart := -1
280 )
281
282 endCapture = (
283     | captured |
284     '' = captureBuffer
285         ifTrue: [ captured := input substringFrom: captureStart to: index
286 - 1 ]
287         ifFalse: [
288             self pauseCapture.
289             captured := captureBuffer.
290             captureBuffer := '' ].
291     captureStart := -1.
292     ^ captured
293 )
294
295 expected: expected = (
296     self isEndOfText ifTrue: [
297         self error: 'Unexpected end of input, expected ' + expected asString
298     ].

```

```

299     self error: 'Expected ' + expected
300 )
301
302 error: message = (
303     'error:' print.
304     message print.
305     ':' print.
306     line print.
307     column println.
308     exceptionBlock value: (ParseException with: message at: index
309                             line: line      column: column )
310 )
311
312 isWhiteSpace = (
313     current = ' ' ifTrue: [^ true].
314     current = '\t' ifTrue: [^ true].
315     current = '\n' ifTrue: [^ true].
316     current = '\r' ifTrue: [^ true].
317     ^ false
318 )
319
320 isDigit = (
321     current = '0' ifTrue: [^ true].
322     current = '1' ifTrue: [^ true].
323     current = '2' ifTrue: [^ true].
324     current = '3' ifTrue: [^ true].
325     current = '4' ifTrue: [^ true].
326     current = '5' ifTrue: [^ true].
327     current = '6' ifTrue: [^ true].
328     current = '7' ifTrue: [^ true].
329     current = '8' ifTrue: [^ true].
330     current = '9' ifTrue: [^ true].
331     ^ false
332 )
333
334 isEndOfText = (
335     ^ current isNil
336 )
337
338 ----
339
340 with: aJsonString = (
341     ^ self new initializeWith: aJsonString
342 )
343 )
344

```



```

1 JsonSmall = Benchmark (
2
3   oneTimeSetup = (
4     JsonLiteral initialize.
5   )
6
7   benchmark = (
8     ^ (JsonParser with: JsonSmall RapBenchmark) parse.
9   )
10
11   verifyResult: result = (
12     result class = ParseException ifTrue: [ result println ].
13     result isObject ifFalse: [ ^ false].
14     (result asObject at: 'head') isObject ifFalse: [ ^ false].
15     (result asObject at: 'operations') isArray ifFalse: [ ^ false].
16     ^ (result asObject at: 'operations') asArray size = 41
17   )
18
19   ----
20
21   RapBenchmark = (
22     ^ '{
23       "head":{"requestCounter":4},
24       "operations":[["destroy","w54"],["set","w2",
25         {"activeControl":"w99"}],
26         ["set","w21",{"customVariant":"variant_navigation"}],
27         ["set","w28",{"customVariant":"variant_selected"}],
28         ["set","w53",{"children":["w95"}]],
29         ["create","w95","rwt.widgets.Composite",
30         {"parent":"w53","style":["NONE"],"bounds":[0,0,1008,586],"children":
31         ["w96","w97"],"tabIndex":-1,"clientArea":[0,0,1008,586]}],
32         ["create","w96","rwt.widgets.Label",{ "parent":"w95","style":
33         ["NONE"],"bounds":[10,30,112,26],"tabIndex":-1,"customVariant":"variant_pageHe
34         adline","text":"TableViewer"}],
35         ["create","w97","rwt.widgets.Composite",
36         {"parent":"w95","style":["NONE"],"bounds":[0,61,1008,525],"children":
37         ["w98","w99","w226","w228"],"tabIndex":-1,"clientArea":[0,0,1008,525]}],
38         ["create","w98","rwt.widgets.Text",{ "parent":"w97","style":
39         ["LEFT","SINGLE","BORDER"],"bounds":[10,10,988,32],"tabIndex":22,"activeKeys":
40         ["#13","#27","#40"]}], ' +
41         '["listen","w98",{ "KeyDown":true,"Modify":true}],
42         ["create","w99","rwt.widgets.Grid",{ "parent":"w97","style":["
43         SINGLE","BORDER"],"appearance":"table","indentationWidth":0,"treeColumn":-1,"mar
44         kupEnabled":false}],
45         ["create","w100","rwt.widgets.ScrollBar",
46         {"parent":"w99","style":["HORIZONTAL"]}],
47         ["create","w101","rwt.widgets.ScrollBar",
48         {"parent":"w99","style":["VERTICAL"]}], ' +
49         '["set","w99",{ "bounds":[10,52,988,402],"children":
50         [], "tabIndex":23,"activeKeys":["CTRL+#70","CTRL+#78","CTRL+#82","CTRL+#89","CT
51         RL+#83","CTRL+#71","CTRL+#69"],"cancelKeys":
52         ["CTRL+#70","CTRL+#78","CTRL+#82","CTRL+#89","CTRL+#83","CTRL+#71","CTRL+#69"]}],
53
54         ["listen","w99",
55         {"MouseDown":true,"MouseUp":true,"MouseDoubleClick":true,"KeyDown":true}],
56         ["set","w99",{ "itemCount":118,"itemHeight":28,"itemMetrics":
57         [[0,0,50,3,0,3,44],[1,50,50,53,0,53,44],[2,100,140,103,0,103,134],
58         [3,240,180,243,0,243,174],[4,420,50,423,0,423,44],[5,470,50,473,0,473,44]], "co

```

```

lumnCount":6,"headerHeight":35,"headerVisible":true,"linesVisible":true,"focus
Item":"w108","selection":[{"w108"}]},
38      ["listen","w99",{ "Selection":true,"DefaultSelection":true}],
39      [{"set","w99",{ "enableCellToolTip":true}},{ "listen","w100",
{"Selection":true}], ' +
40      [{"set","w101",{ "visibility":true}},{ "listen","w101",
{"Selection":true}],
41      ["create","w102","rwt.widgets.GridColumn",
{"parent":"w99","text":"Nr. ","width":50,"moveable":true}], ' +
42      [{"listen","w102",{ "Selection":true}],
43      ["create","w103","rwt.widgets.GridColumn",{ "parent":"w99","te
xt":"Sym. ","index":1,"left":50,"width":50,"moveable":true}],
44      [{"listen","w103",{ "Selection":true}],
45      ["create","w104","rwt.widgets.GridColumn",{ "parent":"w99","te
xt":"Name","index":2,"left":100,"width":140,"moveable":true}],
46      [{"listen","w104",{ "Selection":true}],
47      ["create","w105","rwt.widgets.GridColumn",{ "parent":"w99","te
xt":"Series","index":3,"left":240,"width":180,"moveable":true}],
48      [{"listen","w105",{ "Selection":true}], ' +
49      [{"create","w106","rwt.widgets.GridColumn",{ "parent":"w99","text":
"Group","index":4,"left":420,"width":50,"moveable":true}],
50      [{"listen","w106",{ "Selection":true}],
51      ["create","w107","rwt.widgets.GridColumn",{ "parent":"w99","te
xt":"Period","index":5,"left":470,"width":50,"moveable":true}],
52      [{"listen","w107",{ "Selection":true}],
53      ["create","w108","rwt.widgets.GridItem",
{"parent":"w99","index":0,"texts":
["1","H","Hydrogen","Nonmetal","1","1"],"cellBackgrounds":[null,null,null,
[138,226,52,255],null,null]}], ' +
54      [{"create","w224","rwt.widgets.GridItem",
{"parent":"w99","index":116,"texts":
["117","Uus","Ununseptium","Halogen","17","7"],"cellBackgrounds":
[null,null,null,[252,233,79,255],null,null]}],
55      [{"create","w225","rwt.widgets.GridItem",
{"parent":"w99","index":117,"texts":["118","Uuo","Ununoctium","Noble
gas","18","7"],"cellBackgrounds":[null,null,null,
[114,159,207,255],null,null]}],
56      [{"create","w226","rwt.widgets.Composite",
{"parent":"w97","style":["BORDER"],"bounds":[10,464,988,25],"children":
["w227"],"tabIndex":-1,"clientArea":[0,0,986,23]}],
57      [{"create","w227","rwt.widgets.Label",
{"parent":"w226","style":["NONE"],"bounds":
[10,10,966,3],"tabIndex":-1,"text":"Hydrogen (H)"}], ' +
58      [{"create","w228","rwt.widgets.Label",{ "parent":"w97","style":
["WRAP"],"bounds":[10,499,988,16],"tabIndex":-1,"foreground":
[150,150,150,255],"font":[["Verdana","Lucida Sans","Arial","Helvetica","sans-
serif"],10,false,false],"text":"Shortcuts: [CTRL+F] - Filter | Sort by:
[CTRL+R] - Number, [CTRL+Y] - Symbol, [CTRL+N] - Name, [CTRL+S] - Series,
[CTRL+G] - Group, [CTRL+E] - Period"}],
59      [{"set","w1",{ "focusControl":"w99"}],
["call","rwt.client.BrowserNavigation","addToHistory",{ "entries":
[["tableviewer","TableViewer"]]}]]]
60    )
61  )
62

```

```
1 JsonString = JsonValue (
2   | string |
3
4   initializeWith: str = ( string := str )
5
6   isString = (
7     ^ true
8   )
9
10  asString = (
11    ^ string
12  )
13
14  hashCode = (
15    ^ string hashCode
16  )
17
18  = other = (
19    self == other ifTrue: [ ^ true ].
20    other == nil ifTrue: [ ^ false ].
21
22    self class == other class ifFalse: [ ^ false ].
23
24    ^ string = other asString
25  )
26
27  ----
28
29  new: str = ( ^ self new initializeWith: str )
30
31 )
32
```

```

1 JsonValue = (
2
3   isObject  = ( ^ false )
4   isArray   = ( ^ false )
5   isNumber  = ( ^ false )
6   isString  = ( ^ false )
7   isBoolean = ( ^ false )
8
9   isTrue    = ( ^ false )
10  isFalse   = ( ^ false )
11  isNull    = ( ^ false )
12
13  asObject = (
14    self error: 'Unsupported operation, not an object: ' + self asString
15  )
16
17  asArray = (
18    self error: 'Unsupported operation, not an array: ' + self asString
19  )
20
21  asInteger = (
22    self error: 'Unsupported operation, not a number: ' + self asString
23  )
24
25  asDouble = (
26    self error: 'Unsupported operation, not a number: ' + self asString
27  )
28
29  asString = (
30    self error: 'Unsupported operation, not a string: ' + self asString
31  )
32
33  asBoolean = (
34    self error: 'Unsupported operation, not a boolean: ' + self asString
35  )
36
37  writeTo: writer = (
38    self writeTo: writer with: nil
39  )
40
41 " TODO:
42   public void writeTo( final Writer writer, final WriterConfig config )
throws IOException {
43     WritingBuffer buffer = new WritingBuffer( writer, 128 );
44     write( config == null ? new JsonWriter( buffer ) :
config.createWriter( buffer ) );
45     buffer.flush();
46   }"
47
48 "TODO:
49 @Override
50 public String toString() {
51   return toString( null );
52 }
53
54 public String toString( final WriterConfig config ) {
55   StringWriter writer = new StringWriter();
56   try {

```

```

57     writeTo( writer, config );
58     } catch( IOException exception ) {
59         // StringWriter does not throw IOExceptions
60         throw new RuntimeException( exception );
61     }
62     return writer.toString();
63 }"
64
65
66 writeOn: writer = (
67     ^ self subclassResponsibility
68 )
69
70 ----
71
72 readFrom: str = (
73     ^ (JsonParser with: str) parse
74 )
75
76 integer: anInteger = (
77     ^ JsonNumber new: anInteger asString
78 )
79
80 double: aDouble = (
81     aDouble ifNil: [self error: 'aDouble nil'].
82     ^ JsonNumber new: (self cutOffPointZero: aDouble asString)
83 )
84
85 string: aString = (
86     ^ aString == nil
87         ifTrue: [ JsonLiteral NULL ]
88         ifFalse: [ JsonString new: aString ]
89 )
90
91 boolean: aBoolean = (
92     ^ aBoolean
93         ifTrue: [ JsonLiteral TRUE ]
94         ifFalse: [ JsonLiteral FALSE ]
95 )
96
97 cutOffPointZero: str = (
98     (str endsWith: '.0') ifTrue: [ ^ str substringFrom: 1 to: str length
- 2].
99     ^ str
100 )
101 )

```

```

1 ParseException = (
2   | offset line column msg |
3
4   initializeWith: message at: anOffset line: aLine column: aColumn = (
5     msg      := message.
6     offset   := anOffset.
7     line     := aLine.
8     column   := aColumn.
9   )
10
11   message = ( ^ msg )
12
13   offset = ( ^ offset )
14   line   = ( ^ line   )
15   column = ( ^ column )
16
17   asString = ( ^ msg + ':' + line + ':' + column )
18
19   ----
20
21   with: aMessageString at: offset line: line column: column = (
22     ^ self new initializeWith: aMessageString
23       at: offset
24       line: line
25       column: column
26   )
27 )
28

```

```
1 "  
2  
3 $Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Dispatch = Benchmark (  
27  
28     benchmark = (  
29         | cnt |  
30         cnt := 0.  
31         1 to: 20000 do: [ :i | cnt := cnt + (self method: i) ].  
32         ^ cnt  
33     )  
34  
35     method: argument = ( ^argument )  
36  
37     verifyResult: result = (  
38         ^ 200010000 = result  
39     )  
40  
41 )  
42
```

```
1 "  
2  
3 $Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 DispatchNoArg = Benchmark (  
27  
28   benchmark = (  
29     1 to: 20000 do: [ :i | self method ]  
30   )  
31  
32   method = ( ^ 1 )  
33  
34 )  
35
```



```
1 "  
2  
3 $Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 DispatchPerform = Benchmark (  
27  
28     benchmark = (  
29         1 to: 20000 do: [ :i | self perform: #method: withArguments:  
(Array with: i)]  
30     )  
31  
32     method: argument = ( ^ argument )  
33  
34 )  
35
```

```
1 "  
2  
3 $Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 DispatchPerformNoArg = Benchmark (  
27  
28   benchmark = (  
29     1 to: 20000 do: [ :i | self perform: #method ]  
30   )  
31  
32   method = ( ^ 1 )  
33  
34 )  
35
```

```
1 Calculator = (  
2   | a |  
3  
4   initializeWith: anInt = (  
5     a := anInt  
6   )  
7  
8   inc: aSymbol = (  
9     aSymbol = #once ifTrue: [ a := a + 1 ]  
10  )  
11  
12  a = ( ^ a )  
13  
14  doesNotUnderstand: selector arguments: arguments = (  
15    ^ self inc: #once  
16  )  
17  
18  ----  
19  
20  new: a = (  
21    | calc |  
22    calc := self new.  
23    calc initializeWith: a.  
24    ^ calc  
25  )  
26 )
```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/
CalculatorDelegate.som

```
1 CalculatorDelegate = (  
2   | target |  
3  
4   initializeWith: aCalculator = (  
5       target := aCalculator  
6   )  
7  
8   inc: aSymbol = (  
9       target inc: aSymbol  
10  )  
11  
12  ----  
13  
14  new: a = (  
15      | calc |  
16      calc := self new.  
17      calc initializeWith: a.  
18      ^ calc  
19  )  
20 )
```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/
CalculatorDnuPerform.som

```
1 CalculatorDnuPerform = (  
2   | a |  
3  
4   initializeWith: anInt = (  
5     a := anInt  
6   )  
7  
8   inc: aSymbol = (  
9     aSymbol = #once ifTrue: [ a := a + 1 ]  
10  )  
11  
12  a = ( ^ a )  
13  
14  doesNotUnderstand: selector arguments: arguments = (  
15    ^ self perform: #inc: withArguments: arguments  
16  )  
17  
18  ----  
19  
20  new: a = (  
21    | calc |  
22    calc := self new.  
23    calc initializeWith: a.  
24    ^ calc  
25  )  
26 )
```

```
1 DirectAdd = Benchmark (
2   | calc |
3
4   initialize = (
5     calc := Calculator new.
6   )
7
8   benchmark = (
9     calc initializeWith: 5.
10
11     1 to: 20000 do: [ :i |
12       calc inc: #once
13     ].
14
15     calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result'
println. calc a println. ]
16   )
17
18   ----
19
20   new = (
21     ^ super new initialize
22   )
23 )
24
```

```
1 DnuAdd = Benchmark (
2   | calc |
3
4   initialize = (
5     calc := Calculator new.
6   )
7
8   benchmark = (
9     calc initializeWith: 5.
10
11     1 to: 20000 do: [ :i |
12       calc incDNU: #once
13     ].
14
15     calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result'
println. calc a println. ]
16   )
17
18   ----
19
20   new = (
21     ^ super new initialize
22   )
23 )
24
```

```
1 DnuPerformAdd = Benchmark (
2   | calc |
3
4   initialize = (
5     calc := CalculatorDnuPerform new.
6   )
7
8   benchmark = (
9     calc initializeWith: 5.
10
11     1 to: 20000 do: [ :i |
12       calc incDNU: #once
13     ].
14
15     calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result'
println. calc a println. ]
16   )
17
18   ----
19
20   new = (
21     ^ super new initialize
22   )
23 )
24
```



```
1 IndirectAdd = Benchmark (
2   | delegate calc |
3
4   initialize = (
5     calc := Calculator new.
6     delegate := CalculatorDelegate new: calc
7   )
8
9   benchmark = (
10    calc initializeWith: 5.
11
12    1 to: 20000 do: [ :i |
13      delegate inc: #once
14    ].
15
16    ^ calc a
17  )
18
19  verifyResult: result = (
20    ^ 20005 = result
21  )
22
23  ----
24
25  new = (
26    ^ super new initialize
27  )
28 )
29
```

```
1 PerformAdd = Benchmark (
2   | calc |
3
4   initialize = (
5     calc := Calculator new.
6   )
7
8   benchmark = (
9     calc initializeWith: 5.
10
11     1 to: 20000 do: [ :i |
12       calc perform: #inc: withArguments: (Array with: #once)
13     ].
14
15     calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result'
println. calc a println. ]
16   )
17
18   ----
19
20   new = (
21     ^ super new initialize
22   )
23 )
24
```

```
1 Proxy = (  
2   | target |  
3  
4   initializeWith: anObj = (  
5     target := anObj  
6   )  
7  
8   doesNotUnderstand: selector arguments: arguments = (  
9     ^ target perform: selector withArguments: arguments  
10  )  
11  
12  ----  
13  
14  new: target = (  
15    | proxy |  
16    proxy := self new.  
17    proxy initializeWith: target.  
18    ^ proxy  
19  )  
20 )
```

```
1 ProxyAdd = Benchmark (
2   | proxy calc |
3
4   initialize = (
5     calc := Calculator new.
6     proxy := Proxy new: calc
7   )
8
9   benchmark = (
10    calc initializeWith: 5.
11
12    1 to: 20000 do: [ :i |
13      proxy inc: #once
14    ].
15
16    ^ calc a
17  )
18
19  verifyResult: result = (
20    ^ 20005 = result
21  )
22
23  ----
24
25  new = (
26    ^ super new initialize
27  )
28 )
29
```

```
1 "  
2  
3 $Id: Fibonacci.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Fibonacci = Benchmark (  
27  
28     benchmark = ( | result |  
29         result := self fibonacci: 20.  
30         ^ result  
31     )  
32  
33     fibonacci: n = (  
34         ^(n <= 1)  
35         ifTrue: 1  
36         ifFalse: [ (self fibonacci: (n - 1)) + (self fibonacci: (n -  
2)) ]  
37     )  
38  
39     verifyResult: result = (  
40         ^ 10946 = result  
41     )  
42  
43 )  
44
```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 FieldLoop = Benchmark (  
24     | counter |  
25  
26     benchmark = ( | iter |  
27         counter := 0.  
28         iter := 20000.  
29  
30         [ iter > 0 ] whileTrue: [  
31             iter := iter - 1.  
32             counter := counter + 1.  
33             counter := counter + 1.  
34             counter := counter + 1.  
35             counter := counter + 1.  
36             counter := counter + 1.  
37  
38             counter := counter + 1.  
39             counter := counter + 1.  
40             counter := counter + 1.  
41             counter := counter + 1.  
42             counter := counter + 1.  
43  
44             counter := counter + 1.  
45             counter := counter + 1.  
46             counter := counter + 1.  
47             counter := counter + 1.  
48             counter := counter + 1.  
49  
50             counter := counter + 1.  
51             counter := counter + 1.  
52             counter := counter + 1.  
53             counter := counter + 1.  
54             counter := counter + 1.
```

```

55
56     counter := counter + 1.
57     counter := counter + 1.
58     counter := counter + 1.
59     counter := counter + 1.
60     counter := counter + 1.
61
62     counter := counter + 1.
63     counter := counter + 1.
64     counter := counter + 1.
65     counter := counter + 1.
66     counter := counter + 1.
67
68     l.
69
70     ^ counter
71 )
72
73 verifyResult: result = (
74     ^ 600000 = result
75 )
76
77 )
78

```

```
1 FieldWrite = Benchmark (  
2   | counter |  
3  
4   benchmark = (  
5     | bar |  
6     bar := 1234.  
7  
8     1 to: 20000 do: [:i |  
9       counter := 2122.  
10    ].  
11    ^ counter  
12  )  
13  
14  verifyResult: result = (  
15    ^ 2122 = result and: [counter = result]  
16  )  
17  
18 )  
19
```



```
1 "  
2  
3 $Id: IntegerLoop.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 IntegerLoop = Benchmark (  
27  
28     benchmark = ( | bounds a |  
29         bounds := 20000.  
30         bounds negated to: bounds by: 1 do: [:value | a := value-value].  
31         ^ a  
32     )  
33  
34     verifyResult: result = (  
35         ^ 0 = result  
36     )  
37 )  
38
```

```
1 "  
2  
3 $Id: Loop.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Loop = Benchmark (  
27  
28   singleRun = (  
29     | sum |  
30     sum := 0.  
31     1 to: 100 do: [ :j | sum := sum + 1 ].  
32     (sum = 100)  
33     ifFalse: [  
34       self error: 'Wrong result: ' + sum + ' should be: 100' ].  
35     ^ sum  
36   )  
37  
38   benchmark = (  
39     | sum |  
40     sum := 0.  
41     1 to: 200 do: [ :i | sum := sum + self singleRun ].  
42     ^ sum  
43   )  
44  
45   verifyResult: result = (  
46     ^ 20000 = result  
47   )  
48 )  
49
```

```
1 NonLocalReturn = Benchmark (
2
3   first: a = ( ^ self second: a )
4   second: a = ( ^ self third: a )
5   third: a = ( a value )
6
7   nlr = (
8     self first: [ ^ 1 ]
9   )
10
11  benchmark = (
12    | sum |
13    sum := 0.
14    1 to: 200 do: [ :i | sum := sum + self nlr ].
15    ^ sum
16  )
17
18  verifyResult: result = (
19    ^ 200 = result
20  )
21
22 )
23
```

```
1 "  
2  
3 $Id: Recurse.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Recurse = Benchmark (  
27     benchmark = (  
28         ^ self recurse: 13  
29     )  
30     recurse: n = (  
31         (n > 0) ifTrue: [ self recurse: n - 1. self recurse: n - 1 ].  
32         ^ n  
33     )  
34     verifyResult: result = (  
35         ^ 13 = result  
36     )  
37 )  
38 )  
39 )  
40 )  
41
```

```

1 "
2 modified for SOM by Stefan Marr.
3
4 modified for Squeak by nishis@urban.ne.jp
5 I added one class methods.
6 and modified execute method.  block variables are renamed for Squeak.
7 Thanx to Mr. Tim Olson, Mr. Kohler Markus, Mr. Tim Rowledge, Mr. John
Maloney, Mr. Ian Piumarta.
8
9 original comments
10     NAME             STones80
11     AUTHOR            bruce@utafl1.uta.edu (Bruce Samuelson)
12     FUNCTION          low and medium level benchmarks for ST80 and ST/V
13     ST-VERSIONS       pre R4.0, R4.0, R4.1, ST/V
14     PREREQUISITES     need floating point hardware or emulation
15     CONFLICTS         none
16     DISTRIBUTION      world
17     VERSION           1.0
18     DATE              April 16, 1993
19
20 SUMMARY The filein includes two classes: Slopstones (Smalltalk Low level
21 Operation Stones) and Smopstones (Smalltalk Medium level Operation
Stones).
22 Each includes seven cpu intensive benchmarks. They work equally well with
23 16-bit and 32-bit implementations and are designed to be portable to all
24 Smalltalk versions from ParcPlace and Digitalk. They are normalized to the
25 author's 486/33 Windows 3.1 machine running ParcPlace VisualWorks 1.0.
26 Results have been posted to the Usenet group comp.lang.smalltalk and form
the
27 basis of an article that is scheduled to be published in the June issue
of The
28 Smalltalk Report.
29
30 The only difference between the ST80 (STones80) and ST/V (STonesV) filein
is
31 in the messages that define the classes in the first few lines of code.
The
32 ST80 messages specify the class category and message protocol, which are
not
33 used in ST/V.
34
35 Bruce Samuelson
36 "
37
38 SlopStone = Benchmark (
39     | o obj |
40
41     innerBenchmarkLoop: innerIterations = (
42         "Using the SlopStone benchmarks, but not doing the
43         old style Stone performance number reporting"
44         obj := Object.
45         o := obj new.
46
47         1 to: innerIterations do: [:i | self doIntAdd          ].
48         1 to: innerIterations do: [:i | self doFloatAdd        ].
49         "1 to: innerIterations do: [:i | self doStringAccess    ]." "NOT
SUPPORTED"
50         1 to: innerIterations do: [:i | self doObjectCreation ].

```

```

51      "1 to: innerIterations do: [:i | self doObjectCopy      ]." "NOT
SUPPORTED"
52      1 to: innerIterations do: [:i | self doPerform          ].
53      1 to: innerIterations do: [:i | self doBlockValue       ].
54  )
55
56
57      "STEFAN: use the original benchmarks, but do not use original way
58              of reporting results. Rely on the SMark reporting instead."
59
60      "INTRODUCTION
61
62      Slopstone: Smalltalk Low level OPeration Stones
63      Portable Low Level Benchmarks for ST80 and ST/V (using 16-bit
64      SmallIntegers) Placed in public domain January 1993  (c) Bruce
65      Samuelson Permission is given to place this in public Smalltalk
archives
66
67      Use monospaced fonts if possible to view the methods in this class.
68
69      (1) Collect garbage if supported (2) do 'SlopstoneBenchmark new
70      runBenchmark'. Results are printed in the Transcript window.
71      Post results for your machines to comp.lang.smalltalk or
72      mail them to bruce@ling.uta.edu or bruce@utafll.uta.edu.
73
74      DISCUSSION
75
76      This readme method would normally be in the class comment for ST80.
77      ST/V-DOS doesn't support class comments.
78
79      The benchmarks test strictly low level operations. They do not test
80      higher level operations such as forming sets, sorting, or streaming,
nor
81      do they test
82      applications. They also do not test user interface operations because
of
83      the non-portability of this area of Smalltalk and its sensitivity to
the
84      performance of the video subsystem. The tests are cpu bound. They do
85      not access files and should not cause disk paging.
86
87      The benchmarks use loop counts of 16000 because SmallIntegers cannot
88      exceed 16383 for ST/V-DOS. 16-bit implementations would perform worse
89      with large loop
90      counts. The benchmarks are also suitable for testing 32-bit versions
of
91      Smalltalk.
92
93      DEFINITION OF REFERENCE MACHINE (ONE SLOPSTONE)
94
95      The following machine is the one on which I developed these
96      benchmarks. By
97      convention it is defined to operate at one slopstone. It's a mid
range
98      performer for current ParcPlace versions of Smalltalk.
99
100     Hardware: Amax 486DX/33 (includes internal floating point processor
101     and internal 8K cache), 256K external cache, 16MB RAM.
102
103     Software: ParcPlace VisualWorks 1.0, Windows 3.1, DOS 5.0 (plain
vanilla

```

```

104     setup).
105
106     COMPARISON TO XEROX DORADO
107
108     For reference, the machine runs at 649% of a Dorado on ParcPlace
109     benchmarks for ST80 4.1. Its fast video card helps on these PPS
110     benchmarks. I didn't run
111     them for VisualWorks 1.0. It would be somewhat slower because there
112     are vastly
113     more classes.
114
115     EXAMPLE RESULTS FOR REFERENCE MACHINE
116
117     1000s      time      1000s of
118     itera-    sec-      iterations
119     tions     onds      per sec      slop-
120                                     stones      explanation
121
122     3808      0.577     6600      1.0      add integers
123     544       2.262     240       1.0      add floats
124     960       1.088     882       1.0      access strings
125     320       0.908     352       1.0      create objects
126     160       1.49      107       1.0      copy objects
127     480       1.129     425       1.0      perform selectors
128     896       1.237     724       1.0      evaluate blocks
129
130     640       1.151     555       1.0      geometric mean"
131
132     doBlockValue = (
133     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value.
134     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value.
135     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value.
136     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value.
137     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value.
138     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value.
139     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value.
140     [] value. [] value. [] value. [] value. [] value. [] value. [] value. [] value
141     )
142
143
144     doFloatAdd = (
145     1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+
146     1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0
147     )
148
149     doIntAdd = (
150     1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+
151     "+1+1+1+1+1+1+1+1+1+1+1+1+1+1+
152     1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+
153     1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+
154     1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+
155     1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+
156     1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+
157     1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+"
158     )
159
160     doObjectCopy = (
161     o copy copy copy copy copy copy copy copy copy copy copy
162     )
163
164     doObjectCreation = (

```

```

165     obj new. obj new. obj new. obj new. obj new.
166     obj new. obj new. obj new. obj new. obj new.
167     obj new. obj new. obj new. obj new. obj new.
168     obj new. obj new. obj new. obj new. obj new
169 )
170
171 doPerform = (
172     o perform: #value. o perform: #value. o perform: #value.
173     o perform: #value. o perform: #value. o perform: #value.
174     o perform: #value. o perform: #value. o perform: #value.
175     o perform: #value. o perform: #value. o perform: #value.
176     o perform: #value. o perform: #value. o perform: #value.
177     o perform: #value. o perform: #value. o perform: #value.
178     o perform: #value. o perform: #value. o perform: #value.
179     o perform: #value. o perform: #value. o perform: #value.
180     o perform: #value. o perform: #value. o perform: #value.
181     o perform: #value. o perform: #value. o perform: #value
182 )
183
184 doStringAccess = (
185     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
186     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
187     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
188     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
189     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
190     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
191     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
192     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
193     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
194     'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1
195 )
196 )
197

```



```

1 "
2
3 $Id: Sum.som 31 2009-07-31 12:25:18Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 Sum = Benchmark (
27
28     benchmark = (
29         | result |
30         1 to: 2 do: [ :i | result := self sumFrom: 1 to: 10000 ].
31         ^ result
32     )
33
34     sumFrom: start to: end = (
35         | sum |
36         sum := 0.
37         start to: end do: [ :i | sum := sum + i ].
38         ^sum
39     )
40
41     verifyResult: result = (
42         ^ 50005000 = result
43     )
44
45 )
46

```

```
1 "  
2 Copyright (c) 2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 WhileLoop = Benchmark (  
24  
25     singleRun = (  
26         | sum |  
27         sum := 0.  
28         [sum < 1000]  
29         whileTrue:  
30             [sum := sum + 1].  
31     ^ sum  
32 )  
33  
34     benchmark = (  
35         | sum |  
36         sum := 0.  
37         [sum < 20000]  
38         whileTrue:  
39             [sum := sum + self singleRun].  
40     ^ sum  
41 )  
42  
43     verifyResult: result = (  
44         ^ 20000 = result  
45     )  
46 )  
47  
48
```

```

1 "
2 Copyright (c) 2013 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 WhileLoopPoly = Benchmark (
24
25     singleRun = (
26         | sum poly b |
27         sum := 0.
28         [sum < 1000]
29         whileTrue:
30             [sum := sum + 1.
31             ((sum % 4) = 0) ifTrue: [poly := 1].
32             ((sum % 4) = 1) ifTrue: [poly := 'abc'].
33             ((sum % 4) = 2) ifTrue: [poly := 2222222222222222].
34             ((sum % 4) = 3) ifTrue: [poly := 1//2].
35             b := poly
36             ].
37         b := b + b.
38         ^ sum
39     )
40
41     benchmark = (
42         | sum |
43         sum := 0.
44         [sum < 20000]
45         whileTrue:
46             [sum := sum + self singleRun].
47         ^ sum
48     )
49
50     verifyResult: result = (
51         ^ 20000 = result
52     )
53 )
54

```



```

1 "
2 Copyright (c) 2013 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 WhileLoopVAt = Benchmark (
24
25     singleRun = (
26         | sum v |
27         v := Vector new.
28         v at: 1 put: 1.
29         sum := 0.
30         [sum < 1000]
31         whileTrue:
32             [sum := sum + (v at: 1)].
33         ^ sum
34     )
35
36     benchmark = (
37         | sum |
38         sum := 0.
39         [sum < 20000]
40         whileTrue:
41             [sum := sum + self singleRun].
42         ^ sum
43     )
44
45     verifyResult: result = (
46         ^ 20000 = result
47     )
48 )
49
50

```

```
1 "  
2  
3 $Id: List.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 List = Benchmark (  
27  
28   benchmark = ( | result |  
29     result := self  
30       tailWithX: (self makeList: 15)  
31       withY: (self makeList: 10)  
32       withZ: (self makeList: 6).  
33     ^ result length  
34   )  
35  
36   verifyResult: result = (  
37     ^ self assert: 10 equals: result  
38   )  
39  
40   makeList: length = (  
41     (length = 0)  
42     ifTrue: [ ^nil ]  
43     ifFalse: [  
44       ^(ListElement new: length)  
45       next: (self makeList: (length - 1)) ]  
46   )  
47  
48   isShorter: x than: y = (  
49     | xTail yTail |  
50  
51     xTail := x. yTail := y.  
52     [ yTail isNil ]  
53     whileFalse: [  
54       xTail isNil ifTrue: [ ^true ].
```

```

55         xTail := xTail next.
56         yTail := yTail next ].
57
58     ^false
59 )
60
61 tailWithX: x withY: y withZ: z = (
62     (self isShorter: y than: x)
63     ifTrue: [
64         ^self
65         tailWithX: (self tailWithX: x next withY: y withZ: z)
66         withY: (self tailWithX: y next withY: z withZ: x)
67         withZ: (self tailWithX: z next withY: x withZ: y)) ]
68     ifFalse: [ ^z ].
69 )
70
71 )
72

```

```
1 "  
2  
3 $Id: ListElement.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 ListElement = (  
27   | val next |  
28   length = ( next isNil ifTrue: [ ^1 ] ifFalse: [ ^(1 + next length) ] )  
29   val      = ( ^val )  
30   val: n = ( val := n )  
31  
32   next      = ( ^next )  
33   next: element = ( next := element )  
34  
35   -----  
36  
37   new: n = ( ^super new val: n )  
38  
39  
40  
41  
42 )  
43  
44
```



```
1 " This version is a transcription of the Ruby implementation mandelbrot.rb
2   found with JRuby
3   https://raw.githubusercontent.com/jruby/
jruby/3e43676ee6dc3c13e70fe4a52cce685128c23b8e/bench/truffle/mandelbrot.rb
4
5   The original copyright statement reads as follows:
6
7 # Copyright © 2004-2013 Brent Fulgham
8 #
9 # All rights reserved.
10 #
11 # Redistribution and use in source and binary forms, with or without
12 # modification, are permitted provided that the following conditions are
met:
13 #
14 #   * Redistributions of source code must retain the above copyright
notice,
15 #     this list of conditions and the following disclaimer.
16 #
17 #   * Redistributions in binary form must reproduce the above copyright
notice,
18 #     this list of conditions and the following disclaimer in the
documentation
19 #     and/or other materials provided with the distribution.
20 #
21 #   * Neither the name of 'The Computer Language Benchmarks Game' nor the
name
22 #     of 'The Computer Language Shootout Benchmarks' nor the names of its
23 #     contributors may be used to endorse or promote products derived
from this
24 #     software without specific prior written permission.
25 #
26 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS
IS'
27 # AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
28 # IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE
29 # DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE
30 # FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
31 # DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR
32 # SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER
33 # CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY,
34 # OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
THE USE
35 # OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
36 #
37 # The Computer Language Benchmarks Game
38 # http://benchmarksgame.alioth.debian.org
39 #
40 #   contributed by Karl von Laudermann
41 #   modified by Jeremy Echols
42 #   modified by Detlef Reichl
```

```

43 # modified by Joseph LaFata
44 # modified by Peter Zotov
45 #
46 # http://benchmarksgame.alioth.debian.org/u64q/program.php?
test=mandelbrot&lang=yarv&id=3
47 "
48
49 Mandelbrot = Benchmark (
50     | firstResult |
51
52     innerBenchmarkLoop: innerIterations = (
53         ^ self verify: (self mandelbrot: innerIterations) inner:
innerIterations
54     )
55
56     verify: result inner: innerIterations = (
57         innerIterations = 750 ifTrue: [
58             ^ result = 50
59         ].
60         firstResult isNil ifTrue: [
61             firstResult := result.
62         ].
63         ^ result = firstResult
64     )
65
66     mandelbrot: size = (
67         | sum byteAcc bitNum y |
68         sum      := 0.
69         byteAcc  := 0.
70         bitNum   := 0.
71
72         y := 0.
73
74         [y < size] whileTrue: [
75             | ci x |
76             ci := (2.0 * y // size) - 1.0.
77             x  := 0.
78
79             [x < size] whileTrue: [
80                 | zr zr zr zi zizi cr escape z notDone |
81                 zr zr := 0.0.
82                 zizi := zi := 0.0.
83                 cr   := (2.0 * x // size) - 1.5.
84
85                 z := 0.
86                 notDone := true.
87                 escape := 0.
88                 [notDone and: [z < 50]] whileTrue: [
89                     zr := zr zr - zizi + cr.
90                     zi := 2.0 * zr * zi + ci.
91
92                     "preserve recalculation"
93                     zr zr := zr * zr.
94                     zizi := zi * zi.
95
96                     (zr zr + zizi > 4.0) ifTrue: [
97                         notDone := false.
98                         escape  := 1.
99                     ].
100                     z := z + 1.
101                 ].

```

```

102
103         byteAcc := (byteAcc << 1) + escape.
104         bitNum  := bitNum + 1.
105
106         " Code is very similar for these cases, but using
separate blocks
107         ensures we skip the shifting when it's unnecessary,
108         which is most cases. "
109         bitNum = 8
110         ifTrue: [
111             sum := sum bitXor: byteAcc.
112             byteAcc := 0.
113             bitNum  := 0. ]
114         ifFalse: [
115             (x = (size - 1)) ifTrue: [
116                 byteAcc := byteAcc << (8 - bitNum).
117                 sum := sum bitXor: byteAcc.
118                 byteAcc := 0.
119                 bitNum  := 0. ]].
120             x := x + 1.
121         ].
122         y := y + 1.
123     ].
124
125     ^ sum
126 )
127 )
128

```

```

1 " The Computer Language Benchmarks Game
2 http://shootout.alioth.debian.org/
3
4 contributed by Mark C. Lewis
5 modified slightly by Chad Whipkey
6
7 Based on nbody.java ported to SOM by Stefan Marr.
8 "
9 Body = (
10   | x y z vx vy vz mass |
11
12   x = ( ^ x )
13   y = ( ^ y )
14   z = ( ^ z )
15
16   vx = ( ^ vx )
17   vy = ( ^ vy )
18   vz = ( ^ vz )
19
20   mass = ( ^ mass )
21
22   x: val = ( x := val )
23   y: val = ( y := val )
24   z: val = ( z := val )
25
26   vx: val = ( vx := val )
27   vy: val = ( vy := val )
28   vz: val = ( vz := val )
29
30   mass: val = ( mass := val )
31
32   offsetMomentumX: px y: py z: pz = (
33     vx := 0.0 - (px // Body SolarMass).
34     vy := 0.0 - (py // Body SolarMass).
35     vz := 0.0 - (pz // Body SolarMass).
36   )
37
38   print = (
39     'x: ' print. x println.
40     'y: ' print. y println.
41     'z: ' print. z println.
42
43     'vx: ' print. vx println.
44     'vy: ' print. vy println.
45     'vz: ' print. vz println.
46
47     'mass: ' print. mass println.
48   )
49
50   ----
51
52   | solarMass |
53
54   Pi          = ( ^ 3.141592653589793 )
55   SolarMass   = ( ^ solarMass )
56   DaysPerYear = ( ^ 365.24 )
57
58   initialize = (

```

```

59         solarMass := 4 * self Pi * self Pi.
60     )
61
62     new = ( | b |
63         b := super new.
64         b x: 0.0.  b vx: 0.0.
65         b y: 0.0.  b vy: 0.0.
66         b z: 0.0.  b vz: 0.0.
67         b mass: 0.0.
68         ^ b
69     )
70
71     jupiter = ( | b |
72         b := super new.
73         b x: 4.8414314424647209.
74         b y: -1.16032004402742839.
75         b z: -0.103622044471123109.
76         b vx: 0.00166007664274403694 * self DaysPerYear.
77         b vy: 0.00769901118419740425 * self DaysPerYear.
78         b vz: -0.0000690460016972063023 * self DaysPerYear.
79         b mass: 0.000954791938424326609 * self SolarMass.
80         ^ b
81     )
82
83     saturn = ( | b |
84         b := super new.
85         b x: 8.34336671824457987.
86         b y: 4.12479856412430479.
87         b z: -0.403523417114321381.
88         b vx: -0.00276742510726862411 * self DaysPerYear.
89         b vy: 0.00499852801234917238 * self DaysPerYear.
90         b vz: 0.0000230417297573763929 * self DaysPerYear.
91         b mass: 0.000285885980666130812 * self SolarMass.
92         ^ b
93     )
94
95     uranus = ( | b |
96         b := super new.
97         b x: 12.894369562139131.
98         b y: -15.1111514016986312.
99         b z: -0.223307578892655734.
100        b vx: 0.00296460137564761618 * self DaysPerYear.
101        b vy: 0.0023784717395948095 * self DaysPerYear.
102        b vz: -0.0000296589568540237556 * self DaysPerYear.
103        b mass: 0.0000436624404335156298 * self SolarMass.
104        ^ b
105    )
106
107    neptune = ( | b |
108        b := super new.
109        b x: 15.3796971148509165.
110        b y: -25.9193146099879641.
111        b z: 0.179258772950371181.
112        b vx: 0.00268067772490389322 * self DaysPerYear.
113        b vy: 0.00162824170038242295 * self DaysPerYear.
114        b vz: -0.000095159225451971587 * self DaysPerYear.
115        b mass: 0.0000515138902046611451 * self SolarMass.
116        ^ b
117    )
118
119    sun = ( | b |

```

```
120         b := self new.  
121         b mass: self SolarMass.  
122         ^ b  
123     )  
124 )
```

```
1 NBody = Benchmark (
2   | expectedEnergy |
3
4   initialize = (
5     Body initialize.
6   )
7
8   innerBenchmarkLoop: innerIterations = (
9     | bodies |
10    bodies := NBodySystem new.
11
12    1 to: innerIterations do: [:i |
13      bodies advance: 0.01.
14    ].
15
16    innerIterations = 250000 ifTrue: [
17      ^ bodies energy = -0.1690859889909308
18    ].
19    expectedEnergy isNil ifTrue: [
20      expectedEnergy := bodies energy.
21      ^ true
22    ].
23
24    ^ expectedEnergy = bodies energy.
25  )
26
27  ----
28
29  new = (
30    ^ super new initialize
31  )
32 )
33
```

```
1 " The Computer Language Benchmarks Game
2 http://shootout.alioth.debian.org/
3
4 contributed by Mark C. Lewis
5 modified slightly by Chad Whipkey
6
7 Based on nbody.java ported to SOM by Stefan Marr.
8 "
9 NBodyBench = (
10   run: args = (
11     | n bodies |
12     n := (args at: 2) asInteger.
13     Body initialize.
14
15     bodies := NBodySystem new.
16
17     bodies energy println.
18
19     n timesRepeat: [ bodies advance: 0.01 ].
20
21     bodies energy println.
22   )
23 )
24
```



```

1 " The Computer Language Benchmarks Game
2 http://shootout.alioth.debian.org/
3
4 contributed by Mark C. Lewis
5 modified slightly by Chad Whipkey
6
7 Based on nbody.java ported to SOM by Stefan Marr.
8 "
9 NBodySystem = (
10   | bodies |
11
12   initialize = (
13     | px py pz |
14
15     bodies := Array new: 5.
16     bodies at: 1 put: Body sun.
17     bodies at: 2 put: Body jupiter.
18     bodies at: 3 put: Body saturn.
19     bodies at: 4 put: Body uranus.
20     bodies at: 5 put: Body neptune.
21
22     "bodies do: [:b | b print. '' println ]."
23
24     px := 0.0. py := 0.0. pz := 0.0.
25
26     bodies do: [:b |
27       px := px + (b vx * b mass).
28       py := py + (b vy * b mass).
29       pz := pz + (b vz * b mass).
30     ].
31
32     (bodies at: 1) offsetMomentumX: px y: py z: pz.
33
34     "bodies do: [:b | b print. '' println ]."
35   )
36
37   advance: dt = (
38     1 to: bodies length do: [:i |
39       | iBody |
40       iBody := bodies at: i.
41
42       i + 1 to: bodies length do: [:j |
43         | dx dy dz jBody dSquared distance mag |
44         jBody := bodies at: j.
45         dx := iBody x - jBody x.
46         dy := iBody y - jBody y.
47         dz := iBody z - jBody z.
48
49         dSquared := (dx * dx) + (dy * dy) + (dz * dz).
50         distance := dSquared sqrt.
51         mag      := dt // (dSquared * distance).
52
53         iBody vx: iBody vx - (dx * jBody mass * mag).
54         iBody vy: iBody vy - (dy * jBody mass * mag).
55         iBody vz: iBody vz - (dz * jBody mass * mag).
56
57         jBody vx: jBody vx + (dx * iBody mass * mag).
58         jBody vy: jBody vy + (dy * iBody mass * mag).

```

```

59         jBody vz: jBody vz + (dz * iBody mass * mag).
60     ].
61 ].
62
63     bodies do: [:body |
64         body x: body x + (dt * body vx).
65         body y: body y + (dt * body vy).
66         body z: body z + (dt * body vz).
67     ].
68 )
69
70 energy = (
71     | dx dy dz distance e |
72     e := 0.0.
73
74     1 to: bodies length do: [:i |
75         | iBody |
76         iBody := bodies at: i.
77
78         e := e + (0.5 * iBody mass *
79             ((iBody vx * iBody vx) +
80             (iBody vy * iBody vy) +
81             (iBody vz * iBody vz))).
82
83         i + 1 to: bodies length do: [:j |
84             | jBody |
85             jBody := bodies at: j.
86
87             dx := iBody x - jBody x.
88             dy := iBody y - jBody y.
89             dz := iBody z - jBody z.
90
91             distance := ((dx*dx) + (dy*dy) + (dz*dz)) sqrt.
92             e := e - ((iBody mass * jBody mass) // distance).
93         ].
94     ].
95     ^ e
96 )
97
98 ----
99
100 new = (
101     ^ super new initialize
102 )
103 )
104

```

```

1 "https://github.com/Sable/Ostrich/blob/master/map-reduce/page-rank/js/
pagerank.js"
2 PageRank = Benchmark (
3   | firstResult |
4
5   " generates an array of random pages and their links "
6   generateRandomPagesN: n outLinks: outLinks divisor: divisor = (
7     | pages |
8     " matrix cell i,j means link from j->i "
9     pages := Array new: n * n withAll: 0.
10
11     0 to: n - 1 do: [:i |
12       outLinks at: i + 1 put: 0.
13
14       0 to: n - 1 do: [:j |
15         (i <> j and: [(JenkinsRandom random abs % divisor) = 0]) ifTrue:
16           pages at: (i * n + j) + 1 put: 1.
17           outLinks at: i + 1 put: (outLinks at: i + 1) + 1
18         ]
19       ].
20
21       " the case with no outlinks is afunctioned "
22       (outLinks at: i + 1) = 0 ifTrue: [
23         | k |
24         k := JenkinsRandom random abs % n.
25         [i = k] whileTrue: [
26           k := JenkinsRandom random abs % n.
27         ].
28
29         pages at: (i * n + k) + 1 put: 1.
30         outLinks at: i + 1 put: 1
31       ]
32     ].
33     ^ pages
34   )
35
36   mapPageRanks: pages pageRanks: pageRanks maps: maps outLinks: outLinks
n: n = (
37     0 to: n - 1 do: [:i |
38       | outboundRank |
39       outboundRank := (pageRanks at: i + 1) // (outLinks at: i + 1).
40
41       0 to: n - 1 do: [:j |
42         maps at: (i * n + j) + 1
43         put: ((pages at: (i * n + j) + 1) = 0
44           ifTrue: [0.0]
45           ifFalse: [(pages at: (i * n + j) + 1) *
46             outboundRank])
47       ]
48     )
49
50   reducePageRanks: pageRanks maps: maps n: n = (
51     | dif |
52     dif := 0.0.
53
54     0 to: n - 1 do: [:j |

```

```

55     | oldRank newRank |
56     oldRank := pageRanks at: j + 1.
57     newRank := 0.0.
58
59     0 to: n - 1 do: [:i |
60         newRank := newRank + (maps at: (i * n + j) + 1)
61     ].
62
63     newRank := ((1 - PageRank DFactor) // n) + (PageRank DFactor *
newRank).
64     dif := (newRank - oldRank) abs > dif
65         ifTrue: [(newRank - oldRank) abs] ifFalse: [ dif ].
66     pageRanks at: j + 1 put: newRank
67 ].
68 ^ dif
69 )
70
71 innerBenchmarkLoop: innerIterations = (
72     innerIterations < 2 ifTrue: [
73         self error: 'innerIterations has to be 2 or more, but was ' +
innerIterations ].
74
75     JenkinsRandom seed: 49734321.
76     "standard for verification"
77     "n := 5000
78     iter      := 10.
79     thresh    := 0.00000001.
80     divisor   := 100000."
81     ^ self verify: (self runPageRankN: innerIterations iter: 10 thresh:
0.00000001 divisor: 100000)
82         inner: innerIterations.
83 )
84
85 runPageRankN: n iter: iter thresh: thresh divisor: divisor = (
86     | nbLinks maxDiff pageRanks maps outLinks t pages |
87     maxDiff := Double PositiveInfinity.
88     pageRanks := Array new: n withAll: 1.0 // n.
89     maps      := Array new: n * n withAll: 0.0.
90     outLinks  := Array new: n withAll: 0.
91
92     pages := self generateRandomPagesN: n outLinks: outLinks divisor:
divisor.
93
94     nbLinks := 0.
95     0 to: n - 1 do: [:i |
96         1 to: n do: [:j |
97             nbLinks := nbLinks + (pages at: i * n + j)
98         ]
99     ].
100
101     t := 1.
102     [t <= iter and: [maxDiff >= thresh]] whileTrue: [
103         self mapPageRanks: pages pageRanks: pageRanks maps: maps outLinks:
outLinks n: n.
104         maxDiff := self reducePageRanks: pageRanks maps: maps n: n.
105         t := t + 1
106     ].
107
108     ^ pageRanks
109 )
110

```

```

111   verify: pageRanks inner: innerIterations = (
112       innerIterations = 5000 " and: [iter = 10 and: [thresh = 0.00000001
and: [divisor = 100000]]]"
113       ifTrue: [
114           | expected |
115           expected := PageRank ExpectedPageRanks.
116           pageRanks length = expected length ifFalse: [
117               self error: 'Invalid length of page_ranks array'
118           ].
119
120           pageRanks doIndexes: [:i |
121               (pageRanks at: i) = (expected at: i) ifFalse: [
122                   self error: 'ERROR: page_ranks[' + i asString + ']=' +
123                       (pageRanks at: i) + ' differs from the expected
value: ' +
124                           (expected at: i)
125                       ]
126               ] ]
127       ifFalse: [
128           ^ self checkBasedOnFirstResult: pageRanks inner: innerIterations
129       ].
130   ^ true
131 )
132
133 checkBasedOnFirstResult: pageRanks inner: innerIterations = (
134     pageRanks length = innerIterations ifFalse: [ ^ false ].
135
136     firstResult == nil ifTrue: [
137         firstResult := pageRanks.
138         ^ true
139     ] ifFalse: [
140         1 to: innerIterations do: [:i |
141             (firstResult at: i) = (pageRanks at: i) ifFalse: [
142                 ^ false
143             ]
144         ]
145     ].
146
147     ^ true
148 )
149
150 ----
151 | expected |
152
153 DFactor = ( ^ 0.85 ) "damping factor"
154 ExpectedPageRanks = (
155     | prevSize |
156     expected ifNotNil: [ ^ expected ].
157
158     expected := Vector new: 5000.
159     prevSize := expected size.
160
161     1 to: 53 do: [:i |
162         self perform: ('pageRanks' + i + ':') asSymbol withArguments:
(Array with: expected).
163         expected size = prevSize ifTrue: [ self error: 'The method ' +
('pageRanks' + i) asSymbol + ' did not add elements to expected' ].
164         prevSize := expected size ].
165
166     expected := expected asArray.
167

```

```

168      ^ expected
169    )
170
171    pageRanks1: expected = (
172      expected,
173      0.0000555000000000000001, 0.0000300000000000000004,
0.000030000000000000000004,
174      0.0000988500000000000002, 0.0000300000000000000004,
0.0000555000000000000001,
175      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
176      0.0001937753607433594, 0.0000300000000000000004,
0.0000300000000000000004,
177      0.00026861193950589846, 0.0011626495265046287,
0.00033125968546875005,
178      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
179      0.002419297210359863, 0.0000555000000000000001,
0.0000555000000000000001,
180      0.0000300000000000000004, 0.0000300000000000000004,
0.0000810000000000000002,
181      0.0000300000000000000004, 0.00028382917108593755,
0.0000555000000000000001,
182      0.00022373208234375002, 0.0009500131460436332,
0.0004673045821320703,
183      0.0000555000000000000001, 0.0000300000000000000004,
0.0000555000000000000001,
184      0.0000300000000000000004, 0.000095598750000000003,
0.00029348190124218755,
185      0.000095598750000000003, 0.0000300000000000000004,
0.0000300000000000000004,
186      0.0002522648462121094, 0.0002007225, 0.0005721072315653125,
187      0.0006549038427949217, 0.00027208502253146485,
0.0000300000000000000004,
188      0.000095598750000000003, 0.00022515500625, 0.0000300000000000000004.
189    )
190
191    pageRanks2: expected = (
192      expected,
193      0.0000300000000000000004, 0.00020761099425802734,
0.0000771750000000000002,
194      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
195      0.0014816283407223048, 0.0000300000000000000004,
0.0000771750000000000002,
196      0.00018305958234375, 0.00014299384687500004,
0.00011125893750000003,
197      0.00015412125000000001, 0.0000555000000000000001,
0.0005866682663756834,
198      0.0009038160999648827, 0.0000555000000000000001,
0.0000555000000000000001,
199      0.0000300000000000000004, 0.00014299384687500004,
0.0000300000000000000004,
200      0.0000555000000000000001, 0.0000555000000000000001,
0.0000300000000000000004,
201      0.00012968268750000001, 0.0000300000000000000004,
0.0000300000000000000004,
202      0.000214412971875, 0.0000300000000000000004,
0.0000810000000000000002,
203      0.00016827375000000002, 0.0000555000000000000001,
0.0007612028507183201,

```

```
204      0.00005550000000000001, 0.0000300000000000000004,  
0.0000555000000000000001,  
205      0.0017911603607849024, 0.00012435000000000001, 0.0002074907896875,  
206      0.0000300000000000000004, 0.000620176162217578,  
0.0016205421327844336,  
207      0.0000300000000000000004, 0.0000300000000000000004,  
0.0001772368043671875.  
208  )  
209  
210  pageRanks3: expected = (  
211      expected,  
212      0.0002207814375, 0.0000300000000000000004, 0.0000300000000000000004,  
213      0.00005550000000000001, 0.0009383260308160549,  
0.0000300000000000000004,  
214      0.00007717500000000002, 0.0000300000000000000004,  
0.0000555000000000000001,  
215      0.00007717500000000002, 0.00011727375000000003,  
0.00025645517495238287,  
216      0.00046964726999218764, 0.0000300000000000000004,  
0.0000555000000000000001,  
217      0.0000300000000000000004, 0.00007717500000000002,  
0.0000300000000000000004,  
218      0.00048092971362552716, 0.00022518875783285154,  
0.0002320883721796875,  
219      0.00014550189499218754, 0.0000300000000000000004,  
0.0000300000000000000004,  
220      0.00007717500000000002, 0.0001801089375, 0.0003218540343750001,  
221      0.0000300000000000000004, 0.00024264642949335937,  
0.000146598750000000002,  
222      0.00009885000000000002, 0.00005550000000000001,  
0.0000300000000000000004,  
223      0.00012435000000000001, 0.00005550000000000001,  
0.00009885000000000002,  
224      0.00009559875000000003, 0.00007717500000000002,  
0.00020376888060937495,  
225      0.00005550000000000001, 0.0000300000000000000004,  
0.00005550000000000001,  
226      0.00005550000000000001, 0.00011727375000000003,  
0.00005550000000000001,  
227      0.00023148805436718755, 0.00005550000000000001,  
0.0003663848267566407,  
228      0.0000300000000000000004, 0.00008100000000000002,  
0.0000300000000000000004,  
229      0.00005550000000000001, 0.00020837250000000002,  
0.0000300000000000000004,  
230      0.0000300000000000000004, 0.000128175, 0.00005550000000000001,  
231      0.0007321965646351562, 0.00005550000000000001,  
0.0000300000000000000004,  
232      0.0000300000000000000004, 0.00009559875000000003,  
0.0002918212500000001,  
233      0.00005550000000000001, 0.00007717500000000002,  
0.00018287250000000003,  
234      0.0000300000000000000004, 0.00005550000000000001,  
0.0000300000000000000004,  
235      0.00011727375000000003, 0.00005550000000000001,  
0.0000300000000000000004,  
236      0.0000300000000000000004, 0.0005117003058394531,  
0.0000300000000000000004,  
237      0.0000300000000000000004, 0.0005327917869365431,  
0.0000300000000000000004,  
238      0.00013675893750000003, 0.00007717500000000002,
```

```

0.000030000000000000000004,
239      0.00007717500000000002, 0.000030000000000000000004,
0.000055500000000000000001,
240      0.0008628989988383205, 0.000055500000000000000001,
0.000030000000000000000004,
241      0.00010267500000000002, 0.000030000000000000000004,
0.000030000000000000000004,
242      0.000126919125, 0.00010267500000000002, 0.000030000000000000000004,
243      0.00011125893750000003, 0.000030000000000000000004,
0.000030000000000000000004,
244      0.00007717500000000002, 0.000055500000000000000001,
0.000030000000000000000004,
245      0.000030000000000000000004, 0.00008100000000000002,
0.000030000000000000000004,
246      0.00027766269375, 0.000055500000000000000001, 0.000030000000000000000004.
247  )
248
249  pageRanks4: expected = (
250      expected,
251      0.00018287250000000003, 0.000030000000000000000004, 0.00006825,
252      0.000030000000000000000004, 0.00012052500000000002,
0.000030000000000000000004,
253      0.000055500000000000000001, 0.000030000000000000000004,
0.000030000000000000000004,
254      0.00008100000000000002, 0.00007717500000000002,
0.000055500000000000000001,
255      0.0005574534602777341, 0.0001140225000000000001,
0.000098850000000000000002,
256      0.000030000000000000000004, 0.000030000000000000000004,
0.000055500000000000000001,
257      0.000540919912277207, 0.00008100000000000002,
0.000055500000000000000001,
258      0.00007717500000000002, 0.000030000000000000000004,
0.000055500000000000000001,
259      0.00013588458234375003, 0.000030000000000000000004,
0.00020486333264062499,
260      0.0010616644951779884, 0.000055500000000000000001,
0.00012109875000000002,
261      0.00013894875000000002, 0.0006629172614616794,
0.0015453146662926173,
262      0.000030000000000000000004, 0.0002057215678125,
0.00014277375000000002,
263      0.000030000000000000000004, 0.000030000000000000000004,
0.00012457009687500003,
264      0.000030000000000000000004, 0.000055500000000000000001,
0.000030000000000000000004,
265      0.000030000000000000000004, 0.00022662231875, 0.0002800117218750001,
266      0.000030000000000000000004, 0.000030000000000000000004,
0.000030000000000000000004,
267      0.000030000000000000000004, 0.00012457009687500003,
0.000030000000000000000004,
268      0.000030000000000000000004, 0.000030000000000000000004,
0.000055500000000000000001,
269      0.000030000000000000000004, 0.0009413663007857225,
0.000055500000000000000001,
270      0.000055500000000000000001, 0.000030000000000000000004,
0.000055500000000000000001,
271      0.00025216269375, 0.000030000000000000000004, 0.000055500000000000000001,
272      0.00014299384687500004, 0.000030000000000000000004,
0.0004204378892502149,
273      0.00022315833234375003, 0.001512352057544629,

```



```

0.00040963230479023444,
274      0.00026866422187500004, 0.00009885000000000002,
0.0000300000000000000004,
275      0.0004487230533789453, 0.0000300000000000000004,
0.0001568163804609375,
276      0.00014985, 0.00007717500000000002, 0.00037312038758925784,
277      0.00010267500000000002, 0.00007717500000000002,
0.000935572065964551,
278      0.0000300000000000000004, 0.0000300000000000000004,
0.0006538908974685352,
279      0.00005550000000000001, 0.0000300000000000000004,
0.0006925555784296876,
280      0.00009559875000000003, 0.0000300000000000000004,
0.00005550000000000001,
281      0.0000300000000000000004, 0.000146025, 0.0000300000000000000004,
282      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
283      0.00004275000000000001, 0.0002412005625, 0.00005550000000000001,
284      0.00005550000000000001, 0.000351187703481504,
0.00010267500000000002,
285      0.0000300000000000000004, 0.00019815573560605466, 0.0001356975,
286      0.00007717500000000002, 0.00013588458234375003,
0.0019199758003062113,
287      0.0000300000000000000004, 0.0000300000000000000004,
0.0006042323629701756,
288      0.0000300000000000000004, 0.0008090710683361913,
0.0000300000000000000004.
289      )
290
291      pageRanks5: expected = (
292          expected,
293          0.00021544875, 0.0000300000000000000004, 0.0003507081512421876,
294          0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
295          0.00028123619308710943, 0.00007717500000000002,
0.00022293583249218752,
296          0.0000300000000000000004, 0.00028254900584279304,
0.0004448098933681447,
297          0.0000300000000000000004, 0.0000300000000000000004,
0.00010267500000000002,
298          0.00005550000000000001, 0.000406985079798047,
0.0000300000000000000004,
299          0.00007717500000000002, 0.0000300000000000000004,
0.00005550000000000001,
300          0.00020730370734375002, 0.0000300000000000000004,
0.00008100000000000002,
301          0.00009559875000000003, 0.0000300000000000000004,
0.0006841020911458787,
302          0.0000300000000000000004, 0.0000300000000000000004,
0.00019508699999999998,
303          0.00023373894375, 0.000269050764124043, 0.00007717500000000002,
304          0.0012665335189376758, 0.0003229170826183594,
0.00005550000000000001,
305          0.00010267500000000002, 0.00007717500000000002,
0.0000300000000000000004,
306          0.00005550000000000001, 0.0000300000000000000004,
0.00026877685218750007,
307          0.00007717500000000002, 0.0000300000000000000004,
0.00005550000000000001,
308          0.0000300000000000000004, 0.00035339283501253916,
0.00007717500000000002,

```

```

309      0.00005550000000000001, 0.0000300000000000000004,
0.00007717500000000002,
310      0.000220427784375, 0.0000300000000000000004, 0.000132,
0.00025887973491310547,
311      0.0010559972458375393, 0.00007717500000000002,
0.0000300000000000000004,
312      0.00005550000000000001, 0.00005550000000000001,
0.00008100000000000002,
313      0.0007952039924323437, 0.00009559875000000003,
0.0007619092408695311,
314      0.00003000000000000004, 0.00005550000000000001,
0.00003000000000000004,
315      0.00003000000000000004, 0.00003000000000000004,
0.00005550000000000001,
316      0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
317      0.00003000000000000004, 0.00008100000000000002,
0.00005550000000000001,
318      0.0004741191615037304, 0.00003000000000000004,
0.00003000000000000004,
319      0.00003000000000000004, 0.00005550000000000001,
0.00005550000000000001,
320      0.00044242997182148444, 0.00014277375000000002,
0.00003000000000000004,
321      0.00007717500000000002, 0.00003000000000000004,
0.00003000000000000004,
322      0.0008103090240004296, 0.00003000000000000004,
0.0008127957153562891.
323  )
324
325  pageRanks6: expected = (
326    expected,
327      0.00007717500000000002, 0.0000300000000000000004,
0.0005670410421640625,
328      0.00012435000000000001, 0.0009996132445867678,
0.00005550000000000001,
329      0.000190168846875, 0.00019637074171875002, 0.00005550000000000001,
330      0.0006413808275615429, 0.0008067955708537303,
0.0007001598184784763,
331      0.00007717500000000002, 0.00005550000000000001,
0.00007717500000000002,
332      0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
333      0.00003000000000000004, 0.00019364019374999998,
0.00003000000000000004,
334      0.00003000000000000004, 0.00024570768750000003,
0.00005550000000000001,
335      0.00003000000000000004, 0.00005550000000000001,
0.000816114721648008,
336      0.00016996851984375, 0.000251748159375, 0.00007717500000000002,
337      0.00005550000000000001, 0.00005550000000000001,
0.00005550000000000001,
338      0.00045582617944294916, 0.00003000000000000004,
0.00007717500000000002,
339      0.00009885000000000002, 0.00003000000000000004,
0.00003000000000000004,
340      0.00005550000000000001, 0.00003000000000000004,
0.00003000000000000004,
341      0.00005550000000000001, 0.00003000000000000004,
0.0004087417901358985,
342      0.00003000000000000004, 0.00011125893750000003,

```

```

0.0007096632942357614,
343      0.0004450802223830273, 0.0000300000000000000004,
0.00013788125625000002,
344      0.0000300000000000000004, 0.0000300000000000000004,
0.0007781633581341993,
345      0.0000300000000000000004, 0.0000300000000000000004,
0.0000555000000000000001,
346      0.0000555000000000000001, 0.0000300000000000000004,
0.0007831720377038671,
347      0.000077175000000000002, 0.000146025, 0.0000300000000000000004,
348      0.0004440335706074609, 0.0002797916250000001,
0.000145342875000000002,
349      0.000161417596875, 0.0009436753876707029, 0.00026090069906250005,
350      0.000661540765294531, 0.00029711647746210945,
0.000244833323437501,
351      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
352      0.00021968458249218753, 0.000077175000000000002,
0.000186697500000000003,
353      0.0000300000000000000004, 0.0000300000000000000004,
0.0004852117315546876,
354      0.0003194149033965235, 0.0000300000000000000004,
0.000770995975586406,
355      0.000582465439638496, 0.001229452233367832,
0.000077175000000000002,
356      0.0000300000000000000004, 0.000166433693109375,
0.000969294173404297,
357      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
358      0.0001513576875, 0.000095598750000000003, 0.0000555000000000000001,
359      0.000077175000000000002, 0.0000300000000000000004,
0.00027891726984375005,
360      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004.
361      )
362
363      pageRanks7: expected = (
364          expected,
365          0.0003005925113290822, 0.0000300000000000000004,
0.0000555000000000000001,
366          0.0000300000000000000004, 0.0000300000000000000004,
0.0005980535795580271,
367          0.0000300000000000000004, 0.0000555000000000000001,
0.0006150758829812889,
368          0.0000300000000000000004, 0.00011125893750000003,
0.0000300000000000000004,
369          0.0000555000000000000001, 0.000077175000000000002,
0.00015007009687500002,
370          0.0000300000000000000004, 0.0000555000000000000001,
0.00063777321360625,
371          0.0000300000000000000004, 0.00041378858300664085,
0.0002291377273359375,
372          0.000081000000000000002, 0.0007787174847144531,
0.00011125893750000003,
373          0.000319367505031465, 0.00037540608941236337,
0.0000300000000000000004,
374          0.0008391853330165822, 0.0000300000000000000004, 0.0001952814375,
375          0.000081000000000000002, 0.0000555000000000000001,
0.000098850000000000002,
376          0.0000300000000000000004, 0.0004302220856250001,
0.0000555000000000000001,

```

```

377      0.00007717500000000002, 0.00022148922187500001,
0.0000810000000000000002,
378      0.000168493846875, 0.00005550000000000001, 0.00016444875000000002,
379      0.00022096851984374999, 0.0004387228943070313,
0.0000300000000000000004,
380      0.00018994875000000002, 0.00043396908058710945,
0.0000300000000000000004,
381      0.0000300000000000000004, 0.00119860742323042,
0.00013675893750000003,
382      0.00005550000000000001, 0.00007717500000000002,
0.0000810000000000000002,
383      0.00010267500000000002, 0.0000300000000000000004,
0.0002644775398359375,
384      0.0007738206560650585, 0.0000300000000000000004,
0.0000300000000000000004,
385      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
386      0.0007043146423551758, 0.00005550000000000001,
0.000753936463079121,
387      0.00014277375000000002, 0.00005550000000000001,
0.0000300000000000000004,
388      0.0007755696951406249, 0.0000300000000000000004,
0.00005550000000000001,
389      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004.
390  )
391
392  pageRanks8: expected = (
393      expected,
394      0.0000300000000000000004, 0.00010267500000000002,
0.0000300000000000000004,
395      0.0000300000000000000004, 0.0000300000000000000004,
0.0000810000000000000002,
396      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
397      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
398      0.0000300000000000000004, 0.0007849164249744726,
0.00019180253983593749,
399      0.00013675893750000003, 0.0000300000000000000004,
0.00017895411089179688,
400      0.00013293393750000003, 0.0000300000000000000004,
0.0009846043482525976,
401      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
402      0.0000300000000000000004, 0.0003370130432812501,
0.00005550000000000001,
403      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
404      0.0000300000000000000004, 0.0000300000000000000004,
0.0006119601669933592,
405      0.0005546408271638474, 0.00035661147136054695,
0.0000300000000000000004,
406      0.00008100000000000002, 0.0000300000000000000004,
0.00018305958234375,
407      0.00005550000000000001, 0.0000300000000000000004,
0.0004162326721509962,
408      0.0000300000000000000004, 0.00005550000000000001,
0.000140230284375,
409      0.00007717500000000002, 0.0000300000000000000004,
0.00012435000000000001,

```

```

410      0.0000300000000000000004, 0.00005550000000000001,
0.0000300000000000000004,
411      0.0000300000000000000004, 0.0001752401304609375,
0.0001243500000000000001,
412      0.00005550000000000001, 0.0009916779323594143,
0.0000300000000000000004,
413      0.0001356975, 0.0005381491251261719, 0.000594984439271953,
414      0.0000300000000000000004, 0.00008100000000000002,
0.0009741378132010936,
415      0.0010289986178938673, 0.00009885000000000002,
0.00005550000000000001,
416      0.00005550000000000001, 0.00003000000000000004,
0.00007717500000000002,
417      0.0000300000000000000004, 0.00003000000000000004,
0.0000300000000000000004,
418      0.0000300000000000000004, 0.00003000000000000004,
0.0000300000000000000004,
419      0.0000300000000000000004, 0.00003000000000000004,
0.0003377064978837697,
420      0.0000300000000000000004, 0.00008100000000000002, 0.00014985,
421      0.00007717500000000002, 0.00003000000000000004,
0.00005550000000000001,
422      0.0000300000000000000004, 0.00047936216204097644,
0.00012109875000000002,
423      0.000944214872861504, 0.00005550000000000001,
0.00007717500000000002,
424      0.00010267500000000002, 0.00003000000000000004,
0.0000300000000000000004.
425  )
426
427  pageRanks9: expected = (
428      expected,
429      0.00005550000000000001, 0.0003911262409296876,
0.00016444875000000002,
430      0.0004658189423039455, 0.0006191960454895896,
0.00012109875000000002,
431      0.00008100000000000002, 0.00018994875, 0.0003503022762421876,
432      0.00007717500000000002, 0.00014659875000000002,
0.0000300000000000000004,
433      0.000294136229749043, 0.0001546089375, 0.00003000000000000004,
434      0.00048512037514843756, 0.0000300000000000000004,
0.00010267500000000002,
435      0.00005550000000000001, 0.00003000000000000004,
0.0000300000000000000004,
436      0.00003000000000000004, 0.0009038993515082423,
0.00005550000000000001,
437      0.00003000000000000004, 0.00016792009687500002,
0.00047329793096896486,
438      0.001073084842359102, 0.000227504034375, 0.00007717500000000002,
439      0.0004113979149621094, 0.0000300000000000000004,
0.00005550000000000001,
440      0.0000300000000000000004, 0.00003000000000000004,
0.000994461009736816,
441      0.0000300000000000000004, 0.00003000000000000004,
0.00005550000000000001,
442      0.00009559875000000003, 0.00022421976984375,
0.0004580116392566408,
443      0.00042668597373246103, 0.0000300000000000000004,
0.0005213088860603709,
444      0.00029239500000000004, 0.00003000000000000004,
0.00005550000000000001,

```

```

445      0.0000300000000000000004, 0.000095598750000000003,
0.000117273750000000003,
446      0.000102675000000000002, 0.0000300000000000000004,
0.0004510455397947461,
447      0.000142773750000000002, 0.0000300000000000000004,
0.0000300000000000000004,
448      0.002259585483123663, 0.0010017437556216994,
0.00032997952022560553,
449      0.0000300000000000000004, 0.0000300000000000000004,
0.0000555000000000000001,
450      0.0000300000000000000004, 0.0000300000000000000004,
0.0007017341088929103,
451      0.0000555000000000000001, 0.0005571497378849411,
0.0000300000000000000004,
452      0.0000300000000000000004, 0.0000300000000000000004,
0.00022170931875,
453      0.0006954863535522461, 0.0000555000000000000001,
0.0012527846540595113,
454      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
455      0.0005184859828281249, 0.0000300000000000000004,
0.0000300000000000000004,
456      0.000132933937500000003, 0.000077175000000000002,
0.0008999596475350778,
457      0.00031492419618537123, 0.000081000000000000002,
0.0000300000000000000004,
458      0.0000300000000000000004, 0.0012884279283853029,
0.0000300000000000000004,
459      0.0000300000000000000004, 0.0007648574488383199,
0.0008473654450535153.
460  )
461
462  pageRanks10: expected = (
463      expected,
464      0.000081000000000000002, 0.0000300000000000000004,
0.000180329034375,
465      0.000170842875, 0.0000555000000000000001, 0.000401158314638496,
466      0.0006134211817580272, 0.00015881305436718754,
0.0000300000000000000004,
467      0.0000555000000000000001, 0.000077175000000000002,
0.000081000000000000002,
468      0.0000300000000000000004, 0.000077175000000000002,
0.0000300000000000000004,
469      0.0000300000000000000004, 0.0000555000000000000001,
0.000077175000000000002,
470      0.0005238088503046875, 0.0000555000000000000001,
0.000121098750000000002,
471      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
472      0.000121098750000000002, 0.0000555000000000000001,
0.0000555000000000000001,
473      0.000102675000000000002, 0.0001605102271875, 0.0005087270201025195,
474      0.0009967971475125393, 0.000077175000000000002,
0.000095598750000000003,
475      0.000081000000000000002, 0.0000300000000000000004,
0.0000300000000000000004,
476      0.0000555000000000000001, 0.0000555000000000000001,
0.0000555000000000000001,
477      0.0006039423526103709, 0.0009677339486394532,
0.0000300000000000000004,
478      0.0007675520473719141, 0.0007538680901701758,

```

```

0.000081000000000000002,
479      0.0006169301652431443, 0.000098850000000000002,
0.0005449327357591993,
480      0.00011125893750000003, 0.000124350000000000001,
0.000055500000000000001,
481      0.000055500000000000001, 0.000030000000000000004,
0.000077175000000000002,
482      0.00011125893750000003, 0.00021132074186718748,
0.000077175000000000002,
483      0.000030000000000000004, 0.000030000000000000004,
0.000055500000000000001,
484      0.000077175000000000002, 0.000254291625, 0.000055500000000000001,
485      0.000140230284375, 0.000055500000000000001, 0.0005397573664441406,
486      0.000030000000000000004, 0.0002286781021875,
0.000117273750000000003,
487      0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
488      0.000081000000000000002, 0.000055500000000000001,
0.000030000000000000004,
489      0.000030000000000000004, 0.000055500000000000001,
0.000121098750000000002,
490      0.00014985, 0.0003482766015246095, 0.000077175000000000002,
491      0.00015843393750000003, 0.000030000000000000004,
0.000077175000000000002,
492      0.000030000000000000004, 0.00011125893750000003,
0.00042410386925802743,
493      0.000030000000000000004, 0.000077175000000000002,
0.000030000000000000004,
494      0.000077175000000000002, 0.00037117269868064466,
0.000055500000000000001,
495      0.0001806826875, 0.000055500000000000001, 0.000030000000000000004,
496      0.000081000000000000002, 0.000095598750000000003,
0.000030000000000000004,
497      0.000030000000000000004, 0.000030000000000000004,
0.00025942602609375004.
498      )
499
500      pageRanks11: expected = (
501          expected,
502          0.000030000000000000004, 0.000030000000000000004,
0.000102675000000000002,
503          0.000055500000000000001, 0.000077175000000000002, 0.000245927784375,
504          0.000055500000000000001, 0.0001768576875, 0.000030000000000000004,
505          0.000055500000000000001, 0.000124350000000000001,
0.000030000000000000004,
506          0.000030000000000000004, 0.000055500000000000001,
0.00038169142341406263,
507          0.000030000000000000004, 0.000030000000000000004,
0.000169781437500000002,
508          0.000124350000000000001, 0.000030000000000000004, 0.000218017875,
509          0.0003311542154954884, 0.0023047579029954004,
0.0005710822497378906,
510          0.000077175000000000002, 0.0003164610867890626,
0.000055500000000000001,
511          0.000030000000000000004, 0.00039183545686406263,
0.000030000000000000004,
512          0.000030000000000000004, 0.000102675000000000002,
0.000098850000000000002,
513          0.000030000000000000004, 0.000030000000000000004,
0.000055500000000000001,
514          0.000055500000000000001, 0.0006218280803472657,

```

```

0.00027158680436718753,
515      0.0005329226223534762, 0.0000300000000000000004, 0.0001546089375,
516      0.0000555000000000000001, 0.0000300000000000000004,
0.000121098750000000002,
517      0.0013023196241036914, 0.0000300000000000000004,
0.0005814199422881052,
518      0.0000300000000000000004, 0.0010607720917741798,
0.000111258937500000003,
519      0.0000300000000000000004, 0.0000300000000000000004,
0.0003044889569777344,
520      0.000077175000000000002, 0.0000555000000000000001,
0.0008152654201896286,
521      0.0000555000000000000001, 0.000140230284375, 0.0007082080488527344,
522      0.0007379601076354103, 0.000408482320607461,
0.000129682687500000001,
523      0.0010253114960741213, 0.0000555000000000000001,
0.0000300000000000000004,
524      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
525      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004.
526      )
527
528      pageRanks12: expected = (
529      expected,
530      0.000077175000000000002, 0.000095598750000000003,
0.0000300000000000000004,
531      0.000146025, 0.000205608937500000002, 0.0000300000000000000004,
532      0.0000300000000000000004, 0.000161197500000000003,
0.0000555000000000000001,
533      0.0000300000000000000004, 0.00031838485247324223,
0.0000300000000000000004,
534      0.0000555000000000000001, 0.0014520803604533988,
0.0000300000000000000004,
535      0.0000555000000000000001, 0.000098850000000000002,
0.000364157824359375,
536      0.0000555000000000000001, 0.0013332881297266016,
0.0005557404117852929,
537      0.0000300000000000000004, 0.000102675000000000002,
0.0000300000000000000004,
538      0.0000300000000000000004, 0.0000555000000000000001,
0.0009105278934790038,
539      0.0000300000000000000004, 0.000102675000000000002,
0.0000300000000000000004,
540      0.000095598750000000003, 0.0000555000000000000001,
0.0000300000000000000004,
541      0.0007129847634453126, 0.0000300000000000000004,
0.0000300000000000000004,
542      0.0000300000000000000004, 0.0000555000000000000001,
0.000136758937500000003,
543      0.000950365240418213, 0.0000555000000000000001,
0.0000300000000000000004,
544      0.0000300000000000000004, 0.0000555000000000000001,
0.00014919574171875,
545      0.0000300000000000000004, 0.000102675000000000002,
0.0000300000000000000004,
546      0.0000300000000000000004, 0.0000300000000000000004, 0.0001768576875,
547      0.00043941458861267575, 0.0000555000000000000001,
0.000286185750000000004,
548      0.00020968190109375, 0.000129682687500000001,
0.000121098750000000002,

```



```

549      0.0005675717694396287, 0.0000300000000000000004,
0.0000300000000000000004,
550      0.0000300000000000000004, 0.0002826102781975489,
0.0000300000000000000004,
551      0.00009559875000000003, 0.00005550000000000001,
0.0000300000000000000004.
552  )
553
554  pageRanks13: expected = (
555      expected,
556      0.00009559875000000003, 0.000345610471875,
0.0000300000000000000004,
557      0.0006355078163115429, 0.00005550000000000001,
0.00007717500000000002,
558      0.0003739622193515626, 0.00014299384687500004,
0.00022441180436718748,
559      0.0000300000000000000004, 0.00044359985994126954,
0.0002641154176562501,
560      0.0000300000000000000004, 0.000206951353125,
0.0000555000000000000001,
561      0.00004275000000000001, 0.0000300000000000000004,
0.00008100000000000002,
562      0.0000300000000000000004, 0.00008100000000000002,
0.0000300000000000000004,
563      0.0000300000000000000004, 0.00009559875000000003,
0.00005550000000000001,
564      0.00005550000000000001, 0.00035032903437500005,
0.00008100000000000002,
565      0.0008822237984673634, 0.0007134373373503516,
0.0000300000000000000004,
566      0.0008928574627982618, 0.00018287250000000003,
0.0000300000000000000004,
567      0.00005550000000000001, 0.00005550000000000001,
0.00016827375000000002,
568      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
569      0.00015865403437500001, 0.00008100000000000002,
0.00013293393750000003,
570      0.0000300000000000000004, 0.0000300000000000000004,
0.000511693358096836,
571      0.0000300000000000000004, 0.0000300000000000000004,
0.0008382409215566016,
572      0.0000300000000000000004, 0.00011402250000000001,
0.00005550000000000001,
573      0.0000300000000000000004, 0.00014277375000000002,
0.000590603701889238,
574      0.00011402250000000001, 0.0000300000000000000004,
0.0003913641587121094,
575      0.0000300000000000000004, 0.0000300000000000000004,
0.00012435000000000001,
576      0.0000300000000000000004, 0.0005719241044744334,
0.00005550000000000001,
577      0.0000300000000000000004, 0.00009559875000000003,
0.0000300000000000000004,
578      0.00018612375, 0.00041990431397560547, 0.000128175,
0.00005550000000000001,
579      0.0000300000000000000004, 0.00008100000000000002,
0.00056602218509332,
580      0.00008100000000000002, 0.0000300000000000000004,
0.0000300000000000000004,
581      0.00014919574171875, 0.00005550000000000001,

```

```

0.000030000000000000000004,
582      0.000030000000000000000004, 0.0007010716836625781,
0.000030000000000000000004,
583      0.00027271537500000005, 0.00009559875000000003,
0.000030000000000000000004,
584      0.00015851355328125002, 0.000030000000000000000004,
0.000055500000000000000001,
585      0.000030000000000000000004, 0.0017785457481467384,
0.00035147874703125013,
586      0.000030000000000000000004, 0.000030000000000000000004,
0.000121098750000000002,
587      0.00017087074171875, 0.0000555000000000000001,
0.00029875788750000006,
588      0.00007717500000000002, 0.0000555000000000000001,
0.00018287250000000003,
589      0.00020730370734374997, 0.000030000000000000000004,
0.00046078811633689466,
590      0.00012435000000000001, 0.000030000000000000000004,
0.00010267500000000002.
591      )
592
593      pageRanks14: expected = (
594      expected,
595      0.0011980866815906645, 0.0001897543125, 0.00007717500000000002,
596      0.000030000000000000000004, 0.00007717500000000002,
0.00024398593546875,
597      0.0000555000000000000001, 0.000030000000000000000004,
0.000030000000000000000004,
598      0.0000555000000000000001, 0.000030000000000000000004,
0.000030000000000000000004,
599      0.0000555000000000000001, 0.000030000000000000000004,
0.00009559875000000003,
600      0.0000555000000000000001, 0.000030000000000000000004,
0.0005523386141103321,
601      0.000030000000000000000004, 0.0011369696555575588,
0.00044302649175994153,
602      0.00016225893750000003, 0.000030000000000000000004,
0.0000555000000000000001,
603      0.0000555000000000000001, 0.00009885000000000002,
0.000030000000000000000004,
604      0.0000555000000000000001, 0.00010267500000000002,
0.00034647523371210935,
605      0.0000555000000000000001, 0.0000555000000000000001,
0.00008100000000000002,
606      0.000030000000000000000004, 0.0006266011364933593,
0.0007460569827269042,
607      0.000152419125, 0.0005574828435624999, 0.00045754291748888653,
608      0.0000555000000000000001, 0.00007717500000000002,
0.00012435000000000001,
609      0.0006514320518563474, 0.0007781550062722654,
0.000030000000000000000004,
610      0.00011727375000000003, 0.000030000000000000000004,
0.00016978143750000002,
611      0.0000555000000000000001, 0.0000555000000000000001,
0.000030000000000000000004,
612      0.0000555000000000000001, 0.000030000000000000000004,
0.000030000000000000000004,
613      0.00010267500000000002, 0.000030000000000000000004,
0.00021112870734374997,
614      0.000030000000000000000004, 0.0007241425462692384,
0.000030000000000000000004,

```

```

615      0.00043975632939673826, 0.00026377172812500006,
0.0003192690972632813,
616      0.00007717500000000002, 0.00005550000000000001,
0.001060316398582637,
617      0.00003000000000000004, 0.00016827375, 0.00003000000000000004,
618      0.00008100000000000002, 0.00010267500000000002,
0.00003000000000000004,
619      0.0002944702747961329, 0.00003000000000000004,
0.00005550000000000001,
620      0.000348277539984375, 0.0001546089375, 0.0016954224484599999,
621      0.00003000000000000004, 0.00005550000000000001,
0.000180329034375.
622  )
623
624  pageRanks15: expected = (
625      expected,
626      0.00005550000000000001, 0.00024112459341996093,
0.00003000000000000004,
627      0.00041604693152460934, 0.00007717500000000002,
0.00003000000000000004,
628      0.00003000000000000004, 0.0005496041375411911,
0.00019251787500000001,
629      0.0005949961275392772, 0.0001551826875, 0.000254291625,
630      0.00011727375000000003, 0.00007717500000000002,
0.00012109875000000002,
631      0.00005550000000000001, 0.00015354144375, 0.00015007009687500002,
632      0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
633      0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
634      0.00008100000000000002, 0.00003000000000000004,
0.00027656824171875004,
635      0.00022955245734375, 0.00005550000000000001,
0.00003000000000000004,
636      0.00032534906796093756, 0.00012457009687500003, 0.000215668846875,
637      0.00013894875000000002, 0.000205829034375, 0.0002736906531408399,
638      0.00010267500000000002, 0.00005550000000000001,
0.0007959996360539453,
639      0.00005550000000000001, 0.00005550000000000001,
0.00012052500000000002,
640      0.00003000000000000004, 0.00003000000000000004,
0.00005550000000000001,
641      0.000224252784375, 0.00005550000000000001, 0.00017087074171875,
642      0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
643      0.00005550000000000001, 0.00003000000000000004,
0.00003000000000000004,
644      0.0003375011800228516, 0.00003000000000000004,
0.00005550000000000001,
645      0.00010267500000000002, 0.00003000000000000004,
0.00003000000000000004.
646  )
647
648  pageRanks16: expected = (
649      expected,
650      0.00003000000000000004, 0.00014624509687500003,
0.00005550000000000001,
651      0.00017152500000000004, 0.00003000000000000004,
0.0003609669341552931,
652      0.00007717500000000002, 0.00003000000000000004,
0.00007717500000000002,

```

```

653      0.0000555000000000000001, 0.00003000000000000000004,
0.00003000000000000000004,
654      0.00003000000000000000004, 0.0000555000000000000001,
0.0002042114773359375,
655      0.0008721219106832425, 0.0006369682754531251, 0.0001356975,
656      0.0000555000000000000001, 0.00024838187461171876,
0.000713347958461465,
657      0.0000771750000000000002, 0.00017303268749999998,
0.0000810000000000000002,
658      0.0005128700965591992, 0.00019470905663185548,
0.0000988500000000000002,
659      0.0000810000000000000002, 0.00026435153437500005,
0.0000300000000000000004,
660      0.00028910833249218753, 0.0000555000000000000001,
0.00024367850573554694,
661      0.00015843393750000003, 0.00003000000000000000004,
0.0003614906134310743,
662      0.00003000000000000000004, 0.00035817465068654306,
0.00019702661074335936,
663      0.00003000000000000000004, 0.00003000000000000000004,
0.00031682620734375006,
664      0.00003000000000000000004, 0.00009559875000000003,
0.0009339972378849416,
665      0.00003000000000000000004, 0.00003000000000000000004,
0.00037407066244818355,
666      0.00003000000000000000004, 0.00003000000000000000004,
0.0000555000000000000001,
667      0.00028621361671875, 0.00003000000000000000004,
0.00017557009687500002,
668      0.00003000000000000000004, 0.00017598333234375,
0.0000555000000000000001,
669      0.00003000000000000000004, 0.00003000000000000000004,
0.0005464359416179294,
670      0.00012968268750000001, 0.00003000000000000000004,
0.00003000000000000000004,
671      0.0000988500000000000002, 0.000454723037081924,
0.00003000000000000000004,
672      0.0000988500000000000002, 0.00003000000000000000004,
0.00003000000000000000004,
673      0.00026667045761835943, 0.0010615728509209819,
0.00043192417839472676,
674      0.00014277375000000002, 0.0016763097454050194,
0.0004163827734375001,
675      0.00017174509687500002, 0.00007717500000000002,
0.0005071352788982225,
676      0.00003000000000000000004, 0.0004724088400752538,
0.00003000000000000000004,
677      0.0000555000000000000001, 0.0000555000000000000001,
0.0013976675910372072,
678      0.00003000000000000000004, 0.00003000000000000000004,
0.00003000000000000000004,
679      0.00003000000000000000004, 0.00010267500000000002,
0.00003000000000000000004,
680      0.00010267500000000002, 0.00016827375, 0.0009015184824459963.
681  )
682
683  pageRanks17: expected = (
684      expected,
685      0.00011402250000000001, 0.00003000000000000000004,
0.00020620815124218746,
686      0.0006429872559689648, 0.00009559875000000003,

```

0.000030000000000000000004,
 687 0.0000555000000000000001, 0.000030000000000000000004,
 0.0002761068750000000006,
 688 0.000030000000000000000004, 0.00030289569021933597,
 0.0006080530205542576,
 689 0.000117273750000000003, 0.000030000000000000000004,
 0.000030000000000000000004,
 690 0.0010484356591858599, 0.000030000000000000000004,
 0.0004212905427699219,
 691 0.000077175000000000002, 0.000030000000000000000004,
 0.00043143789891640633,
 692 0.000146598750000000002, 0.000030000000000000000004,
 0.000030000000000000000004,
 693 0.000030000000000000000004, 0.0002032035485179687,
 0.0007551751218261719,
 694 0.000030000000000000000004, 0.0000555000000000000001,
 0.0003794698669933595,
 695 0.000146598750000000002, 0.000077175000000000002,
 0.000030000000000000000004,
 696 0.000030000000000000000004, 0.000117273750000000003,
 0.000030000000000000000004,
 697 0.000030000000000000000004, 0.0004934760543513475,
 0.000030000000000000000004,
 698 0.000030000000000000000004, 0.000030000000000000000004, 0.000128175,
 699 0.0000555000000000000001, 0.0008547118390340233,
 0.000030000000000000000004,
 700 0.0014346082973754881, 0.00021695643750000001,
 0.0000555000000000000001,
 701 0.0015211104302111527, 0.000117273750000000003,
 0.0000555000000000000001,
 702 0.0004392343555509962, 0.000030000000000000000004,
 0.000030000000000000000004,
 703 0.000102675000000000002, 0.0000555000000000000001,
 0.0009416894236694923,
 704 0.000077175000000000002, 0.0000555000000000000001,
 0.00022873191724904294,
 705 0.0008242737845741408, 0.00025836458859375004,
 0.000095598750000000003,
 706 0.000030000000000000000004, 0.00031687600301942386,
 0.000030000000000000000004,
 707 0.000030000000000000000004, 0.000077175000000000002,
 0.0000555000000000000001,
 708 0.000158433937500000003, 0.000390462103125,
 0.000030000000000000000004,
 709 0.0000555000000000000001, 0.0004362053242109375,
 0.00019035592921875,
 710 0.000030000000000000000004, 0.0005769638100933201, 0.00006825,
 711 0.0000555000000000000001, 0.000030000000000000000004,
 0.00022241513046093752,
 712 0.000030000000000000000004, 0.0000555000000000000001,
 0.001006261613162637.
 713)
 714
 715 pageRanks18: expected = (
 716 expected,
 717 0.0007857088789003512, 0.000030000000000000000004,
 0.00025498721718750006,
 718 0.0000555000000000000001, 0.0011432981682779491,
 0.000030000000000000000004,
 719 0.000030000000000000000004, 0.000077175000000000002,
 0.000030000000000000000004,

720 0.00005550000000000001, 0.00007717500000000002,
0.0006757280227431446,
721 0.000600739772793574, 0.00014919574171875,
0.0000300000000000000004,
722 0.0004217083662265626, 0.0000300000000000000004,
0.0021483529248317776,
723 0.00005550000000000001, 0.0006842449449634961,
0.00005550000000000001,
724 0.0000300000000000000004, 0.00013588458234375003,
0.0000300000000000000004,
725 0.0000300000000000000004, 0.00008100000000000002,
0.0000300000000000000004,
726 0.0003539867951484376, 0.0000300000000000000004,
0.0000300000000000000004,
727 0.00005550000000000001, 0.00005550000000000001,
0.0000300000000000000004,
728 0.0000300000000000000004, 0.00009559875000000003,
0.0005256094171827733,
729 0.0000300000000000000004, 0.00014985, 0.00007717500000000002,
730 0.00008100000000000002, 0.0000300000000000000004,
0.0000300000000000000004,
731 0.0006064650511017775, 0.0000300000000000000004,
0.00005550000000000001,
732 0.0000300000000000000004, 0.00023480038124999998,
0.0000300000000000000004,
733 0.000386993224139668, 0.0007025938102384176,
0.00007717500000000002,
734 0.0000300000000000000004, 0.0000300000000000000004,
0.0008015986250978318,
735 0.0011856587198157474, 0.001220542597781563,
0.00005550000000000001,
736 0.001103845676174756, 0.0000300000000000000004,
0.0005001217209945313,
737 0.000164668846875, 0.0000300000000000000004,
0.00021766422187500002,
738 0.00009559875000000003, 0.0002708821616396681,
0.0008902189805322072,
739 0.0002728723524621094, 0.0001928969918671875,
0.0000300000000000000004,
740 0.00005550000000000001, 0.00008100000000000002,
0.00012435000000000001,
741 0.0000300000000000000004, 0.00005550000000000001,
0.0000300000000000000004,
742 0.00020673125625000002, 0.0000300000000000000004,
0.00009559875000000003,
743 0.00012109875000000002, 0.0000300000000000000004,
0.0001795858324921875,
744 0.00014659875000000002, 0.0000300000000000000004,
0.0000300000000000000004,
745 0.0000300000000000000004, 0.0000300000000000000004,
0.0006738107530478319,
746 0.00012435000000000001, 0.0000300000000000000004,
0.0000300000000000000004,
747 0.00005550000000000001, 0.00005550000000000001,
0.0005546722821167576,
748 0.00013675893750000003, 0.00005550000000000001,
0.00007717500000000002,
749 0.0000300000000000000004, 0.00005550000000000001,
0.0000300000000000000004,
750 0.0007719215799593356, 0.00009559875000000003,
0.00021162375000000001,

```
751      0.0000555000000000000001, 0.0000300000000000000004,  
0.000030000000000000000004,  
752      0.0000300000000000000004, 0.0000555000000000000001,  
0.000030000000000000000004,  
753      0.0000555000000000000001, 0.0000300000000000000004,  
0.00012968268750000001.  
754  )  
755  
756  pageRanks19: expected = (  
757      expected,  
758      0.0000555000000000000001, 0.00027164788125, 0.00005550000000000001,  
759      0.0014103401638943945, 0.00015154476984375004,  
0.0005737931960558594,  
760      0.0008436621533431052, 0.00012457009687500003,  
0.00019145643750000002,  
761      0.0000300000000000000004, 0.0000300000000000000004,  
0.0000300000000000000004,  
762      0.000102675000000000002, 0.0000300000000000000004,  
0.0002807485055871094,  
763      0.0005178079962573242, 0.0000300000000000000004,  
0.00042718429824335937,  
764      0.0000300000000000000004, 0.0000300000000000000004,  
0.0000988500000000000002,  
765      0.000081000000000000002, 0.00007717500000000002,  
0.0000300000000000000004,  
766      0.0009962234120309377, 0.0000300000000000000004, 0.0002712076875,  
767      0.0007484966368463475, 0.0000300000000000000004,  
0.0000300000000000000004,  
768      0.00021782324186718748, 0.0007975849074420704,  
0.0000555000000000000001,  
769      0.0014440971395321093, 0.0004298739782575976,  
0.0000300000000000000004,  
770      0.00016827375, 0.0000300000000000000004, 0.00005550000000000001,  
771      0.0000300000000000000004, 0.0000300000000000000004,  
0.0010549562700050393,  
772      0.0000300000000000000004, 0.00009559875000000003,  
0.0000300000000000000004,  
773      0.000126919125, 0.000183092596875, 0.00045689434595333993,  
774      0.0000300000000000000004, 0.0006202840925404099,  
0.0000300000000000000004,  
775      0.0000300000000000000004, 0.0000555000000000000001,  
0.0000300000000000000004,  
776      0.0000555000000000000001, 0.0000300000000000000004,  
0.0000555000000000000001,  
777      0.00041851013033476574, 0.0009673196422284764,  
0.0001140225000000000001,  
778      0.0000300000000000000004, 0.0000555000000000000001,  
0.0000300000000000000004,  
779      0.0000300000000000000004, 0.00024139500000000002,  
0.00015881305436718754,  
780      0.0000300000000000000004, 0.0000300000000000000004,  
0.0000300000000000000004,  
781      0.0000555000000000000001, 0.0005228328479950585,  
0.0000300000000000000004,  
782      0.00021283083234375, 0.0000300000000000000004,  
0.0007436318898950389,  
783      0.0004801294072429297, 0.00010267500000000002,  
0.000502228642298047,  
784      0.0000300000000000000004, 0.000572949911904863,  
0.0000555000000000000001,  
785      0.0000300000000000000004, 0.0000555000000000000001,
```

0.00007717500000000002,
786 0.00005550000000000001, 0.00005550000000000001,
0.0000300000000000000004,
787 0.0000300000000000000004, 0.000269925119280293,
0.0007819586751550391,
788 0.00005550000000000001, 0.0000300000000000000004,
0.0007033901825952341,
789 0.00007717500000000002, 0.000165730284375, 0.00008100000000000002,
790 0.0010263752493400197, 0.0000300000000000000004,
0.00014299384687500004,
791 0.0000300000000000000004, 0.0002240326875, 0.0000300000000000000004,
792 0.0000300000000000000004, 0.0001584339375, 0.00045708025931367194,
793 0.00007717500000000002, 0.00047436583249218747,
0.0000300000000000000004,
794 0.0002730541168671876, 0.0000300000000000000004,
0.00016225893750000003,
795 0.0000300000000000000004, 0.00040197340380373055,
0.0000300000000000000004.
796)
797
798 pageRanks20: expected = (
799 expected,
800 0.000261367875, 0.0000300000000000000004, 0.0000300000000000000004,
801 0.0000555000000000000001, 0.00013894875000000002,
0.0000300000000000000004,
802 0.0002513864773359376, 0.0000555000000000000001,
0.0000300000000000000004,
803 0.00047612431834921886, 0.00007717500000000002,
0.00007717500000000002,
804 0.0000300000000000000004, 0.0006952168221796875,
0.00012109875000000002,
805 0.0000555000000000000001, 0.00007717500000000002,
0.0000300000000000000004,
806 0.001107573809079551, 0.0000300000000000000004,
0.00007717500000000002,
807 0.0000300000000000000004, 0.0000300000000000000004,
0.0000555000000000000001,
808 0.00018431305436718753, 0.0000300000000000000004,
0.000165730284375,
809 0.00030354806250000013, 0.00040802521613302745,
0.0007408627669822461,
810 0.0003570662890816407, 0.00028091394375000006,
0.0000555000000000000001,
811 0.0000300000000000000004, 0.0000300000000000000004,
0.00007717500000000002,
812 0.0001356975, 0.0004396857337185352, 0.0000300000000000000004,
813 0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
814 0.00011125893750000003, 0.0000300000000000000004,
0.0000555000000000000001,
815 0.00008100000000000002, 0.0000300000000000000004,
0.00008100000000000002,
816 0.00038478386950781257, 0.0000300000000000000004,
0.0007459325985840232,
817 0.0003567720534166603, 0.0000300000000000000004,
0.00008100000000000002,
818 0.0001356975, 0.0000555000000000000001, 0.0000300000000000000004,
819 0.0000300000000000000004, 0.00017798000625, 0.0014359547638245706,
820 0.00009559875000000003, 0.0000300000000000000004,
0.0000300000000000000004,
821 0.00046789477015955094, 0.0000300000000000000004,


```

0.00010650000000000001,
822      0.000030000000000000004, 0.000030000000000000004,
0.000226919852484375,
823      0.00022297125, 0.000596758121936328, 0.000030000000000000004,
824      0.0006366992817095507, 0.000030000000000000004,
0.000030000000000000004,
825      0.00010267500000000002, 0.00007717500000000002, 0.00017196519375,
826      0.00007717500000000002, 0.000030000000000000004, 0.0002011018125,
827      0.000030000000000000004, 0.0002167318951183594,
0.000030000000000000004,
828      0.000030000000000000004, 0.000030000000000000004,
0.00012109875000000002,
829      0.0008290133266141014, 0.000030000000000000004, 0.000183580284375,
830      0.00023110893750000005, 0.00005550000000000001,
0.000030000000000000004,
831      0.00028930645988302735, 0.0001649910962121094,
0.00005550000000000001,
832      0.000030000000000000004, 0.0003763601051910157,
0.00010267500000000002,
833      0.000030000000000000004, 0.0001573725, 0.00007717500000000002,
834      0.000030000000000000004, 0.000030000000000000004,
0.00007717500000000002,
835      0.000030000000000000004, 0.00007717500000000002,
0.00005550000000000001,
836      0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
837      0.0004523801286034764, 0.00005550000000000001,
0.000030000000000000004,
838      0.0004951815598534765, 0.00008100000000000002,
0.00010267500000000002.
839      )
840
841      pageRanks21: expected = (
842          expected,
843          0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
844          0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
845          0.000030000000000000004, 0.00010267500000000002,
0.00009559875000000003,
846          0.000030000000000000004, 0.001175440344576426,
0.000030000000000000004,
847          0.000030000000000000004, 0.00035443770990195323,
0.000030000000000000004,
848          0.0003747032963148926, 0.00005550000000000001,
0.0005116082614710938,
849          0.000030000000000000004, 0.000030000000000000004,
0.00021331851984375,
850          0.00007717500000000002, 0.0008077201393309957,
0.000030000000000000004,
851          0.000030000000000000004, 0.000994534956437754,
0.000030000000000000004,
852          0.00011727375000000003, 0.000030000000000000004,
0.0009117984778980079,
853          0.00005550000000000001, 0.00007717500000000002,
0.00009559875000000003,
854          0.000030000000000000004, 0.000030000000000000004,
0.00036226694577816426,
855          0.000030000000000000004, 0.000030000000000000004,
0.00032663461605585945,
856          0.000030000000000000004, 0.000030000000000000004,

```

```

0.000030000000000000000004,
857      0.00005550000000000001, 0.0000300000000000000004,
0.00009559875000000003,
858      0.0006578930123802537, 0.00013293393750000003,
0.000030000000000000000004,
859      0.0000300000000000000004, 0.000146025, 0.0006895069584232615,
860      0.00012968268750000001, 0.0000300000000000000004,
0.00007717500000000002,
861      0.00034956950625, 0.00005550000000000001, 0.00007717500000000002,
862      0.0000300000000000000004, 0.00047335328654826193,
0.000030000000000000000004,
863      0.00005550000000000001, 0.0003485537326737892,
0.00005550000000000001,
864      0.0000300000000000000004, 0.0000300000000000000004,
0.00020124834327152344,
865      0.0000300000000000000004, 0.0005462430679701755,
0.0010715045568435355,
866      0.00007717500000000002, 0.0001356975, 0.00008100000000000002,
867      0.0000300000000000000004, 0.0000300000000000000004,
0.000030000000000000000004,
868      0.00009559875000000003, 0.00005550000000000001,
0.001050337302045796,
869      0.0005798148140722265, 0.0000300000000000000004,
0.00005550000000000001,
870      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
871      0.00008100000000000002, 0.0000300000000000000004,
0.0005860187924074219,
872      0.000695229028248086, 0.00036895237500000005,
0.00005550000000000001,
873      0.00009559875000000003, 0.0000300000000000000004,
0.00010267500000000002,
874      0.00022698333234375003, 0.0000300000000000000004,
0.00018994875000000002,
875      0.00030363530370429697, 0.00005550000000000001,
0.0000300000000000000004.
876      )
877
878      pageRanks22: expected = (
879          expected,
880          0.0000300000000000000004, 0.0012613613445382034,
0.0009061000272912988,
881          0.0000300000000000000004, 0.0000300000000000000004,
0.00044870342936718753,
882          0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
883          0.00005550000000000001, 0.0000300000000000000004,
0.00030216397748437506,
884          0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
885          0.0003228767387781642, 0.00009885000000000002, 0.00016338125625,
886          0.0000300000000000000004, 0.00007717500000000002,
0.0000300000000000000004,
887          0.00005550000000000001, 0.0000300000000000000004,
0.0000300000000000000004,
888          0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
889          0.0000300000000000000004, 0.0005762014488482618,
0.00005550000000000001,
890          0.000252516346875, 0.0000300000000000000004,
0.00007717500000000002,

```

```

891      0.00043890087460225595, 0.0000300000000000000004,
0.0000300000000000000004,
892      0.0000810000000000000002, 0.00016701787500000002,
0.0000771750000000000002,
893      0.0000300000000000000004, 0.0009819741968800392,
0.00047788861671875,
894      0.0000555000000000000001, 0.0000810000000000000002,
0.0000300000000000000004,
895      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
896      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
897      0.0000300000000000000004, 0.001424334953569922,
0.0000300000000000000004,
898      0.0000300000000000000004, 0.0003072850223830275,
0.00019267689499218754,
899      0.0002809656928125, 0.000174094125, 0.0001650225,
0.0000771750000000000002,
900      0.0000300000000000000004, 0.0011707606836439263,
0.0000300000000000000004,
901      0.0000810000000000000002, 0.00014277375000000002,
0.0011823230263219532,
902      0.0000300000000000000004, 0.0000300000000000000004,
0.0000810000000000000002,
903      0.0000300000000000000004, 0.00013244625000000002,
0.0006514792329670507,
904      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
905      0.0000300000000000000004, 0.0000300000000000000004,
0.0003684399653628907.
906  )
907
908  pageRanks23: expected = (
909      expected,
910      0.0000300000000000000004, 0.00038739278765625003,
0.0000300000000000000004,
911      0.00022297125000000003, 0.00028660922160039077,
0.00022708709554921871,
912      0.0000555000000000000001, 0.0000300000000000000004,
0.0006567602159249608,
913      0.0000555000000000000001, 0.0000555000000000000001,
0.0005183511222142771,
914      0.00201874207261627, 0.00037233539805243184,
0.0000555000000000000001,
915      0.0004911306645736522, 0.0000300000000000000004,
0.0000300000000000000004,
916      0.0005789639421744336, 0.0000300000000000000004,
0.0000300000000000000004,
917      0.00023881851984375, 0.0000300000000000000004,
0.0000300000000000000004,
918      0.0000300000000000000004, 0.000258336721875,
0.0000300000000000000004,
919      0.0000300000000000000004, 0.0000300000000000000004,
0.00046646993601609374,
920      0.0000555000000000000001, 0.0000555000000000000001,
0.0000300000000000000004,
921      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
922      0.00015865403437500001, 0.0000300000000000000004,
0.0000300000000000000004,
923      0.0000300000000000000004, 0.0000300000000000000004,

```

```

0.00017174509687500002,
924      0.00005550000000000001, 0.00019164351984375002,
0.0000300000000000000004,
925      0.0000300000000000000004, 0.0004902321882890625,
0.00011402250000000001,
926      0.00012968268750000001, 0.0000300000000000000004,
0.0000300000000000000004,
927      0.00021589500000000002, 0.0000300000000000000004,
0.00005550000000000001,
928      0.0010836219592849413, 0.0010891681019760942,
0.0000300000000000000004,
929      0.00012435000000000001, 0.00005550000000000001,
0.0007079486145866796,
930      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004.
931  )
932
933  pageRanks24: expected = (
934      expected,
935      0.00007717500000000002, 0.0003419444918671875,
0.0000300000000000000004,
936      0.00005550000000000001, 0.001505804961977207,
0.0000300000000000000004,
937      0.00008100000000000002, 0.00032973896826562506,
0.00013675893750000003,
938      0.00007717500000000002, 0.0007428987645248242,
0.0000300000000000000004,
939      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
940      0.00009559875000000003, 0.0000300000000000000004,
0.0004649452599635351,
941      0.000128175, 0.0000300000000000000004, 0.0012289260041186134,
942      0.0005547659253248241, 0.0012735475711920707,
0.0000300000000000000004,
943      0.0000300000000000000004, 0.0006304702346261715,
0.00005550000000000001,
944      0.0000300000000000000004, 0.00010267500000000002,
0.0006962683298203127,
945      0.0015407160072108007, 0.0000300000000000000004,
0.00005550000000000001,
946      0.0000300000000000000004, 0.00007717500000000002,
0.00005550000000000001,
947      0.000575346237916621, 0.0008075947471042188,
0.00005550000000000001,
948      0.00009559875000000003, 0.0000300000000000000004, 0.0001513576875,
949      0.00007717500000000002, 0.00016444875, 0.0000300000000000000004,
950      0.0000300000000000000004, 0.00012109875000000002,
0.0000300000000000000004,
951      0.0012133833776340236, 0.00016701787500000002,
0.0006743516353132419,
952      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
953      0.00028358187446328134, 0.0005657072281472655, 0.000128175,
954      0.00005550000000000001, 0.0000300000000000000004,
0.00010267500000000002,
955      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
956      0.00005550000000000001, 0.0000300000000000000004,
0.00007717500000000002,
957      0.00005550000000000001, 0.00010267500000000002,
0.00010267500000000002,

```

```

958      0.0005714230341351562, 0.0003404881216429688,
0.0005203748888714452,
959      0.0000300000000000000004, 0.00039713555163551777,
0.0003546995317031251,
960      0.0004783518928479491, 0.0000300000000000000004,
0.000120525000000000002,
961      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
962      0.000055500000000000001, 0.00044235231887617196,
0.0000300000000000000004,
963      0.00016444875, 0.00019609428398671873, 0.000126919125,
964      0.0000300000000000000004, 0.0000300000000000000004,
0.000077175000000000002,
965      0.000055500000000000001, 0.000055500000000000001,
0.0000300000000000000004,
966      0.000081000000000000002, 0.0000300000000000000004,
0.0000300000000000000004,
967      0.0001677, 0.0000300000000000000004, 0.000055500000000000001.
968  )
969
970  pageRanks25: expected = (
971      expected,
972      0.0000300000000000000004, 0.0000300000000000000004,
0.00034297342150898445,
973      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
974      0.000055500000000000001, 0.000077175000000000002,
0.00016792009687500002,
975      0.0000300000000000000004, 0.0000300000000000000004,
0.000102675000000000002,
976      0.000128175, 0.00040291446016097664, 0.000077175000000000002,
977      0.0000300000000000000004, 0.000055500000000000001,
0.0000300000000000000004,
978      0.000077175000000000002, 0.00037065834674240237,
0.0000300000000000000004,
979      0.0006004148802589451, 0.0000300000000000000004,
0.000055500000000000001,
980      0.000102675000000000002, 0.00019016884687500003,
0.0000300000000000000004,
981      0.000077175000000000002, 0.000081000000000000002,
0.0006253264456486521,
982      0.000055500000000000001, 0.0000300000000000000004,
0.0010081750473936526,
983      0.0001584339375, 0.00015007009687500002, 0.000425316405686543,
984      0.00032775693480585947, 0.00016792009687500002,
0.000055500000000000001,
985      0.0000300000000000000004, 0.000770613605453125,
0.0000300000000000000004,
986      0.0000300000000000000004, 0.0010328207671150588, 0.0001801089375,
987      0.0000300000000000000004, 0.000055500000000000001,
0.0005880984259152342,
988      0.000098850000000000002, 0.000055500000000000001,
0.0002693696017992188,
989      0.0000300000000000000004, 0.0000300000000000000004,
0.0005425831398122849,
990      0.0000300000000000000004, 0.00028088092921875,
0.000102675000000000002,
991      0.0006764355701484372, 0.000081000000000000002,
0.0003437430909375001,
992      0.0009308938220033006, 0.0009178192984247847,
0.0007719799011481251,

```

```

993      0.00007717500000000002, 0.00013675893750000003,
0.0000300000000000000004,
994      0.0000300000000000000004, 0.0000300000000000000004,
0.0000988500000000000002,
995      0.00014810643750000002, 0.0000300000000000000004,
0.0000300000000000000004,
996      0.0000300000000000000004, 0.0000300000000000000004,
0.0006161561722175781,
997      0.002341625584141152, 0.0002463233892566407,
0.0014496255159246877,
998      0.00039820903892660153, 0.0000300000000000000004,
0.00007717500000000002,
999      0.00005500000000000001, 0.0004494569376072462,
0.00005500000000000001.
1000  )
1001
1002  pageRanks26: expected = (
1003      expected,
1004      0.0007375958831105469, 0.00005500000000000001,
0.0000300000000000000004,
1005      0.00019691513046093753, 0.0000300000000000000004,
0.0006960112701089452,
1006      0.0000300000000000000004, 0.0001356975, 0.00010267500000000002,
1007      0.00012457009687500003, 0.00009559875000000003,
0.00028669488604804693,
1008      0.0005773723220977929, 0.0025455574471148447,
0.00005500000000000001,
1009      0.0000300000000000000004, 0.0000300000000000000004,
0.0008289915908865819,
1010      0.00012435000000000001, 0.000254291625, 0.0000300000000000000004,
1011      0.0000300000000000000004, 0.0003034902985243946,
0.00031878986671875003,
1012      0.0006200913580648829, 0.00089866348489084,
0.0000300000000000000004,
1013      0.00024847125000000003, 0.0000300000000000000004,
0.00044653863884943365,
1014      0.0009588353839884179, 0.0000300000000000000004,
0.00005500000000000001,
1015      0.0004802001794933595, 0.0000300000000000000004,
0.00007717500000000002,
1016      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1017      0.0005993897213728708, 0.0000300000000000000004,
0.0000300000000000000004,
1018      0.00012435000000000001, 0.0006565863834296874,
0.0004414859974607228,
1019      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1020      0.00010267500000000002, 0.0000300000000000000004,
0.00011125893750000003,
1021      0.0000300000000000000004, 0.0000300000000000000004,
0.00046621688976466807,
1022      0.0000300000000000000004, 0.00007717500000000002,
0.0000300000000000000004,
1023      0.0000300000000000000004, 0.00005500000000000001,
0.0000300000000000000004,
1024      0.0000300000000000000004, 0.00005500000000000001,
0.0000300000000000000004,
1025      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1026      0.0003956887727812501, 0.000187405284375, 0.00008100000000000002,

```

```

1027      0.000081000000000000002, 0.0000300000000000000004,
0.0000300000000000000004,
1028      0.0000300000000000000004, 0.000322784383514668,
0.0000555000000000000001,
1029      0.00013588458234375003, 0.00005550000000000001,
0.0000300000000000000004,
1030      0.00013293393750000003, 0.00012109875000000002,
0.0000300000000000000004,
1031      0.0000300000000000000004, 0.00005550000000000001,
0.0000300000000000000004,
1032      0.0000300000000000000004, 0.0002027230162402734,
0.00007717500000000002,
1033      0.0000300000000000000004, 0.00005550000000000001,
0.00011727375000000003,
1034      0.0000300000000000000004, 0.0002919278240625,
0.0000300000000000000004,
1035      0.0000300000000000000004, 0.00007717500000000002,
0.0000300000000000000004.
1036      )
1037
1038      pageRanks27: expected = (
1039      expected,
1040      0.00005550000000000001, 0.0011767052391037309,
0.0000300000000000000004,
1041      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
1042      0.00005550000000000001, 0.0000300000000000000004,
0.0000300000000000000004,
1043      0.00005550000000000001, 0.00007717500000000002,
0.00007717500000000002,
1044      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1045      0.0000300000000000000004, 0.0000300000000000000004, 0.00015361125,
1046      0.0017282723083981836, 0.00005550000000000001,
0.0000300000000000000004,
1047      0.0007795432040711617, 0.00007717500000000002,
0.000081000000000000002,
1048      0.00014985, 0.00005550000000000001, 0.0000300000000000000004,
1049      0.0003018932018828127, 0.00035541894550021494,
0.0000300000000000000004,
1050      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
1051      0.0007141020085688866, 0.0000300000000000000004,
0.0000300000000000000004,
1052      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1053      0.0000300000000000000004, 0.0000300000000000000004,
0.00011125893750000003,
1054      0.001132409535121133, 0.00024089480953125002,
0.00022514327545312499,
1055      0.0000300000000000000004, 0.00005550000000000001,
0.0004311059424464845,
1056      0.000128175, 0.00019035592921875, 0.0000300000000000000004,
0.0001768576875,
1057      0.0009238328865187112, 0.0000300000000000000004,
0.00005550000000000001,
1058      0.000081000000000000002, 0.00012435000000000001,
0.00012109875000000002,
1059      0.00005550000000000001, 0.0000300000000000000004,
0.0001965018949921875,
1060      0.00007717500000000002, 0.00005550000000000001,

```

0.000340432358860547,
1061 0.0000300000000000000004, 0.00009885000000000002,
0.00009885000000000002,
1062 0.0000300000000000000004, 0.00008100000000000002,
0.0006406874962545115,
1063 0.00047452478709895517, 0.0010634564139578517,
0.0000300000000000000004,
1064 0.0000300000000000000004, 0.0000300000000000000004,
0.000164668846875,
1065 0.00007717500000000002, 0.00008100000000000002,
0.0000300000000000000004,
1066 0.0000300000000000000004, 0.00005550000000000001,
0.00014076403125000002,
1067 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004.
1068)
1069
1070 pageRanks28: expected = (
1071 expected,
1072 0.0000300000000000000004, 0.00031425467936718754,
0.0000300000000000000004,
1073 0.0000300000000000000004, 0.00013244625000000002,
0.00011125893750000003,
1074 0.00019459416468749997, 0.001184899122552129,
0.0006817205711949704,
1075 0.00005550000000000001, 0.00007717500000000002,
0.00005550000000000001,
1076 0.0005271594965196095, 0.0010400870586408399,
0.0004752608268892381,
1077 0.0000300000000000000004, 0.00020129625, 0.0000300000000000000004,
1078 0.00064671966671541, 0.00029821537500000001,
0.0000300000000000000004,
1079 0.0000300000000000000004, 0.0003997243869443556,
0.00048734821359374986,
1080 0.0004759093776406251, 0.00011727375000000003,
0.0000300000000000000004,
1081 0.0005440875227589844, 0.00005550000000000001,
0.00019582394999999997,
1082 0.00037308581183710947, 0.00005550000000000001,
0.00005550000000000001,
1083 0.00008100000000000002, 0.00005550000000000001,
0.0000300000000000000004,
1084 0.0004962196445816407, 0.00011727375000000003, 0.000164668846875,
1085 0.00005550000000000001, 0.00013588458234375003,
0.00005550000000000001,
1086 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1087 0.0000300000000000000004, 0.00012109875000000002,
0.00005550000000000001,
1088 0.0000300000000000000004, 0.00007717500000000002,
0.0000300000000000000004,
1089 0.00020380225558710936, 0.00010267500000000002,
0.0000300000000000000004,
1090 0.00009559875000000003, 0.0000300000000000000004,
0.00022728393750000005,
1091 0.00011402250000000001, 0.0000300000000000000004,
0.0006183836193906247,
1092 0.0001730326875, 0.00008100000000000002, 0.0012089610287725784,
1093 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1094 0.0010155726653473343, 0.0000300000000000000004,


```

0.000030000000000000000004,
1095      0.00016119750000000003, 0.000030000000000000000004,
0.000030000000000000000004,
1096      0.0000555000000000000001, 0.0000555000000000000001, 0.000192737971875,
1097      0.000030000000000000000004, 0.0006722068841278708,
0.0010718436839313866,
1098      0.000030000000000000000004, 0.001128313168464766,
0.002193723830047618,
1099      0.00022640958234375002, 0.000030000000000000000004,
0.00025480590886054686,
1100      0.0008077473013638865, 0.000030000000000000000004,
0.000055500000000000000001,
1101      0.000030000000000000000004, 0.000030000000000000000004,
0.000081000000000000000002,
1102      0.0005865207417187499, 0.000030000000000000000004, 0.00017579625,
1103      0.000081000000000000000002, 0.000030000000000000000004,
0.000055500000000000000001,
1104      0.00022315833234375003, 0.0001427737500000000002,
0.000055500000000000000001,
1105      0.0001026750000000000002, 0.0003722072014756055,
0.000030000000000000000004,
1106      0.000030000000000000000004, 0.00017598333234375,
0.0006452517980099412,
1107      0.000174094125, 0.00024649398358593755, 0.000030000000000000000004,
1108      0.00011125893750000003, 0.00044401822041662117,
0.000030000000000000000004.
1109      )
1110
1111      pageRanks29: expected = (
1112          expected,
1113          0.0001210987500000000002, 0.000081000000000000000002,
0.0004345341423233984,
1114          0.0016198246023303518, 0.0000771750000000000002,
0.00024390100558710937,
1115          0.00023174226984375002, 0.0001682737500000000002,
0.000030000000000000000004,
1116          0.00023969287499999998, 0.000030000000000000000004,
0.00029862255421875006,
1117          0.00028843038750000001, 0.000030000000000000000004,
0.000030000000000000000004,
1118          0.000055500000000000000001, 0.0012822416701354098,
0.00011727375000000003,
1119          0.00019875278437500002, 0.0000555000000000000001,
0.000030000000000000000004,
1120          0.000055500000000000000001, 0.00013293393750000003,
0.000030000000000000000004,
1121          0.000030000000000000000004, 0.000231320375, 0.00011727375000000003,
1122          0.000030000000000000000004, 0.00013293393750000003,
0.000030000000000000000004,
1123          0.0000771750000000000002, 0.000030000000000000000004,
0.0006375577699699217,
1124          0.00013293393750000003, 0.0005119848858394532,
0.00012968268750000001,
1125          0.000055500000000000000001, 0.0000988500000000000002,
0.00019747261913185545,
1126          0.000055500000000000000001, 0.000030000000000000000004,
0.000055500000000000000001,
1127          0.0002687487898359375, 0.000030000000000000000004,
0.000055500000000000000001,
1128          0.000030000000000000000004, 0.000030000000000000000004,
0.000055500000000000000001,

```

```

1129      0.00005550000000000001, 0.00007717500000000002,
0.00005550000000000001,
1130      0.00005550000000000001, 0.00010267500000000002,
0.00012052500000000002,
1131      0.00018287250000000003, 0.00003000000000000004,
0.00027437842921875004,
1132      0.00025469880421875006, 0.00021870000000000003,
0.001054292033974219,
1133      0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
1134      0.00008100000000000002, 0.00005550000000000001,
0.0002522608324921875,
1135      0.00017904144375, 0.0000880125, 0.00008100000000000002,
1136      0.0008665907698721879, 0.00009559875000000003,
0.00011727375000000003,
1137      0.00005550000000000001, 0.00005550000000000001,
0.000195405039984375,
1138      0.00003000000000000004, 0.00015154476984375004,
0.00003000000000000004,
1139      0.00005550000000000001, 0.00005550000000000001,
0.00003000000000000004,
1140      0.00005550000000000001, 0.00003000000000000004,
0.00003000000000000004,
1141      0.00009559875000000003, 0.00003000000000000004,
0.0004193817696888867,
1142      0.00003000000000000004, 0.00005550000000000001,
0.00003000000000000004,
1143      0.00007717500000000002, 0.00020620815124218752,
0.00011402250000000001,
1144      0.00034031658501562504, 0.00008100000000000002,
0.00016225893750000003,
1145      0.00003000000000000004, 0.00016444875000000002,
0.00003000000000000004,
1146      0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
1147      0.00003000000000000004, 0.00005550000000000001,
0.00003000000000000004,
1148      0.00012457009687500003, 0.00003000000000000004,
0.00003000000000000004,
1149      0.001323955484927151, 0.00003000000000000004,
0.00005550000000000001,
1150      0.00003000000000000004, 0.0003249621588382814,
0.00003000000000000004,
1151      0.00003000000000000004, 0.002044314609656622,
0.00003000000000000004.
1152      )
1153
1154      pageRanks30: expected = (
1155          expected,
1156          0.00003000000000000004, 0.00003000000000000004,
0.0002445528608917969,
1157          0.00003000000000000004, 0.0005265423747665818, 0.000140230284375,
1158          0.00005550000000000001, 0.00011125893750000003,
0.00003000000000000004,
1159          0.00009559875000000003, 0.00003000000000000004,
0.00035159226984375006,
1160          0.00009885000000000002, 0.00003000000000000004,
0.00003000000000000004,
1161          0.00012435000000000001, 0.00003000000000000004,
0.00003000000000000004,
1162          0.0006776933182608983, 0.00003000000000000004,

```

```

0.000030000000000000000004,
1163      0.000030000000000000000004, 0.000114022500000000001,
0.000030000000000000000004,
1164      0.000111258937500000003, 0.00022071992883046875,
0.000030000000000000000004,
1165      0.000030000000000000000004, 0.000154121250000000001,
0.000030000000000000000004,
1166      0.000030000000000000000004, 0.000030000000000000000004,
0.00005550000000000000001,
1167      0.00005550000000000000001, 0.0003944472271875,
0.000142773750000000002,
1168      0.000030000000000000000004, 0.0003893728637121095,
0.00005550000000000000001,
1169      0.0005825809400868945, 0.000030000000000000000004,
0.00005550000000000000001,
1170      0.00019546851984375, 0.000030000000000000000004,
0.00005550000000000000001,
1171      0.000030000000000000000004, 0.00022476706823554686,
0.000095598750000000003,
1172      0.000077175000000000002, 0.000030000000000000000004,
0.0004692964517755665,
1173      0.000030000000000000000004, 0.000030000000000000000004,
0.000030000000000000000004,
1174      0.000030000000000000000004, 0.000081000000000000002,
0.0006707026050272654,
1175      0.000030000000000000000004, 0.000128175, 0.0010464823826843753,
1176      0.000132933937500000003, 0.00005550000000000000001,
0.00009885000000000000002,
1177      0.000030000000000000000004, 0.000030000000000000000004,
0.00005550000000000000001,
1178      0.00005550000000000000001, 0.000030000000000000000004,
0.000030000000000000000004,
1179      0.000030000000000000000004, 0.00005550000000000000001,
0.000030000000000000000004,
1180      0.00009885000000000000002, 0.000095598750000000003,
0.000111258937500000003,
1181      0.00005550000000000000001, 0.0011688491673253515,
0.00038341672349904315,
1182      0.00033088877587617195, 0.000030000000000000000004,
0.00040050305411792956,
1183      0.0003604310280146681, 0.000168273750000000002,
0.00005550000000000000001,
1184      0.00005550000000000000001, 0.00005550000000000000001,
0.00005550000000000000001,
1185      0.000030000000000000000004, 0.000095598750000000003,
0.000030000000000000000004.
1186      )
1187
1188      pageRanks31: expected = (
1189          expected,
1190          0.00060983427609375, 0.000030000000000000000004,
0.00005550000000000000001,
1191          0.00014985, 0.000030000000000000000004, 0.00048470548947291976,
1192          0.000030000000000000000004, 0.00005550000000000000001, 0.00016827375,
1193          0.000077175000000000002, 0.00005550000000000000001, 0.00020129625,
1194          0.000081000000000000002, 0.00018820518749999997,
0.00022042778437499996,
1195          0.000030000000000000000004, 0.00005550000000000000001,
0.000030000000000000000004,
1196          0.00005550000000000000001, 0.000077175000000000002,
0.0007755538548972265,

```

```

1197      0.0000300000000000000004, 0.0006086423521652343,
0.000102675000000000002,
1198      0.0006501657417822068, 0.0002758531021875, 0.00005550000000000001,
1199      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
1200      0.0009579667596042187, 0.0000300000000000000004,
0.000077175000000000002,
1201      0.0000300000000000000004, 0.0000300000000000000004,
0.000081000000000000002,
1202      0.00021845292093749998, 0.00040177039710841806,
0.00043681785095626963,
1203      0.0003784534574986133, 0.0000300000000000000004, 0.00026525375625,
1204      0.0000300000000000000004, 0.0002016704146875,
0.0016676698321384962,
1205      0.0016026092253459765, 0.0000555000000000000001,
0.000081000000000000002,
1206      0.0000555000000000000001, 0.00011125893750000003,
0.0000300000000000000004,
1207      0.0000300000000000000004, 0.000268820838890625,
0.0000300000000000000004,
1208      0.0000555000000000000001, 0.0000555000000000000001,
0.00040044269215550797,
1209      0.00013788125625000002, 0.0010699961603220117,
0.0000300000000000000004,
1210      0.000146025, 0.000081000000000000002, 0.0001801089375,
0.000927249592168359,
1211      0.00025420596890625007, 0.00030216799120429697,
0.0003564819935984766,
1212      0.0000300000000000000004, 0.0000300000000000000004,
0.0005913996994322457,
1213      0.0000300000000000000004, 0.0000300000000000000004,
0.00031262326397560547,
1214      0.0008478974921164453, 0.0000555000000000000001,
0.0000300000000000000004,
1215      0.00024811154062500003, 0.0000300000000000000004,
0.0000555000000000000001,
1216      0.00009559875000000003, 0.0000300000000000000004,
0.0000555000000000000001,
1217      0.0000300000000000000004, 0.0000300000000000000004,
0.000077175000000000002,
1218      0.000081000000000000002, 0.00046281358580802727,
0.0000555000000000000001,
1219      0.0000300000000000000004, 0.0000300000000000000004,
0.0004620340538085939,
1220      0.0000300000000000000004, 0.000081000000000000002,
0.000102675000000000002,
1221      0.0000300000000000000004, 0.0004924743943451367,
0.0000300000000000000004,
1222      0.0000300000000000000004, 0.0000300000000000000004,
0.0006910466454180467,
1223      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1224      0.0000300000000000000004, 0.0008784156453262499,
0.0000300000000000000004.
1225      )
1226
1227      pageRanks32: expected = (
1228          expected,
1229          0.00009559875000000003, 0.0000555000000000000001,
0.0000300000000000000004,
1230          0.0003290759292187501, 0.00007717500000000002,

```

0.0000300000000000000004,
1231 0.00009885000000000002, 0.0002467391346093751,
0.000770629669361074,
1232 0.0000300000000000000004, 0.00005550000000000001,
0.00035637197413966804,
1233 0.0006510181400392773, 0.0001932, 0.0000300000000000000004,
1234 0.00013293393750000003, 0.00012109875000000002,
0.0000300000000000000004,
1235 0.0000300000000000000004, 0.0005881109660193553,
0.0006697311012689257,
1236 0.0000300000000000000004, 0.0000300000000000000004,
0.0019048124731323435,
1237 0.0000300000000000000004, 0.0012570390195532622,
0.000817076816654082,
1238 0.0000300000000000000004, 0.0000300000000000000004,
0.0000555000000000000001,
1239 0.00037593731782833994, 0.0003176617982433595,
0.0000555000000000000001,
1240 0.0000555000000000000001, 0.0009558905795487891, 0.0001356975,
1241 0.00007717500000000002, 0.002363161256569453,
0.0000300000000000000004,
1242 0.00008100000000000002, 0.0003379288264756056,
0.0000300000000000000004,
1243 0.0000300000000000000004, 0.0000300000000000000004,
0.0000555000000000000001,
1244 0.00010267500000000002, 0.0000300000000000000004,
0.0002601929146875,
1245 0.0002584977130570313, 0.0000300000000000000004,
0.0000300000000000000004,
1246 0.0000300000000000000004, 0.0006118566301355858,
0.00013675893750000003,
1247 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1248 0.0000300000000000000004, 0.00019853268750000003,
0.0000300000000000000004,
1249 0.0000555000000000000001, 0.0000555000000000000001,
0.0000300000000000000004,
1250 0.00008100000000000002, 0.00008100000000000002,
0.0000300000000000000004,
1251 0.0000555000000000000001, 0.0000555000000000000001,
0.0000300000000000000004,
1252 0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
1253 0.00007717500000000002, 0.0000300000000000000004,
0.0000300000000000000004,
1254 0.0000300000000000000004, 0.0000555000000000000001,
0.00011125893750000003,
1255 0.00007717500000000002, 0.00017174509687500002,
0.0000300000000000000004,
1256 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1257 0.0000300000000000000004, 0.00015843393750000003,
0.0000555000000000000001,
1258 0.0000300000000000000004, 0.00012968268750000001,
0.0000555000000000000001,
1259 0.00007717500000000002, 0.00031165253983593755,
0.0004989153507442577,
1260 0.0002587445625, 0.0000555000000000000001, 0.00038396702839960947,
1261 0.00045063995922013693, 0.0009222551727412307,
0.0000300000000000000004,
1262 0.0000555000000000000001, 0.00012457009687500003,

0.000030000000000000000004,
1263 0.000081000000000000002, 0.000081000000000000002,
0.000030000000000000000004,
1264 0.00014330527412167969, 0.000030000000000000000004,
0.000030000000000000000004,
1265 0.000030000000000000000004, 0.00047010862846427726,
0.00035199944906250005,
1266 0.0003710352612187501, 0.000117273750000000003,
0.0012424059491165434.
1267)
1268
1269 pageRanks33: expected = (
1270 expected,
1271 0.000030000000000000000004, 0.000077175000000000002,
0.0004125731167599415,
1272 0.000081000000000000002, 0.0002748134751562501,
0.000030000000000000000004,
1273 0.000030000000000000000004, 0.00023476058810156252,
0.000030000000000000000004,
1274 0.0000555000000000000001, 0.000030000000000000000004,
0.000095598750000000003,
1275 0.000030000000000000000004, 0.000030000000000000000004,
0.000030000000000000000004,
1276 0.000332562522531465, 0.000030000000000000000004,
0.000030000000000000000004,
1277 0.000030000000000000000004, 0.0000555000000000000001,
0.000030000000000000000004,
1278 0.000077175000000000002, 0.000030000000000000000004,
0.000030000000000000000004,
1279 0.0000555000000000000001, 0.000030000000000000000004, 0.00015354144375,
1280 0.000121098750000000002, 0.0002386314375, 0.000030000000000000000004,
1281 0.000030000000000000000004, 0.000030000000000000000004,
0.000030000000000000000004,
1282 0.000077175000000000002, 0.000030000000000000000004,
0.000098850000000000002,
1283 0.000030000000000000000004, 0.0009038323984436134,
0.000077175000000000002,
1284 0.0000555000000000000001, 0.0009495415982433593,
0.00039800205690646495,
1285 0.000030000000000000000004, 0.000095598750000000003,
0.000030000000000000000004,
1286 0.000030000000000000000004, 0.000030000000000000000004,
0.00039053699136197274,
1287 0.0000555000000000000001, 0.00012968268750000001,
0.000030000000000000000004,
1288 0.0006424087503536033, 0.000117273750000000003,
0.0007974356282954882,
1289 0.0000555000000000000001, 0.00013293393750000003,
0.0000555000000000000001,
1290 0.000030000000000000000004, 0.002784270476832207,
0.000077175000000000002,
1291 0.0010764892763057425, 0.000030000000000000000004,
0.000030000000000000000004,
1292 0.000030000000000000000004, 0.0002833907783996095,
0.0007036298547836328,
1293 0.0000555000000000000001, 0.00046026382205075194,
0.0013872956926727733,
1294 0.000077175000000000002, 0.000095598750000000003,
0.000081000000000000002,
1295 0.0000555000000000000001, 0.000077175000000000002,
0.000030000000000000000004,

```

1296      0.0003395341507054688, 0.0000300000000000000004,
0.0005002082528993947,
1297      0.0000300000000000000004, 0.00007717500000000002,
0.0007778580435433202,
1298      0.0000300000000000000004, 0.00005550000000000001,
0.0007483684404247458,
1299      0.00011727375000000003, 0.00009559875000000003,
0.00005550000000000001,
1300      0.0000300000000000000004, 0.0000300000000000000004,
0.00012109875000000002,
1301      0.0006657233743404492, 0.0000300000000000000004,
0.000548018287406992,
1302      0.00042246069565507797, 0.001355053843909375,
0.0008567852761221873,
1303      0.00016119750000000003, 0.00016701787500000002,
0.00007717500000000002,
1304      0.00007717500000000002, 0.00020443397718749996,
0.00011402250000000001,
1305      0.00009559875000000003, 0.00014277375000000002,
0.00005550000000000001,
1306      0.0000300000000000000004, 0.00009559875000000003,
0.0000300000000000000004,
1307      0.0000300000000000000004, 0.00005550000000000001,
0.00005550000000000001,
1308      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1309      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1310      0.00008100000000000002, 0.0000300000000000000004,
0.00009559875000000003.
1311  )
1312
1313  pageRanks34: expected = (
1314      expected,
1315      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
1316      0.00017152500000000004, 0.0000300000000000000004,
0.0005834868354039061,
1317      0.0000300000000000000004, 0.00008100000000000002,
0.00005550000000000001,
1318      0.0000300000000000000004, 0.0000300000000000000004,
0.00028394420437500005,
1319      0.00010267500000000002, 0.00005550000000000001,
0.0000300000000000000004,
1320      0.0000300000000000000004, 0.0003260878243371094,
0.00005550000000000001,
1321      0.0000300000000000000004, 0.0000300000000000000004, 0.000128175,
1322      0.0000300000000000000004, 0.00005550000000000001,
0.0000300000000000000004,
1323      0.0000300000000000000004, 0.0000300000000000000004,
0.000192737971875,
1324      0.00009885000000000002, 0.0005593131134316015,
0.00007717500000000002,
1325      0.0003434709373674024, 0.0000300000000000000004,
0.0000300000000000000004,
1326      0.00033521453983593764, 0.00010267500000000002,
0.00011727375000000003,
1327      0.0010148410295049998, 0.0000300000000000000004,
0.0000300000000000000004,
1328      0.0000300000000000000004, 0.0000300000000000000004,
0.00007717500000000002,

```

1329 0.00009885000000000002, 0.000030000000000000004,
 0.000030000000000000004,
 1330 0.00019744738492187498, 0.00019875278437500002,
 0.00013031062500000001,
 1331 0.000030000000000000004, 0.00020403750000000002,
 0.0005788974922679687,
 1332 0.000030000000000000004, 0.000030000000000000004,
 0.00017521644374999997,
 1333 0.0003113064375, 0.000030000000000000004, 0.00007717500000000002,
 1334 0.000030000000000000004, 0.00040593012538185554,
 0.00015119241562500002,
 1335 0.0020183828489022167, 0.000030000000000000004,
 0.000030000000000000004,
 1336 0.0010626563778712694, 0.00005550000000000001,
 0.00005550000000000001,
 1337 0.000030000000000000004, 0.00005550000000000001,
 0.000030000000000000004,
 1338 0.00005550000000000001, 0.000030000000000000004,
 0.00005550000000000001,
 1339 0.000030000000000000004, 0.00010267500000000002,
 0.000030000000000000004,
 1340 0.0010527559032889554, 0.000030000000000000004,
 0.000030000000000000004,
 1341 0.000030000000000000004, 0.00011727375000000003, 0.00014985,
 1342 0.000030000000000000004, 0.000030000000000000004,
 0.0008434553828328515,
 1343 0.0004977945566349413, 0.00005550000000000001,
 0.000672223484265624,
 1344 0.000030000000000000004, 0.000030000000000000004,
 0.00005550000000000001,
 1345 0.00005550000000000001, 0.000030000000000000004,
 0.0006747054431943555,
 1346 0.000210941625, 0.0008963466727096389, 0.00005550000000000001,
 1347 0.00005550000000000001, 0.000030000000000000004,
 0.000030000000000000004,
 1348 0.0001605102271875, 0.00005550000000000001,
 0.000030000000000000004,
 1349 0.00009559875000000003, 0.000030000000000000004,
 0.000030000000000000004,
 1350 0.000030000000000000004, 0.00023775102609375, 0.0001932,
 1351 0.00005550000000000001, 0.000030000000000000004,
 0.00005550000000000001,
 1352 0.00005550000000000001, 0.000030000000000000004,
 0.00012109875000000002,
 1353 0.000030000000000000004, 0.000030000000000000004,
 0.0013960562834876764,
 1354 0.00009885000000000002, 0.000030000000000000004,
 0.000030000000000000004,
 1355 0.00007717500000000002, 0.00010267500000000002,
 0.00005550000000000001.
 1356)
 1357
 1358 pageRanks35: expected = (
 1359 expected,
 1360 0.00005550000000000001, 0.00009885000000000002,
 0.000030000000000000004,
 1361 0.000030000000000000004, 0.000030000000000000004,
 0.000030000000000000004,
 1362 0.00043246640566427745, 0.0002664178528851563,
 0.000030000000000000004,
 1363 0.000030000000000000004, 0.0005632257407151952,


```

0.00047896735528576165,
1364      0.000632508809124043, 0.0000300000000000000004,
0.0016506434779794729,
1365      0.00024059509687500002, 0.0000300000000000000004, 0.000192517875,
1366      0.0000300000000000000004, 0.0000300000000000000004,
0.00019853268750000003,
1367      0.00005550000000000001, 0.0012807454091226561,
0.0006797589345724413,
1368      0.00019668014138871093, 0.00038820125014843763,
0.00009885000000000002,
1369      0.00005550000000000001, 0.00018010893750000003,
0.0012861574391040038,
1370      0.00019382727609374998, 0.000803065772465078,
0.0000300000000000000004,
1371      0.0000300000000000000004, 0.0004803804035068555,
0.0000300000000000000004,
1372      0.0004718426704453124, 0.00005550000000000001,
0.00010267500000000002,
1373      0.0004208408546006056, 0.0001828725, 0.001204916585841094,
1374      0.000687985769325869, 0.00021094162500000002, 0.00016827375,
1375      0.00008100000000000002, 0.0000300000000000000004,
0.00011727375000000003,
1376      0.0003829687037402735, 0.0000300000000000000004, 0.0002555475,
1377      0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
1378      0.00011402250000000001, 0.00027699795761835936,
0.00010267500000000002,
1379      0.00007717500000000002, 0.0000300000000000000004,
0.0000300000000000000004,
1380      0.0000300000000000000004, 0.00012457009687500003,
0.0000300000000000000004,
1381      0.000181390471875, 0.00007717500000000002, 0.00026633888046093747,
1382      0.0000300000000000000004, 0.0000300000000000000004,
0.0002201722699921875,
1383      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1384      0.00008100000000000002, 0.00005550000000000001,
0.00007717500000000002,
1385      0.00030823986686718754, 0.00007717500000000002,
0.00009559875000000003,
1386      0.0000300000000000000004, 0.00007717500000000002,
0.00011125893750000003,
1387      0.00005550000000000001, 0.00021162375000000001,
0.0000300000000000000004,
1388      0.0000300000000000000004, 0.00010650000000000001,
0.00019303215886054686,
1389      0.00011125893750000003, 0.0000300000000000000004,
0.00009885000000000002,
1390      0.0000300000000000000004, 0.00005550000000000001,
0.00005550000000000001,
1391      0.00026469084375000003, 0.0000300000000000000004,
0.0000300000000000000004.
1392  )
1393
1394  pageRanks36: expected = (
1395    expected,
1396      0.00005550000000000001, 0.0000300000000000000004,
0.00019486807662404297,
1397      0.0000300000000000000004, 0.00005550000000000001,
0.00005550000000000001,
1398      0.001238094285992715, 0.00005550000000000001,

```

```

0.000030000000000000000004,
1399      0.00007717500000000002, 0.0009582389058128515,
0.000030000000000000000004,
1400      0.000055500000000000001, 0.000095598750000000003,
0.000030000000000000000004,
1401      0.00024953268749999997, 0.000030000000000000000004,
0.000030000000000000000004,
1402      0.000055500000000000001, 0.000102675000000000002,
0.000030000000000000000004,
1403      0.00011125893750000003, 0.0001513576875, 0.000030000000000000000004,
1404      0.000030000000000000000004, 0.000030000000000000000004,
0.000102675000000000002,
1405      0.00014257931250000002, 0.000030000000000000000004,
0.001362380613464414,
1406      0.000030000000000000000004, 0.000077175000000000002,
0.00023734384687500003,
1407      0.00031125545949474615, 0.00016887406781249998,
0.000030000000000000000004,
1408      0.00017962125, 0.000030000000000000000004, 0.00032175329238164074,
1409      0.000030000000000000000004, 0.000030000000000000000004,
0.000030000000000000000004,
1410      0.000030000000000000000004, 0.001066234109780723, 0.0001395225,
1411      0.000102675000000000002, 0.00013293393750000003,
0.0004844613860603709,
1412      0.0002699217076183594, 0.000030000000000000000004,
0.00009559875000000003,
1413      0.000030000000000000000004, 0.000030000000000000000004,
0.0005309502408943944,
1414      0.000030000000000000000004, 0.000030000000000000000004, 0.0001356975,
1415      0.00011402250000000001, 0.000030000000000000000004,
0.000030000000000000000004,
1416      0.000030000000000000000004, 0.0010622049395273834,
0.00023004750000000002,
1417      0.000030000000000000000004, 0.0002491371986964844,
0.000030000000000000000004,
1418      0.000124350000000000001, 0.000055500000000000001,
0.00022138175531249998,
1419      0.00007717500000000002, 0.0017936928663086918,
0.00042966960194765636,
1420      0.000030000000000000000004, 0.000030000000000000000004,
0.0009841252989410156,
1421      0.000261941499592793, 0.000030000000000000000004,
0.000055500000000000001,
1422      0.000030000000000000000004, 0.00007717500000000002,
0.000055500000000000001,
1423      0.000030000000000000000004, 0.000055500000000000001,
0.000030000000000000000004,
1424      0.0001932, 0.000030000000000000000004, 0.000030000000000000000004.
1425      )
1426
1427      pageRanks37: expected = (
1428          expected,
1429          0.00010267500000000002, 0.00043106509355255866,
0.0010437505899590234,
1430          0.0002792762970273438, 0.000030000000000000000004,
0.000055500000000000001,
1431          0.000030000000000000000004, 0.00018047532249609375,
0.000055500000000000001,
1432          0.000030000000000000000004, 0.00048641498056345683,
0.00024958621359375,
1433          0.0003643005830224805, 0.000030000000000000000004,

```

```

0.000030000000000000000004,
1434      0.0007504075031519922, 0.000030000000000000000004,
0.0002680099635937501,
1435      0.0000555000000000000001, 0.0003932948035558594,
0.000030000000000000000004,
1436      0.000172545, 0.0003614907562500001, 0.000030000000000000000004,
1437      0.00022164824186718748, 0.0006228411419443552,
0.000030000000000000000004,
1438      0.000030000000000000000004, 0.0005299478300855074,
0.000030000000000000000004,
1439      0.0000771750000000000002, 0.0000555000000000000001,
0.000081000000000000000002,
1440      0.000030000000000000000004, 0.0000555000000000000001,
0.000030000000000000000004,
1441      0.000129682687500000001, 0.0003085340538527344,
0.000198532687500000003,
1442      0.0000555000000000000001, 0.0009653476063435351,
0.000030000000000000000004,
1443      0.0011345384625932814, 0.0008455575109275976,
0.000081000000000000000002,
1444      0.000030000000000000000004, 0.000030000000000000000004,
0.000055500000000000000001,
1445      0.000128175, 0.0000555000000000000001, 0.0000555000000000000001,
1446      0.0000555000000000000001, 0.0004085092553125001,
0.0001026750000000000002,
1447      0.0000555000000000000001, 0.00019645035749999996,
0.000030000000000000000004,
1448      0.000030000000000000000004, 0.000030000000000000000004,
0.000030000000000000000004,
1449      0.0003811410199921876, 0.0000771750000000000002,
0.0012464304642064259,
1450      0.0002083725000000000002, 0.0007725661289884177,
0.0005878685803759568,
1451      0.00015865403437500001, 0.000114022500000000001,
0.00017303268749999998,
1452      0.0001828725, 0.0000988500000000000002, 0.0001205250000000000002,
1453      0.0001026750000000000002, 0.0002678228812500001,
0.000030000000000000000004,
1454      0.000030000000000000000004, 0.0000771750000000000002,
0.000030000000000000000004,
1455      0.0002396928750000000004, 0.0008607545337244333,
0.0000555000000000000001,
1456      0.000030000000000000000004, 0.0000771750000000000002,
0.000030000000000000000004,
1457      0.000030000000000000000004, 0.0000988500000000000002,
0.0000555000000000000001,
1458      0.000030000000000000000004, 0.0001210987500000000002,
0.0000555000000000000001,
1459      0.0000771750000000000002, 0.0000555000000000000001,
0.00026326623179687505,
1460      0.000030000000000000000004, 0.0000555000000000000001,
0.0000555000000000000001,
1461      0.000030000000000000000004, 0.0007833576583764941,
0.000030000000000000000004,
1462      0.000030000000000000000004, 0.0000555000000000000001,
0.0000555000000000000001.
1463      )
1464
1465      pageRanks38: expected = (
1466          expected,
1467          0.0000555000000000000001, 0.0001026750000000000002,

```

0.0005110525401105469,
1468 0.00019871976984375, 0.0000300000000000000004,
0.0000300000000000000004,
1469 0.0003483386168671875, 0.00005550000000000001,
0.00005550000000000001,
1470 0.00012435000000000001, 0.0000300000000000000004,
0.0000300000000000000004,
1471 0.0000300000000000000004, 0.0000300000000000000004,
0.00042438202159986323,
1472 0.00019396244308710937, 0.0000300000000000000004,
0.0000300000000000000004,
1473 0.0000300000000000000004, 0.00019698280387499996,
0.0000300000000000000004,
1474 0.00007717500000000002, 0.00011125893750000003,
0.0000300000000000000004,
1475 0.00008100000000000002, 0.0000300000000000000004,
0.00005550000000000001,
1476 0.00005550000000000001, 0.00014299384687500004,
0.0000300000000000000004,
1477 0.0000300000000000000004, 0.0000300000000000000004,
0.00047081371679001953,
1478 0.0000300000000000000004, 0.0009089884096401953,
0.0000300000000000000004,
1479 0.000224376386859375, 0.00011402250000000001,
0.00005550000000000001,
1480 0.0009667603782073827, 0.0000300000000000000004,
0.00011125893750000003,
1481 0.0000300000000000000004, 0.0000300000000000000004,
0.00005550000000000001,
1482 0.0000300000000000000004, 0.00005550000000000001,
0.001241030682654688,
1483 0.0000300000000000000004, 0.0004823641678046876,
0.00009559875000000003,
1484 0.00021509509687500002, 0.00034904545746992195,
0.0008725268335975779,
1485 0.0000300000000000000004, 0.0004102900824509962,
0.0002089040241216797,
1486 0.0000300000000000000004, 0.0003740019745820314,
0.0012997076386943947,
1487 0.00005550000000000001, 0.00018994875000000002,
0.0006884943091336133,
1488 0.00005550000000000001, 0.00010267500000000002,
0.00012435000000000001,
1489 0.0000300000000000000004, 0.00005550000000000001,
0.0011576753198899027,
1490 0.0000300000000000000004, 0.000649733662027949,
0.0000300000000000000004,
1491 0.0000300000000000000004, 0.000492883398434199,
0.0007212162063932225,
1492 0.000128175, 0.0000300000000000000004, 0.0000300000000000000004,
1493 0.00036528014310937504, 0.0009957260430857229,
0.0000300000000000000004,
1494 0.001329195085524004, 0.0006927975738414062,
0.0000300000000000000004,
1495 0.0000300000000000000004, 0.0000300000000000000004,
0.00008100000000000002,
1496 0.0000300000000000000004, 0.001100317863523008,
0.00005550000000000001,
1497 0.00010267500000000002, 0.0000300000000000000004, 0.000146025,
1498 0.00008100000000000002, 0.0000300000000000000004,
0.0006030464054154491.

```
1499  )
1500
1501  pageRanks39: expected = (
1502    expected,
1503      0.00007717500000000002, 0.00014277375000000002,
1504      0.00011727375000000003,
1505      0.00003000000000000004, 0.0008181163542995311,
1506      0.00003000000000000004,
1507      0.00014277375000000002, 0.00003000000000000004,
1508      0.00003000000000000004,
1509      0.0014100132311993948, 0.00010267500000000002,
1510      0.00009559875000000003,
1511      0.0006551366364775194, 0.00007717500000000002,
1512      0.00009559875000000003,
1513      0.00008100000000000002, 0.00010267500000000002,
1514      0.00005550000000000001,
1515      0.0007480018415248242, 0.0005355432135937501,
1516      0.00007717500000000002,
1517      0.00003000000000000004, 0.00005550000000000001,
1518      0.00003000000000000004,
1519      0.00027271537500000005, 0.00003000000000000004,
1520      0.00003000000000000004,
1521      0.00046871357410369156, 0.00003000000000000004,
1522      0.00005550000000000001,
1523      0.000128175, 0.00003000000000000004, 0.00005550000000000001,
1524      0.00003000000000000004, 0.00009559875000000003,
1525      0.00012968268750000001,
1526      0.00013675893750000003, 0.0006451623310560742,
1527      0.00003000000000000004,
1528      0.00020930038125, 0.00003000000000000004,
1529      0.00003000000000000004,
1530      0.00003000000000000004, 0.00003000000000000004,
1531      0.00003000000000000004,
1532      0.00025474342050342774, 0.00003000000000000004,
1533      0.00003000000000000004,
1534      0.0006523670799398827, 0.00003000000000000004,
1535      0.00014919574171875,
1536      0.00003000000000000004, 0.00009559875000000003,
1537      0.00009559875000000003,
1538      0.000204767596875, 0.00003000000000000004,
1539      0.00009559875000000003,
1540      0.00003000000000000004, 0.00009885000000000002,
1541      0.00007717500000000002,
1542      0.00003000000000000004, 0.00007717500000000002,
1543      0.0004906871529635349,
1544      0.00003000000000000004, 0.00014277375000000002,
1545      0.00003000000000000004,
1546      0.00030763942364748055, 0.00003000000000000004,
1547      0.00003000000000000004,
1548      0.00003000000000000004, 0.00003000000000000004,
1549      0.00003000000000000004,
1550      0.0001677, 0.00005550000000000001, 0.00003000000000000004,
1551      0.00003000000000000004, 0.00003000000000000004,
1552      0.00005550000000000001,
1553      0.00007717500000000002, 0.00005550000000000001,
1554      0.00003000000000000004,
1555      0.00008100000000000002, 0.00008100000000000002,
1556      0.00014810643750000002,
1557      0.00008100000000000002, 0.00003000000000000004,
1558      0.0004704629513872852,
1559      0.000205829034375, 0.00009885000000000002, 0.00019853268750000003,
```

```

1533      0.00012968268750000001, 0.0000300000000000000004,
0.0000300000000000000004,
1534      0.00013588458234375003, 0.0000300000000000000004,
0.0007767642435570312,
1535      0.0000300000000000000004, 0.0006168027873406638,
0.0000300000000000000004,
1536      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004.
1537    )
1538
1539    pageRanks40: expected = (
1540      expected,
1541      0.000081000000000000002, 0.0000300000000000000004, 0.00023329875,
1542      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1543      0.0000300000000000000004, 0.000077175000000000002,
0.0000300000000000000004,
1544      0.0005132322826406251, 0.0005522987924888865,
0.000098850000000000002,
1545      0.000098850000000000002, 0.0002266018125, 0.0015834187107174223,
1546      0.00033898444402460947, 0.0008324012817950196,
0.0000300000000000000004,
1547      0.000180329034375, 0.0008978864941825587, 0.00005550000000000001,
1548      0.00014299384687500004, 0.0001801089375, 0.00005550000000000001,
1549      0.0000300000000000000004, 0.0005692153571463084,
0.0005368253732539843,
1550      0.000055500000000000001, 0.000055500000000000001,
0.0000300000000000000004,
1551      0.0006468409577320312, 0.0000300000000000000004,
0.0003211811310781251,
1552      0.0000300000000000000004, 0.0006582275594396678,
0.0007064921391304686,
1553      0.0000300000000000000004, 0.0000300000000000000004,
0.000106500000000000001,
1554      0.001123349208692012, 0.00013894875000000002,
0.00017174509687500002,
1555      0.000768254994630996, 0.0000300000000000000004,
0.000077175000000000002,
1556      0.0000300000000000000004, 0.001407102531453867,
0.0000300000000000000004,
1557      0.0009750694000490038, 0.000055500000000000001,
0.000055500000000000001,
1558      0.00024731642339179686, 0.00015154476984375004,
0.0000300000000000000004,
1559      0.00035790707901488293, 0.00022954649988632815,
0.0000300000000000000004,
1560      0.0000300000000000000004, 0.000055500000000000001,
0.0000300000000000000004,
1561      0.00027896925010724617, 0.0000300000000000000004,
0.0000300000000000000004,
1562      0.0000300000000000000004, 0.0000300000000000000004,
0.00037515426984375004,
1563      0.0000300000000000000004, 0.00010267500000000002,
0.0000300000000000000004,
1564      0.0000300000000000000004, 0.00007717500000000002,
0.0000300000000000000004,
1565      0.0009195100252817186, 0.00010267500000000002,
0.0011678366105352931,
1566      0.0004596703178347658, 0.0000300000000000000004,
0.0000300000000000000004,
1567      0.0004559106924523828, 0.0000300000000000000004,

```

0.0001823163804609375,
1568 0.0000300000000000000004, 0.0000300000000000000004,
0.001121086223968906,
1569 0.0000300000000000000004, 0.00005550000000000001,
0.00004275000000000001,
1570 0.00007717500000000002, 0.00011125893750000003,
0.0000300000000000000004,
1571 0.00007717500000000002, 0.0000300000000000000004,
0.00014534287500000002,
1572 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004.
1573)
1574
1575 pageRanks41: expected = (
1576 expected,
1577 0.0017556940398932423, 0.00015007009687500002,
0.0000300000000000000004,
1578 0.00011727375000000003, 0.0000300000000000000004,
0.0000300000000000000004,
1579 0.00005550000000000001, 0.00005550000000000001,
0.0000300000000000000004,
1580 0.000168493846875, 0.0000300000000000000004,
0.0000300000000000000004,
1581 0.000636605465767539, 0.0006222834791867967,
0.0000300000000000000004,
1582 0.00016225893750000003, 0.00020495467921875,
0.00019540503998437497,
1583 0.0000300000000000000004, 0.00008100000000000002,
0.00009559875000000003,
1584 0.00008100000000000002, 0.000163766625, 0.0004007745255792969,
1585 0.00005550000000000001, 0.00005550000000000001,
0.00007717500000000002,
1586 0.0000300000000000000004, 0.0000300000000000000004,
0.00023427967921875,
1587 0.00005550000000000001, 0.00005550000000000001,
0.00040445321618064444,
1588 0.0002949046588605469, 0.00052845754028125,
0.0000300000000000000004,
1589 0.0004532404760482618, 0.00028161129623437505,
0.00023373894375000001,
1590 0.0000300000000000000004, 0.00005550000000000001,
0.0012445951359051468,
1591 0.0000300000000000000004, 0.000168493846875, 0.00057090017059375,
1592 0.0000300000000000000004, 0.0005946650262327733,
0.00005550000000000001,
1593 0.00007717500000000002, 0.0001973778608917969,
0.0004028958548752149,
1594 0.00010267500000000002, 0.0000300000000000000004,
0.00012109875000000002,
1595 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1596 0.0004724473638984376, 0.00007717500000000002,
0.00025497271615529296,
1597 0.0000300000000000000004, 0.00005550000000000001,
0.0000300000000000000004,
1598 0.0003436367774209181, 0.0006515472775947068,
0.0000300000000000000004,
1599 0.0007925057115101563, 0.00005550000000000001,
0.0000300000000000000004,
1600 0.0000300000000000000004, 0.0000300000000000000004,
0.00025811662500000007,

```

1601      0.0000555000000000000001, 0.0000300000000000000004,
0.0003093791396384961,
1602      0.0000300000000000000004, 0.0004234529288368946,
0.0005038604170281249,
1603      0.000121098750000000002, 0.000102675000000000002,
0.0001784913804609375,
1604      0.0006189891913622458, 0.000055500000000000001,
0.0000555000000000000001,
1605      0.0000555000000000000001, 0.0000300000000000000004,
0.0000810000000000000002,
1606      0.0001065000000000000001, 0.0000300000000000000004,
0.0000300000000000000004,
1607      0.0000300000000000000004, 0.000055500000000000001,
0.0010110427299035549,
1608      0.0000555000000000000001, 0.000077175000000000002,
0.0012826222617107622,
1609      0.0009419121841931444, 0.000106500000000000001,
0.0000555000000000000001,
1610      0.000132933937500000003, 0.0005449953795705271,
0.0000300000000000000004,
1611      0.0000300000000000000004, 0.0000300000000000000004,
0.000095598750000000003,
1612      0.0000555000000000000001, 0.0000300000000000000004,
0.0000300000000000000004,
1613      0.0000555000000000000001, 0.000098850000000000002,
0.0012708980017750392,
1614      0.0000555000000000000001, 0.00020881269375, 0.00035306008668643563,
1615      0.0000300000000000000004, 0.0005407651450046874, 0.0001356975.
1616  )
1617
1618  pageRanks42: expected = (
1619      expected,
1620      0.0000555000000000000001, 0.0000555000000000000001,
0.00022511452490337893,
1621      0.0000300000000000000004, 0.00021046892339179687,
0.00043287358488302733,
1622      0.0000300000000000000004, 0.00019783027718359373,
0.0000300000000000000004,
1623      0.00031611087217968756, 0.0000555000000000000001,
0.0003520324635937501,
1624      0.000111258937500000003, 0.0000300000000000000004,
0.0000555000000000000001,
1625      0.0004311003574218751, 0.000102675000000000002,
0.0000300000000000000004,
1626      0.0000300000000000000004, 0.0003175084025693555,
0.0011122127446247852,
1627      0.0001243500000000000001, 0.0000300000000000000004,
0.0000300000000000000004,
1628      0.0009516672924987891, 0.0007951100024036912,
0.0000300000000000000004,
1629      0.0000300000000000000004, 0.0000300000000000000004,
0.0004405921018214845,
1630      0.0006293696852640232, 0.0000300000000000000004,
0.0000555000000000000001,
1631      0.0000300000000000000004, 0.000142773750000000002,
0.0000300000000000000004,
1632      0.0000300000000000000004, 0.0000810000000000000002,
0.000111258937500000003,
1633      0.000221269125, 0.0002654379938572461, 0.000055500000000000001,
1634      0.0001065000000000000001, 0.0000300000000000000004,
0.00015154476984375004,

```



```

1635      0.00010267500000000002, 0.0003013663127651465,
0.0000300000000000000004,
1636      0.00009559875000000003, 0.00005550000000000001,
0.0000988500000000000002,
1637      0.0000300000000000000004, 0.00028801585312500014,
0.00051896699171875,
1638      0.0000300000000000000004, 0.0000300000000000000004,
0.0000810000000000000002,
1639      0.00005550000000000001, 0.00010267500000000002,
0.00007717500000000002,
1640      0.00027813865045312505, 0.00005550000000000001,
0.0000300000000000000004,
1641      0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1642      0.0000300000000000000004, 0.0002295803240625, 0.000591690735639453,
1643      0.00040644273358593757, 0.0000300000000000000004,
0.00005550000000000001,
1644      0.0004418071279275587, 0.00005550000000000001,
0.0004835387763812108,
1645      0.001059485402328809, 0.000998035093129512,
0.00012052500000000002,
1646      0.00010267500000000002, 0.0007237203894777343,
0.0000300000000000000004,
1647      0.00005550000000000001, 0.0000300000000000000004,
0.00010267500000000002,
1648      0.00005550000000000001, 0.00021695643750000001,
0.00007717500000000002,
1649      0.00012968268750000001, 0.00012109875000000002,
0.0000300000000000000004,
1650      0.00005550000000000001, 0.0022970199819720605,
0.00005550000000000001,
1651      0.00047827658859375005, 0.00005550000000000001, 0.00006825,
1652      0.0005510104412360739, 0.0010187398686616798,
0.00011125893750000003,
1653      0.00005550000000000001, 0.0000300000000000000004,
0.0000810000000000000002,
1654      0.0000300000000000000004, 0.00012968268750000001,
0.0000300000000000000004,
1655      0.00005550000000000001, 0.00013293393750000003,
0.00017321976984375,
1656      0.00016225893750000003, 0.00019145643750000002,
0.00007717500000000002,
1657      0.00031651012275117196, 0.00014659875000000002,
0.00039644005107951184.
1658      )
1659
1660      pageRanks43: expected = (
1661          expected,
1662          0.0002660132896875, 0.00005550000000000001,
0.00005550000000000001,
1663          0.000152419125, 0.0016891438247597268, 0.00012457009687500003,
1664          0.00016119750000000003, 0.00010267500000000002,
0.0000300000000000000004,
1665          0.00010267500000000002, 0.0009333136792756053,
0.0000300000000000000004,
1666          0.00019528143750000004, 0.00009885000000000002,
0.00005550000000000001,
1667          0.0003461528180871094, 0.0000300000000000000004,
0.00039424190589769544,
1668          0.00033577812302578136, 0.0000300000000000000004,
0.00005550000000000001,

```

1669 0.00007717500000000002, 0.0002877893380792969,
 0.0000300000000000000004,
 1670 0.0000300000000000000004, 0.0002240326875, 0.0000300000000000000004,
 1671 0.00007717500000000002, 0.0000300000000000000004,
 0.0000810000000000000002,
 1672 0.0000300000000000000004, 0.0005661583932553319,
 0.0000300000000000000004,
 1673 0.0000300000000000000004, 0.0000555000000000000001,
 0.0000300000000000000004,
 1674 0.0010921575166542774, 0.0014712415030663285,
 0.0000300000000000000004,
 1675 0.0000300000000000000004, 0.00023186717123437498,
 0.0011153109768279107,
 1676 0.00037087997203593765, 0.0000555000000000000001,
 0.0000810000000000000002,
 1677 0.0000300000000000000004, 0.0000300000000000000004,
 0.0000300000000000000004,
 1678 0.0000300000000000000004, 0.0000555000000000000001,
 0.0005527790258255663,
 1679 0.0000810000000000000002, 0.0000300000000000000004,
 0.0000300000000000000004,
 1680 0.0000300000000000000004, 0.00009559875000000003, 0.000128175,
 1681 0.0000555000000000000001, 0.00009559875000000003,
 0.0003304736308681447,
 1682 0.0000300000000000000004, 0.00011125893750000003,
 0.0018335073900723833,
 1683 0.0000300000000000000004, 0.0000300000000000000004,
 0.0000771750000000000002,
 1684 0.0000300000000000000004, 0.0000300000000000000004,
 0.0000555000000000000001,
 1685 0.0000771750000000000002, 0.00010267500000000002,
 0.0000300000000000000004,
 1686 0.0001513576875, 0.00032466091776349617, 0.0000300000000000000004,
 1687 0.0000300000000000000004, 0.00022403268749999998,
 0.0006712320912878907,
 1688 0.00024917903437500006, 0.0000300000000000000004,
 0.00025780846905468756,
 1689 0.0000300000000000000004, 0.0008138743728837696,
 0.0000300000000000000004,
 1690 0.0000300000000000000004, 0.0005576755996828122,
 0.0006588155804808006,
 1691 0.0000771750000000000002, 0.0005866007615345508,
 0.0000555000000000000001,
 1692 0.00029290500012617196, 0.0007023281972303903,
 0.0000300000000000000004,
 1693 0.0015980741978907033, 0.0000300000000000000004,
 0.0000771750000000000002,
 1694 0.0000555000000000000001, 0.00016827375000000002,
 0.0000300000000000000004,
 1695 0.00010267500000000002, 0.0000300000000000000004,
 0.0000300000000000000004.
 1696)
 1697
 1698 pageRanks44: expected = (
 1699 expected,
 1700 0.0000300000000000000004, 0.00012109875000000002,
 0.0000300000000000000004,
 1701 0.0000555000000000000001, 0.0000300000000000000004,
 0.0009396548305022267,
 1702 0.00012109875000000002, 0.0000300000000000000004,
 0.0000300000000000000004,

1703 0.00009559875000000003, 0.001814793173989063,
0.0000300000000000000004,
1704 0.00009559875000000003, 0.00010267500000000002,
0.0000300000000000000004,
1705 0.002758934391622441, 0.0000300000000000000004,
0.0000300000000000000004,
1706 0.0000300000000000000004, 0.0003839836037885841,
0.0000300000000000000004,
1707 0.0008133136555286917, 0.00005550000000000001,
0.00007717500000000002,
1708 0.0000300000000000000004, 0.00010267500000000002,
0.00038753493443324235,
1709 0.0000300000000000000004, 0.00019609428398671875,
0.001827727958064629,
1710 0.0007766777549231443, 0.002192770461927129,
0.0000300000000000000004,
1711 0.0000300000000000000004, 0.0000300000000000000004,
0.0006964689514820309,
1712 0.0000300000000000000004, 0.00009559875000000003, 0.0001356975,
1713 0.000789890542953125, 0.00005550000000000001,
0.0000300000000000000004,
1714 0.00021036787500000001, 0.0000300000000000000004,
0.00016444875000000002,
1715 0.00008100000000000002, 0.00005550000000000001,
0.00017146863914296875,
1716 0.00009559875000000003, 0.00043470567059683606, 0.000209654034375,
1717 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1718 0.0000300000000000000004, 0.0002679983892566408,
0.0000300000000000000004,
1719 0.000172545, 0.00012109875000000002, 0.00024865833234375003,
1720 0.0000300000000000000004, 0.00007717500000000002,
0.0000300000000000000004,
1721 0.00008100000000000002, 0.00015007009687500002,
0.00005550000000000001,
1722 0.0005220628684277733, 0.0007966927069299414,
0.00018691759687499999,
1723 0.0008192211742001366, 0.0000300000000000000004,
0.0011461627584204495,
1724 0.00012435000000000001, 0.00005550000000000001,
0.0000300000000000000004,
1725 0.00017303268749999998, 0.0005645964837343749,
0.0003273738042187501,
1726 0.00013293393750000003, 0.0000300000000000000004,
0.0011958616266875002,
1727 0.00011727375000000003, 0.0006563778608917968,
0.00038789903437500006,
1728 0.0000300000000000000004, 0.00007717500000000002,
0.00005550000000000001,
1729 0.0000300000000000000004, 0.0000300000000000000004,
0.00014277375000000002,
1730 0.0000300000000000000004, 0.00009559875000000003,
0.0000300000000000000004,
1731 0.0003000125285068556, 0.00009885000000000002, 0.0001945941646875,
1732 0.00007717500000000002, 0.0000300000000000000004,
0.0000300000000000000004,
1733 0.0001611975, 0.0000300000000000000004, 0.00019422,
0.00005550000000000001.
1734)
1735
1736 pageRanks45: expected = (

1737 expected,
1738 0.0000300000000000000004, 0.0000300000000000000004,
0.0003215274082826368,
1739 0.00009559875000000003, 0.00008100000000000002,
0.0005578175908965235,
1740 0.00007717500000000002, 0.00010650000000000001,
0.00007717500000000002,
1741 0.0002650830054609375, 0.0000300000000000000004,
0.00012109875000000002,
1742 0.0003952191616555079, 0.00007717500000000002,
0.00009559875000000003,
1743 0.00010650000000000001, 0.00030403575000000006,
0.00020817806250000002,
1744 0.00011125893750000003, 0.0007993951805930079,
0.0009647913134157617,
1745 0.0000300000000000000004, 0.00007717500000000002,
0.00005550000000000001,
1746 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1747 0.0000300000000000000004, 0.000140230284375,
0.00015843393750000003,
1748 0.0001573725, 0.00022963383274453124, 0.0000300000000000000004,
1749 0.00005550000000000001, 0.0000300000000000000004,
0.00010267500000000002,
1750 0.00005550000000000001, 0.00012109875000000002,
0.0005682106623220115,
1751 0.0000300000000000000004, 0.00007717500000000002,
0.00048354904214099615,
1752 0.000171525, 0.0000300000000000000004, 0.00008100000000000002,
1753 0.00005550000000000001, 0.0000300000000000000004,
0.0011088292365873393,
1754 0.00005550000000000001, 0.00019399384687500003,
0.0000300000000000000004,
1755 0.00007717500000000002, 0.00010267500000000002,
0.0000300000000000000004,
1756 0.000478586195888496, 0.0000300000000000000004,
0.00027036634687500004,
1757 0.00013293393750000003, 0.000985747305325,
0.0000300000000000000004,
1758 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1759 0.00007717500000000002, 0.00005550000000000001,
0.0000300000000000000004,
1760 0.0000300000000000000004, 0.00005550000000000001,
0.0000300000000000000004,
1761 0.00005550000000000001, 0.00005550000000000001,
0.0000300000000000000004,
1762 0.000187405284375, 0.00009559875000000003, 0.00037878833859375006,
1763 0.0000300000000000000004, 0.00025573217936718755,
0.0000300000000000000004,
1764 0.000422222140789454, 0.00005550000000000001,
0.0000300000000000000004,
1765 0.0000300000000000000004, 0.0000300000000000000004,
0.00007717500000000002,
1766 0.00009885000000000002, 0.0000300000000000000004,
0.0000300000000000000004,
1767 0.0008828566442090232, 0.0000300000000000000004,
0.0000300000000000000004,
1768 0.0000300000000000000004, 0.0000300000000000000004,
0.00007717500000000002,
1769 0.0000300000000000000004, 0.0000300000000000000004,

```

0.00005550000000000001,
1770      0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
1771      0.000126919125, 0.000030000000000000004, 0.000030000000000000004,
1772      0.0008706173606547655, 0.000030000000000000004,
0.000030000000000000004.
1773  )
1774
1775  pageRanks46: expected = (
1776      expected,
1777      0.000030000000000000004, 0.000030000000000000004,
0.001097905835669102,
1778      0.000030000000000000004, 0.000030000000000000004,
0.0012911886927233205,
1779      0.00010267500000000002, 0.000030000000000000004,
0.000030000000000000004,
1780      0.0014471317232076367, 0.00007717500000000002,
0.00032919842311718756,
1781      0.0006736807611742187, 0.00005550000000000001,
0.000030000000000000004,
1782      0.000030000000000000004, 0.0005083643571588086,
0.00020877967921875,
1783      0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
1784      0.00005550000000000001, 0.00010267500000000002,
0.00005550000000000001,
1785      0.000030000000000000004, 0.000243164221875, 0.000267602784375,
1786      0.00005550000000000001, 0.000030000000000000004,
0.00047918890923906253,
1787      0.000030000000000000004, 0.0004517866978943946,
0.0015112480414817186,
1788      0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
1789      0.00021090620749218748, 0.0005817136509740037,
0.000030000000000000004,
1790      0.00005550000000000001, 0.0006902496070234765,
0.00005550000000000001,
1791      0.00005550000000000001, 0.000030000000000000004,
0.00005550000000000001,
1792      0.000030000000000000004, 0.00023004750000000002,
0.000030000000000000004,
1793      0.00014810643750000002, 0.000030000000000000004,
0.000030000000000000004,
1794      0.0006170154330910936, 0.000165730284375, 0.00020927027636835936,
1795      0.00042570324346896486, 0.0005386219151164453,
0.00007717500000000002,
1796      0.0013819125144174025, 0.000030000000000000004,
0.0010520113375331644,
1797      0.00009559875000000003, 0.0005797662982845508,
0.00015007009687500002,
1798      0.00005550000000000001, 0.00005550000000000001,
0.00026522826441796886,
1799      0.0001356975, 0.00009885000000000002, 0.0012027005989846294,
1800      0.00005550000000000001, 0.00017685768750000004,
0.00005550000000000001,
1801      0.000030000000000000004, 0.00016119750000000003,
0.000030000000000000004,
1802      0.00011727375000000003, 0.00015755958234375,
0.000030000000000000004,
1803      0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,

```

1804 0.0000300000000000000004, 0.0000300000000000000004,
0.0010036956054313869,
1805 0.0001866975, 0.0000300000000000000004, 0.0000300000000000000004,
1806 0.0000300000000000000004, 0.0004468573575634569,
0.0007568851183328513,
1807 0.00042272894573730486, 0.0000300000000000000004,
0.0000555000000000000001,
1808 0.0003257730341351564, 0.0005692704913527343,
0.00019777118175802734,
1809 0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
1810 0.0000300000000000000004, 0.0000300000000000000004,
0.0002544981050078126,
1811 0.0000300000000000000004, 0.0000300000000000000004,
0.0000555000000000000001,
1812 0.0009998740718150587, 0.0000300000000000000004,
0.0000300000000000000004,
1813 0.00028684279252365243, 0.0000300000000000000004,
0.0000555000000000000001,
1814 0.00022644259687500004, 0.0000555000000000000001,
0.0000300000000000000004,
1815 0.0000300000000000000004, 0.00011125893750000003,
0.00029678490030468754.
1816)
1817
1818 pageRanks47: expected = (
1819 expected,
1820 0.0000300000000000000004, 0.0000300000000000000004,
0.0000771750000000000002,
1821 0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
1822 0.0000555000000000000001, 0.0000300000000000000004,
0.0000555000000000000001,
1823 0.0000300000000000000004, 0.0000555000000000000001,
0.00090625993512791,
1824 0.0000771750000000000002, 0.0000300000000000000004, 0.00021531519375,
1825 0.0004893327048039453, 0.0000810000000000000002,
0.00028449085474904296,
1826 0.0001140225000000000001, 0.0000300000000000000004,
0.0004426083352713087,
1827 0.0000300000000000000004, 0.000168493846875, 0.000225540375,
1828 0.00024176661889201175, 0.0001065000000000000001,
0.00038301536889201183,
1829 0.0003264409174306447, 0.0000300000000000000004,
0.0007187389780083399,
1830 0.0006355851256347263, 0.0000300000000000000004,
0.0000300000000000000004,
1831 0.000944545681565801, 0.00044651261407503907,
0.0001172737500000000003,
1832 0.0000810000000000000002, 0.0005137126510087695,
0.0000771750000000000002,
1833 0.0000300000000000000004, 0.0001392675, 0.0000300000000000000004,
1834 0.0000810000000000000002, 0.0000555000000000000001,
0.0000810000000000000002,
1835 0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1836 0.0000300000000000000004, 0.0001427737500000000002,
0.00020859259687499998,
1837 0.0001806826875, 0.0000955987500000000003, 0.0001210987500000000002,
1838 0.0000300000000000000004, 0.00006825, 0.0001112589375000000003,
1839 0.0001140225000000000001, 0.0002121975000000000002,

```

0.00005550000000000001,
1840      0.0005975487986158006, 0.00005550000000000001,
0.000030000000000000004,
1841      0.0006294703299214255, 0.00010267500000000002,
0.00005550000000000001,
1842      0.000030000000000000004, 0.00024502556250000004,
0.000030000000000000004,
1843      0.00021736361671874997, 0.000030000000000000004,
0.000478267900330039,
1844      0.000030000000000000004, 0.000030000000000000004,
0.0005636191341747459,
1845      0.000030000000000000004, 0.000030000000000000004, 0.0001610030625,
1846      0.0003478678864488282, 0.000030000000000000004,
0.0008226126343261718,
1847      0.00044772343749357427, 0.0010844268495790823,
0.00027328157421093753,
1848      0.000030000000000000004, 0.00007717500000000002,
0.0004377152188212696,
1849      0.00005550000000000001, 0.00007717500000000002,
0.0003810599095457033,
1850      0.000030000000000000004, 0.0007371604510937501,
0.0010461534389142773,
1851      0.00024611486671875007, 0.0008474182917279883,
0.00038903171229613285.
1852      )
1853
1854      pageRanks48: expected = (
1855          expected,
1856          0.00005550000000000001, 0.000030000000000000004,
0.00007717500000000002,
1857          0.0007478200571626757, 0.000030000000000000004,
0.00013293393750000003,
1858          0.0003705205402589845, 0.000030000000000000004,
0.00009559875000000003,
1859          0.00015007009687500002, 0.00026467378371210936,
0.00013330032249609376,
1860          0.00047262199203455097, 0.000168493846875, 0.00009559875000000003,
1861          0.000030000000000000004, 0.00005550000000000001,
0.00019035592921875,
1862          0.00005550000000000001, 0.0004023563457193946,
0.000792128087533164,
1863          0.00005550000000000001, 0.000030000000000000004,
0.0004883296733917968,
1864          0.00005550000000000001, 0.0004559052944855469,
0.000497331361295918,
1865          0.00005550000000000001, 0.000400097954671875,
0.00005550000000000001,
1866          0.000030000000000000004, 0.00005550000000000001,
0.00007717500000000002,
1867          0.0001513576875, 0.000030000000000000004, 0.00011402250000000001,
1868          0.00005550000000000001, 0.00010267500000000002,
0.0005869410960447462,
1869          0.00005550000000000001, 0.00011727375000000003, 0.00014985,
1870          0.00046397884894101556, 0.00013293393750000003,
0.00005550000000000001,
1871          0.000030000000000000004, 0.000030000000000000004,
0.0004985444978679295,
1872          0.00009559875000000003, 0.00029019763671093757, 0.00024614788125,
1873          0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
1874          0.00004275000000000001, 0.00005550000000000001,

```

```

0.0002890909435165822,
1875      0.0000300000000000000004, 0.0009232520230490042,
0.0000771750000000000002,
1876      0.0000771750000000000002, 0.0000300000000000000004, 0.0002699518125,
1877      0.0000300000000000000004, 0.000095598750000000003,
0.0000300000000000000004.
1878      )
1879
1880      pageRanks49: expected = (
1881          expected,
1882          0.0004820662698785156, 0.000055500000000000001,
0.0000300000000000000004,
1883          0.0000300000000000000004, 0.0000300000000000000004, 0.0001584339375,
1884          0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1885          0.000117273750000000003, 0.0004580961996339844,
0.000055500000000000001,
1886          0.0034371926123786135, 0.0007171233431230858,
0.0000300000000000000004,
1887          0.0000300000000000000004, 0.0004940070602542576,
0.0000300000000000000004,
1888          0.0000555000000000000001, 0.0020155847563304292,
0.00026344306061718757,
1889          0.00025103522718749997, 0.00047311409130681634,
0.0000300000000000000004,
1890          0.0000300000000000000004, 0.0000300000000000000004,
0.0000555000000000000001,
1891          0.0000300000000000000004, 0.0000300000000000000004,
0.0000300000000000000004,
1892          0.0005740974556124608, 0.000055500000000000001,
0.000055500000000000001,
1893          0.0000300000000000000004, 0.000121098750000000002,
0.0000555000000000000001,
1894          0.0000300000000000000004, 0.0003655747524131056,
0.0003691095187500001,
1895          0.000128175, 0.0002538039375000001, 0.000185441625,
0.0004509094982433595,
1896          0.0000300000000000000004, 0.000095598750000000003,
0.0000300000000000000004,
1897          0.00012968268750000001, 0.000055500000000000001,
0.0013434953179166213,
1898          0.00019364019375, 0.000081000000000000002, 0.000152419125,
1899          0.0000300000000000000004, 0.000055500000000000001,
0.0000300000000000000004,
1900          0.00036471634687500003, 0.0000300000000000000004,
0.00017557009687500002,
1901          0.0000555000000000000001, 0.0000300000000000000004,
0.000095598750000000003,
1902          0.00016225893750000003, 0.000140230284375, 0.0010686188597835936,
1903          0.0005728017023122849, 0.0000300000000000000004,
0.00028052439499218754,
1904          0.0003154114045719142, 0.0003410776875000001,
0.0000300000000000000004,
1905          0.000077175000000000002, 0.000077175000000000002,
0.000585874040565371,
1906          0.0008652517462578516, 0.0006117752634263864,
0.0006415838093136718,
1907          0.0000555000000000000001, 0.000987794997580957,
0.000077175000000000002,
1908          0.0003370707326484376, 0.000077175000000000002,
0.000077175000000000002,

```


1909 0.0011201223724816604, 0.0000300000000000000004,
0.0000300000000000000004,
1910 0.0000555000000000000001, 0.0000300000000000000004,
0.00041637400038828116,
1911 0.00030097138671093755, 0.0005901878716493943,
0.0009725682330679693,
1912 0.0005054563544739061, 0.000081000000000000002,
0.0001744732418671875,
1913 0.0000300000000000000004, 0.0000555000000000000001,
0.0011511606224924999,
1914 0.0000555000000000000001, 0.00017469574171875,
0.0000300000000000000004,
1915 0.0000555000000000000001, 0.0000555000000000000001,
0.000081000000000000002.
1916)
1917
1918 pageRanks50: expected = (
1919 expected,
1920 0.000077175000000000002, 0.00017174509687500002,
0.0005585642724306445,
1921 0.0004331871046195312, 0.0000300000000000000004,
0.000077175000000000002,
1922 0.0006780410787286913, 0.00019145643750000002,
0.0006755674858472655,
1923 0.0004212642249228322, 0.0000555000000000000001,
0.0000300000000000000004,
1924 0.00020085, 0.0000300000000000000004, 0.0000300000000000000004,
1925 0.0000300000000000000004, 0.00011727375000000003,
0.0000300000000000000004,
1926 0.0000555000000000000001, 0.00017730948300342773,
0.000077175000000000002,
1927 0.0000300000000000000004, 0.000081000000000000002,
0.0000300000000000000004,
1928 0.0000300000000000000004, 0.00012109875000000002,
0.0000300000000000000004,
1929 0.0009763612694554688, 0.00024463889484375003,
0.00021184384687500003,
1930 0.0000300000000000000004, 0.00013675893750000003,
0.000077175000000000002,
1931 0.0000300000000000000004, 0.00036788708859375,
0.0012347261757795606,
1932 0.0000300000000000000004, 0.00023885153437500003, 0.0002056089375,
1933 0.00020476759687499999, 0.0000300000000000000004,
0.0009038579981077738,
1934 0.00020235768750000003, 0.0000300000000000000004,
0.00020859259687500004,
1935 0.0000555000000000000001, 0.00036820402530468756,
0.0000555000000000000001,
1936 0.00066419930620291, 0.00010267500000000002,
0.0003713825254372853,
1937 0.0000555000000000000001, 0.00010267500000000002,
0.0000300000000000000004,
1938 0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
1939 0.0000555000000000000001, 0.0004132730735068556,
0.00011402250000000001,
1940 0.0000300000000000000004, 0.0000300000000000000004,
0.000610537969854375,
1941 0.0000555000000000000001, 0.0000555000000000000001,
0.0000300000000000000004,
1942 0.0000555000000000000001, 0.0000555000000000000001,

```

0.00231323575590872,
1943      0.0000300000000000000004, 0.000102675000000000002,
0.0000555000000000000001,
1944      0.0000300000000000000004, 0.0000300000000000000004,
0.0007635236087597262,
1945      0.0000555000000000000001, 0.0000300000000000000004,
0.000121098750000000002,
1946      0.0000810000000000000002, 0.000077175000000000002,
0.0000300000000000000004,
1947      0.0000300000000000000004, 0.0006835465769273435,
0.000102675000000000002,
1948      0.0000300000000000000004, 0.000111258937500000003,
0.0002674281846796875,
1949      0.00089338520952625, 0.000077175000000000002,
0.0000555000000000000001,
1950      0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
1951      0.000102675000000000002, 0.001359938627840293,
0.0000810000000000000002,
1952      0.000145342875000000002, 0.0000555000000000000001,
0.000102675000000000002,
1953      0.0000300000000000000004, 0.000081000000000000002,
0.00030859339458498054.
1954      )
1955
1956      pageRanks51: expected = (
1957          expected,
1958          0.00025607358488302737, 0.0000300000000000000004,
0.00040411500814865246,
1959          0.0000300000000000000004, 0.000129682687500000001,
0.0000300000000000000004,
1960          0.000077175000000000002, 0.0000300000000000000004,
0.00034456086915078134,
1961          0.0000300000000000000004, 0.0000300000000000000004,
0.000102675000000000002,
1962          0.000117273750000000003, 0.000095598750000000003,
0.0000300000000000000004,
1963          0.0000300000000000000004, 0.0000555000000000000001,
0.0000300000000000000004,
1964          0.000121098750000000002, 0.000132933937500000003,
0.0000300000000000000004,
1965          0.0000300000000000000004, 0.0000300000000000000004,
0.000111258937500000003,
1966          0.0000300000000000000004, 0.0000555000000000000001,
0.0007807470633976951,
1967          0.0000555000000000000001, 0.0000300000000000000004,
0.0000555000000000000001,
1968          0.000077175000000000002, 0.0000555000000000000001,
0.0000555000000000000001,
1969          0.0000300000000000000004, 0.0003173643442031251,
0.000095598750000000003,
1970          0.0000300000000000000004, 0.000106500000000000001,
0.0000300000000000000004,
1971          0.0006441722416652344, 0.00024267653437499997,
0.0000300000000000000004,
1972          0.0000810000000000000002, 0.0000555000000000000001,
0.0000300000000000000004,
1973          0.0000300000000000000004, 0.000111258937500000003,
0.0000300000000000000004,
1974          0.000168273750000000002, 0.0002450773350314649,
0.0000300000000000000004,

```

1975 0.0000300000000000000004, 0.0000300000000000000004,
 0.0005768196484567382,
 1976 0.0000300000000000000004, 0.00007717500000000002,
 0.0000300000000000000004,
 1977 0.0000300000000000000004, 0.0000300000000000000004,
 0.0014809880060556444,
 1978 0.0003982600444914454, 0.0000300000000000000004,
 0.0000555000000000000001,
 1979 0.00007717500000000002, 0.0006242420395359766,
 0.0000300000000000000004,
 1980 0.00016444875, 0.0000300000000000000004, 0.0003311918156250001,
 1981 0.0000300000000000000004, 0.00010267500000000002,
 0.0000300000000000000004,
 1982 0.0000300000000000000004, 0.0008059152801275586,
 0.0012004930497145903,
 1983 0.0000300000000000000004, 0.0004951419985664452,
 0.0004380695932937892,
 1984 0.0000300000000000000004, 0.00010267500000000002,
 0.00011727375000000003,
 1985 0.0000300000000000000004, 0.0000300000000000000004,
 0.0000555000000000000001,
 1986 0.0000300000000000000004, 0.0005296988262009961,
 0.0000300000000000000004,
 1987 0.0000300000000000000004, 0.0000300000000000000004,
 0.0000300000000000000004,
 1988 0.00010650000000000001, 0.0000300000000000000004,
 0.0005620807610697853,
 1989 0.0005286613671070311, 0.0000300000000000000004,
 0.0000300000000000000004,
 1990 0.0000300000000000000004, 0.00005550000000000001,
 0.0000300000000000000004.
 1991)
 1992
 1993 pageRanks52: expected = (
 1994 expected,
 1995 0.00011125893750000003, 0.0000300000000000000004,
 0.0000555000000000000001,
 1996 0.0000300000000000000004, 0.00005550000000000001,
 0.0000300000000000000004,
 1997 0.000760817167551992, 0.0000300000000000000004,
 0.00010650000000000001,
 1998 0.0000300000000000000004, 0.0000300000000000000004,
 0.0003138477416337697,
 1999 0.0000300000000000000004, 0.00005550000000000001,
 0.0000300000000000000004,
 2000 0.0000300000000000000004, 0.0008371103585068553,
 0.00007717500000000002,
 2001 0.0004518288427982617, 0.0000300000000000000004,
 0.0000300000000000000004,
 2002 0.0000300000000000000004, 0.0000300000000000000004, 0.0002608801875,
 2003 0.00005550000000000001, 0.0000300000000000000004,
 0.00005550000000000001,
 2004 0.0000300000000000000004, 0.0000300000000000000004,
 0.00005550000000000001,
 2005 0.000164668846875, 0.00011727375000000003, 0.0003858458493481542,
 2006 0.00007717500000000002, 0.0000300000000000000004,
 0.0000300000000000000004,
 2007 0.0000300000000000000004, 0.001094358015352618,
 0.00035263408031250003,
 2008 0.000370735670154336, 0.00008100000000000002,
 0.0007597855260648826,

2009 0.000030000000000000000004, 0.000030000000000000000004,
 0.000030000000000000000004,
 2010 0.000030000000000000000004, 0.000030000000000000000004,
 0.000098850000000000000002,
 2011 0.0000555000000000000001, 0.00014277375000000002,
 0.00012457009687500003,
 2012 0.0000555000000000000001, 0.00016444875000000002,
 0.000030000000000000000004,
 2013 0.000030000000000000000004, 0.00007717500000000002,
 0.00009559875000000003,
 2014 0.0004551034628453516, 0.00017174509687500002,
 0.00016119750000000003,
 2015 0.0000555000000000000001, 0.000030000000000000000004,
 0.0003727185005080275,
 2016 0.0001243500000000000001, 0.00048311825031271483,
 0.000030000000000000000004,
 2017 0.0012598353609723147, 0.000030000000000000000004,
 0.000030000000000000000004,
 2018 0.00038769586070216803, 0.0000555000000000000001,
 0.00010267500000000002,
 2019 0.00013588458234375003, 0.000568914242375215,
 0.0006243928897521678,
 2020 0.000030000000000000000004, 0.000030000000000000000004,
 0.00007717500000000002,
 2021 0.0010377028924124902, 0.00008100000000000002,
 0.0000555000000000000001.
 2022)
 2023
 2024 pageRanks53: expected = (
 2025 expected,
 2026 0.00010267500000000002, 0.000030000000000000000004,
 0.0005503465372175779,
 2027 0.000030000000000000000004, 0.001355548606467051,
 0.0003556893808617188,
 2028 0.000030000000000000000004, 0.00016827375, 0.000030000000000000000004,
 2029 0.0008246139931943555, 0.000030000000000000000004,
 0.0010154740938932223,
 2030 0.00040518822771779304, 0.00045871095761835937,
 0.0000555000000000000001,
 2031 0.00016701787500000002, 0.002071953365629297,
 0.000030000000000000000004,
 2032 0.000030000000000000000004, 0.000030000000000000000004,
 0.0000555000000000000001,
 2033 0.000030000000000000000004, 0.000030000000000000000004,
 0.0005470457547568554,
 2034 0.0005157095790148827, 0.000030000000000000000004,
 0.000688324568213281,
 2035 0.0006465114741332422, 0.0010292594947507033,
 0.0000555000000000000001,
 2036 0.000030000000000000000004, 0.0000555000000000000001,
 0.0001649910962121094,
 2037 0.000030000000000000000004, 0.00009559875000000003,
 0.00014277375000000002,
 2038 0.00010267500000000002, 0.000030000000000000000004,
 0.00031039666483593755,
 2039 0.0005295671026200583, 0.000030000000000000000004,
 0.00008100000000000002,
 2040 0.0004951876449921876, 0.000146025, 0.0005186043726941406,
 2041 0.0000555000000000000001, 0.00007717500000000002,
 0.00011727375000000003,
 2042 0.0004914393382277344, 0.00030981865964960947,

0.0005172251503359373,
2043 0.00007717500000000002, 0.00014624509687500003,
0.0009398635399813478,
2044 0.0014674101670753515, 0.00005550000000000001,
0.0000300000000000000004,
2045 0.0000300000000000000004.
2046)
2047)
2048

```

1 "
2
3 $Id: Permute.som 31 2009-07-31 12:25:18Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 Permute = Benchmark (
27   | count v |
28   benchmark = (
29     count := 0.
30     v      := Array new: 7.
31     self permute: 6.
32     ^ count
33   )
34   verifyResult: result = (
35     ^ self assert: 8660 equals: result
36   )
37   permute: n = (
38     count := count + 1.
39     (n <> 0)
40     ifTrue: [
41       self permute: n - 1.
42       n downTo: 1 do: [ :i |
43         self swap: n with: i.
44         self permute: n - 1.
45         self swap: n with: i ] ]
46   )
47   swap: i with: j = (
48     | tmp |
49     tmp := v at: i.

```

```
55      v at: i put: (v at: j).  
56      v at: j put: tmp  
57  )  
58  
59 )  
60
```

```
1 Poly1 = (  
2   | sub |  
3   initializeWith: anObject = (  
4     sub := anObject  
5   )  
6  
7   score = (  
8     ^ sub score  
9   )  
10  ----  
11  with: anObject = (  
12    ^ super new initializeWith: anObject  
13  )  
14  
15 )
```



```

1 "Port of https://github.com/dropbox/pyston/blob/master/microbenchmarks/
polymorphism.py
2 # This microbenchmark is inspired by the ICBD type checker, which has a
'scoring' phase at
3 # the end of the analysis.
4 # I'm not sure how representative this file is of the type checker, or
even if
5 # this is really a polymorphism test or maybe just a tree test, or some
cross of the two.
6 "
7
8 Polymorphism = Benchmark (
9   | d |
10   rand = (
11     d := (d * 1.24591 + 0.195) % 1.0.
12     ^ d
13   )
14
15   makeRandom: x = (
16     self rand > x   ifTrue: [^ Simple new].
17     self rand < 0.3 ifTrue: [
18       ^ Union with: (
19         Array with: (self makeRandom: 0.5 * (x - 1))
20         with: (self makeRandom: 0.5 * (x - 1))]
21       ^ Poly1 with: (self makeRandom: x - 1)
22     )
23
24   benchmark = (
25     | r |
26     d := 0.0.
27     r := self makeRandom: 10000.
28     1000 timesRepeat: [r score]
29   )
30 )

```

Examples/Benchmarks/Polymorphism/Simple.som

```
1 Simple = (  
2   score = ( ^ 1.0 )  
3 )
```

```
1 Union = (  
2   | subs |  
3  
4   initializeWith: anArray = ( subs := anArray )  
5   score = (  
6     | t |  
7     t := 0.0.  
8     subs do: [:s | t := t + s score].  
9     t := t // (subs length * subs length).  
10    ^ t  
11  )  
12  
13  ----  
14  with: anArray = (  
15    ^ super new initializeWith: anArray  
16  )  
17 )
```

```
1 "  
2  
3 $Id: Queens.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Queens = Benchmark (  
27  
28   | freeMaxs freeRows freeMins queenRows |  
29  
30   benchmark = (  
31     | result |  
32     result := true.  
33     1 to: 10 do: [ :i | result := result and: self queens ].  
34     ^ result  
35   )  
36  
37   verifyResult: result = (  
38     ^ result  
39   )  
40  
41   queens = (  
42     freeRows := Array new: 8 withAll: true.  
43     freeMaxs := Array new: 16 withAll: true.  
44     freeMins := Array new: 16 withAll: true.  
45     queenRows := Array new: 8 withAll: -1.  
46     ^ self placeQueen: 1  
47   )  
48  
49   placeQueen: c = (  
50     1 to: 8 do: [ :r |  
51       (self row: r column: c)  
52         ifTrue: [  
53           queenRows at: r put: c.  
54           self row: r column: c put: false.
```

```

55             (c = 8) ifTrue: [ ^true ].
56             (self placeQueen: c + 1) ifTrue: [ ^true ].
57             self row: r column: c put: true ] ].
58         ^false
59     )
60
61     row: r column: c = (
62         ^(freeRows at: r) && (freeMaxs at: c + r) && (freeMins at: c - r +
63         8)
64     )
65     row: r column: c put: v = (
66         freeRows at: r          put: v.
67         freeMaxs at: c + r      put: v.
68         freeMins at: c - r + 8 put: v.
69     )
70
71 )
72
73

```

```

1 "
2
3 $Id: QuickSort.som 31 2009-07-31 12:25:18Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 QuickSort = Sort (
27
28     sort: array = (
29         self sort: array low: 1 high: self dataSize.
30         ^ array
31     )
32
33     sort: array low: low high: high = (
34         | pivot i j |
35
36         pivot := array at: (low + high) / 2.
37         i := low.
38         j := high.
39         [ i <= j ]
40         whileTrue: [
41             [ (array at: i) < pivot ] whileTrue: [ i := i + 1 ].
42             [ pivot < (array at: j) ] whileTrue: [ j := j - 1 ].
43             ( i <= j )
44             ifTrue: [
45                 | tmp |
46                 tmp := array at: i.
47                 array at: i put: (array at: j).
48                 array at: j put: tmp.
49                 i := i + 1.
50                 j := j - 1. ] ].
51
52         (low < j) ifTrue: [ self sort: array low: low high: j ].
53         (i < high) ifTrue: [ self sort: array low: i high: high ]
54     )

```

```
55
56     dataSize = ( ^800 )
57
58 )
59
```

```
1 "  
2  
3 $Id: Random.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Random = (  
27   | seed |  
28  
29   initialize = (  
30     seed := 74755  
31   )  
32  
33   next = (  
34     seed := ((seed * 1309) + 13849) & 65535.  
35     ^seed  
36   )  
37  
38   run = (  
39     | fail |  
40     'Testing random number generator ... ' print.  
41     fail := [ 'FAILED:' println. ^nil ].  
42     (self next <> 22896) ifTrue: fail.  
43     (self next <> 34761) ifTrue: fail.  
44     (self next <> 34014) ifTrue: fail.  
45     (self next <> 39231) ifTrue: fail.  
46     (self next <> 52540) ifTrue: fail.  
47     (self next <> 41445) ifTrue: fail.  
48     (self next <> 1546) ifTrue: fail.  
49     (self next <> 5947) ifTrue: fail.  
50     (self next <> 65224) ifTrue: fail.  
51     'PASSED' println  
52   )  
53 )  
54
```



```
55  -----
56
57  | random |
58
59  new          = ( ^super new initialize )
60  next         = ( ^self random next )
61  initialize = ( ^random := Random new )
62
63  random = ( ^random )
64
65 )
66
```

```
1 DeviceTaskDataRecord = RObject (
2   |pending|
3
4   pending = (^pending)
5
6   pending: packet = (pending := packet)
7
8   create = (pending := RObject NoWork)
9 ----
10  create = (
11    ^super new create
12  )
13 )
14
```

```

1 HandlerTaskDataRecord = RObject (
2   |workIn deviceIn|
3
4   deviceIn = (^deviceIn)
5
6   deviceIn: aPacket = (deviceIn := aPacket)
7
8   deviceInAdd: packet = (
9     deviceIn := self append: packet head: deviceIn
10  )
11
12  workIn = (^workIn)
13
14  workIn: aWorkQueue = (workIn := aWorkQueue)
15
16  workInAdd: packet = (
17    workIn := self append: packet head: workIn
18  )
19
20  create = (
21    workIn := deviceIn := RObject NoWork
22  )
23  asString = (
24    ^ 'HandlerTaskDataRecord(' + workIn asString + ', ' + deviceIn
25    asString + ' )'
26  )
27  ----
28  create = (^super new create)
29  )

```

```
1 IdleTaskDataRecord = RObject (
2
3   |control count|
4
5   control = (^control)
6
7   control: aNumber = (
8     control := aNumber
9   )
10
11  count = (^count)
12
13  count: aCount = (
14    count := aCount
15  )
16
17  create = (
18    control := 1.
19    count := 10000
20  )
21
22  ----
23
24  create = (^super new create)
25 )
26
```

```

1 Packet = RObject (
2   | link identity kind datum data |
3
4   data = ( ^data)
5
6   datum = ( ^datum)
7   datum: someData = (datum := someData)
8
9   identity = ( ^identity)
10  identity: anIdentity = ( identity := anIdentity)
11
12  kind = ( ^kind)
13  link = ( ^link)
14
15  link: aWorkQueue = ( link := aWorkQueue )
16  link: aLink identity: anIdentity kind: aKind = (
17      link := aLink.
18      identity := anIdentity.
19      kind := aKind.
20      datum := 1.
21      data := Array new: 4 withAll: 0
22  )
23
24  asString = (
25      ^ 'Packet(' +
26          link asString + ', ' +
27          identity asString + ', ' +
28          kind asString + ', ' +
29          datum asString + ', ' +
30          data asString +
31      ') '
32  )
33
34  ----
35
36  create: link identity: identity kind: kind = (
37      ^super new
38          link: link
39          identity: identity
40          kind: kind
41  )
42 )
43

```

```

1 RBObject = Object (
2
3   append: packet head: queueHead = (
4     | mouse link |
5     packet link: RBObject NoWork.
6     RBObject NoWork == queueHead ifTrue: [^packet].
7     mouse := queueHead.
8     [RBObject NoWork == (link := mouse link)]
9       whileFalse: [mouse := link].
10    mouse link: packet.
11    ^queueHead
12 ' .
13
14 ----
15
16 NoTask  = ( ^ nil )
17 Idler   = ( ^ 1   )
18 NoWork  = ( ^ nil )
19 Worker  = ( ^ 2   )
20 WorkPacketKind = ( ^ 2 )
21 HandlerA = ( ^ 3 )
22 HandlerB = ( ^ 4 )
23 DeviceA  = ( ^ 5 )
24 DeviceB  = ( ^ 6 )
25 DevicePacketKind = ( ^ 1 )
26 )
27

```

```
1 Richards = Benchmark (  
2   benchmark = (  
3     ^ RichardsBenchmarks new reBenchStart.  
4   )  
5   verifyResult: result = (  
6     ^ result  
7   )  
8 )
```

```

1 RichardsBenchmarks = RObject (
2   |taskList currentTask currentTaskIdentity taskTable tracing layout
queuePacketCount holdCount|
3
4   createDevice: identity priority: priority work: work state: state = (
5       | data |
6       data := DeviceTaskDataRecord create.
7       self
8           createTask: identity
9           priority: priority
10          work: work
11          state: state
12          function:
13              [:work :word |
14                  | data functionWork |
15                  data := word.
16                  functionWork := work.
17                  RObject NoWork == functionWork
18                      ifTrue:
19                          [RObject NoWork ==
(functionWork := data pending)
20                              ifTrue: [self wait]
21                              ifFalse:
22                                  [data pending:
RObject NoWork.
23                                      self queuePacket:
functionWork]]
24                      ifFalse:
25                          [data pending: functionWork.
26                          tracing
27                              ifTrue:
28                                  [self trace:
functionWork datum].
29                              self holdSelf]]
30          data: data
31   )
32
33   createHandler: identity priority: priority work: work state: state = (
34       | data |
35       data := HandlerTaskDataRecord create.
36       self
37           createTask: identity
38           priority: priority
39           work: work
40           state: state
41           function:
42               [:work :word |
43                   | data workPacket |
44                   data := word.
45                   RObject NoWork == work
46                       ifFalse: [RObject WorkPacketKind == work
kind
47                           ifTrue: [data workInAdd: work]
48                           ifFalse: [data deviceInAdd:
work]].
49                   RObject NoWork == (workPacket := data workIn)
50                       ifTrue: [self wait]
51                       ifFalse:

```



```

52                                     [ |count|
53                                     count := workPacket datum.
54                                     count > 4
55                                     ifTrue:
56                                     [data workIn:
workPacket link.                                     self queuePacket:
57                                     self queuePacket:
workPacket]
58                                     ifFalse:
59                                     [ | devicePacket |
60                                     RObject NoWork
== (devicePacket := data deviceIn)
61                                     ifTrue:
[self wait]
62                                     ifFalse:
63
64 [data deviceIn: devicePacket link.
65
66 devicePacket datum: (workPacket data at: count).
67
68 workPacket datum: count + 1.
69
70 self queuePacket: devicePacket]]]]
71                                     data: data
72                                     )
73
74 createIdler: identity priority: priority work: work state: state = (
75     | data |
76     data := IdleTaskDataRecord create.
77     self
78     createTask: identity
79     priority: priority
80     work: work
81     state: state
82     function:
83     [:work :word |
84     | data |
85     data := word.
86     data count: data count - 1.
87     0 = data count
88     ifTrue: [self holdSelf]
89     ifFalse:
90     [0 = (data control & 1)
91     ifTrue:
92     [data control:
93     self release:
RObject DeviceA]
94     ifFalse:
95     [data control:
96     self release:
(data control / 2 bitXor: 53256).
97
98 RObject DeviceB]]]
99                                     data: data
100                                     )
101
102 createPacket: link identity: identity kind: kind = (
103     ^Packet
104     create: link
105     identity: identity
106     kind: kind

```

```

101     )
102
103     createTask: identity priority: priority work: work state: state
function: aBlock data: data = (
104         | t |
105
106         t := TaskControlBlock
107             link: taskList
108             create: identity
109             priority: priority
110             initialWorkQueue: work
111             initialState: state
112             function: aBlock
113             privateData: data.
114
115         taskList := t.
116         taskTable at: identity put: t
117     )
118
119     createWorker: identity priority: priority work: work state: state = (
120         | data |
121         data := WorkerTaskDataRecord create.
122         self
123             createTask: identity
124             priority: priority
125             work: work
126             state: state
127             function:
128                 [:work :word |
129                     | data |
130                     data := word.
131                     RObject NoWork == work
132                     ifTrue: [self wait]
133                     ifFalse:
134                         [data destination: (RObject
135                             HandlerA = data destination
136                             ifTrue: [RObject
137                                 HandlerB]
138                                 ifFalse:
139                                     [RObject HandlerA]).
140                                     work identity: data destination.
141                                     work datum: 1.
142                                     1 to: 4 do:
143                                         [:i |
144                                             data count: data count +
145                                             1.
146                                             data count > 26 ifTrue:
147                                                 "work data at: i put: $A
148                                                 work data at: i put: 65 +
149                                                 self queuePacket: work]]
150                                     data: data
151                                 )
152
153         run = (
154             RObject initialize.
155             self start.
156         )
157
158         start = (

```

```

154 | workQ mark1 mark2 mark3 mark4|
155 self initTrace.
156 self initScheduler.
157 mark1 := Time millisecondClockValue.
158 Transcript show: 'Bench mark starting'.
159 Transcript cr.
160 self
161     createIdler: RObject Idler
162     priority: 0
163     work: RObject NoWork
164     state: TaskState running.
165 workQ := self
166     createPacket: RObject NoWork
167     identity: RObject Worker
168     kind: RObject WorkPacketKind.
169 workQ := self
170     createPacket: workQ
171     identity: RObject Worker
172     kind: RObject WorkPacketKind.
173 self
174     createWorker: RObject Worker
175     priority: 1000
176     work: workQ
177     state: TaskState waitingWithPacket.
178 workQ := self
179     createPacket: RObject NoWork
180     identity: RObject DeviceA
181     kind: RObject DevicePacketKind.
182 workQ := self
183     createPacket: workQ
184     identity: RObject DeviceA
185     kind: RObject DevicePacketKind.
186 workQ := self
187     createPacket: workQ
188     identity: RObject DeviceA
189     kind: RObject DevicePacketKind.
190 self
191     createHandler: RObject HandlerA
192     priority: 2000
193     work: workQ
194     state: TaskState waitingWithPacket.
195 workQ := self
196     createPacket: RObject NoWork
197     identity: RObject DeviceB
198     kind: RObject DevicePacketKind.
199 workQ := self
200     createPacket: workQ
201     identity: RObject DeviceB
202     kind: RObject DevicePacketKind.
203 workQ := self
204     createPacket: workQ
205     identity: RObject DeviceB
206     kind: RObject DevicePacketKind.
207 self
208     createHandler: RObject HandlerB
209     priority: 3000
210     work: workQ
211     state: TaskState waitingWithPacket.
212 self
213     createDevice: RObject DeviceA
214     priority: 4000

```

```

215         work: RObject NoWork
216         state: TaskState waiting.
217     self
218         createDevice: RObject DeviceB
219         priority: 5000
220         work: RObject NoWork
221         state: TaskState waiting.
222     Transcript show: 'Starting'.
223     Transcript cr.
224     mark2 := Time millisecondClockValue.
225     self schedule.
226     mark3 := Time millisecondClockValue.
227     Transcript show: 'Finished'.
228     Transcript cr.
229     Transcript show: 'QueuePacket count = '.
230     Transcript show: queuePacketCount asString.
231     Transcript show: ' HoldCount = '.
232     Transcript show: holdCount asString.
233     Transcript cr.
234     Transcript show: 'These results are '.
235     (((queuePacketCount = 23246) and: (holdCount = 9297))
236         ifTrue: [Transcript show: 'correct']
237         ifFalse: [Transcript show: 'wrong'])).
238     Transcript cr.
239     Transcript show: 'End of run'.
240     Transcript cr.
241     mark4 := Time millisecondClockValue.
242     Transcript show: '*****Scheduler time = '.
243     Transcript show: (mark3 - mark2) asString.
244     Transcript show: ' Total time = '.
245     Transcript show: (mark4 - mark1) asString.
246     Transcript cr
247 )
248
249 "This is start simply duplicated, removing all output and making the
250 correctness check exiting with an error."
251 reBenchStart = (
252     | workQ |
253     self initTrace.
254     self initScheduler.
255
256     self
257         createIdler: RObject Idler
258         priority: 0
259         work: RObject NoWork
260         state: TaskState running.
261     workQ := self
262         createPacket: RObject NoWork
263         identity: RObject Worker
264         kind: RObject WorkPacketKind.
265     workQ := self
266         createPacket: workQ
267         identity: RObject Worker
268         kind: RObject WorkPacketKind.
269     self
270         createWorker: RObject Worker
271         priority: 1000
272         work: workQ
273         state: TaskState waitingWithPacket.
274     workQ := self
275         createPacket: RObject NoWork

```

```

276             identity: RObject DeviceA
277             kind: RObject DevicePacketKind.
278     workQ := self
279             createPacket: workQ
280             identity: RObject DeviceA
281             kind: RObject DevicePacketKind.
282     workQ := self
283             createPacket: workQ
284             identity: RObject DeviceA
285             kind: RObject DevicePacketKind.
286     self
287         createHandler: RObject HandlerA
288         priority: 2000
289         work: workQ
290         state: TaskState waitingWithPacket.
291     workQ := self
292             createPacket: RObject NoWork
293             identity: RObject DeviceB
294             kind: RObject DevicePacketKind.
295     workQ := self
296             createPacket: workQ
297             identity: RObject DeviceB
298             kind: RObject DevicePacketKind.
299     workQ := self
300             createPacket: workQ
301             identity: RObject DeviceB
302             kind: RObject DevicePacketKind.
303     self
304         createHandler: RObject HandlerB
305         priority: 3000
306         work: workQ
307         state: TaskState waitingWithPacket.
308     self
309         createDevice: RObject DeviceA
310         priority: 4000
311         work: RObject NoWork
312         state: TaskState waiting.
313     self
314         createDevice: RObject DeviceB
315         priority: 5000
316         work: RObject NoWork
317         state: TaskState waiting.
318
319     self schedule.
320
321     ^ ((queuePacketCount = 23246) and: (holdCount = 9297))
322 )
323
324
325 findTask: identity = (
326     | t |
327     t := taskTable at: identity.
328     RObject NoTask == t ifTrue: [self error: 'findTask failed'].
329     ^t
330 )
331
332 holdSelf = (
333     holdCount := holdCount + 1.
334     currentTask taskHolding: true.
335     ^currentTask link
336 )

```

```

337
338   initScheduler = (
339       queuePacketCount := 0.
340       holdCount := 0.
341       taskTable := Array new: 6 withAll: RObject NoTask.
342       taskList := RObject NoTask
343   )
344
345   initTrace = (
346       "îâ      &-æ '"6†ö-6R
347       message: 'Trace?'
348       displayAt: Sensor mousePoint
349       centered: true
350       ifTrue: [tracing := true]
351       ifFalse: [tracing := false].
352       "
353       "does not work in V. 4"
354
355       tracing := false.
356       layout := 0
357   )
358
359   queuePacket: packet = (
360       | t |
361       t := self findTask: packet identity.
362       RObject NoTask == t ifTrue: [^RObject NoTask].
363       queuePacketCount := queuePacketCount + 1.
364       packet link: RObject NoWork.
365       packet identity: currentTaskIdentity.
366       ^t addInput: packet checkPriority: currentTask
367   )
368
369   release: identity = (
370       | t |
371       t := self findTask: identity.
372       RObject NoTask == t ifTrue: [^RObject NoTask].
373       t taskHolding: false.
374       t priority > currentTask priority
375           ifTrue: [^t]
376           ifFalse: [^currentTask]
377   )
378
379   trace: id = (
380       layout := layout - 1.
381       0 >= layout
382           ifTrue:
383               [Transcript cr.
384                layout := 50].
385       Transcript show: id asString
386   )
387
388   wait = (
389       currentTask taskWaiting: true.
390       ^currentTask
391   )
392
393   schedule = (
394       currentTask := taskList.
395       [RObject NoTask == currentTask]
396       whileFalse:
397           [currentTask isTaskHoldingOrWaiting

```

```

398
399
400
currentTask identity.
401
currentTaskIdentity].
402
runTask]]
403     )
404
405     ----
406
407     start = (
408         super new start
409     )
410 )
411

```

```

ifTrue: [currentTask := currentTask link]
ifFalse:
    [currentTaskIdentity :=
tracing ifTrue: [self trace:
currentTask := currentTask

```

```

1
2 TaskControlBlock = TaskState (
3   |link identity priority input function handle|
4
5   identity = (^identity)
6
7   link = (^link)
8
9   priority = (^priority)
10
11   link: aLink identity: anIdentity priority: aPriority initialWorkQueue:
anInitialWorkQueue initialState: anInitialState function: aBlock privateData:
aPrivateData = (
12     link := aLink.
13     identity := anIdentity.
14     priority := aPriority.
15     input := anInitialWorkQueue.
16     self packetPending: anInitialState isPacketPending.
17     self taskWaiting: anInitialState isTaskWaiting.
18     self taskHolding: anInitialState isTaskHolding.
19     function := aBlock.
20     handle := aPrivateData.
21   )
22
23   addInput: packet checkPriority: oldTask = (
24     RObject NoWork == input
25       ifTrue:
26         [input := packet.
27          self packetPending: true.
28          priority > oldTask priority ifTrue: [^self]]
29       ifFalse:
30         [input := self append: packet head: input].
31     ^oldTask
32   )
33
34   runTask = (
35     | message |
36     self isWaitingWithPacket
37       ifTrue:
38         [message := input.
39          input := message link.
40          RObject NoWork == input
41            ifTrue: [self running]
42            ifFalse: [self packetPending]]
43       ifFalse: [message := RObject NoWork].
44     ^function value: message with: handle
45   )
46
47 ----
48
49   link: link create: identity priority: priority initialWorkQueue:
initialWorkQueue initialState: initialState function: aBlock privateData:
privateData = (
50     ^super new
51     link: link
52     identity: identity
53     priority: priority
54     initialWorkQueue: initialWorkQueue

```



```
55         initialState: initialState
56         function: aBlock
57         privateData: privateData
58     )
59 )
60
```

```

1 TaskState = RObject (
2   |packetPending taskWaiting taskHolding|
3
4   isPacketPending = (^packetPending)
5
6   isTaskHolding = (^taskHolding)
7
8   isTaskWaiting = (^taskWaiting)
9
10  taskHolding: aBoolean = (taskHolding := aBoolean)
11  taskWaiting: aBoolean = (taskWaiting := aBoolean)
12  packetPending: aBoolean = (packetPending := aBoolean)
13
14  packetPending = (
15    packetPending := true.
16    taskWaiting := false.
17    taskHolding := false
18  )
19
20  running = (
21    packetPending := taskWaiting := taskHolding := false
22  )
23
24  waiting = (
25    packetPending := taskHolding := false.
26    taskWaiting := true
27  )
28
29  waitingWithPacket = (
30    taskHolding := false.
31    taskWaiting := packetPending := true
32  )
33
34  isRunning = (^packetPending not and: [taskWaiting not and:
[taskHolding not]])
35
36  isTaskHoldingOrWaiting = (^taskHolding or: [packetPending not and:
[taskWaiting]])
37
38  isWaiting = (^packetPending not and: [taskWaiting and: [taskHolding
not]])
39
40  isWaitingWithPacket = (^packetPending and: [taskWaiting and:
[taskHolding not]])
41
42
43  ----
44  packetPending = (^super new packetPending)
45
46  running = (
47    ^super new running
48  )
49
50  waiting = (^super new waiting)
51
52  waitingWithPacket = (^super new waitingWithPacket)
53 )
54

```

```
1 Time = Object (  
2     ----  
3     millisecondClockValue = (^system time)  
4 )  
5
```

```
1 Transcript = (  
2     ----  
3     cr = (system printNewline)  
4     show: text = (system printString: text)  
5 )  
6  
7
```

```
1 WorkerTaskDataRecord = RObject (
2   |destination count|
3   count = (^count)
4
5   count: aCount =(count := aCount)
6
7   destination = (^destination)
8
9   destination: aHandler = (destination := aHandler)
10
11   create = (
12       destination := RObject HandlerA.
13       count := 0
14   )
15 ----
16 create = (^super new create)
17 )
18
```

```
1 "  
2  
3 $Id: Sieve.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Sieve = Benchmark (  
27  
28   benchmark = (  
29     | flags |  
30     flags := Array new: 5000.  
31     ^ self sieve: flags size: 5000.  
32   )  
33  
34   verifyResult: result = (  
35     ^ self assert: 669 equals: result  
36   )  
37  
38   sieve: flags size: size = (  
39     | primeCount |  
40     primeCount := 0.  
41     flags putAll: true.  
42     2 to: size do: [ :i |  
43       (flags at: i - 1)  
44         ifTrue: [  
45           | k |  
46           primeCount := primeCount + 1.  
47           k := i + i.  
48           [ k <= size ]  
49             whileTrue: [  
50               flags at: k - 1 put: false. k := k + i ]. ] ].  
51     ^primeCount  
52   )  
53  
54 )
```



```
1 "  
2 Copyright (c) 2022 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22 SomInit = Benchmark (  
23   | rnd vec benchHarn planner jsonLit jsonNum jsonStr jsonP parEx edge  
24   | lexer sblock sclass sdouble sinteger smethod sprimitive sstring |  
25  
26   oneTimeSetup = (  
27     rnd      := Random new.  
28     vec      := Vector new.  
29     benchHarn := BenchmarkHarness new.  
30     planner  := Planner new.  
31     jsonLit  := JsonLiteral new.  
32     jsonNum  := JsonNumber new.  
33     jsonStr  := JsonString new.  
34     jsonP    := JsonParser new.  
35     parEx    := ParseException new.  
36     edge     := Edge new.  
37  
38     lexer    := Lexer new.  
39     sblock   := SBlock new.  
40     sclass   := SClass new.  
41  
42     sdouble  := SDouble new.  
43     sinteger := SInteger new.  
44     smethod  := SMethod new.  
45     sprimitive := SPrimitive new.  
46     sstring  := SString new.  
47   )  
48  
49   benchmark = (  
50     "Fannkuch new initialize: 1."  
51     rnd initialize.  
52     vec initialize: 1.  
53     benchHarn initialize.  
54     planner initialize.
```



```

55
56     jsonLit initializeWith: 'null'.
57     jsonLit initializeWith: 'true'.
58     jsonLit initializeWith: 'false'.
59     jsonNum initializeWith: '123'.
60     jsonStr initializeWith: '123'.
61     jsonP initializeWith: '123'.
62
63     parEx initializeWith: 'msg' at: 1 line: 3 column: 55.
64
65     edge initializeWith: self and: self.
66     lexer initialize: ''.
67
68     sblock initialize: self in: self with: self.
69     sclass initialize: self.
70     sdouble initialize: 0.0.
71     sinteger initialize: 10.
72     smethod initializeWith: self bc: self literals: self numLocals: 1
maxStack: 1.
73     sprimitive initialize: #sym with: self.
74     sstring initializeWith: 'www'.
75 )
76
77     verifyResult: result = (
78         ^ true
79     )
80 )
81

```

```

1 "
2 Copyright (c) 2021 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22 SomParse = Benchmark (
23   | classNamees first |
24   oneTimeSetup = (
25     | v |
26     first := true.
27
28     v := Vector new: 100.
29     self classNamees1: v.
30     self classNamees2: v.
31     classNamees := v asArray
32   )
33
34   classNamees1: v = (
35     v, #Echo
36     , #Hello
37     , #QuickSort
38     , #Ball
39     , #ListElement
40     , #JenkinsRandom
41     , #NBodyBench
42     , #NBody
43     , #Body
44     , #NBodySystem
45     , #Queens
46     , #List
47     , #BubbleSort
48     , #TestHarness
49     , #SuperTest
50     , #ReflectionTest
51     , #ClassStructureTest
52     , #BlockTest
53     , #SuperTestSuperClass

```

```

54      , #EmptyTest
55      , #StringTest
56      , #ClassLoadingTest
57      , #CompilerReturnTest
58      , #ClassC
59      , #ClassB
60
61      , #SelfBlockTest
62      , #HashTest
63      , #ClassA
64      , #SetTest
65      , #GlobalTest
66      , #ClosureTest
67      , #DoesNotUnderstandMessage
68      , #SpecialSelectorsTest
69      , #PreliminaryTest
70      , #TestRunner
71      , #DictionaryTest
72      , #VectorTest
73      , #TestCase
74      , #DoesNotUnderstandTest
75      , #CoercionTest
76      , #ArrayTest
77      , #SystemTest
78      , #DoubleTest
79      , #BooleanTest
80  )
81
82  classNames2: v = (
83      v, #SString
84      , #SObject
85      , #SAbstractObject
86      , #SSymbol
87      , #SBlock
88      , #SDouble
89      , #SArray
90      , #SPrimitive
91      , #SMethod
92      , #SClass
93      , #SInteger
94      , #SystemPrimitives
95      , #ClassPrimitives
96      , #DoublePrimitives
97      , #Primitives
98      , #IntegerPrimitives
99      , #PrimitivePrimitives
100     , #SymbolPrimitives
101     , #MethodPrimitives
102     , #StringPrimitives
103     , #BlockPrimitives
104     , #ObjectPrimitives
105     , #ArrayPrimitives
106
107     , #Parser
108     , #BytecodeGenerator
109     , #ClassGenerationContext
110     , #Lexer
111     , #SourcecodeCompiler
112     , #Disassembler
113     , #MethodGenerationContext
114     , #BasicInterpreterTests

```

```
115         , #SomSomTests
116         , #SomTests
117         , #FrameTests
118         , #LexerTests
119         , #ParserWithError
120         , #ParserTests
121     )
122
123     benchmark = (
124         classNames do: [:name |
125             system load: name ].
126         ^ first
127     )
128
129     verifyResult: result = (
130         first := false.
131         "Can only be execute once"
132         ^ result
133     )
134 )
135
```

```

1 "
2
3 $Id: Sort.som 31 2009-07-31 12:25:18Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 Sort = Benchmark (
27     | smallest largest |
28
29     benchmark = (
30         | array |
31         array := self randomArray: self dataSize.
32         ^ self sort: array.
33     )
34
35     dataSize      = ( self subclassResponsibility )
36     sort: array = ( self subclassResponsibility )
37
38     verifyResult: array = (
39         ((array at: 1) <> smallest)
40         || ((array at: (array length)) <> largest)
41         ifTrue: [ self error: 'Array is not sorted. smallest: ' +
smallest asString + ' largest: ' + largest asString + ' [1]: ' + (array at:
1) asString + ' [1]: ' + (array at: array length) asString ].
42         3 to: (array length) do: [ :i |
43             (array at: i - 1) > (array at: i)
44             ifTrue: [ self error: 'Array is not sorted. [' + i
asString + ' - 1]: ' + (array at: i - 1) asString + ' [' + i asString + ']: '
+ (array at: i) asString ]. ].
45         ^ true
46     )
47
48     randomArray: size = (
49         | array |

```

```
51      Random initialize.
52      array := Array new: size withAll: [ Random next ].
53      smallest := largest := array at: 1.
54      array do: [ :elm |
55          (elm > largest) ifTrue: [ largest := elm ].
56          (elm < smallest) ifTrue: [ smallest := elm ]. ].
57      ^array
58  )
59
60 )
61
```

```
1 "  
2  
3 $Id: Storage.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Storage = Benchmark (  
27  
28   | count |  
29  
30   benchmark = (  
31     Random initialize.  
32     count := 0.  
33     self buildTreeDepth: 7.  
34     ^ count  
35   )  
36  
37   verifyResult: result = (  
38     ^ self assert: 5461 equals: result  
39   )  
40  
41   buildTreeDepth: depth = (  
42     count := count + 1.  
43     ^(depth = 1)  
44       ifTrue: [ Array new: Random next % 10 + 1 ]  
45       ifFalse: [  
46         Array new: 4 withAll: [ self buildTreeDepth: depth - 1 ] ]  
47   )  
48  
49 )  
50
```

```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Array2Test = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Array new: 3.  
33     a at: 1 put: 'hello'.  
34     a at: 2 put: #world.  
35     a at: 3 put: 23.  
36   )  
37  
38   testAt = (  
39     self assert: 'hello' equals: (a at: 1).  
40     self assert: #world equals: (a at: 2).  
41     self assert: 23 equals:      (a at: 3).  
42   )  
43  
44   testLength = (  
45     self assert: 3 equals: a length  
46   )  
47  
48   testPutAllBlock = (  
49     | arr i |  
50     arr := Array new: 10.  
51  
52     i := 0.  
53     arr putAll: [ i := i + 1. i ].
```



```

54
55     1 to: 10 do: [:j |
56         self assert: j equals: (arr at: j) ].
57 )
58
59 testPutAllInt = (
60     | arr i |
61     arr := Array new: 10.
62     arr putAll: 0.
63
64     1 to: 10 do: [:j |
65         self assert: 0 equals: (arr at: j) ].
66 )
67
68 testPutAllSym = (
69     | arr i |
70     arr := Array new: 10.
71     arr putAll: #sym.
72
73     1 to: 10 do: [:j |
74         self assert: #sym is: (arr at: j) ].
75 )
76
77 testFirst = (
78     self assert: 'hello' equals: a first.
79 )
80
81 testLast = (
82     self assert: 23 equals: a last.
83 )
84
85 testContains = (
86     self assert: (a contains: 23).
87     self deny: (a contains: #notInThere).
88 )
89
90 testDo = (
91     | j |
92     j := 1.
93
94     a do: [:i |
95         self assert: (a at: j) is: i.
96         j := j + 1.
97     ]
98 )
99
100 testDoIndexes = (
101     | i arr |
102     i := 1.
103     a doIndexes: [:j |
104         self assert: i equals: j.
105         i := i + 1.
106     ].
107     self assert: 4 equals: i.
108
109     arr := Array new: 10.
110     i := 1.
111     arr doIndexes: [:j |
112         self assert: i equals: j.
113         i := i + 1.
114     ].

```

```

115     self assert: 11 equals: i.
116 )
117
118 testFromToDo = (
119     | arr i |
120     a from: 2 to: 2 do: [:e | self assert: #world is: e ].
121
122     i := 0.
123     arr := Array new: 10 withAll: [ i := i + 1. i ].
124
125     i := 3.
126     arr from: 3 to: 7 do: [:e |
127         self assert: i equals: e.
128         i := i + 1 ].
129
130     self assert: 8 equals: i.
131 )
132
133 testSumAndAverage = (
134     | arr |
135     arr := Array new: 3.
136     1 to: 3 do: [:i | arr at: i put: i ].
137
138     self assert: 6 equals: arr sum.
139     self assert: 2 equals: arr average.
140 )
141
142 testCopyFrom = (
143     | arr b |
144     arr := Array new: 5.
145     1 to: 5 do: [:i | arr at: i put: i ].
146
147     b := arr copyFrom: 2 to: 4.
148     self assert: 2 equals: (b at: 1).
149     self assert: 3 equals: (b at: 2).
150     self assert: 4 equals: (b at: 3).
151
152     b := arr copyFrom: 3.
153     self assert: 3 equals: (b at: 1).
154     self assert: 4 equals: (b at: 2).
155     self assert: 5 equals: (b at: 3).
156 )
157
158 testCopy = (
159     | arr |
160     arr := a copy.
161     self assert: 3 equals: arr length.
162     self assert: 'hello' equals: (arr at: 1).
163     self assert: #world equals: (arr at: 2).
164     self assert: 23 equals: (arr at: 3).
165 )
166
167 testReplaceFrom = (
168     | arr1 arr2 i |
169     arr1 := Array new: 10 withAll: 0.
170
171     i := 0.
172     arr2 := Array new: 10 withAll: [ i := i + 1. i ].
173
174     arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.
175

```

```

176     i := 1.
177     3 to: 7 do: [:j |
178         self assert: i equals: (arr1 at: j).
179         i := i + 1.
180     ]
181 )
182
183 testExtendedWith = (
184     | arr newArr |
185     arr := Array new: 0.
186     newArr := arr extendedWith: 33.
187
188     self assert: 1 equals: newArr length.
189     self assert: 0 equals: arr length.
190     self assert: 33 equals: (newArr at: 1).
191
192     self testAt. "confirm a is correct"
193     self testLength.
194
195     newArr := a extendedWith: 44.
196
197     self testAt. "confirm a is correct"
198     self testLength.
199
200     self assert: 4 equals: newArr length.
201     self assert: 0 equals: arr length.
202     self assert: 44 equals: (newArr at: 4).
203 )
204
205 testPrependedWith = (
206     | arr newArr |
207     arr := Array new: 0.
208     newArr := arr prependedWith: 33.
209
210     self assert: 1 equals: newArr length.
211     self assert: 0 equals: arr length.
212     self assert: 33 equals: (newArr at: 1).
213
214     self testAt. "confirm a is correct"
215     self testLength.
216
217     newArr := a prependedWith: 44.
218
219     self testAt. "confirm a is correct"
220     self testLength.
221
222     self assert: 4 equals: newArr length.
223     self assert: 0 equals: arr length.
224     self assert: 44 equals: (newArr at: 1).
225 )
226
227 testIndexOf = (
228     | arr |
229     arr := Array new: 6.
230     arr at: 1 put: #one.
231     arr at: 2 put: #two.
232     arr at: 3 put: #three.
233     arr at: 4 put: #four.
234     arr at: 5 put: #five.
235     arr at: 6 put: #one.
236

```

```

237     self assert: 2 equals: (arr indexOf: #two).
238     self assert: 4 equals: (arr indexOf: #four).
239     self assert: 5 equals: (arr indexOf: #five).
240
241     self assert: nil equals: (arr indexOf: #notIncluded).
242
243     self assert: 1 equals: (arr indexOf: #one).
244 )
245
246 testLastIndexOf = (
247     | arr |
248     arr := Array new: 6.
249     arr at: 1 put: #one.
250     arr at: 2 put: #two.
251     arr at: 3 put: #three.
252     arr at: 4 put: #four.
253     arr at: 5 put: #five.
254     arr at: 6 put: #one.
255
256     self assert: 2 equals: (arr lastIndexOf: #two).
257     self assert: 4 equals: (arr lastIndexOf: #four).
258     self assert: 5 equals: (arr lastIndexOf: #five).
259
260     self assert: nil equals: (arr indexOf: #notIncluded).
261
262     self assert: 6 equals: (arr lastIndexOf: #one).
263 )
264
265 testCollect = (
266     | arr i col |
267     i := 0.
268     arr := Array new: 10 withAll: [ i := i + 1. i ].
269     col := arr collect: [:e | e + 1 ].
270
271     self assert: 10 equals: col length.
272
273     1 to: 10 do: [:i |
274         self assert: i + 1 equals: (col at: i) ].
275 )
276
277 testInject = (
278     | arr result |
279     arr := Array new: 10 withAll: 1.
280
281     result := arr inject: 100 into: [:sum :e | sum + e ].
282
283     self assert: 110 equals: result.
284 )
285
286 testReject = (
287     | arr i result |
288     i := 0.
289     arr := Array new: 10 withAll: [ i := i + 1. i ].
290
291     result := arr reject: [:e | e % 2 = 0 ].
292
293     self assert: 5 equals: result size.
294     self assert: 1 equals: (result at: 1).
295     self assert: 3 equals: (result at: 2).
296     self assert: 5 equals: (result at: 3).
297     self assert: 7 equals: (result at: 4).

```

```

298     self assert: 9 equals: (result at: 5).
299 )
300
301 testRejectEmpty = (
302     | result |
303     result := (Array new: 0) reject: [:e | false ].
304     self assert: 0 equals: result size.
305
306     result := (Array new: 0) reject: [:e | true ].
307     self assert: 0 equals: result size.
308
309     result := (Array new: 10 withAll: 4) reject: [:e | true ].
310     self assert: 0 equals: result size.
311
312     result := (Array new: 10 withAll: 4) reject: [:e | true ].
313     self assert: 0 equals: result size.
314 )
315
316 testDontRejectAny = (
317     | result |
318     result := (Array new: 10 withAll: 0) reject: [:e | false ].
319     self assert: 10 equals: result size.
320
321     self assert: 0 equals: (result at: 5).
322 )
323
324 testSelect = (
325     | arr i result |
326     i := 0.
327     arr := Array new: 10 withAll: [ i := i + 1. i ].
328
329     result := arr select: [:e | e % 2 = 0 ].
330
331     self assert: 5 equals: result size.
332     self assert: 2 equals: (result at: 1).
333     self assert: 4 equals: (result at: 2).
334     self assert: 6 equals: (result at: 3).
335     self assert: 8 equals: (result at: 4).
336     self assert: 10 equals: (result at: 5).
337 )
338
339 testSelectEmpty = (
340     | result |
341     result := (Array new: 0) select: [:e | false ].
342     self assert: 0 equals: result size.
343
344     result := (Array new: 0) select: [:e | true ].
345     self assert: 0 equals: result size.
346
347     result := (Array new: 10 withAll: 4) select: [:e | false ].
348     self assert: 0 equals: result size.
349
350     result := (Array new: 10 withAll: 4) select: [:e | false ].
351     self assert: 0 equals: result size.
352 )
353
354 testSelectAll = (
355     | result |
356     result := (Array new: 10 withAll: 0) select: [:e | true ].
357     self assert: 10 equals: result size.
358

```

```

359     self assert: 0 equals: (result at: 5).
360 )
361
362 testUnion = (
363     | result |
364     result := a union: a.
365     self assert: 3 equals: result size.
366
367     self assert: (result contains: #world).
368     self assert: (result contains: 23).
369
370     result := a union: #(21 22 23 #world).
371
372     self assert: 5 equals: result size.
373     self assert: (result contains: 21).
374     self assert: (result contains: 22).
375     self assert: (result contains: 23).
376     self assert: (result contains: #world).
377 )
378
379 testNewWithAll = (
380     | arr |
381     arr := Array new: 5 withAll: [1].
382     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
383
384     arr := Array new: 5 withAll: 1.
385     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
386 )
387
388 testNewWithAllIntAndObjects = (
389     | arr o |
390     arr := Array new: 5 withAll: 5.
391     self assert: 5 equals: (arr at: 3).
392     arr at: 3 put: nil.
393     self assert: nil equals: (arr at: 3).
394
395     o := Object new.
396     arr at: 2 put: o.
397     self assert: o equals: (arr at: 2).
398 )
399
400 testLiteralArrays = (
401     -6VÆb 76W'Cç ,2f " ' Cç ' W V Ç3ç à
402     -6VÆb 76W'Cç ,2f " ' Cç " ' W V Ç3ç "à
403
404     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç ' W V Ç3ç Ó à
405     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç " ' W V Ç3ç Ó# ã à
406 )
407
408 testJoin = (
409     | arr |
410     arr := Array new: 0.
411     self assert: nil is: (arr join: ', ').
412
413     arr := Array with: 1 with: 10 with: 100.
414     self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).
415
416     arr := Array with: 'a' with: 'b' with: 'c'.
417     self assert: 'a, b, c' equals: (arr join: ', ').
418 )
419 )

```



```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Array3Test = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Array new: 3.  
33     a at: 1 put: 'hello'.  
34     a at: 2 put: #world.  
35     a at: 3 put: 23.  
36   )  
37  
38   testAt = (  
39     self assert: 'hello' equals: (a at: 1).  
40     self assert: #world equals: (a at: 2).  
41     self assert: 23 equals:      (a at: 3).  
42   )  
43  
44   testLength = (  
45     self assert: 3 equals: a length  
46   )  
47  
48   testPutAllBlock = (  
49     | arr i |  
50     arr := Array new: 10.  
51  
52     i := 0.  
53     arr putAll: [ i := i + 1. i ].
```



```

54
55     1 to: 10 do: [:j |
56         self assert: j equals: (arr at: j) ].
57 )
58
59 testPutAllInt = (
60     | arr i |
61     arr := Array new: 10.
62     arr putAll: 0.
63
64     1 to: 10 do: [:j |
65         self assert: 0 equals: (arr at: j) ].
66 )
67
68 testPutAllSym = (
69     | arr i |
70     arr := Array new: 10.
71     arr putAll: #sym.
72
73     1 to: 10 do: [:j |
74         self assert: #sym is: (arr at: j) ].
75 )
76
77 testFirst = (
78     self assert: 'hello' equals: a first.
79 )
80
81 testLast = (
82     self assert: 23 equals: a last.
83 )
84
85 testContains = (
86     self assert: (a contains: 23).
87     self deny:    (a contains: #notInThere).
88 )
89
90 testDo = (
91     | j |
92     j := 1.
93
94     a do: [:i |
95         self assert: (a at: j) is: i.
96         j := j + 1.
97     ]
98 )
99
100 testDoIndexes = (
101     | i arr |
102     i := 1.
103     a doIndexes: [:j |
104         self assert: i equals: j.
105         i := i + 1.
106     ].
107     self assert: 4 equals: i.
108
109     arr := Array new: 10.
110     i := 1.
111     arr doIndexes: [:j |
112         self assert: i equals: j.
113         i := i + 1.
114     ].

```

```

115     self assert: 11 equals: i.
116 )
117
118 testFromToDo = (
119     | arr i |
120     a from: 2 to: 2 do: [:e | self assert: #world is: e ].
121
122     i := 0.
123     arr := Array new: 10 withAll: [ i := i + 1. i ].
124
125     i := 3.
126     arr from: 3 to: 7 do: [:e |
127         self assert: i equals: e.
128         i := i + 1 ].
129
130     self assert: 8 equals: i.
131 )
132
133 testSumAndAverage = (
134     | arr |
135     arr := Array new: 3.
136     1 to: 3 do: [:i | arr at: i put: i ].
137
138     self assert: 6 equals: arr sum.
139     self assert: 2 equals: arr average.
140 )
141
142 testCopyFrom = (
143     | arr b |
144     arr := Array new: 5.
145     1 to: 5 do: [:i | arr at: i put: i ].
146
147     b := arr copyFrom: 2 to: 4.
148     self assert: 2 equals: (b at: 1).
149     self assert: 3 equals: (b at: 2).
150     self assert: 4 equals: (b at: 3).
151
152     b := arr copyFrom: 3.
153     self assert: 3 equals: (b at: 1).
154     self assert: 4 equals: (b at: 2).
155     self assert: 5 equals: (b at: 3).
156 )
157
158 testCopy = (
159     | arr |
160     arr := a copy.
161     self assert: 3 equals: arr length.
162     self assert: 'hello' equals: (arr at: 1).
163     self assert: #world equals: (arr at: 2).
164     self assert: 23 equals: (arr at: 3).
165 )
166
167 testReplaceFrom = (
168     | arr1 arr2 i |
169     arr1 := Array new: 10 withAll: 0.
170
171     i := 0.
172     arr2 := Array new: 10 withAll: [ i := i + 1. i ].
173
174     arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.
175

```

```

176     i := 1.
177     3 to: 7 do: [:j |
178         self assert: i equals: (arr1 at: j).
179         i := i + 1.
180     ]
181 )
182
183 testExtendedWith = (
184     | arr newArr |
185     arr := Array new: 0.
186     newArr := arr extendedWith: 33.
187
188     self assert: 1 equals: newArr length.
189     self assert: 0 equals: arr length.
190     self assert: 33 equals: (newArr at: 1).
191
192     self testAt. "confirm a is correct"
193     self testLength.
194
195     newArr := a extendedWith: 44.
196
197     self testAt. "confirm a is correct"
198     self testLength.
199
200     self assert: 4 equals: newArr length.
201     self assert: 0 equals: arr length.
202     self assert: 44 equals: (newArr at: 4).
203 )
204
205 testPrependedWith = (
206     | arr newArr |
207     arr := Array new: 0.
208     newArr := arr prependedWith: 33.
209
210     self assert: 1 equals: newArr length.
211     self assert: 0 equals: arr length.
212     self assert: 33 equals: (newArr at: 1).
213
214     self testAt. "confirm a is correct"
215     self testLength.
216
217     newArr := a prependedWith: 44.
218
219     self testAt. "confirm a is correct"
220     self testLength.
221
222     self assert: 4 equals: newArr length.
223     self assert: 0 equals: arr length.
224     self assert: 44 equals: (newArr at: 1).
225 )
226
227 testIndexOf = (
228     | arr |
229     arr := Array new: 6.
230     arr at: 1 put: #one.
231     arr at: 2 put: #two.
232     arr at: 3 put: #three.
233     arr at: 4 put: #four.
234     arr at: 5 put: #five.
235     arr at: 6 put: #one.
236

```

```

237     self assert: 2 equals: (arr indexOf: #two).
238     self assert: 4 equals: (arr indexOf: #four).
239     self assert: 5 equals: (arr indexOf: #five).
240
241     self assert: nil equals: (arr indexOf: #notIncluded).
242
243     self assert: 1 equals: (arr indexOf: #one).
244 )
245
246 testLastIndexOf = (
247     | arr |
248     arr := Array new: 6.
249     arr at: 1 put: #one.
250     arr at: 2 put: #two.
251     arr at: 3 put: #three.
252     arr at: 4 put: #four.
253     arr at: 5 put: #five.
254     arr at: 6 put: #one.
255
256     self assert: 2 equals: (arr lastIndexOf: #two).
257     self assert: 4 equals: (arr lastIndexOf: #four).
258     self assert: 5 equals: (arr lastIndexOf: #five).
259
260     self assert: nil equals: (arr indexOf: #notIncluded).
261
262     self assert: 6 equals: (arr lastIndexOf: #one).
263 )
264
265 testCollect = (
266     | arr i col |
267     i := 0.
268     arr := Array new: 10 withAll: [ i := i + 1. i ].
269     col := arr collect: [:e | e + 1 ].
270
271     self assert: 10 equals: col length.
272
273     1 to: 10 do: [:i |
274         self assert: i + 1 equals: (col at: i) ].
275 )
276
277 testInject = (
278     | arr result |
279     arr := Array new: 10 withAll: 1.
280
281     result := arr inject: 100 into: [:sum :e | sum + e ].
282
283     self assert: 110 equals: result.
284 )
285
286 testReject = (
287     | arr i result |
288     i := 0.
289     arr := Array new: 10 withAll: [ i := i + 1. i ].
290
291     result := arr reject: [:e | e % 2 = 0 ].
292
293     self assert: 5 equals: result size.
294     self assert: 1 equals: (result at: 1).
295     self assert: 3 equals: (result at: 2).
296     self assert: 5 equals: (result at: 3).
297     self assert: 7 equals: (result at: 4).

```

```

298     self assert: 9 equals: (result at: 5).
299 )
300
301 testRejectEmpty = (
302     | result |
303     result := (Array new: 0) reject: [:e | false ].
304     self assert: 0 equals: result size.
305
306     result := (Array new: 0) reject: [:e | true ].
307     self assert: 0 equals: result size.
308
309     result := (Array new: 10 withAll: 4) reject: [:e | true ].
310     self assert: 0 equals: result size.
311
312     result := (Array new: 10 withAll: 4) reject: [:e | true ].
313     self assert: 0 equals: result size.
314 )
315
316 testDontRejectAny = (
317     | result |
318     result := (Array new: 10 withAll: 0) reject: [:e | false ].
319     self assert: 10 equals: result size.
320
321     self assert: 0 equals: (result at: 5).
322 )
323
324 testSelect = (
325     | arr i result |
326     i := 0.
327     arr := Array new: 10 withAll: [ i := i + 1. i ].
328
329     result := arr select: [:e | e % 2 = 0 ].
330
331     self assert: 5 equals: result size.
332     self assert: 2 equals: (result at: 1).
333     self assert: 4 equals: (result at: 2).
334     self assert: 6 equals: (result at: 3).
335     self assert: 8 equals: (result at: 4).
336     self assert: 10 equals: (result at: 5).
337 )
338
339 testSelectEmpty = (
340     | result |
341     result := (Array new: 0) select: [:e | false ].
342     self assert: 0 equals: result size.
343
344     result := (Array new: 0) select: [:e | true ].
345     self assert: 0 equals: result size.
346
347     result := (Array new: 10 withAll: 4) select: [:e | false ].
348     self assert: 0 equals: result size.
349
350     result := (Array new: 10 withAll: 4) select: [:e | false ].
351     self assert: 0 equals: result size.
352 )
353
354 testSelectAll = (
355     | result |
356     result := (Array new: 10 withAll: 0) select: [:e | true ].
357     self assert: 10 equals: result size.
358

```

```

359     self assert: 0 equals: (result at: 5).
360 )
361
362 testUnion = (
363     | result |
364     result := a union: a.
365     self assert: 3 equals: result size.
366
367     self assert: (result contains: #world).
368     self assert: (result contains: 23).
369
370     result := a union: #(21 22 23 #world).
371
372     self assert: 5 equals: result size.
373     self assert: (result contains: 21).
374     self assert: (result contains: 22).
375     self assert: (result contains: 23).
376     self assert: (result contains: #world).
377 )
378
379 testNewWithAll = (
380     | arr |
381     arr := Array new: 5 withAll: [1].
382     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
383
384     arr := Array new: 5 withAll: 1.
385     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
386 )
387
388 testNewWithAllIntAndObjects = (
389     | arr o |
390     arr := Array new: 5 withAll: 5.
391     self assert: 5 equals: (arr at: 3).
392     arr at: 3 put: nil.
393     self assert: nil equals: (arr at: 3).
394
395     o := Object new.
396     arr at: 2 put: o.
397     self assert: o equals: (arr at: 2).
398 )
399
400 testLiteralArrays = (
401     -6VÆb 76W'Cç ,2f " ' Cç ' W V Ç3ç à
402     -6VÆb 76W'Cç ,2f " ' Cç " ' W V Ç3ç "à
403
404     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç ' W V Ç3ç Ó à
405     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç " ' W V Ç3ç Ó# ã à
406 )
407
408 testJoin = (
409     | arr |
410     arr := Array new: 0.
411     self assert: nil is: (arr join: ', ').
412
413     arr := Array with: 1 with: 10 with: 100.
414     self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).
415
416     arr := Array with: 'a' with: 'b' with: 'c'.
417     self assert: 'a, b, c' equals: (arr join: ', ').
418 )
419 )

```



```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Array4Test = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Array new: 3.  
33     a at: 1 put: 'hello'.  
34     a at: 2 put: #world.  
35     a at: 3 put: 23.  
36   )  
37  
38   testAt = (  
39     self assert: 'hello' equals: (a at: 1).  
40     self assert: #world equals: (a at: 2).  
41     self assert: 23 equals:      (a at: 3).  
42   )  
43  
44   testLength = (  
45     self assert: 3 equals: a length  
46   )  
47  
48   testPutAllBlock = (  
49     | arr i |  
50     arr := Array new: 10.  
51  
52     i := 0.  
53     arr putAll: [ i := i + 1. i ].
```



```

54
55     1 to: 10 do: [:j |
56         self assert: j equals: (arr at: j) ].
57 )
58
59 testPutAllInt = (
60     | arr i |
61     arr := Array new: 10.
62     arr putAll: 0.
63
64     1 to: 10 do: [:j |
65         self assert: 0 equals: (arr at: j) ].
66 )
67
68 testPutAllSym = (
69     | arr i |
70     arr := Array new: 10.
71     arr putAll: #sym.
72
73     1 to: 10 do: [:j |
74         self assert: #sym is: (arr at: j) ].
75 )
76
77 testFirst = (
78     self assert: 'hello' equals: a first.
79 )
80
81 testLast = (
82     self assert: 23 equals: a last.
83 )
84
85 testContains = (
86     self assert: (a contains: 23).
87     self deny:    (a contains: #notInThere).
88 )
89
90 testDo = (
91     | j |
92     j := 1.
93
94     a do: [:i |
95         self assert: (a at: j) is: i.
96         j := j + 1.
97     ]
98 )
99
100 testDoIndexes = (
101     | i arr |
102     i := 1.
103     a doIndexes: [:j |
104         self assert: i equals: j.
105         i := i + 1.
106     ].
107     self assert: 4 equals: i.
108
109     arr := Array new: 10.
110     i := 1.
111     arr doIndexes: [:j |
112         self assert: i equals: j.
113         i := i + 1.
114     ].

```

```

115     self assert: 11 equals: i.
116 )
117
118 testFromToDo = (
119     | arr i |
120     a from: 2 to: 2 do: [:e | self assert: #world is: e ].
121
122     i := 0.
123     arr := Array new: 10 withAll: [ i := i + 1. i ].
124
125     i := 3.
126     arr from: 3 to: 7 do: [:e |
127         self assert: i equals: e.
128         i := i + 1 ].
129
130     self assert: 8 equals: i.
131 )
132
133 testSumAndAverage = (
134     | arr |
135     arr := Array new: 3.
136     1 to: 3 do: [ :i | arr at: i put: i ].
137
138     self assert: 6 equals: arr sum.
139     self assert: 2 equals: arr average.
140 )
141
142 testCopyFrom = (
143     | arr b |
144     arr := Array new: 5.
145     1 to: 5 do: [ :i | arr at: i put: i ].
146
147     b := arr copyFrom: 2 to: 4.
148     self assert: 2 equals: (b at: 1).
149     self assert: 3 equals: (b at: 2).
150     self assert: 4 equals: (b at: 3).
151
152     b := arr copyFrom: 3.
153     self assert: 3 equals: (b at: 1).
154     self assert: 4 equals: (b at: 2).
155     self assert: 5 equals: (b at: 3).
156 )
157
158 testCopy = (
159     | arr |
160     arr := a copy.
161     self assert: 3 equals: arr length.
162     self assert: 'hello' equals: (arr at: 1).
163     self assert: #world equals: (arr at: 2).
164     self assert: 23 equals: (arr at: 3).
165 )
166
167 testReplaceFrom = (
168     | arr1 arr2 i |
169     arr1 := Array new: 10 withAll: 0.
170
171     i := 0.
172     arr2 := Array new: 10 withAll: [ i := i + 1. i ].
173
174     arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.
175

```

```

176     i := 1.
177     3 to: 7 do: [:j |
178         self assert: i equals: (arr1 at: j).
179         i := i + 1.
180     ]
181 )
182
183 testExtendedWith = (
184     | arr newArr |
185     arr := Array new: 0.
186     newArr := arr extendedWith: 33.
187
188     self assert: 1 equals: newArr length.
189     self assert: 0 equals: arr length.
190     self assert: 33 equals: (newArr at: 1).
191
192     self testAt. "confirm a is correct"
193     self testLength.
194
195     newArr := a extendedWith: 44.
196
197     self testAt. "confirm a is correct"
198     self testLength.
199
200     self assert: 4 equals: newArr length.
201     self assert: 0 equals: arr length.
202     self assert: 44 equals: (newArr at: 4).
203 )
204
205 testPrependedWith = (
206     | arr newArr |
207     arr := Array new: 0.
208     newArr := arr prependedWith: 33.
209
210     self assert: 1 equals: newArr length.
211     self assert: 0 equals: arr length.
212     self assert: 33 equals: (newArr at: 1).
213
214     self testAt. "confirm a is correct"
215     self testLength.
216
217     newArr := a prependedWith: 44.
218
219     self testAt. "confirm a is correct"
220     self testLength.
221
222     self assert: 4 equals: newArr length.
223     self assert: 0 equals: arr length.
224     self assert: 44 equals: (newArr at: 1).
225 )
226
227 testIndexOf = (
228     | arr |
229     arr := Array new: 6.
230     arr at: 1 put: #one.
231     arr at: 2 put: #two.
232     arr at: 3 put: #three.
233     arr at: 4 put: #four.
234     arr at: 5 put: #five.
235     arr at: 6 put: #one.
236

```

```

237     self assert: 2 equals: (arr indexOf: #two).
238     self assert: 4 equals: (arr indexOf: #four).
239     self assert: 5 equals: (arr indexOf: #five).
240
241     self assert: nil equals: (arr indexOf: #notIncluded).
242
243     self assert: 1 equals: (arr indexOf: #one).
244 )
245
246 testLastIndexOf = (
247     | arr |
248     arr := Array new: 6.
249     arr at: 1 put: #one.
250     arr at: 2 put: #two.
251     arr at: 3 put: #three.
252     arr at: 4 put: #four.
253     arr at: 5 put: #five.
254     arr at: 6 put: #one.
255
256     self assert: 2 equals: (arr lastIndexOf: #two).
257     self assert: 4 equals: (arr lastIndexOf: #four).
258     self assert: 5 equals: (arr lastIndexOf: #five).
259
260     self assert: nil equals: (arr indexOf: #notIncluded).
261
262     self assert: 6 equals: (arr lastIndexOf: #one).
263 )
264
265 testCollect = (
266     | arr i col |
267     i := 0.
268     arr := Array new: 10 withAll: [ i := i + 1. i ].
269     col := arr collect: [:e | e + 1 ].
270
271     self assert: 10 equals: col length.
272
273     1 to: 10 do: [:i |
274         self assert: i + 1 equals: (col at: i) ].
275 )
276
277 testInject = (
278     | arr result |
279     arr := Array new: 10 withAll: 1.
280
281     result := arr inject: 100 into: [:sum :e | sum + e ].
282
283     self assert: 110 equals: result.
284 )
285
286 testReject = (
287     | arr i result |
288     i := 0.
289     arr := Array new: 10 withAll: [ i := i + 1. i ].
290
291     result := arr reject: [:e | e % 2 = 0 ].
292
293     self assert: 5 equals: result size.
294     self assert: 1 equals: (result at: 1).
295     self assert: 3 equals: (result at: 2).
296     self assert: 5 equals: (result at: 3).
297     self assert: 7 equals: (result at: 4).

```

```

298     self assert: 9 equals: (result at: 5).
299 )
300
301 testRejectEmpty = (
302     | result |
303     result := (Array new: 0) reject: [:e | false ].
304     self assert: 0 equals: result size.
305
306     result := (Array new: 0) reject: [:e | true ].
307     self assert: 0 equals: result size.
308
309     result := (Array new: 10 withAll: 4) reject: [:e | true ].
310     self assert: 0 equals: result size.
311
312     result := (Array new: 10 withAll: 4) reject: [:e | true ].
313     self assert: 0 equals: result size.
314 )
315
316 testDontRejectAny = (
317     | result |
318     result := (Array new: 10 withAll: 0) reject: [:e | false ].
319     self assert: 10 equals: result size.
320
321     self assert: 0 equals: (result at: 5).
322 )
323
324 testSelect = (
325     | arr i result |
326     i := 0.
327     arr := Array new: 10 withAll: [ i := i + 1. i ].
328
329     result := arr select: [:e | e % 2 = 0 ].
330
331     self assert: 5 equals: result size.
332     self assert: 2 equals: (result at: 1).
333     self assert: 4 equals: (result at: 2).
334     self assert: 6 equals: (result at: 3).
335     self assert: 8 equals: (result at: 4).
336     self assert: 10 equals: (result at: 5).
337 )
338
339 testSelectEmpty = (
340     | result |
341     result := (Array new: 0) select: [:e | false ].
342     self assert: 0 equals: result size.
343
344     result := (Array new: 0) select: [:e | true ].
345     self assert: 0 equals: result size.
346
347     result := (Array new: 10 withAll: 4) select: [:e | false ].
348     self assert: 0 equals: result size.
349
350     result := (Array new: 10 withAll: 4) select: [:e | false ].
351     self assert: 0 equals: result size.
352 )
353
354 testSelectAll = (
355     | result |
356     result := (Array new: 10 withAll: 0) select: [:e | true ].
357     self assert: 10 equals: result size.
358

```

```

359     self assert: 0 equals: (result at: 5).
360 )
361
362 testUnion = (
363     | result |
364     result := a union: a.
365     self assert: 3 equals: result size.
366
367     self assert: (result contains: #world).
368     self assert: (result contains: 23).
369
370     result := a union: #(21 22 23 #world).
371
372     self assert: 5 equals: result size.
373     self assert: (result contains: 21).
374     self assert: (result contains: 22).
375     self assert: (result contains: 23).
376     self assert: (result contains: #world).
377 )
378
379 testNewWithAll = (
380     | arr |
381     arr := Array new: 5 withAll: [1].
382     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
383
384     arr := Array new: 5 withAll: 1.
385     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
386 )
387
388 testNewWithAllIntAndObjects = (
389     | arr o |
390     arr := Array new: 5 withAll: 5.
391     self assert: 5 equals: (arr at: 3).
392     arr at: 3 put: nil.
393     self assert: nil equals: (arr at: 3).
394
395     o := Object new.
396     arr at: 2 put: o.
397     self assert: o equals: (arr at: 2).
398 )
399
400 testLiteralArrays = (
401     -6VÆb 76W'Cç ,2f " ' Cç ' W V Ç3ç à
402     -6VÆb 76W'Cç ,2f " ' Cç " ' W V Ç3ç "à
403
404     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç ' W V Ç3ç Ó à
405     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç " ' W V Ç3ç Ó# ã à
406 )
407
408 testJoin = (
409     | arr |
410     arr := Array new: 0.
411     self assert: nil is: (arr join: ', ').
412
413     arr := Array with: 1 with: 10 with: 100.
414     self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).
415
416     arr := Array with: 'a' with: 'b' with: 'c'.
417     self assert: 'a, b, c' equals: (arr join: ', ').
418 )
419 )

```



```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Array5Test = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Array new: 3.  
33     a at: 1 put: 'hello'.  
34     a at: 2 put: #world.  
35     a at: 3 put: 23.  
36   )  
37  
38   testAt = (  
39     self assert: 'hello' equals: (a at: 1).  
40     self assert: #world equals: (a at: 2).  
41     self assert: 23 equals:      (a at: 3).  
42   )  
43  
44   testLength = (  
45     self assert: 3 equals: a length  
46   )  
47  
48   testPutAllBlock = (  
49     | arr i |  
50     arr := Array new: 10.  
51  
52     i := 0.  
53     arr putAll: [ i := i + 1. i ].
```



```

54
55     1 to: 10 do: [:j |
56         self assert: j equals: (arr at: j) ].
57 )
58
59 testPutAllInt = (
60     | arr i |
61     arr := Array new: 10.
62     arr putAll: 0.
63
64     1 to: 10 do: [:j |
65         self assert: 0 equals: (arr at: j) ].
66 )
67
68 testPutAllSym = (
69     | arr i |
70     arr := Array new: 10.
71     arr putAll: #sym.
72
73     1 to: 10 do: [:j |
74         self assert: #sym is: (arr at: j) ].
75 )
76
77 testFirst = (
78     self assert: 'hello' equals: a first.
79 )
80
81 testLast = (
82     self assert: 23 equals: a last.
83 )
84
85 testContains = (
86     self assert: (a contains: 23).
87     self deny:    (a contains: #notInThere).
88 )
89
90 testDo = (
91     | j |
92     j := 1.
93
94     a do: [:i |
95         self assert: (a at: j) is: i.
96         j := j + 1.
97     ]
98 )
99
100 testDoIndexes = (
101     | i arr |
102     i := 1.
103     a doIndexes: [:j |
104         self assert: i equals: j.
105         i := i + 1.
106     ].
107     self assert: 4 equals: i.
108
109     arr := Array new: 10.
110     i := 1.
111     arr doIndexes: [:j |
112         self assert: i equals: j.
113         i := i + 1.
114     ].

```

```

115     self assert: 11 equals: i.
116 )
117
118 testFromToDo = (
119     | arr i |
120     a from: 2 to: 2 do: [:e | self assert: #world is: e ].
121
122     i := 0.
123     arr := Array new: 10 withAll: [ i := i + 1. i ].
124
125     i := 3.
126     arr from: 3 to: 7 do: [:e |
127         self assert: i equals: e.
128         i := i + 1 ].
129
130     self assert: 8 equals: i.
131 )
132
133 testSumAndAverage = (
134     | arr |
135     arr := Array new: 3.
136     1 to: 3 do: [:i | arr at: i put: i ].
137
138     self assert: 6 equals: arr sum.
139     self assert: 2 equals: arr average.
140 )
141
142 testCopyFrom = (
143     | arr b |
144     arr := Array new: 5.
145     1 to: 5 do: [:i | arr at: i put: i ].
146
147     b := arr copyFrom: 2 to: 4.
148     self assert: 2 equals: (b at: 1).
149     self assert: 3 equals: (b at: 2).
150     self assert: 4 equals: (b at: 3).
151
152     b := arr copyFrom: 3.
153     self assert: 3 equals: (b at: 1).
154     self assert: 4 equals: (b at: 2).
155     self assert: 5 equals: (b at: 3).
156 )
157
158 testCopy = (
159     | arr |
160     arr := a copy.
161     self assert: 3 equals: arr length.
162     self assert: 'hello' equals: (arr at: 1).
163     self assert: #world equals: (arr at: 2).
164     self assert: 23 equals: (arr at: 3).
165 )
166
167 testReplaceFrom = (
168     | arr1 arr2 i |
169     arr1 := Array new: 10 withAll: 0.
170
171     i := 0.
172     arr2 := Array new: 10 withAll: [ i := i + 1. i ].
173
174     arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.
175

```

```

176     i := 1.
177     3 to: 7 do: [:j |
178         self assert: i equals: (arr1 at: j).
179         i := i + 1.
180     ]
181 )
182
183 testExtendedWith = (
184     | arr newArr |
185     arr := Array new: 0.
186     newArr := arr extendedWith: 33.
187
188     self assert: 1 equals: newArr length.
189     self assert: 0 equals: arr length.
190     self assert: 33 equals: (newArr at: 1).
191
192     self testAt. "confirm a is correct"
193     self testLength.
194
195     newArr := a extendedWith: 44.
196
197     self testAt. "confirm a is correct"
198     self testLength.
199
200     self assert: 4 equals: newArr length.
201     self assert: 0 equals: arr length.
202     self assert: 44 equals: (newArr at: 4).
203 )
204
205 testPrependedWith = (
206     | arr newArr |
207     arr := Array new: 0.
208     newArr := arr prependedWith: 33.
209
210     self assert: 1 equals: newArr length.
211     self assert: 0 equals: arr length.
212     self assert: 33 equals: (newArr at: 1).
213
214     self testAt. "confirm a is correct"
215     self testLength.
216
217     newArr := a prependedWith: 44.
218
219     self testAt. "confirm a is correct"
220     self testLength.
221
222     self assert: 4 equals: newArr length.
223     self assert: 0 equals: arr length.
224     self assert: 44 equals: (newArr at: 1).
225 )
226
227 testIndexOf = (
228     | arr |
229     arr := Array new: 6.
230     arr at: 1 put: #one.
231     arr at: 2 put: #two.
232     arr at: 3 put: #three.
233     arr at: 4 put: #four.
234     arr at: 5 put: #five.
235     arr at: 6 put: #one.
236

```

```

237     self assert: 2 equals: (arr indexOf: #two).
238     self assert: 4 equals: (arr indexOf: #four).
239     self assert: 5 equals: (arr indexOf: #five).
240
241     self assert: nil equals: (arr indexOf: #notIncluded).
242
243     self assert: 1 equals: (arr indexOf: #one).
244 )
245
246 testLastIndexOf = (
247     | arr |
248     arr := Array new: 6.
249     arr at: 1 put: #one.
250     arr at: 2 put: #two.
251     arr at: 3 put: #three.
252     arr at: 4 put: #four.
253     arr at: 5 put: #five.
254     arr at: 6 put: #one.
255
256     self assert: 2 equals: (arr lastIndexOf: #two).
257     self assert: 4 equals: (arr lastIndexOf: #four).
258     self assert: 5 equals: (arr lastIndexOf: #five).
259
260     self assert: nil equals: (arr indexOf: #notIncluded).
261
262     self assert: 6 equals: (arr lastIndexOf: #one).
263 )
264
265 testCollect = (
266     | arr i col |
267     i := 0.
268     arr := Array new: 10 withAll: [ i := i + 1. i ].
269     col := arr collect: [:e | e + 1 ].
270
271     self assert: 10 equals: col length.
272
273     1 to: 10 do: [:i |
274         self assert: i + 1 equals: (col at: i) ].
275 )
276
277 testInject = (
278     | arr result |
279     arr := Array new: 10 withAll: 1.
280
281     result := arr inject: 100 into: [:sum :e | sum + e ].
282
283     self assert: 110 equals: result.
284 )
285
286 testReject = (
287     | arr i result |
288     i := 0.
289     arr := Array new: 10 withAll: [ i := i + 1. i ].
290
291     result := arr reject: [:e | e % 2 = 0 ].
292
293     self assert: 5 equals: result size.
294     self assert: 1 equals: (result at: 1).
295     self assert: 3 equals: (result at: 2).
296     self assert: 5 equals: (result at: 3).
297     self assert: 7 equals: (result at: 4).

```

```

298     self assert: 9 equals: (result at: 5).
299 )
300
301 testRejectEmpty = (
302     | result |
303     result := (Array new: 0) reject: [:e | false ].
304     self assert: 0 equals: result size.
305
306     result := (Array new: 0) reject: [:e | true ].
307     self assert: 0 equals: result size.
308
309     result := (Array new: 10 withAll: 4) reject: [:e | true ].
310     self assert: 0 equals: result size.
311
312     result := (Array new: 10 withAll: 4) reject: [:e | true ].
313     self assert: 0 equals: result size.
314 )
315
316 testDontRejectAny = (
317     | result |
318     result := (Array new: 10 withAll: 0) reject: [:e | false ].
319     self assert: 10 equals: result size.
320
321     self assert: 0 equals: (result at: 5).
322 )
323
324 testSelect = (
325     | arr i result |
326     i := 0.
327     arr := Array new: 10 withAll: [ i := i + 1. i ].
328
329     result := arr select: [:e | e % 2 = 0 ].
330
331     self assert: 5 equals: result size.
332     self assert: 2 equals: (result at: 1).
333     self assert: 4 equals: (result at: 2).
334     self assert: 6 equals: (result at: 3).
335     self assert: 8 equals: (result at: 4).
336     self assert: 10 equals: (result at: 5).
337 )
338
339 testSelectEmpty = (
340     | result |
341     result := (Array new: 0) select: [:e | false ].
342     self assert: 0 equals: result size.
343
344     result := (Array new: 0) select: [:e | true ].
345     self assert: 0 equals: result size.
346
347     result := (Array new: 10 withAll: 4) select: [:e | false ].
348     self assert: 0 equals: result size.
349
350     result := (Array new: 10 withAll: 4) select: [:e | false ].
351     self assert: 0 equals: result size.
352 )
353
354 testSelectAll = (
355     | result |
356     result := (Array new: 10 withAll: 0) select: [:e | true ].
357     self assert: 10 equals: result size.
358

```

```

359     self assert: 0 equals: (result at: 5).
360 )
361
362 testUnion = (
363     | result |
364     result := a union: a.
365     self assert: 3 equals: result size.
366
367     self assert: (result contains: #world).
368     self assert: (result contains: 23).
369
370     result := a union: #(21 22 23 #world).
371
372     self assert: 5 equals: result size.
373     self assert: (result contains: 21).
374     self assert: (result contains: 22).
375     self assert: (result contains: 23).
376     self assert: (result contains: #world).
377 )
378
379 testNewWithAll = (
380     | arr |
381     arr := Array new: 5 withAll: [1].
382     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
383
384     arr := Array new: 5 withAll: 1.
385     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
386 )
387
388 testNewWithAllIntAndObjects = (
389     | arr o |
390     arr := Array new: 5 withAll: 5.
391     self assert: 5 equals: (arr at: 3).
392     arr at: 3 put: nil.
393     self assert: nil equals: (arr at: 3).
394
395     o := Object new.
396     arr at: 2 put: o.
397     self assert: o equals: (arr at: 2).
398 )
399
400 testLiteralArrays = (
401     -6VÆb 76W'Cç ,2f " ' Cç ' W V Ç3ç à
402     -6VÆb 76W'Cç ,2f " ' Cç " ' W V Ç3ç "à
403
404     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç ' W V Ç3ç Ó à
405     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç " ' W V Ç3ç Ó# ã à
406 )
407
408 testJoin = (
409     | arr |
410     arr := Array new: 0.
411     self assert: nil is: (arr join: ', ').
412
413     arr := Array with: 1 with: 10 with: 100.
414     self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).
415
416     arr := Array with: 'a' with: 'b' with: 'c'.
417     self assert: 'a, b, c' equals: (arr join: ', ').
418 )
419 )

```



```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 ArrayTest = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Array new: 3.  
33     a at: 1 put: 'hello'.  
34     a at: 2 put: #world.  
35     a at: 3 put: 23.  
36   )  
37  
38   testAt = (  
39     self assert: 'hello' equals: (a at: 1).  
40     self assert: #world equals: (a at: 2).  
41     self assert: 23 equals:      (a at: 3).  
42   )  
43  
44   testLength = (  
45     self assert: 3 equals: a length  
46   )  
47  
48   testPutAllBlock = (  
49     | arr i |  
50     arr := Array new: 10.  
51  
52     i := 0.  
53     arr putAll: [ i := i + 1. i ].
```



```

54
55     1 to: 10 do: [:j |
56         self assert: j equals: (arr at: j) ].
57 )
58
59 testPutAllInt = (
60     | arr i |
61     arr := Array new: 10.
62     arr putAll: 0.
63
64     1 to: 10 do: [:j |
65         self assert: 0 equals: (arr at: j) ].
66 )
67
68 testPutAllSym = (
69     | arr i |
70     arr := Array new: 10.
71     arr putAll: #sym.
72
73     1 to: 10 do: [:j |
74         self assert: #sym is: (arr at: j) ].
75 )
76
77 testFirst = (
78     self assert: 'hello' equals: a first.
79 )
80
81 testLast = (
82     self assert: 23 equals: a last.
83 )
84
85 testContains = (
86     self assert: (a contains: 23).
87     self deny:    (a contains: #notInThere).
88 )
89
90 testDo = (
91     | j |
92     j := 1.
93
94     a do: [:i |
95         self assert: (a at: j) is: i.
96         j := j + 1.
97     ]
98 )
99
100 testDoIndexes = (
101     | i arr |
102     i := 1.
103     a doIndexes: [:j |
104         self assert: i equals: j.
105         i := i + 1.
106     ].
107     self assert: 4 equals: i.
108
109     arr := Array new: 10.
110     i := 1.
111     arr doIndexes: [:j |
112         self assert: i equals: j.
113         i := i + 1.
114     ].

```

```

115     self assert: 11 equals: i.
116 )
117
118 testFromToDo = (
119     | arr i |
120     a from: 2 to: 2 do: [:e | self assert: #world is: e ].
121
122     i := 0.
123     arr := Array new: 10 withAll: [ i := i + 1. i ].
124
125     i := 3.
126     arr from: 3 to: 7 do: [:e |
127         self assert: i equals: e.
128         i := i + 1 ].
129
130     self assert: 8 equals: i.
131 )
132
133 testSumAndAverage = (
134     | arr |
135     arr := Array new: 3.
136     1 to: 3 do: [ :i | arr at: i put: i ].
137
138     self assert: 6 equals: arr sum.
139     self assert: 2 equals: arr average.
140 )
141
142 testCopyFrom = (
143     | arr b |
144     arr := Array new: 5.
145     1 to: 5 do: [ :i | arr at: i put: i ].
146
147     b := arr copyFrom: 2 to: 4.
148     self assert: 2 equals: (b at: 1).
149     self assert: 3 equals: (b at: 2).
150     self assert: 4 equals: (b at: 3).
151
152     b := arr copyFrom: 3.
153     self assert: 3 equals: (b at: 1).
154     self assert: 4 equals: (b at: 2).
155     self assert: 5 equals: (b at: 3).
156 )
157
158 testCopy = (
159     | arr |
160     arr := a copy.
161     self assert: 3 equals: arr length.
162     self assert: 'hello' equals: (arr at: 1).
163     self assert: #world equals: (arr at: 2).
164     self assert: 23 equals: (arr at: 3).
165 )
166
167 testReplaceFrom = (
168     | arr1 arr2 i |
169     arr1 := Array new: 10 withAll: 0.
170
171     i := 0.
172     arr2 := Array new: 10 withAll: [ i := i + 1. i ].
173
174     arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.
175

```

```

176     i := 1.
177     3 to: 7 do: [:j |
178         self assert: i equals: (arr1 at: j).
179         i := i + 1.
180     ]
181 )
182
183 testExtendedWith = (
184     | arr newArr |
185     arr := Array new: 0.
186     newArr := arr extendedWith: 33.
187
188     self assert: 1 equals: newArr length.
189     self assert: 0 equals: arr length.
190     self assert: 33 equals: (newArr at: 1).
191
192     self testAt. "confirm a is correct"
193     self testLength.
194
195     newArr := a extendedWith: 44.
196
197     self testAt. "confirm a is correct"
198     self testLength.
199
200     self assert: 4 equals: newArr length.
201     self assert: 0 equals: arr length.
202     self assert: 44 equals: (newArr at: 4).
203 )
204
205 testPrependedWith = (
206     | arr newArr |
207     arr := Array new: 0.
208     newArr := arr prependedWith: 33.
209
210     self assert: 1 equals: newArr length.
211     self assert: 0 equals: arr length.
212     self assert: 33 equals: (newArr at: 1).
213
214     self testAt. "confirm a is correct"
215     self testLength.
216
217     newArr := a prependedWith: 44.
218
219     self testAt. "confirm a is correct"
220     self testLength.
221
222     self assert: 4 equals: newArr length.
223     self assert: 0 equals: arr length.
224     self assert: 44 equals: (newArr at: 1).
225 )
226
227 testIndexOf = (
228     | arr |
229     arr := Array new: 6.
230     arr at: 1 put: #one.
231     arr at: 2 put: #two.
232     arr at: 3 put: #three.
233     arr at: 4 put: #four.
234     arr at: 5 put: #five.
235     arr at: 6 put: #one.
236

```

```

237     self assert: 2 equals: (arr indexOf: #two).
238     self assert: 4 equals: (arr indexOf: #four).
239     self assert: 5 equals: (arr indexOf: #five).
240
241     self assert: nil equals: (arr indexOf: #notIncluded).
242
243     self assert: 1 equals: (arr indexOf: #one).
244 )
245
246 testLastIndexOf = (
247     | arr |
248     arr := Array new: 6.
249     arr at: 1 put: #one.
250     arr at: 2 put: #two.
251     arr at: 3 put: #three.
252     arr at: 4 put: #four.
253     arr at: 5 put: #five.
254     arr at: 6 put: #one.
255
256     self assert: 2 equals: (arr lastIndexOf: #two).
257     self assert: 4 equals: (arr lastIndexOf: #four).
258     self assert: 5 equals: (arr lastIndexOf: #five).
259
260     self assert: nil equals: (arr indexOf: #notIncluded).
261
262     self assert: 6 equals: (arr lastIndexOf: #one).
263 )
264
265 testCollect = (
266     | arr i col |
267     i := 0.
268     arr := Array new: 10 withAll: [ i := i + 1. i ].
269     col := arr collect: [:e | e + 1 ].
270
271     self assert: 10 equals: col length.
272
273     1 to: 10 do: [:i |
274         self assert: i + 1 equals: (col at: i) ].
275 )
276
277 testInject = (
278     | arr result |
279     arr := Array new: 10 withAll: 1.
280
281     result := arr inject: 100 into: [:sum :e | sum + e ].
282
283     self assert: 110 equals: result.
284 )
285
286 testReject = (
287     | arr i result |
288     i := 0.
289     arr := Array new: 10 withAll: [ i := i + 1. i ].
290
291     result := arr reject: [:e | e % 2 = 0 ].
292
293     self assert: 5 equals: result size.
294     self assert: 1 equals: (result at: 1).
295     self assert: 3 equals: (result at: 2).
296     self assert: 5 equals: (result at: 3).
297     self assert: 7 equals: (result at: 4).

```

```

298     self assert: 9 equals: (result at: 5).
299 )
300
301 testRejectEmpty = (
302     | result |
303     result := (Array new: 0) reject: [:e | false ].
304     self assert: 0 equals: result size.
305
306     result := (Array new: 0) reject: [:e | true ].
307     self assert: 0 equals: result size.
308
309     result := (Array new: 10 withAll: 4) reject: [:e | true ].
310     self assert: 0 equals: result size.
311
312     result := (Array new: 10 withAll: 4) reject: [:e | true ].
313     self assert: 0 equals: result size.
314 )
315
316 testDontRejectAny = (
317     | result |
318     result := (Array new: 10 withAll: 0) reject: [:e | false ].
319     self assert: 10 equals: result size.
320
321     self assert: 0 equals: (result at: 5).
322 )
323
324 testSelect = (
325     | arr i result |
326     i := 0.
327     arr := Array new: 10 withAll: [ i := i + 1. i ].
328
329     result := arr select: [:e | e % 2 = 0 ].
330
331     self assert: 5 equals: result size.
332     self assert: 2 equals: (result at: 1).
333     self assert: 4 equals: (result at: 2).
334     self assert: 6 equals: (result at: 3).
335     self assert: 8 equals: (result at: 4).
336     self assert: 10 equals: (result at: 5).
337 )
338
339 testSelectEmpty = (
340     | result |
341     result := (Array new: 0) select: [:e | false ].
342     self assert: 0 equals: result size.
343
344     result := (Array new: 0) select: [:e | true ].
345     self assert: 0 equals: result size.
346
347     result := (Array new: 10 withAll: 4) select: [:e | false ].
348     self assert: 0 equals: result size.
349
350     result := (Array new: 10 withAll: 4) select: [:e | false ].
351     self assert: 0 equals: result size.
352 )
353
354 testSelectAll = (
355     | result |
356     result := (Array new: 10 withAll: 0) select: [:e | true ].
357     self assert: 10 equals: result size.
358

```

```

359     self assert: 0 equals: (result at: 5).
360 )
361
362 testUnion = (
363     | result |
364     result := a union: a.
365     self assert: 3 equals: result size.
366
367     self assert: (result contains: #world).
368     self assert: (result contains: 23).
369
370     result := a union: #(21 22 23 #world).
371
372     self assert: 5 equals: result size.
373     self assert: (result contains: 21).
374     self assert: (result contains: 22).
375     self assert: (result contains: 23).
376     self assert: (result contains: #world).
377 )
378
379 testNewWithAll = (
380     | arr |
381     arr := Array new: 5 withAll: [1].
382     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
383
384     arr := Array new: 5 withAll: 1.
385     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
386 )
387
388 testNewWithAllIntAndObjects = (
389     | arr o |
390     arr := Array new: 5 withAll: 5.
391     self assert: 5 equals: (arr at: 3).
392     arr at: 3 put: nil.
393     self assert: nil equals: (arr at: 3).
394
395     o := Object new.
396     arr at: 2 put: o.
397     self assert: o equals: (arr at: 2).
398 )
399
400 testLiteralArrays = (
401     -6VÆb 76W'Cç ,2f " ' Cç ' W V Ç3ç à
402     -6VÆb 76W'Cç ,2f " ' Cç " ' W V Ç3ç "à
403
404     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç ' W V Ç3ç Ó à
405     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç " ' W V Ç3ç Ó# ã à
406 )
407
408 testJoin = (
409     | arr |
410     arr := Array new: 0.
411     self assert: nil is: (arr join: ', ').
412
413     arr := Array with: 1 with: 10 with: 100.
414     self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).
415
416     arr := Array with: 'a' with: 'b' with: 'c'.
417     self assert: 'a, b, c' equals: (arr join: ', ').
418 )
419 )

```



```

1 Block2Test = TestCase (
2   |escape_count escaped_block|
3
4   simpleBlock = (
5     ^ [ 42 ]
6   )
7
8   incBlock = (
9     ^ [:val | val + 1 ]
10  )
11
12  "This requires a closure"
13  adderBlock: amount = (
14    ^ [:val | amount + val ]
15  )
16
17  "Closure with mutable state in block"
18  counterBlock = (
19    |count|
20    count := 0.
21    ^ [ count := count + 1. count ]
22  )
23
24  selfKeeper = (
25    ^ [ self ]
26  )
27
28  escapingBlock = (
29    ^ [ ^ 42 ]
30  )
31
32  escapingNestedBlock = (
33    [ [ ^ [ ^ 43 ] ] value ] value
34  )
35
36  testSimpleBlocks = (
37    self assert: 42 equals: self simpleBlock value.
38    self assert: 4  equals: (self incBlock value: 3).
39    self assert: 43 equals: ((self adderBlock: 13) value: 30).
40  )
41
42  testClosure = (
43    | counter |
44    counter := self counterBlock.
45    self assert: 1 equals: counter value.
46    self assert: 2 equals: counter value.
47    self assert: 1 equals: self counterBlock value. "make sure each
copy is independent"
48    self assert: 3 equals: counter value.
49  )
50
51  testSelfInBlock = (
52    | test_inst |
53    test_inst := BlockTest new.
54    self assert: test_inst equals: test_inst selfKeeper value.
55    self assert: self      equals: self selfKeeper value.
56  )
57

```



```

58     testEscapedBlock = (
59         | escaping_block |
60
61         escape_count := 0.
62
63         escaping_block := self escapingBlock.
64
65         self assert: 0 equals: escape_count.
66         self assert: 666 equals: escaping_block value.
67         self assert: 1 equals: escape_count.
68
69         self assert: escaping_block is: escaped_block.
70
71
72         escaping_block := self escapingNestedBlock.
73         self assert: 1 equals: escape_count.
74         self assert: 666 equals: escaping_block value.
75         self assert: 2 equals: escape_count.
76         self assert: escaping_block is: escaped_block.
77     )
78
79     escapedBlock: block = (
80         escape_count := escape_count + 1.
81         escaped_block := block.
82
83         "return some dummy value to the object that sent 'value' to block"
84         ^666
85     )
86
87     testWhileTrue = (
88         | i |
89         i := 1.
90         [ i < 10 ] whileTrue: [ i := i + 1 ].
91         self assert: 10 equals: i.
92     )
93
94     testWhileFalse = (
95         | i |
96         i := 1.
97         [ i >= 10 ] whileFalse: [ i := i + 1 ].
98         self assert: 10 equals: i.
99     )
100 )
101

```

```

1 Block3Test = TestCase (
2   |escape_count escaped_block|
3
4   simpleBlock = (
5     ^ [ 42 ]
6   )
7
8   incBlock = (
9     ^ [:val | val + 1 ]
10  )
11
12  "This requires a closure"
13  adderBlock: amount = (
14    ^ [:val | amount + val ]
15  )
16
17  "Closure with mutable state in block"
18  counterBlock = (
19    |count|
20    count := 0.
21    ^ [ count := count + 1. count ]
22  )
23
24  selfKeeper = (
25    ^ [ self ]
26  )
27
28  escapingBlock = (
29    ^ [ ^ 42 ]
30  )
31
32  escapingNestedBlock = (
33    [ [ ^ [ ^ 43 ] ] value ] value
34  )
35
36  testSimpleBlocks = (
37    self assert: 42 equals: self simpleBlock value.
38    self assert: 4  equals: (self incBlock value: 3).
39    self assert: 43 equals: ((self adderBlock: 13) value: 30).
40  )
41
42  testClosure = (
43    | counter |
44    counter := self counterBlock.
45    self assert: 1 equals: counter value.
46    self assert: 2 equals: counter value.
47    self assert: 1 equals: self counterBlock value. "make sure each
copy is independent"
48    self assert: 3 equals: counter value.
49  )
50
51  testSelfInBlock = (
52    | test_inst |
53    test_inst := BlockTest new.
54    self assert: test_inst equals: test_inst selfKeeper value.
55    self assert: self      equals: self selfKeeper value.
56  )
57

```

```

58     testEscapedBlock = (
59         | escaping_block |
60
61         escape_count := 0.
62
63         escaping_block := self escapingBlock.
64
65         self assert: 0 equals: escape_count.
66         self assert: 666 equals: escaping_block value.
67         self assert: 1 equals: escape_count.
68
69         self assert: escaping_block is: escaped_block.
70
71
72         escaping_block := self escapingNestedBlock.
73         self assert: 1 equals: escape_count.
74         self assert: 666 equals: escaping_block value.
75         self assert: 2 equals: escape_count.
76         self assert: escaping_block is: escaped_block.
77     )
78
79     escapedBlock: block = (
80         escape_count := escape_count + 1.
81         escaped_block := block.
82
83         "return some dummy value to the object that sent 'value' to block"
84         ^666
85     )
86
87     testWhileTrue = (
88         | i |
89         i := 1.
90         [ i < 10 ] whileTrue: [ i := i + 1 ].
91         self assert: 10 equals: i.
92     )
93
94     testWhileFalse = (
95         | i |
96         i := 1.
97         [ i >= 10 ] whileFalse: [ i := i + 1 ].
98         self assert: 10 equals: i.
99     )
100 )
101

```

```

1 Block4Test = TestCase (
2   |escape_count escaped_block|
3
4   simpleBlock = (
5     ^ [ 42 ]
6   )
7
8   incBlock = (
9     ^ [:val | val + 1 ]
10  )
11
12  "This requires a closure"
13  adderBlock: amount = (
14    ^ [:val | amount + val ]
15  )
16
17  "Closure with mutable state in block"
18  counterBlock = (
19    |count|
20    count := 0.
21    ^ [ count := count + 1. count ]
22  )
23
24  selfKeeper = (
25    ^ [ self ]
26  )
27
28  escapingBlock = (
29    ^ [ ^ 42 ]
30  )
31
32  escapingNestedBlock = (
33    [ [ ^ [ ^ 43 ] ] value ] value
34  )
35
36  testSimpleBlocks = (
37    self assert: 42 equals: self simpleBlock value.
38    self assert: 4  equals: (self incBlock value: 3).
39    self assert: 43 equals: ((self adderBlock: 13) value: 30).
40  )
41
42  testClosure = (
43    | counter |
44    counter := self counterBlock.
45    self assert: 1 equals: counter value.
46    self assert: 2 equals: counter value.
47    self assert: 1 equals: self counterBlock value. "make sure each
copy is independent"
48    self assert: 3 equals: counter value.
49  )
50
51  testSelfInBlock = (
52    | test_inst |
53    test_inst := BlockTest new.
54    self assert: test_inst equals: test_inst selfKeeper value.
55    self assert: self      equals: self selfKeeper value.
56  )
57

```

```

58     testEscapedBlock = (
59         | escaping_block |
60
61         escape_count := 0.
62
63         escaping_block := self escapingBlock.
64
65         self assert: 0 equals: escape_count.
66         self assert: 666 equals: escaping_block value.
67         self assert: 1 equals: escape_count.
68
69         self assert: escaping_block is: escaped_block.
70
71
72         escaping_block := self escapingNestedBlock.
73         self assert: 1 equals: escape_count.
74         self assert: 666 equals: escaping_block value.
75         self assert: 2 equals: escape_count.
76         self assert: escaping_block is: escaped_block.
77     )
78
79     escapedBlock: block = (
80         escape_count := escape_count + 1.
81         escaped_block := block.
82
83         "return some dummy value to the object that sent 'value' to block"
84         ^666
85     )
86
87     testWhileTrue = (
88         | i |
89         i := 1.
90         [ i < 10 ] whileTrue: [ i := i + 1 ].
91         self assert: 10 equals: i.
92     )
93
94     testWhileFalse = (
95         | i |
96         i := 1.
97         [ i >= 10 ] whileFalse: [ i := i + 1 ].
98         self assert: 10 equals: i.
99     )
100 )
101

```

```

1 Block5Test = TestCase (
2   |escape_count escaped_block|
3
4   simpleBlock = (
5     ^ [ 42 ]
6   )
7
8   incBlock = (
9     ^ [:val | val + 1 ]
10  )
11
12  "This requires a closure"
13  adderBlock: amount = (
14    ^ [:val | amount + val ]
15  )
16
17  "Closure with mutable state in block"
18  counterBlock = (
19    |count|
20    count := 0.
21    ^ [ count := count + 1. count ]
22  )
23
24  selfKeeper = (
25    ^ [ self ]
26  )
27
28  escapingBlock = (
29    ^ [ ^ 42 ]
30  )
31
32  escapingNestedBlock = (
33    [ [ ^ [ ^ 43 ] ] value ] value
34  )
35
36  testSimpleBlocks = (
37    self assert: 42 equals: self simpleBlock value.
38    self assert: 4  equals: (self incBlock value: 3).
39    self assert: 43 equals: ((self adderBlock: 13) value: 30).
40  )
41
42  testClosure = (
43    | counter |
44    counter := self counterBlock.
45    self assert: 1 equals: counter value.
46    self assert: 2 equals: counter value.
47    self assert: 1 equals: self counterBlock value. "make sure each
copy is independent"
48    self assert: 3 equals: counter value.
49  )
50
51  testSelfInBlock = (
52    | test_inst |
53    test_inst := BlockTest new.
54    self assert: test_inst equals: test_inst selfKeeper value.
55    self assert: self      equals: self selfKeeper value.
56  )
57

```

```

58     testEscapedBlock = (
59         | escaping_block |
60
61         escape_count := 0.
62
63         escaping_block := self escapingBlock.
64
65         self assert: 0 equals: escape_count.
66         self assert: 666 equals: escaping_block value.
67         self assert: 1 equals: escape_count.
68
69         self assert: escaping_block is: escaped_block.
70
71
72         escaping_block := self escapingNestedBlock.
73         self assert: 1 equals: escape_count.
74         self assert: 666 equals: escaping_block value.
75         self assert: 2 equals: escape_count.
76         self assert: escaping_block is: escaped_block.
77     )
78
79     escapedBlock: block = (
80         escape_count := escape_count + 1.
81         escaped_block := block.
82
83         "return some dummy value to the object that sent 'value' to block"
84         ^666
85     )
86
87     testWhileTrue = (
88         | i |
89         i := 1.
90         [ i < 10 ] whileTrue: [ i := i + 1 ].
91         self assert: 10 equals: i.
92     )
93
94     testWhileFalse = (
95         | i |
96         i := 1.
97         [ i >= 10 ] whileFalse: [ i := i + 1 ].
98         self assert: 10 equals: i.
99     )
100 )
101

```

```

1 BlockTest = TestCase (
2   |escape_count escaped_block|
3
4   simpleBlock = (
5     ^ [ 42 ]
6   )
7
8   incBlock = (
9     ^ [:val | val + 1 ]
10  )
11
12  "This requires a closure"
13  adderBlock: amount = (
14    ^ [:val | amount + val ]
15  )
16
17  "Closure with mutable state in block"
18  counterBlock = (
19    |count|
20    count := 0.
21    ^ [ count := count + 1. count ]
22  )
23
24  selfKeeper = (
25    ^ [ self ]
26  )
27
28  escapingBlock = (
29    ^ [ ^ 42 ]
30  )
31
32  escapingNestedBlock = (
33    [ [ ^ [ ^ 43 ] ] value ] value
34  )
35
36  testSimpleBlocks = (
37    self assert: 42 equals: self simpleBlock value.
38    self assert: 4  equals: (self incBlock value: 3).
39    self assert: 43 equals: ((self adderBlock: 13) value: 30).
40  )
41
42  testClosure = (
43    | counter |
44    counter := self counterBlock.
45    self assert: 1 equals: counter value.
46    self assert: 2 equals: counter value.
47    self assert: 1 equals: self counterBlock value. "make sure each
copy is independent"
48    self assert: 3 equals: counter value.
49  )
50
51  testSelfInBlock = (
52    | test_inst |
53    test_inst := BlockTest new.
54    self assert: test_inst equals: test_inst selfKeeper value.
55    self assert: self      equals: self selfKeeper value.
56  )
57

```



```

58     testEscapedBlock = (
59         | escaping_block |
60
61         escape_count := 0.
62
63         escaping_block := self escapingBlock.
64
65         self assert: 0 equals: escape_count.
66         self assert: 666 equals: escaping_block value.
67         self assert: 1 equals: escape_count.
68
69         self assert: escaping_block is: escaped_block.
70
71
72         escaping_block := self escapingNestedBlock.
73         self assert: 1 equals: escape_count.
74         self assert: 666 equals: escaping_block value.
75         self assert: 2 equals: escape_count.
76         self assert: escaping_block is: escaped_block.
77     )
78
79     escapedBlock: block = (
80         escape_count := escape_count + 1.
81         escaped_block := block.
82
83         "return some dummy value to the object that sent 'value' to block"
84         ^666
85     )
86
87     testWhileTrue = (
88         | i |
89         i := 1.
90         [ i < 10 ] whileTrue: [ i := i + 1 ].
91         self assert: 10 equals: i.
92     )
93
94     testWhileFalse = (
95         | i |
96         i := 1.
97         [ i >= 10 ] whileFalse: [ i := i + 1 ].
98         self assert: 10 equals: i.
99     )
100 )
101

```

```

1 Boolean2Test = TestCase (
2
3   testIfTrueIfFalse = (
4     | b1 b2 |
5     b1 := false.
6     b2 := false.
7
8     true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
9     self assert: b1.
10    self deny: b2.
11
12    b1 := false.
13    b2 := false.
14    false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
15    self assert: b2.
16    self deny: b1.
17  )
18
19  testIfTrue = (
20    | b |
21    b := false.
22
23    true ifTrue: [ b := true ].
24    self assert: b.
25
26    b := false.
27    false ifTrue: [ b := true ].
28    self deny: b.
29  )
30
31  testIfFalse = (
32    | b |
33    b := false.
34
35    true ifFalse: [ b := true ].
36    self deny: b.
37
38    b := false.
39    false ifFalse: [ b := true ].
40    self assert: b.
41  )
42
43  testNot = (
44    self deny: true not.
45    self assert: false not.
46  )
47
48  andBoolTrueTrue   = ( ^ true   and: [ true ] )
49  andBoolTrueFalse  = ( ^ true   and: [ false ] )
50  andBoolFalseTrue  = ( ^ false  and: [ true ] )
51  andBoolFalseFalse = ( ^ false  and: [ false ] )
52
53  testAnd = (
54    self assert: self andBoolTrueTrue.
55    self deny:   self andBoolTrueFalse.
56    self deny:   self andBoolFalseTrue.
57    self deny:   self andBoolFalseFalse.
58  )

```

```

59
60     ampBoolTrueTrue   = ( ^ true  && [ true  ] )
61     ampBoolTrueFalse  = ( ^ true  && [ false ] )
62     ampBoolFalseTrue  = ( ^ false && [ true  ] )
63     ampBoolFalseFalse = ( ^ false && [ false ] )
64
65     testAmp = (
66         self assert: self ampBoolTrueTrue.
67         self deny:   self ampBoolTrueFalse.
68         self deny:   self ampBoolFalseTrue.
69         self deny:   self ampBoolFalseFalse.
70     )
71
72     orBoolTrueTrue    = ( ^ true  or: [ true  ] )
73     orBoolTrueFalse   = ( ^ true  or: [ false ] )
74     orBoolFalseTrue   = ( ^ false or: [ true  ] )
75     orBoolFalseFalse  = ( ^ false or: [ false ] )
76
77     testOr = (
78         self assert: self orBoolTrueTrue.
79         self assert: self orBoolTrueFalse.
80         self assert: self orBoolFalseTrue.
81         self deny:   self orBoolFalseFalse.
82     )
83
84     pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
85     pipeBoolTrueFalse  = ( ^ true  || [ false ] )
86     pipeBoolFalseTrue  = ( ^ false || [ true  ] )
87     pipeBoolFalseFalse = ( ^ false || [ false ] )
88
89     testPipe = (
90         self assert: self pipeBoolTrueTrue.
91         self assert: self pipeBoolTrueFalse.
92         self assert: self pipeBoolFalseTrue.
93         self deny:   self pipeBoolFalseFalse.
94     )
95
96     testAsString = (
97         self assert: 'true'  equals: true asString.
98         self assert: 'false' equals: false asString.
99     )
100 )
101

```

```

1 Boolean3Test = TestCase (
2
3   testIfTrueIfFalse = (
4     | b1 b2 |
5     b1 := false.
6     b2 := false.
7
8     true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
9     self assert: b1.
10    self deny: b2.
11
12    b1 := false.
13    b2 := false.
14    false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
15    self assert: b2.
16    self deny: b1.
17  )
18
19  testIfTrue = (
20    | b |
21    b := false.
22
23    true ifTrue: [ b := true ].
24    self assert: b.
25
26    b := false.
27    false ifTrue: [ b := true ].
28    self deny: b.
29  )
30
31  testIfFalse = (
32    | b |
33    b := false.
34
35    true ifFalse: [ b := true ].
36    self deny: b.
37
38    b := false.
39    false ifFalse: [ b := true ].
40    self assert: b.
41  )
42
43  testNot = (
44    self deny: true not.
45    self assert: false not.
46  )
47
48  andBoolTrueTrue   = ( ^ true  and: [ true  ] )
49  andBoolTrueFalse  = ( ^ true  and: [ false ] )
50  andBoolFalseTrue  = ( ^ false and: [ true  ] )
51  andBoolFalseFalse = ( ^ false and: [ false ] )
52
53  testAnd = (
54    self assert: self andBoolTrueTrue.
55    self deny:   self andBoolTrueFalse.
56    self deny:   self andBoolFalseTrue.
57    self deny:   self andBoolFalseFalse.
58  )

```

```

59
60     ampBoolTrueTrue   = ( ^ true  && [ true  ] )
61     ampBoolTrueFalse  = ( ^ true  && [ false ] )
62     ampBoolFalseTrue  = ( ^ false && [ true  ] )
63     ampBoolFalseFalse = ( ^ false && [ false ] )
64
65     testAmp = (
66         self assert: self ampBoolTrueTrue.
67         self deny:   self ampBoolTrueFalse.
68         self deny:   self ampBoolFalseTrue.
69         self deny:   self ampBoolFalseFalse.
70     )
71
72     orBoolTrueTrue    = ( ^ true  or: [ true  ] )
73     orBoolTrueFalse   = ( ^ true  or: [ false ] )
74     orBoolFalseTrue   = ( ^ false or: [ true  ] )
75     orBoolFalseFalse  = ( ^ false or: [ false ] )
76
77     testOr = (
78         self assert: self orBoolTrueTrue.
79         self assert: self orBoolTrueFalse.
80         self assert: self orBoolFalseTrue.
81         self deny:   self orBoolFalseFalse.
82     )
83
84     pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
85     pipeBoolTrueFalse  = ( ^ true  || [ false ] )
86     pipeBoolFalseTrue  = ( ^ false || [ true  ] )
87     pipeBoolFalseFalse = ( ^ false || [ false ] )
88
89     testPipe = (
90         self assert: self pipeBoolTrueTrue.
91         self assert: self pipeBoolTrueFalse.
92         self assert: self pipeBoolFalseTrue.
93         self deny:   self pipeBoolFalseFalse.
94     )
95
96     testAsString = (
97         self assert: 'true'  equals: true asString.
98         self assert: 'false' equals: false asString.
99     )
100 )
101

```

```

1 Boolean4Test = TestCase (
2
3   testIfTrueIfFalse = (
4     | b1 b2 |
5     b1 := false.
6     b2 := false.
7
8     true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
9     self assert: b1.
10    self deny: b2.
11
12    b1 := false.
13    b2 := false.
14    false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
15    self assert: b2.
16    self deny: b1.
17  )
18
19  testIfTrue = (
20    | b |
21    b := false.
22
23    true ifTrue: [ b := true ].
24    self assert: b.
25
26    b := false.
27    false ifTrue: [ b := true ].
28    self deny: b.
29  )
30
31  testIfFalse = (
32    | b |
33    b := false.
34
35    true ifFalse: [ b := true ].
36    self deny: b.
37
38    b := false.
39    false ifFalse: [ b := true ].
40    self assert: b.
41  )
42
43  testNot = (
44    self deny: true not.
45    self assert: false not.
46  )
47
48  andBoolTrueTrue   = ( ^ true   and: [ true ] )
49  andBoolTrueFalse  = ( ^ true   and: [ false ] )
50  andBoolFalseTrue  = ( ^ false  and: [ true ] )
51  andBoolFalseFalse = ( ^ false  and: [ false ] )
52
53  testAnd = (
54    self assert: self andBoolTrueTrue.
55    self deny:   self andBoolTrueFalse.
56    self deny:   self andBoolFalseTrue.
57    self deny:   self andBoolFalseFalse.
58  )

```

```

59
60     ampBoolTrueTrue   = ( ^ true  && [ true ] )
61     ampBoolTrueFalse  = ( ^ true  && [ false ] )
62     ampBoolFalseTrue  = ( ^ false && [ true ] )
63     ampBoolFalseFalse = ( ^ false && [ false ] )
64
65     testAmp = (
66         self assert: self ampBoolTrueTrue.
67         self deny:   self ampBoolTrueFalse.
68         self deny:   self ampBoolFalseTrue.
69         self deny:   self ampBoolFalseFalse.
70     )
71
72     orBoolTrueTrue    = ( ^ true  or: [ true ] )
73     orBoolTrueFalse   = ( ^ true  or: [ false ] )
74     orBoolFalseTrue   = ( ^ false or: [ true ] )
75     orBoolFalseFalse  = ( ^ false or: [ false ] )
76
77     testOr = (
78         self assert: self orBoolTrueTrue.
79         self assert: self orBoolTrueFalse.
80         self assert: self orBoolFalseTrue.
81         self deny:   self orBoolFalseFalse.
82     )
83
84     pipeBoolTrueTrue   = ( ^ true  || [ true ] )
85     pipeBoolTrueFalse  = ( ^ true  || [ false ] )
86     pipeBoolFalseTrue  = ( ^ false || [ true ] )
87     pipeBoolFalseFalse = ( ^ false || [ false ] )
88
89     testPipe = (
90         self assert: self pipeBoolTrueTrue.
91         self assert: self pipeBoolTrueFalse.
92         self assert: self pipeBoolFalseTrue.
93         self deny:   self pipeBoolFalseFalse.
94     )
95
96     testAsString = (
97         self assert: 'true'  equals: true asString.
98         self assert: 'false' equals: false asString.
99     )
100 )
101

```

```

1 Boolean5Test = TestCase (
2
3   testIfTrueIfFalse = (
4     | b1 b2 |
5     b1 := false.
6     b2 := false.
7
8     true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
9     self assert: b1.
10    self deny: b2.
11
12    b1 := false.
13    b2 := false.
14    false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
15    self assert: b2.
16    self deny: b1.
17  )
18
19  testIfTrue = (
20    | b |
21    b := false.
22
23    true ifTrue: [ b := true ].
24    self assert: b.
25
26    b := false.
27    false ifTrue: [ b := true ].
28    self deny: b.
29  )
30
31  testIfFalse = (
32    | b |
33    b := false.
34
35    true ifFalse: [ b := true ].
36    self deny: b.
37
38    b := false.
39    false ifFalse: [ b := true ].
40    self assert: b.
41  )
42
43  testNot = (
44    self deny: true not.
45    self assert: false not.
46  )
47
48  andBoolTrueTrue   = ( ^ true   and: [ true ] )
49  andBoolTrueFalse  = ( ^ true   and: [ false ] )
50  andBoolFalseTrue  = ( ^ false  and: [ true ] )
51  andBoolFalseFalse = ( ^ false  and: [ false ] )
52
53  testAnd = (
54    self assert: self andBoolTrueTrue.
55    self deny:   self andBoolTrueFalse.
56    self deny:   self andBoolFalseTrue.
57    self deny:   self andBoolFalseFalse.
58  )

```



```

59
60     ampBoolTrueTrue   = ( ^ true  && [ true  ] )
61     ampBoolTrueFalse  = ( ^ true  && [ false ] )
62     ampBoolFalseTrue  = ( ^ false && [ true  ] )
63     ampBoolFalseFalse = ( ^ false && [ false ] )
64
65     testAmp = (
66         self assert: self ampBoolTrueTrue.
67         self deny:   self ampBoolTrueFalse.
68         self deny:   self ampBoolFalseTrue.
69         self deny:   self ampBoolFalseFalse.
70     )
71
72     orBoolTrueTrue    = ( ^ true  or: [ true  ] )
73     orBoolTrueFalse   = ( ^ true  or: [ false ] )
74     orBoolFalseTrue   = ( ^ false or: [ true  ] )
75     orBoolFalseFalse  = ( ^ false or: [ false ] )
76
77     testOr = (
78         self assert: self orBoolTrueTrue.
79         self assert: self orBoolTrueFalse.
80         self assert: self orBoolFalseTrue.
81         self deny:   self orBoolFalseFalse.
82     )
83
84     pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
85     pipeBoolTrueFalse  = ( ^ true  || [ false ] )
86     pipeBoolFalseTrue  = ( ^ false || [ true  ] )
87     pipeBoolFalseFalse = ( ^ false || [ false ] )
88
89     testPipe = (
90         self assert: self pipeBoolTrueTrue.
91         self assert: self pipeBoolTrueFalse.
92         self assert: self pipeBoolFalseTrue.
93         self deny:   self pipeBoolFalseFalse.
94     )
95
96     testAsString = (
97         self assert: 'true'  equals: true asString.
98         self assert: 'false' equals: false asString.
99     )
100 )
101

```

```

1 BooleanTest = TestCase (
2
3   testIfTrueIfFalse = (
4     | b1 b2 |
5     b1 := false.
6     b2 := false.
7
8     true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
9     self assert: b1.
10    self deny: b2.
11
12    b1 := false.
13    b2 := false.
14    false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
15    self assert: b2.
16    self deny: b1.
17  )
18
19  testIfTrue = (
20    | b |
21    b := false.
22
23    true ifTrue: [ b := true ].
24    self assert: b.
25
26    b := false.
27    false ifTrue: [ b := true ].
28    self deny: b.
29  )
30
31  testIfFalse = (
32    | b |
33    b := false.
34
35    true ifFalse: [ b := true ].
36    self deny: b.
37
38    b := false.
39    false ifFalse: [ b := true ].
40    self assert: b.
41  )
42
43  testNot = (
44    self deny: true not.
45    self assert: false not.
46  )
47
48  andBoolTrueTrue   = ( ^ true  and: [ true  ] )
49  andBoolTrueFalse  = ( ^ true  and: [ false ] )
50  andBoolFalseTrue  = ( ^ false and: [ true  ] )
51  andBoolFalseFalse = ( ^ false and: [ false ] )
52
53  testAnd = (
54    self assert: self andBoolTrueTrue.
55    self deny:   self andBoolTrueFalse.
56    self deny:   self andBoolFalseTrue.
57    self deny:   self andBoolFalseFalse.
58  )

```

```

59
60     ampBoolTrueTrue   = ( ^ true  && [ true  ] )
61     ampBoolTrueFalse  = ( ^ true  && [ false ] )
62     ampBoolFalseTrue  = ( ^ false && [ true  ] )
63     ampBoolFalseFalse = ( ^ false && [ false ] )
64
65     testAmp = (
66         self assert: self ampBoolTrueTrue.
67         self deny:   self ampBoolTrueFalse.
68         self deny:   self ampBoolFalseTrue.
69         self deny:   self ampBoolFalseFalse.
70     )
71
72     orBoolTrueTrue    = ( ^ true  or: [ true  ] )
73     orBoolTrueFalse   = ( ^ true  or: [ false ] )
74     orBoolFalseTrue   = ( ^ false or: [ true  ] )
75     orBoolFalseFalse  = ( ^ false or: [ false ] )
76
77     testOr = (
78         self assert: self orBoolTrueTrue.
79         self assert: self orBoolTrueFalse.
80         self assert: self orBoolFalseTrue.
81         self deny:   self orBoolFalseFalse.
82     )
83
84     pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
85     pipeBoolTrueFalse  = ( ^ true  || [ false ] )
86     pipeBoolFalseTrue  = ( ^ false || [ true  ] )
87     pipeBoolFalseFalse = ( ^ false || [ false ] )
88
89     testPipe = (
90         self assert: self pipeBoolTrueTrue.
91         self assert: self pipeBoolTrueFalse.
92         self assert: self pipeBoolFalseTrue.
93         self deny:   self pipeBoolFalseFalse.
94     )
95
96     testAsString = (
97         self assert: 'true'  equals: true asString.
98         self assert: 'false' equals: false asString.
99     )
100 )
101

```

```
1 ClassA = (  
2   | a b |  
3   result = (  
4     ^42  
5   )  
6   ----  
7   | c1 c2 c3 |  
8 )  
9
```

```
1 ClassB = ClassA (  
2   | c d |  
3   ----  
4   | c4 c5 c6 |  
5 )  
6
```

```

1 ClassC = ClassB (
2   | e f |
3   a      = ( ^ a )
4   a: val = ( a := val )
5
6   f      = ( ^ f )
7   f: val = ( f := val )
8
9   ----
10
11  | c7 c8 c9 |
12
13  setAllAndInc: anInt = (
14    c1 := anInt.
15    c2 := c1 + 1.
16    c3 := c2 + 1.
17    c4 := c3 + 1.
18    c5 := c4 + 1.
19    c6 := c5 + 1.
20    c7 := c6 + 1.
21    c8 := c7 + 1.
22    c9 := c8 + 1.
23  )
24
25  getAll = (
26    | arr |
27    arr := Array new: 9.
28    arr at: 1 put: c1.
29    arr at: 2 put: c2.
30    arr at: 3 put: c3.
31    arr at: 4 put: c4.
32    arr at: 5 put: c5.
33    arr at: 6 put: c6.
34    arr at: 7 put: c7.
35    arr at: 8 put: c8.
36    arr at: 9 put: c9.
37    ^ arr
38  )
39 )
40

```

```
1 ClassLoading2Test = TestCase (
2   testEqualityOfClasses = (
3     | a b c |
4     b := ClassB new.
5     a := ClassA new.
6     c := ClassC new.
7
8     self assert: 42 equals: b result.
9     self assert: 42 equals: c result.
10
11     self assert: a class equals: b class superclass.
12     self assert: b class equals: c class superclass.
13   )
14 )
15
```

```
1 ClassLoading3Test = TestCase (
2   testEqualityOfClasses = (
3     | a b c |
4     b := ClassB new.
5     a := ClassA new.
6     c := ClassC new.
7
8     self assert: 42 equals: b result.
9     self assert: 42 equals: c result.
10
11     self assert: a class equals: b class superclass.
12     self assert: b class equals: c class superclass.
13   )
14 )
15
```



```
1 ClassLoading4Test = TestCase (
2   testEqualityOfClasses = (
3     | a b c |
4     b := ClassB new.
5     a := ClassA new.
6     c := ClassC new.
7
8     self assert: 42 equals: b result.
9     self assert: 42 equals: c result.
10
11     self assert: a class equals: b class superclass.
12     self assert: b class equals: c class superclass.
13   )
14 )
15
```

```
1 ClassLoading5Test = TestCase (
2   testEqualityOfClasses = (
3     | a b c |
4     b := ClassB new.
5     a := ClassA new.
6     c := ClassC new.
7
8     self assert: 42 equals: b result.
9     self assert: 42 equals: c result.
10
11     self assert: a class equals: b class superclass.
12     self assert: b class equals: c class superclass.
13   )
14 )
15
```

```
1 ClassLoadingTest = TestCase (  
2   testEqualityOfClasses = (  
3     | a b c |  
4     b := ClassB new.  
5     a := ClassA new.  
6     c := ClassC new.  
7  
8     self assert: 42 equals: b result.  
9     self assert: 42 equals: c result.  
10  
11     self assert: a class equals: b class superclass.  
12     self assert: b class equals: c class superclass.  
13   )  
14 )  
15
```

```

1 ClassStructure2Test = TestCase (
2
3   testClassIdentity = (
4     self assert: Array equals: Array new class.
5     self assert: Integer equals: 1 class.
6     self assert: Integer equals: 10000000000 class.
7     self assert: Double equals: (1 // 2) class.
8     self assert: Double equals: 0.5 class.
9     self assert: Block1 equals: [42] class.
10    self assert: Object equals: Object new class.
11    self assert: Set equals: Set new class.
12    self assert: String equals: 'foo' class.
13    self assert: Symbol equals: #foo class.
14    self assert: True equals: true class.
15    self assert: False equals: false class.
16    self assert: Nil equals: nil class.
17
18    self assert: True superclass equals: False superclass.
19    self assert: True superclass equals: Boolean.
20    self assert: True superclass equals: Boolean.
21  )
22
23  testThatCertainMethodsArePrimitives = (
24    | m |
25    "This is a little fragile.
26     Index needs to be adapted with changing Class definition."
27    m := Object methods at: 1.
28    "self expect: #class equals: m signature."
29
30    self optional: #invokableTypes assert: Primitive equals: m class.
31    "Class>>#name should be a primitive."
32
33    m := Object methods at: 7.
34    "self expect: #asString equals: m signature."
35
36    self optional: #invokableTypes assert: Method equals: m class.
37    "Class>>#asString should be a normal method."
38  )
39
40  testAccessToInstanceFields = (
41    | o |
42    o := ClassC new.
43    o a: 333.
44    self assert: 333 equals: o a.
45
46    o f: 333.
47    self assert: 333 equals: o f.
48  )
49
50  testAccessToClassFields = (
51    | arr |
52    ClassC setAllAndInc: 4.
53    arr := ClassC getAll.
54    1 to: 9 do: [:i |
55      self assert: i + (4 - 1) equals: (arr at: i).
56    ].
57
58    "We do that here to make sure that class fields do not interfere with

```

```

57     other class properties."
58     self assert: ClassB      is: ClassC superclass.
59     self assert: Metaclass is: ClassC class class.
60     self assert: #ClassC     equals: ClassC name.
61 )
62
63 testMetaClasses = (
64     self assert: nil          is: Object superclass.
65     self assert: Integer      is: 1 class.
66     self assert: #'Integer class' is: 1 class class name.
67     self assert: Metaclass    is: 1 class class class.
68
69     self assert: #'Metaclass class' is: Metaclass class name.
70     self assert: Metaclass        is: Metaclass class class.
71
72     self assert: Object          is: 1 class superclass.
73     self assert: #'Object class' is: 1 class class superclass name.
74     self assert: Class          is: Object class superclass.
75     self assert: Metaclass      is: Class class class.
76 )
77
78 testInstanceFields = (
79     self assert: 2 equals: ClassA fields length.
80     self assert: 4 equals: ClassB fields length.
81     self assert: 6 equals: ClassC fields length.
82 )
83 )
84

```

```

1 ClassStructure3Test = TestCase (
2
3   testClassIdentity = (
4     self assert: Array equals: Array new class.
5     self assert: Integer equals: 1 class.
6     self assert: Integer equals: 10000000000 class.
7     self assert: Double equals: (1 // 2) class.
8     self assert: Double equals: 0.5 class.
9     self assert: Block1 equals: [42] class.
10    self assert: Object equals: Object new class.
11    self assert: Set equals: Set new class.
12    self assert: String equals: 'foo' class.
13    self assert: Symbol equals: #foo class.
14    self assert: True equals: true class.
15    self assert: False equals: false class.
16    self assert: Nil equals: nil class.
17
18    self assert: True superclass equals: False superclass.
19    self assert: True superclass equals: Boolean.
20    self assert: True superclass equals: Boolean.
21  )
22
23  testThatCertainMethodsArePrimitives = (
24    | m |
25    "This is a little fragile.
26     Index needs to be adapted with changing Class definition."
27    m := Object methods at: 1.
28    "self expect: #class equals: m signature."
29
30    self optional: #invokableTypes assert: Primitive equals: m class.
31    "Class>>#name should be a primitive."
32
33    m := Object methods at: 7.
34    "self expect: #asString equals: m signature."
35
36    self optional: #invokableTypes assert: Method equals: m class.
37    "Class>>#asString should be a normal method."
38  )
39
40  testAccessToInstanceFields = (
41    | o |
42    o := ClassC new.
43    o a: 333.
44    self assert: 333 equals: o a.
45
46    o f: 333.
47    self assert: 333 equals: o f.
48  )
49
50  testAccessToClassFields = (
51    | arr |
52    ClassC setAllAndInc: 4.
53    arr := ClassC getAll.
54    1 to: 9 do: [:i |
55      self assert: i + (4 - 1) equals: (arr at: i).
56    ].
57
58    "We do that here to make sure that class fields do not interfere with

```

```

57     other class properties."
58     self assert: ClassB      is: ClassC superclass.
59     self assert: Metaclass is: ClassC class class.
60     self assert: #ClassC     equals: ClassC name.
61 )
62
63 testMetaClasses = (
64     self assert: nil          is: Object superclass.
65     self assert: Integer      is: 1 class.
66     self assert: #'Integer class' is: 1 class class name.
67     self assert: Metaclass     is: 1 class class class.
68
69     self assert: #'Metaclass class' is: Metaclass class name.
70     self assert: Metaclass        is: Metaclass class class.
71
72     self assert: Object          is: 1 class superclass.
73     self assert: #'Object class' is: 1 class class superclass name.
74     self assert: Class           is: Object class superclass.
75     self assert: Metaclass       is: Class class class.
76 )
77
78 testInstanceFields = (
79     self assert: 2 equals: ClassA fields length.
80     self assert: 4 equals: ClassB fields length.
81     self assert: 6 equals: ClassC fields length.
82 )
83 )
84

```

```

1 ClassStructure4Test = TestCase (
2
3   testClassIdentity = (
4     self assert: Array equals: Array new class.
5     self assert: Integer equals: 1 class.
6     self assert: Integer equals: 10000000000 class.
7     self assert: Double equals: (1 // 2) class.
8     self assert: Double equals: 0.5 class.
9     self assert: Block1 equals: [42] class.
10    self assert: Object equals: Object new class.
11    self assert: Set equals: Set new class.
12    self assert: String equals: 'foo' class.
13    self assert: Symbol equals: #foo class.
14    self assert: True equals: true class.
15    self assert: False equals: false class.
16    self assert: Nil equals: nil class.
17
18    self assert: True superclass equals: False superclass.
19    self assert: True superclass equals: Boolean.
20    self assert: True superclass equals: Boolean.
21  )
22
23  testThatCertainMethodsArePrimitives = (
24    | m |
25    "This is a little fragile.
26     Index needs to be adapted with changing Class definition."
27    m := Object methods at: 1.
28    "self expect: #class equals: m signature."
29
30    self optional: #invokableTypes assert: Primitive equals: m class.
31    "Class>>#name should be a primitive."
32
33    m := Object methods at: 7.
34    "self expect: #asString equals: m signature."
35
36    self optional: #invokableTypes assert: Method equals: m class.
37    "Class>>#asString should be a normal method."
38  )
39
40  testAccessToInstanceFields = (
41    | o |
42    o := ClassC new.
43    o a: 333.
44    self assert: 333 equals: o a.
45
46    o f: 333.
47    self assert: 333 equals: o f.
48  )
49
50  testAccessToClassFields = (
51    | arr |
52    ClassC setAllAndInc: 4.
53    arr := ClassC getAll.
54    1 to: 9 do: [:i |
55      self assert: i + (4 - 1) equals: (arr at: i).
56    ].
57
58    "We do that here to make sure that class fields do not interfere with

```



```

57     other class properties."
58     self assert: ClassB      is: ClassC superclass.
59     self assert: Metaclass is: ClassC class class.
60     self assert: #ClassC     equals: ClassC name.
61 )
62
63 testMetaClasses = (
64     self assert: nil          is: Object superclass.
65     self assert: Integer      is: 1 class.
66     self assert: #'Integer class' is: 1 class class name.
67     self assert: Metaclass    is: 1 class class class.
68
69     self assert: #'Metaclass class' is: Metaclass class name.
70     self assert: Metaclass        is: Metaclass class class.
71
72     self assert: Object          is: 1 class superclass.
73     self assert: #'Object class' is: 1 class class superclass name.
74     self assert: Class          is: Object class superclass.
75     self assert: Metaclass      is: Class class class.
76 )
77
78 testInstanceFields = (
79     self assert: 2 equals: ClassA fields length.
80     self assert: 4 equals: ClassB fields length.
81     self assert: 6 equals: ClassC fields length.
82 )
83 )
84

```

```

1 ClassStructure5Test = TestCase (
2
3   testClassIdentity = (
4     self assert: Array equals: Array new class.
5     self assert: Integer equals: 1 class.
6     self assert: Integer equals: 10000000000 class.
7     self assert: Double equals: (1 // 2) class.
8     self assert: Double equals: 0.5 class.
9     self assert: Block1 equals: [42] class.
10    self assert: Object equals: Object new class.
11    self assert: Set equals: Set new class.
12    self assert: String equals: 'foo' class.
13    self assert: Symbol equals: #foo class.
14    self assert: True equals: true class.
15    self assert: False equals: false class.
16    self assert: Nil equals: nil class.
17
18    self assert: True superclass equals: False superclass.
19    self assert: True superclass equals: Boolean.
20    self assert: True superclass equals: Boolean.
21  )
22
23  testThatCertainMethodsArePrimitives = (
24    | m |
25    "This is a little fragile.
26     Index needs to be adapted with changing Class definition."
27    m := Object methods at: 1.
28    "self expect: #class equals: m signature."
29
30    self optional: #invokableTypes assert: Primitive equals: m class.
31    "Class>>#name should be a primitive."
32
33    m := Object methods at: 7.
34    "self expect: #asString equals: m signature."
35
36    self optional: #invokableTypes assert: Method equals: m class.
37    "Class>>#asString should be a normal method."
38  )
39
40  testAccessToInstanceFields = (
41    | o |
42    o := ClassC new.
43    o a: 333.
44    self assert: 333 equals: o a.
45
46    o f: 333.
47    self assert: 333 equals: o f.
48  )
49
50  testAccessToClassFields = (
51    | arr |
52    ClassC setAllAndInc: 4.
53    arr := ClassC getAll.
54    1 to: 9 do: [:i |
55      self assert: i + (4 - 1) equals: (arr at: i).
56    ].
57
58    "We do that here to make sure that class fields do not interfere with

```

```

57     other class properties."
58     self assert: ClassB      is: ClassC superclass.
59     self assert: Metaclass is: ClassC class class.
60     self assert: #ClassC     equals: ClassC name.
61 )
62
63 testMetaClasses = (
64     self assert: nil          is: Object superclass.
65     self assert: Integer      is: 1 class.
66     self assert: #'Integer class' is: 1 class class name.
67     self assert: Metaclass     is: 1 class class class.
68
69     self assert: #'Metaclass class' is: Metaclass class name.
70     self assert: Metaclass        is: Metaclass class class.
71
72     self assert: Object          is: 1 class superclass.
73     self assert: #'Object class' is: 1 class class superclass name.
74     self assert: Class           is: Object class superclass.
75     self assert: Metaclass       is: Class class class.
76 )
77
78 testInstanceFields = (
79     self assert: 2 equals: ClassA fields length.
80     self assert: 4 equals: ClassB fields length.
81     self assert: 6 equals: ClassC fields length.
82 )
83 )
84

```

```

1 ClassStructureTest = TestCase (
2
3   testClassIdentity = (
4     self assert: Array equals: Array new class.
5     self assert: Integer equals: 1 class.
6     self assert: Integer equals: 10000000000 class.
7     self assert: Double equals: (1 // 2) class.
8     self assert: Double equals: 0.5 class.
9     self assert: Block1 equals: [42] class.
10    self assert: Object equals: Object new class.
11    self assert: Set equals: Set new class.
12    self assert: String equals: 'foo' class.
13    self assert: Symbol equals: #foo class.
14    self assert: True equals: true class.
15    self assert: False equals: false class.
16    self assert: Nil equals: nil class.
17
18    self assert: True superclass equals: False superclass.
19    self assert: True superclass equals: Boolean.
20    self assert: True superclass equals: Boolean.
21  )
22
23  testThatCertainMethodsArePrimitives = (
24    | m |
25    "This is a little fragile.
26     Index needs to be adapted with changing Class definition."
27    m := Object methods at: 1.
28    "self expect: #class equals: m signature."
29
30    self optional: #invokableTypes assert: Primitive equals: m class.
31    "Class>>#name should be a primitive."
32
33    m := Object methods at: 7.
34    "self expect: #asString equals: m signature."
35
36    self optional: #invokableTypes assert: Method equals: m class.
37    "Class>>#asString should be a normal method."
38  )
39
40  testAccessToInstanceFields = (
41    | o |
42    o := ClassC new.
43    o a: 333.
44    self assert: 333 equals: o a.
45
46    o f: 333.
47    self assert: 333 equals: o f.
48  )
49
50  testAccessToClassFields = (
51    | arr |
52    ClassC setAllAndInc: 4.
53    arr := ClassC getAll.
54    1 to: 9 do: [:i |
55      self assert: i + (4 - 1) equals: (arr at: i).
56    ].
57
58    "We do that here to make sure that class fields do not interfere with

```

```

57     other class properties."
58     self assert: ClassB      is: ClassC superclass.
59     self assert: Metaclass is: ClassC class class.
60     self assert: #ClassC     equals: ClassC name.
61 )
62
63 testMetaClasses = (
64     self assert: nil          is: Object superclass.
65     self assert: Integer      is: 1 class.
66     self assert: #'Integer class' is: 1 class class name.
67     self assert: Metaclass     is: 1 class class class.
68
69     self assert: #'Metaclass class' is: Metaclass class name.
70     self assert: Metaclass        is: Metaclass class class.
71
72     self assert: Object          is: 1 class superclass.
73     self assert: #'Object class' is: 1 class class superclass name.
74     self assert: Class           is: Object class superclass.
75     self assert: Metaclass       is: Class class class.
76 )
77
78 testInstanceFields = (
79     self assert: 2 equals: ClassA fields length.
80     self assert: 4 equals: ClassB fields length.
81     self assert: 6 equals: ClassC fields length.
82 )
83 )
84

```

```
1 "  
2  
3 $Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "This test verifies that SOM blocks are indeed closures. The test was  
found on  
27 Eliot Miranda's Cog Blog."  
28  
29 Closure2Test = TestCase (  
30   testClosureProperty = (  
31     | factorial result facts |  
32  
33     facts := Array new: 10.  
34     facts at: 1 put: 1.  
35     facts at: 2 put: 2.  
36     facts at: 3 put: 6.  
37     facts at: 4 put: 24.  
38     facts at: 5 put: 120.  
39     facts at: 6 put: 720.  
40     facts at: 7 put: 5040.  
41     facts at: 8 put: 40320.  
42     facts at: 9 put: 362880.  
43     facts at: 10 put: 3628800.  
44  
45     factorial := [ :n |  
46       n = 1  
47         ifTrue: [ 1 ]  
48         ifFalse: [ (factorial value: n - 1) * n ] ].  
49  
50     result := (1 to: 10) collect: factorial.  
51     result doIndexes: [ :i |  
52       self assert: (facts at: i) equals: (result at: i) ]  
53   )
```

54)
55

```
1 "  
2  
3 $Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "This test verifies that SOM blocks are indeed closures. The test was  
found on  
27 Eliot Miranda's Cog Blog."  
28  
29 Closure3Test = TestCase (  
30   testClosureProperty = (  
31     | factorial result facts |  
32  
33     facts := Array new: 10.  
34     facts at: 1 put: 1.  
35     facts at: 2 put: 2.  
36     facts at: 3 put: 6.  
37     facts at: 4 put: 24.  
38     facts at: 5 put: 120.  
39     facts at: 6 put: 720.  
40     facts at: 7 put: 5040.  
41     facts at: 8 put: 40320.  
42     facts at: 9 put: 362880.  
43     facts at: 10 put: 3628800.  
44  
45     factorial := [ :n |  
46       n = 1  
47         ifTrue: [ 1 ]  
48         ifFalse: [ (factorial value: n - 1) * n ] ].  
49  
50     result := (1 to: 10) collect: factorial.  
51     result doIndexes: [ :i |  
52       self assert: (facts at: i) equals: (result at: i) ]  
53   )
```


54)
55

```

1 "
2
3 $Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 "This test verifies that SOM blocks are indeed closures. The test was
found on
27 Eliot Miranda's Cog Blog."
28
29 Closure4Test = TestCase (
30   testClosureProperty = (
31     | factorial result facts |
32
33     facts := Array new: 10.
34     facts at: 1 put: 1.
35     facts at: 2 put: 2.
36     facts at: 3 put: 6.
37     facts at: 4 put: 24.
38     facts at: 5 put: 120.
39     facts at: 6 put: 720.
40     facts at: 7 put: 5040.
41     facts at: 8 put: 40320.
42     facts at: 9 put: 362880.
43     facts at: 10 put: 3628800.
44
45     factorial := [ :n |
46       n = 1
47         ifTrue: [ 1 ]
48         ifFalse: [ (factorial value: n - 1) * n ] ].
49
50     result := (1 to: 10) collect: factorial.
51     result doIndexes: [ :i |
52       self assert: (facts at: i) equals: (result at: i) ]
53   )

```

54)
55

```
1 "  
2  
3 $Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "This test verifies that SOM blocks are indeed closures. The test was  
found on  
27 Eliot Miranda's Cog Blog."  
28  
29 Closure5Test = TestCase (  
30   testClosureProperty = (  
31     | factorial result facts |  
32  
33     facts := Array new: 10.  
34     facts at: 1 put: 1.  
35     facts at: 2 put: 2.  
36     facts at: 3 put: 6.  
37     facts at: 4 put: 24.  
38     facts at: 5 put: 120.  
39     facts at: 6 put: 720.  
40     facts at: 7 put: 5040.  
41     facts at: 8 put: 40320.  
42     facts at: 9 put: 362880.  
43     facts at: 10 put: 3628800.  
44  
45     factorial := [ :n |  
46       n = 1  
47         ifTrue: [ 1 ]  
48         ifFalse: [ (factorial value: n - 1) * n ] ].  
49  
50     result := (1 to: 10) collect: factorial.  
51     result doIndexes: [ :i |  
52       self assert: (facts at: i) equals: (result at: i) ]  
53   )
```

54)
55

```
1 "  
2  
3 $Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "This test verifies that SOM blocks are indeed closures. The test was  
found on  
27 Eliot Miranda's Cog Blog."  
28  
29 ClosureTest = TestCase (  
30   testClosureProperty = (  
31     | factorial result facts |  
32  
33     facts := Array new: 10.  
34     facts at: 1 put: 1.  
35     facts at: 2 put: 2.  
36     facts at: 3 put: 6.  
37     facts at: 4 put: 24.  
38     facts at: 5 put: 120.  
39     facts at: 6 put: 720.  
40     facts at: 7 put: 5040.  
41     facts at: 8 put: 40320.  
42     facts at: 9 put: 362880.  
43     facts at: 10 put: 3628800.  
44  
45     factorial := [ :n |  
46       n = 1  
47         ifTrue: [ 1 ]  
48         ifFalse: [ (factorial value: n - 1) * n ] ].  
49  
50     result := (1 to: 10) collect: factorial.  
51     result doIndexes: [ :i |  
52       self assert: (facts at: i) equals: (result at: i) ]  
53   )
```

54)
55

```
1 "  
2  
3 $Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Coercion2Test = TestCase (  
29  
30     testBasicNumberCoercion = (  
31         self assert: 5 equals: 25 sqrt.  
32         self assert: 1 equals: (2 // 4) * 2.  
33         self assert: 1 equals: 2 * (2 // 4).  
34     )  
35 )  
36
```



```
1 "  
2  
3 $Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Coercion3Test = TestCase (  
29  
30   testBasicNumberCoercion = (  
31     self assert: 5 equals: 25 sqrt.  
32     self assert: 1 equals: (2 // 4) * 2.  
33     self assert: 1 equals: 2 * (2 // 4).  
34   )  
35 )  
36
```

```
1 "  
2  
3 $Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Coercion4Test = TestCase (  
29  
30   testBasicNumberCoercion = (  
31     self assert: 5 equals: 25 sqrt.  
32     self assert: 1 equals: (2 // 4) * 2.  
33     self assert: 1 equals: 2 * (2 // 4).  
34   )  
35 )  
36
```

```
1 "  
2  
3 $Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Coercion5Test = TestCase (  
29  
30     testBasicNumberCoercion = (  
31         self assert: 5 equals: 25 sqrt.  
32         self assert: 1 equals: (2 // 4) * 2.  
33         self assert: 1 equals: 2 * (2 // 4).  
34     )  
35 )  
36
```

```
1 "  
2  
3 $Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 CoercionTest = TestCase (  
29  
30     testBasicNumberCoercion = (  
31         self assert: 5 equals: 25 sqrt.  
32         self assert: 1 equals: (2 // 4) * 2.  
33         self assert: 1 equals: 2 * (2 // 4).  
34     )  
35 )  
36
```

```

1  "
2
3  $Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5  Copyright (c) 2009-2013 see AUTHORS file
6  Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7  http://www.hpi.uni-potsdam.de/swa/
8
9  Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 CompilerReturn2Test = TestCase (
29
30     return1 = ( ^self )
31     return2 = (      )
32
33     return3: arg = ( ^self )
34     return4: arg = (      )
35
36     testExplicitAndImplicitReturns = (
37         self assert: self is: self return1.
38         self assert: self is: self return2.
39         self assert: self is: (self return3: 23).
40         self assert: self is: (self return4: 23).
41     )
42
43
44     "In SOM++, code after the #ifTrue: does not seem to be executed, if
the
45     block expression ends with a dot."
46     testIfTrueWithDot = (
47         | arr |
48         arr := Array new: 3.
49         self usesIfTrueWithDot: arr.
50         self assertArrayCorrectness: arr.
51     )
52

```

```

53     assertArrayCorrectness: arr = (
54         self assert: 1 equals: (arr at: 1). "method was not executed"
55         self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
56         self assert: 3 equals: (arr at: 3). "remainder was not
executed"
57     )
58
59     testIfTrueWithoutDot = (
60         | arr |
61         arr := Array new: 3.
62         self usesIfTrueWithoutDot: arr.
63         self assertArrayCorrectness: arr.
64     )
65
66     testIfFalseWithDot = (
67         | arr |
68         arr := Array new: 3.
69         self usesIfFalseWithDot: arr.
70         self assertArrayCorrectness: arr.
71     )
72
73     testIfFalseWithoutDot = (
74         | arr |
75         arr := Array new: 3.
76         self usesIfFalseWithoutDot: arr.
77         self assertArrayCorrectness: arr.
78     )
79
80     usesIfTrueWithDot: arr = (
81         arr at: 1 put: 1.
82         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
83         arr at: 3 put: 3.
84     )
85
86     usesIfTrueWithoutDot: arr = (
87         arr at: 1 put: 1.
88         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
89         arr at: 3 put: 3.
90     )
91
92     usesIfFalseWithDot: arr = (
93         arr at: 1 put: 1.
94         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
95         arr at: 3 put: 3.
96     )
97
98     usesIfFalseWithoutDot: arr = (
99         arr at: 1 put: 1.
100        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
101        arr at: 3 put: 3.
102    )
103
104    testWriteArgument = (
105        self assert: 42 equals: (self dec: 43).
106    )
107
108    dec: anInt = (
109        anInt := anInt - 1.
110        ^ anInt
111    )
112 )

```



```

1  "
2
3  $Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5  Copyright (c) 2009-2013 see AUTHORS file
6  Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7  http://www.hpi.uni-potsdam.de/swa/
8
9  Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 CompilerReturn3Test = TestCase (
29
30     return1 = ( ^self )
31     return2 = (      )
32
33     return3: arg = ( ^self )
34     return4: arg = (      )
35
36     testExplicitAndImplicitReturns = (
37         self assert: self is: self return1.
38         self assert: self is: self return2.
39         self assert: self is: (self return3: 23).
40         self assert: self is: (self return4: 23).
41     )
42
43
44     "In SOM++, code after the #ifTrue: does not seem to be executed, if
the
45     block expression ends with a dot."
46     testIfTrueWithDot = (
47         | arr |
48         arr := Array new: 3.
49         self usesIfTrueWithDot: arr.
50         self assertArrayCorrectness: arr.
51     )
52

```



```

53     assertArrayCorrectness: arr = (
54         self assert: 1 equals: (arr at: 1). "method was not executed"
55         self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
56         self assert: 3 equals: (arr at: 3). "remainder was not
executed"
57     )
58
59     testIfTrueWithoutDot = (
60         | arr |
61         arr := Array new: 3.
62         self usesIfTrueWithoutDot: arr.
63         self assertArrayCorrectness: arr.
64     )
65
66     testIfFalseWithDot = (
67         | arr |
68         arr := Array new: 3.
69         self usesIfFalseWithDot: arr.
70         self assertArrayCorrectness: arr.
71     )
72
73     testIfFalseWithoutDot = (
74         | arr |
75         arr := Array new: 3.
76         self usesIfFalseWithoutDot: arr.
77         self assertArrayCorrectness: arr.
78     )
79
80     usesIfTrueWithDot: arr = (
81         arr at: 1 put: 1.
82         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
83         arr at: 3 put: 3.
84     )
85
86     usesIfTrueWithoutDot: arr = (
87         arr at: 1 put: 1.
88         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
89         arr at: 3 put: 3.
90     )
91
92     usesIfFalseWithDot: arr = (
93         arr at: 1 put: 1.
94         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
95         arr at: 3 put: 3.
96     )
97
98     usesIfFalseWithoutDot: arr = (
99         arr at: 1 put: 1.
100        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
101        arr at: 3 put: 3.
102    )
103
104    testWriteArgument = (
105        self assert: 42 equals: (self dec: 43).
106    )
107
108    dec: anInt = (
109        anInt := anInt - 1.
110        ^ anInt
111    )
112 )

```



```

1 "
2
3 $Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2009-2013 see AUTHORS file
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7 http://www.hpi.uni-potsdam.de/swa/
8
9 Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 CompilerReturn4Test = TestCase (
29
30     return1 = ( ^self )
31     return2 = (      )
32
33     return3: arg = ( ^self )
34     return4: arg = (      )
35
36     testExplicitAndImplicitReturns = (
37         self assert: self is: self return1.
38         self assert: self is: self return2.
39         self assert: self is: (self return3: 23).
40         self assert: self is: (self return4: 23).
41     )
42
43
44     "In SOM++, code after the #ifTrue: does not seem to be executed, if
the
45     block expression ends with a dot."
46     testIfTrueWithDot = (
47         | arr |
48         arr := Array new: 3.
49         self usesIfTrueWithDot: arr.
50         self assertArrayCorrectness: arr.
51     )
52

```

```

53     assertArrayCorrectness: arr = (
54         self assert: 1 equals: (arr at: 1). "method was not executed"
55         self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
56         self assert: 3 equals: (arr at: 3). "remainder was not
executed"
57     )
58
59     testIfTrueWithoutDot = (
60         | arr |
61         arr := Array new: 3.
62         self usesIfTrueWithoutDot: arr.
63         self assertArrayCorrectness: arr.
64     )
65
66     testIfFalseWithDot = (
67         | arr |
68         arr := Array new: 3.
69         self usesIfFalseWithDot: arr.
70         self assertArrayCorrectness: arr.
71     )
72
73     testIfFalseWithoutDot = (
74         | arr |
75         arr := Array new: 3.
76         self usesIfFalseWithoutDot: arr.
77         self assertArrayCorrectness: arr.
78     )
79
80     usesIfTrueWithDot: arr = (
81         arr at: 1 put: 1.
82         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
83         arr at: 3 put: 3.
84     )
85
86     usesIfTrueWithoutDot: arr = (
87         arr at: 1 put: 1.
88         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
89         arr at: 3 put: 3.
90     )
91
92     usesIfFalseWithDot: arr = (
93         arr at: 1 put: 1.
94         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
95         arr at: 3 put: 3.
96     )
97
98     usesIfFalseWithoutDot: arr = (
99         arr at: 1 put: 1.
100        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
101        arr at: 3 put: 3.
102    )
103
104    testWriteArgument = (
105        self assert: 42 equals: (self dec: 43).
106    )
107
108    dec: anInt = (
109        anInt := anInt - 1.
110        ^ anInt
111    )
112 )

```



```

1  "
2
3  $Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5  Copyright (c) 2009-2013 see AUTHORS file
6  Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7  http://www.hpi.uni-potsdam.de/swa/
8
9  Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 CompilerReturn5Test = TestCase (
29
30     return1 = ( ^self )
31     return2 = (      )
32
33     return3: arg = ( ^self )
34     return4: arg = (      )
35
36     testExplicitAndImplicitReturns = (
37         self assert: self is: self return1.
38         self assert: self is: self return2.
39         self assert: self is: (self return3: 23).
40         self assert: self is: (self return4: 23).
41     )
42
43
44     "In SOM++, code after the #ifTrue: does not seem to be executed, if
the
45     block expression ends with a dot."
46     testIfTrueWithDot = (
47         | arr |
48         arr := Array new: 3.
49         self usesIfTrueWithDot: arr.
50         self assertArrayCorrectness: arr.
51     )
52

```

```

53     assertArrayCorrectness: arr = (
54         self assert: 1 equals: (arr at: 1). "method was not executed"
55         self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
56         self assert: 3 equals: (arr at: 3). "remainder was not
executed"
57     )
58
59     testIfTrueWithoutDot = (
60         | arr |
61         arr := Array new: 3.
62         self usesIfTrueWithoutDot: arr.
63         self assertArrayCorrectness: arr.
64     )
65
66     testIfFalseWithDot = (
67         | arr |
68         arr := Array new: 3.
69         self usesIfFalseWithDot: arr.
70         self assertArrayCorrectness: arr.
71     )
72
73     testIfFalseWithoutDot = (
74         | arr |
75         arr := Array new: 3.
76         self usesIfFalseWithoutDot: arr.
77         self assertArrayCorrectness: arr.
78     )
79
80     usesIfTrueWithDot: arr = (
81         arr at: 1 put: 1.
82         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
83         arr at: 3 put: 3.
84     )
85
86     usesIfTrueWithoutDot: arr = (
87         arr at: 1 put: 1.
88         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
89         arr at: 3 put: 3.
90     )
91
92     usesIfFalseWithDot: arr = (
93         arr at: 1 put: 1.
94         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
95         arr at: 3 put: 3.
96     )
97
98     usesIfFalseWithoutDot: arr = (
99         arr at: 1 put: 1.
100        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
101        arr at: 3 put: 3.
102    )
103
104    testWriteArgument = (
105        self assert: 42 equals: (self dec: 43).
106    )
107
108    dec: anInt = (
109        anInt := anInt - 1.
110        ^ anInt
111    )
112 )

```



```

1  "
2
3  $Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5  Copyright (c) 2009-2013 see AUTHORS file
6  Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7  http://www.hpi.uni-potsdam.de/swa/
8
9  Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 CompilerReturnTest = TestCase (
29
30     return1 = ( ^self )
31     return2 = (      )
32
33     return3: arg = ( ^self )
34     return4: arg = (      )
35
36     testExplicitAndImplicitReturns = (
37         self assert: self is: self return1.
38         self assert: self is: self return2.
39         self assert: self is: (self return3: 23).
40         self assert: self is: (self return4: 23).
41     )
42
43
44     "In SOM++, code after the #ifTrue: does not seem to be executed, if
the
45     block expression ends with a dot."
46     testIfTrueWithDot = (
47         | arr |
48         arr := Array new: 3.
49         self usesIfTrueWithDot: arr.
50         self assertArrayCorrectness: arr.
51     )
52

```

```

53     assertArrayCorrectness: arr = (
54         self assert: 1 equals: (arr at: 1). "method was not executed"
55         self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
56         self assert: 3 equals: (arr at: 3). "remainder was not
executed"
57     )
58
59     testIfTrueWithoutDot = (
60         | arr |
61         arr := Array new: 3.
62         self usesIfTrueWithoutDot: arr.
63         self assertArrayCorrectness: arr.
64     )
65
66     testIfFalseWithDot = (
67         | arr |
68         arr := Array new: 3.
69         self usesIfFalseWithDot: arr.
70         self assertArrayCorrectness: arr.
71     )
72
73     testIfFalseWithoutDot = (
74         | arr |
75         arr := Array new: 3.
76         self usesIfFalseWithoutDot: arr.
77         self assertArrayCorrectness: arr.
78     )
79
80     usesIfTrueWithDot: arr = (
81         arr at: 1 put: 1.
82         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
83         arr at: 3 put: 3.
84     )
85
86     usesIfTrueWithoutDot: arr = (
87         arr at: 1 put: 1.
88         (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
89         arr at: 3 put: 3.
90     )
91
92     usesIfFalseWithDot: arr = (
93         arr at: 1 put: 1.
94         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
95         arr at: 3 put: 3.
96     )
97
98     usesIfFalseWithoutDot: arr = (
99         arr at: 1 put: 1.
100        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
101        arr at: 3 put: 3.
102    )
103
104    testWriteArgument = (
105        self assert: 42 equals: (self dec: 43).
106    )
107
108    dec: anInt = (
109        anInt := anInt - 1.
110        ^ anInt
111    )
112 )

```



```

1 Dictionary2Test = TestCase (
2   testAtAndAtPut = (
3     | dict val |
4     dict := Dictionary new.
5     val := dict at: 1.
6
7     self assert: nil is: val.
8
9     val := dict at: 1 put: #foo.
10    self assert: dict is: val.
11    val := dict at: 1.
12    self assert: #foo is: val.
13
14    val := dict at: 1 put: #foo.
15    self assert: dict is: val.
16    val := dict at: 1.
17    self assert: #foo is: val.
18
19    val := dict at: 1 put: 42.
20    self assert: dict is: val.
21    val := dict at: 1.
22    self assert: 42 equals: val.
23  )
24
25  testContainsKey = (
26    | dict |
27    dict := Dictionary new.
28    self deny: (dict containsKey: 4).
29
30    dict at: 4 put: 34234.
31    self assert: (dict containsKey: 4).
32  )
33
34  testKeys = (
35    | dict keys |
36    dict := Dictionary new.
37
38    self assert: 0 equals: dict keys size.
39
40    dict at: 4 put: 423.
41    self assert: 1 equals: dict keys size.
42
43    dict at: 4 put: #gdfgd.
44    self assert: 1 equals: dict keys size.
45
46    dict at: 'as' put: Object new.
47    self assert: 2 equals: dict keys size.
48
49    keys := dict keys.
50    self assert: 4 equals: (keys at: 1).
51    self assert: 'as' equals: (keys at: 2).
52  )
53
54  testValues = (
55    | dict values v2 |
56    dict := Dictionary new.
57
58    self assert: 0 equals: dict values size.

```

```

59
60 dict at: 4 put: 423.
61 self assert: 1 equals: dict values size.
62
63 dict at: 4 put: #gdfgd.
64 self assert: 1 equals: dict values size.
65
66 dict at: 'as' put: #(1 2 3).
67 self assert: 2 equals: dict values size.
68
69 values := dict values.
70 self assert: #gdfgd is: (values at: 1).
71
72 v2 := values at: 2.
73 self assert: 1 equals: (v2 at: 1).
74 self assert: 2 equals: (v2 at: 2).
75 self assert: 3 equals: (v2 at: 3).
76 )
77
78 testDo = (
79 | dict expectedKs expectedVs i |
80 i := 1.
81 dict := Dictionary new.
82 dict at: #e put: 344.
83 dict at: 1 put: 545.
84 dict at: 0 put: 123.
85
86 expectedKs := #(#e 1 0).
87 expectedVs := #(344 545 123).
88
89 dict do: [:p |
90 self assert: (expectedKs at: i) equals: p key.
91 self assert: (expectedVs at: i) equals: p value.
92 i := i + 1.
93 ].
94
95 self assert: 4 equals: i.
96 )
97 )

```

```

1 Dictionary3Test = TestCase (
2   testAtAndAtPut = (
3     | dict val |
4     dict := Dictionary new.
5     val := dict at: 1.
6
7     self assert: nil is: val.
8
9     val := dict at: 1 put: #foo.
10    self assert: dict is: val.
11    val := dict at: 1.
12    self assert: #foo is: val.
13
14    val := dict at: 1 put: #foo.
15    self assert: dict is: val.
16    val := dict at: 1.
17    self assert: #foo is: val.
18
19    val := dict at: 1 put: 42.
20    self assert: dict is: val.
21    val := dict at: 1.
22    self assert: 42 equals: val.
23  )
24
25  testContainsKey = (
26    | dict |
27    dict := Dictionary new.
28    self deny: (dict containsKey: 4).
29
30    dict at: 4 put: 34234.
31    self assert: (dict containsKey: 4).
32  )
33
34  testKeys = (
35    | dict keys |
36    dict := Dictionary new.
37
38    self assert: 0 equals: dict keys size.
39
40    dict at: 4 put: 423.
41    self assert: 1 equals: dict keys size.
42
43    dict at: 4 put: #gdfgd.
44    self assert: 1 equals: dict keys size.
45
46    dict at: 'as' put: Object new.
47    self assert: 2 equals: dict keys size.
48
49    keys := dict keys.
50    self assert: 4 equals: (keys at: 1).
51    self assert: 'as' equals: (keys at: 2).
52  )
53
54  testValues = (
55    | dict values v2 |
56    dict := Dictionary new.
57
58    self assert: 0 equals: dict values size.

```

```

59
60 dict at: 4 put: 423.
61 self assert: 1 equals: dict values size.
62
63 dict at: 4 put: #gdfgd.
64 self assert: 1 equals: dict values size.
65
66 dict at: 'as' put: #(1 2 3).
67 self assert: 2 equals: dict values size.
68
69 values := dict values.
70 self assert: #gdfgd is: (values at: 1).
71
72 v2 := values at: 2.
73 self assert: 1 equals: (v2 at: 1).
74 self assert: 2 equals: (v2 at: 2).
75 self assert: 3 equals: (v2 at: 3).
76 )
77
78 testDo = (
79 | dict expectedKs expectedVs i |
80 i := 1.
81 dict := Dictionary new.
82 dict at: #e put: 344.
83 dict at: 1 put: 545.
84 dict at: 0 put: 123.
85
86 expectedKs := #(#e 1 0).
87 expectedVs := #(344 545 123).
88
89 dict do: [:p |
90 self assert: (expectedKs at: i) equals: p key.
91 self assert: (expectedVs at: i) equals: p value.
92 i := i + 1.
93 ].
94
95 self assert: 4 equals: i.
96 )
97 )

```

```

1 Dictionary4Test = TestCase (
2   testAtAndAtPut = (
3     | dict val |
4     dict := Dictionary new.
5     val := dict at: 1.
6
7     self assert: nil is: val.
8
9     val := dict at: 1 put: #foo.
10    self assert: dict is: val.
11    val := dict at: 1.
12    self assert: #foo is: val.
13
14    val := dict at: 1 put: #foo.
15    self assert: dict is: val.
16    val := dict at: 1.
17    self assert: #foo is: val.
18
19    val := dict at: 1 put: 42.
20    self assert: dict is: val.
21    val := dict at: 1.
22    self assert: 42 equals: val.
23  )
24
25  testContainsKey = (
26    | dict |
27    dict := Dictionary new.
28    self deny: (dict containsKey: 4).
29
30    dict at: 4 put: 34234.
31    self assert: (dict containsKey: 4).
32  )
33
34  testKeys = (
35    | dict keys |
36    dict := Dictionary new.
37
38    self assert: 0 equals: dict keys size.
39
40    dict at: 4 put: 423.
41    self assert: 1 equals: dict keys size.
42
43    dict at: 4 put: #gdfgd.
44    self assert: 1 equals: dict keys size.
45
46    dict at: 'as' put: Object new.
47    self assert: 2 equals: dict keys size.
48
49    keys := dict keys.
50    self assert: 4 equals: (keys at: 1).
51    self assert: 'as' equals: (keys at: 2).
52  )
53
54  testValues = (
55    | dict values v2 |
56    dict := Dictionary new.
57
58    self assert: 0 equals: dict values size.

```



```

59
60 dict at: 4 put: 423.
61 self assert: 1 equals: dict values size.
62
63 dict at: 4 put: #gdfgd.
64 self assert: 1 equals: dict values size.
65
66 dict at: 'as' put: #(1 2 3).
67 self assert: 2 equals: dict values size.
68
69 values := dict values.
70 self assert: #gdfgd is: (values at: 1).
71
72 v2 := values at: 2.
73 self assert: 1 equals: (v2 at: 1).
74 self assert: 2 equals: (v2 at: 2).
75 self assert: 3 equals: (v2 at: 3).
76 )
77
78 testDo = (
79 | dict expectedKs expectedVs i |
80 i := 1.
81 dict := Dictionary new.
82 dict at: #e put: 344.
83 dict at: 1 put: 545.
84 dict at: 0 put: 123.
85
86 expectedKs := #(#e 1 0).
87 expectedVs := #(344 545 123).
88
89 dict do: [:p |
90 self assert: (expectedKs at: i) equals: p key.
91 self assert: (expectedVs at: i) equals: p value.
92 i := i + 1.
93 ].
94
95 self assert: 4 equals: i.
96 )
97 )

```

```

1 Dictionary5Test = TestCase (
2   testAtAndAtPut = (
3     | dict val |
4     dict := Dictionary new.
5     val := dict at: 1.
6
7     self assert: nil is: val.
8
9     val := dict at: 1 put: #foo.
10    self assert: dict is: val.
11    val := dict at: 1.
12    self assert: #foo is: val.
13
14    val := dict at: 1 put: #foo.
15    self assert: dict is: val.
16    val := dict at: 1.
17    self assert: #foo is: val.
18
19    val := dict at: 1 put: 42.
20    self assert: dict is: val.
21    val := dict at: 1.
22    self assert: 42 equals: val.
23  )
24
25  testContainsKey = (
26    | dict |
27    dict := Dictionary new.
28    self deny: (dict containsKey: 4).
29
30    dict at: 4 put: 34234.
31    self assert: (dict containsKey: 4).
32  )
33
34  testKeys = (
35    | dict keys |
36    dict := Dictionary new.
37
38    self assert: 0 equals: dict keys size.
39
40    dict at: 4 put: 423.
41    self assert: 1 equals: dict keys size.
42
43    dict at: 4 put: #gdfgd.
44    self assert: 1 equals: dict keys size.
45
46    dict at: 'as' put: Object new.
47    self assert: 2 equals: dict keys size.
48
49    keys := dict keys.
50    self assert: 4 equals: (keys at: 1).
51    self assert: 'as' equals: (keys at: 2).
52  )
53
54  testValues = (
55    | dict values v2 |
56    dict := Dictionary new.
57
58    self assert: 0 equals: dict values size.

```

```

59
60     dict at: 4 put: 423.
61     self assert: 1 equals: dict values size.
62
63     dict at: 4 put: #gdfgd.
64     self assert: 1 equals: dict values size.
65
66     dict at: 'as' put: #(1 2 3).
67     self assert: 2 equals: dict values size.
68
69     values := dict values.
70     self assert: #gdfgd is: (values at: 1).
71
72     v2 := values at: 2.
73     self assert: 1 equals: (v2 at: 1).
74     self assert: 2 equals: (v2 at: 2).
75     self assert: 3 equals: (v2 at: 3).
76 )
77
78 testDo = (
79     | dict expectedKs expectedVs i |
80     i := 1.
81     dict := Dictionary new.
82     dict at: #e put: 344.
83     dict at: 1 put: 545.
84     dict at: 0 put: 123.
85
86     expectedKs := #(#e 1 0).
87     expectedVs := #(344 545 123).
88
89     dict do: [:p |
90         self assert: (expectedKs at: i) equals: p key.
91         self assert: (expectedVs at: i) equals: p value.
92         i := i + 1.
93     ].
94
95     self assert: 4 equals: i.
96 )
97 )

```

```

1 DictionaryTest = TestCase (
2   testAtAndAtPut = (
3     | dict val |
4     dict := Dictionary new.
5     val := dict at: 1.
6
7     self assert: nil is: val.
8
9     val := dict at: 1 put: #foo.
10    self assert: dict is: val.
11    val := dict at: 1.
12    self assert: #foo is: val.
13
14    val := dict at: 1 put: #foo.
15    self assert: dict is: val.
16    val := dict at: 1.
17    self assert: #foo is: val.
18
19    val := dict at: 1 put: 42.
20    self assert: dict is: val.
21    val := dict at: 1.
22    self assert: 42 equals: val.
23  )
24
25  testContainsKey = (
26    | dict |
27    dict := Dictionary new.
28    self deny: (dict containsKey: 4).
29
30    dict at: 4 put: 34234.
31    self assert: (dict containsKey: 4).
32  )
33
34  testKeys = (
35    | dict keys |
36    dict := Dictionary new.
37
38    self assert: 0 equals: dict keys size.
39
40    dict at: 4 put: 423.
41    self assert: 1 equals: dict keys size.
42
43    dict at: 4 put: #gdfgd.
44    self assert: 1 equals: dict keys size.
45
46    dict at: 'as' put: Object new.
47    self assert: 2 equals: dict keys size.
48
49    keys := dict keys.
50    self assert: 4 equals: (keys at: 1).
51    self assert: 'as' equals: (keys at: 2).
52  )
53
54  testValues = (
55    | dict values v2 |
56    dict := Dictionary new.
57
58    self assert: 0 equals: dict values size.

```

```

59
60 dict at: 4 put: 423.
61 self assert: 1 equals: dict values size.
62
63 dict at: 4 put: #gdfgd.
64 self assert: 1 equals: dict values size.
65
66 dict at: 'as' put: #(1 2 3).
67 self assert: 2 equals: dict values size.
68
69 values := dict values.
70 self assert: #gdfgd is: (values at: 1).
71
72 v2 := values at: 2.
73 self assert: 1 equals: (v2 at: 1).
74 self assert: 2 equals: (v2 at: 2).
75 self assert: 3 equals: (v2 at: 3).
76 )
77
78 testDo = (
79 | dict expectedKs expectedVs i |
80 i := 1.
81 dict := Dictionary new.
82 dict at: #e put: 344.
83 dict at: 1 put: 545.
84 dict at: 0 put: 123.
85
86 expectedKs := #(#e 1 0).
87 expectedVs := #(344 545 123).
88
89 dict do: [:p |
90 self assert: (expectedKs at: i) equals: p key.
91 self assert: (expectedVs at: i) equals: p value.
92 i := i + 1.
93 ].
94
95 self assert: 4 equals: i.
96 )
97 )

```

```

1 DoesNotUnderstand2Test = TestCase (
2
3   testSimpleUnknownFoo = (
4     | result |
5     result := self foo.
6     self assert: DoesNotUnderstandMessage is: result class.
7     self assert: self is: result target.
8     self assert: #foo is: result selector.
9   )
10
11  testArguments = (
12    | result |
13    result := self foo.
14    self assert: Array is: result arguments class.
15    self assert: 0 equals: result arguments length.
16
17    result := self foo: 1.
18    self assert: 1 equals: result arguments length.
19    self assert: 1 equals: (result arguments at: 1).
20
21    result := self foo: 1 bar: 2 baz: 3.
22    self assert: 3 equals: result arguments length.
23    self assert: 1 equals: (result arguments at: 1).
24    self assert: 2 equals: (result arguments at: 2).
25    self assert: 3 equals: (result arguments at: 3).
26  )
27
28  testRepeat = (
29    | result |
30    result := Array new: 5.
31    1 to: result length do: [:i |
32      result at: i put: self foo.
33
34      i > 1 ifTrue: [
35        self assert: (result at: i - 1) ~= (result at: i).
36      ]
37    ].
38  )
39
40  doesNotUnderstand: selector arguments: arguments = (
41    ^ DoesNotUnderstandMessage to: self selector: selector arguments:
arguments.
42  )
43 )
44

```

```

1 DoesNotUnderstand3Test = TestCase (
2
3   testSimpleUnknownFoo = (
4     | result |
5     result := self foo.
6     self assert: DoesNotUnderstandMessage is: result class.
7     self assert: self is: result target.
8     self assert: #foo is: result selector.
9   )
10
11  testArguments = (
12    | result |
13    result := self foo.
14    self assert: Array is: result arguments class.
15    self assert: 0 equals: result arguments length.
16
17    result := self foo: 1.
18    self assert: 1 equals: result arguments length.
19    self assert: 1 equals: (result arguments at: 1).
20
21    result := self foo: 1 bar: 2 baz: 3.
22    self assert: 3 equals: result arguments length.
23    self assert: 1 equals: (result arguments at: 1).
24    self assert: 2 equals: (result arguments at: 2).
25    self assert: 3 equals: (result arguments at: 3).
26  )
27
28  testRepeat = (
29    | result |
30    result := Array new: 5.
31    1 to: result length do: [:i |
32      result at: i put: self foo.
33
34      i > 1 ifTrue: [
35        self assert: (result at: i - 1) ~= (result at: i).
36      ]
37    ].
38  )
39
40  doesNotUnderstand: selector arguments: arguments = (
41    ^ DoesNotUnderstandMessage to: self selector: selector arguments:
arguments.
42  )
43 )
44

```

```

1 DoesNotUnderstand4Test = TestCase (
2
3   testSimpleUnknownFoo = (
4     | result |
5     result := self foo.
6     self assert: DoesNotUnderstandMessage is: result class.
7     self assert: self is: result target.
8     self assert: #foo is: result selector.
9   )
10
11  testArguments = (
12    | result |
13    result := self foo.
14    self assert: Array is: result arguments class.
15    self assert: 0 equals: result arguments length.
16
17    result := self foo: 1.
18    self assert: 1 equals: result arguments length.
19    self assert: 1 equals: (result arguments at: 1).
20
21    result := self foo: 1 bar: 2 baz: 3.
22    self assert: 3 equals: result arguments length.
23    self assert: 1 equals: (result arguments at: 1).
24    self assert: 2 equals: (result arguments at: 2).
25    self assert: 3 equals: (result arguments at: 3).
26  )
27
28  testRepeat = (
29    | result |
30    result := Array new: 5.
31    1 to: result length do: [:i |
32      result at: i put: self foo.
33
34      i > 1 ifTrue: [
35        self assert: (result at: i - 1) ~= (result at: i).
36      ]
37    ].
38  )
39
40  doesNotUnderstand: selector arguments: arguments = (
41    ^ DoesNotUnderstandMessage to: self selector: selector arguments:
arguments.
42  )
43 )
44

```



```

1 DoesNotUnderstand5Test = TestCase (
2
3   testSimpleUnknownFoo = (
4     | result |
5     result := self foo.
6     self assert: DoesNotUnderstandMessage is: result class.
7     self assert: self is: result target.
8     self assert: #foo is: result selector.
9   )
10
11  testArguments = (
12    | result |
13    result := self foo.
14    self assert: Array is: result arguments class.
15    self assert: 0 equals: result arguments length.
16
17    result := self foo: 1.
18    self assert: 1 equals: result arguments length.
19    self assert: 1 equals: (result arguments at: 1).
20
21    result := self foo: 1 bar: 2 baz: 3.
22    self assert: 3 equals: result arguments length.
23    self assert: 1 equals: (result arguments at: 1).
24    self assert: 2 equals: (result arguments at: 2).
25    self assert: 3 equals: (result arguments at: 3).
26  )
27
28  testRepeat = (
29    | result |
30    result := Array new: 5.
31    1 to: result length do: [:i |
32      result at: i put: self foo.
33
34      i > 1 ifTrue: [
35        self assert: (result at: i - 1) ~= (result at: i).
36      ]
37    ].
38  )
39
40  doesNotUnderstand: selector arguments: arguments = (
41    ^ DoesNotUnderstandMessage to: self selector: selector arguments:
arguments.
42  )
43 )
44

```

```
1 DoesNotUnderstandMessage = (  
2   | target selector arguments |  
3  
4   initializeWith: targetObj selector: aSelector arguments: argArray = (  
5     target      := targetObj.  
6     selector    := aSelector.  
7     arguments   := argArray.  
8   )  
9  
10  target      = ( ^ target )  
11  selector    = ( ^ selector )  
12  arguments   = ( ^ arguments )  
13  
14  ----  
15  
16  to: target selector: selector arguments: args = (  
17    | m |  
18    m := self new.  
19    m initializeWith: target selector: selector arguments: args.  
20    ^ m  
21  )  
22 )
```

```

1 DoesNotUnderstandTest = TestCase (
2
3   testSimpleUnknownFoo = (
4     | result |
5     result := self foo.
6     self assert: DoesNotUnderstandMessage is: result class.
7     self assert: self is: result target.
8     self assert: #foo is: result selector.
9   )
10
11  testArguments = (
12    | result |
13    result := self foo.
14    self assert: Array is: result arguments class.
15    self assert: 0 equals: result arguments length.
16
17    result := self foo: 1.
18    self assert: 1 equals: result arguments length.
19    self assert: 1 equals: (result arguments at: 1).
20
21    result := self foo: 1 bar: 2 baz: 3.
22    self assert: 3 equals: result arguments length.
23    self assert: 1 equals: (result arguments at: 1).
24    self assert: 2 equals: (result arguments at: 2).
25    self assert: 3 equals: (result arguments at: 3).
26  )
27
28  testRepeat = (
29    | result |
30    result := Array new: 5.
31    1 to: result length do: [:i |
32      result at: i put: self foo.
33
34      i > 1 ifTrue: [
35        self assert: (result at: i - 1) ~= (result at: i).
36      ]
37    ].
38  )
39
40  doesNotUnderstand: selector arguments: arguments = (
41    ^ DoesNotUnderstandMessage to: self selector: selector arguments:
arguments.
42  )
43 )
44

```

```
1 "  
2  
3 $Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Double2Test = TestCase (  
27  
28     testAdd = (  
29         self assert: 1.0 equals: 0.5 + 0.5.  
30         self assert: 0.0 equals: 0.5 + -0.5.  
31  
32         self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.  
33         self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.  
34     )  
35  
36     testSubtract = (  
37         self assert: 0.0 equals: 0.5 - 0.5.  
38         self assert: 1.0 equals: 0.5 - -0.5.  
39  
40         self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.  
41         self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.  
42     )  
43  
44     testMultiply = (  
45         self assert: 4.0 equals: 2.0 * 2.0.  
46         self assert: -4.0 equals: 2.0 * -2.0.  
47  
48         self assert: 1.0 equals: 4.0 * 0.25.  
49         self assert: -1.0 equals: -4.0 * 0.25.  
50     )  
51  
52     testIntegerDivision = (  
53         self assert: 1 equals: (4/3) + (4/5)
```

```

54 )
55
56 testDoubleDivision = (
57     self assert: 32 // 15 equals: (4//3) + (4//5)
58 )
59
60 testModulo = (
61     self assert: 1.0 equals: 3.0 % 2.0.
62     self assert: 0.0 equals: 3.0 % 3.0.
63
64     self assert: -1.0 equals: -3.0 % 2.0.
65     self assert: -1.0 equals: -3.0 % -2.0.
66     self assert: 1.0 equals: 3.0 % -2.0.
67
68     self assert: 0.0 equals: 3.0 % 3.0.
69     self assert: 0.0 equals: -3.0 % -3.0.
70     self assert: 0.0 equals: 3.0 % -3.0.
71 )
72
73 testAbs = (
74     self assert: 1.0 equals: 1.0 abs.
75     self assert: 1.0 equals: -1.0 abs.
76
77     self assert: 9007199254740992.0 equals: 9007199254740992.0 abs.
78     self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.
79
80     self assert: 19007199254740992.0 equals: 19007199254740992.0 abs.
81     self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
82 )
83
84 testSqrt = (
85     self assert: 3.0 equals: 9.0 sqrt.
86     self assert: 16.0 equals: 256.0 sqrt.
87
88     self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
89 )
90
91 testNegated = (
92     self assert: 0.0 equals: 0.0 negated.
93     self assert: -1.0 equals: 1.0 negated.
94     self assert: 1.0 equals: -1.0 negated.
95
96     self assert: -9007199254740992.0 equals: 9007199254740992.0 negated.
97     self assert: 9007199254740992.0 equals: -9007199254740992.0 negated.
98
99     self assert: -19007199254740992.0 equals: 19007199254740992.0
negated.
100     self assert: 19007199254740992.0 equals: -19007199254740992.0
negated.
101 )
102
103 testAsString = (
104     self assert: '0.5' equals: (1//2) asString.
105     self assert: '0.5' equals: 0.5 asString.
106 )
107
108 testEquals = (
109     self assert: (1.0 = 1).
110 )
111
112 testRound = (

```

```

113     self assert: 1 equals: 1.0 round.
114     self assert: 1 equals: 1.4 round.
115     self assert: 1 equals: 1.4999 round.
116     self assert: 2 equals: 1.5 round.
117     self assert: 2 equals: 1.5000001 round.
118     self assert: 1 equals: (5//10) round.
119     self assert: 1 equals: (14//10) round.
120     self assert: 445 equals: (44534//100) round.
121 )
122
123 testAsInteger = (
124     self assert: 1 equals: 1.0 asInteger.
125     self assert: 1 equals: 1.1 asInteger.
126     self assert: 1 equals: 1.999 asInteger.
127
128     self assert: -1 equals: -1.0 asInteger.
129     self assert: -1 equals: -1.999 asInteger.
130 )
131
132 testSin = (
133     | pi |
134     pi := 3.141592653589.
135     self assert: 0.0 equals: 0.0 sin.
136     self assert: pi sin abs < 0.000000000001.
137     self assert: (pi // 2.0) sin > 0.9999999999.
138 )
139
140 testCos = (
141     | pi |
142     pi := 3.141592653589.
143     self assert: 1.0 equals: 0.0 cos.
144     self assert: (pi // 2.0) cos abs < 0.000000000001.
145     self assert: pi cos < -0.9999999999.
146 )
147
148 testInfinity = (
149     self assert: Double PositiveInfinity > 1.
150     self assert: Double PositiveInfinity equals: Double PositiveInfinity
151 + 1.
151     self assert: Double PositiveInfinity equals: Double PositiveInfinity
152 - 1.
152
153     self assert: Double PositiveInfinity > (999999 * 999999 * 999999 *
154 999999).
154 )
155
156 testFromString = (
157     self assert: 0.0 equals: (Double fromString: '0.0').
158     self assert: -1.1 equals: (Double fromString: '-1.1').
159
160     self assert: 3423.54656 equals: (Double fromString: '3423.54656').
161     self assert: -672.433244 equals: (Double fromString: '-672.433244').
162 )
163
164 testEqual = (
165     self assert: 0.0 = 0.0.
166     self assert: 1.0 = 1.0.
167     self assert: 0.0 = -0.0.
168     self assert: -0.0 = 0.0.
169 )
170

```

```

171 testLessThan = (
172     self assert: 0.0 < 1.0.
173     self assert: 0.499999999 < 0.5.
174     self deny: 1.0 < 0.0.
175     self deny: 0.5 < 0.499999999.
176 )
177
178 testGreaterThan = (
179     self deny: 0.0 > 1.0.
180     self deny: 0.499999999 > 0.5.
181     self assert: 1.0 > 0.0.
182     self assert: 0.5 > 0.499999999.
183 )
184
185 testLessThanOrEqual = (
186     self assert: 0.0 <= 1.0.
187     self assert: 0.499999999 <= 0.5.
188     self assert: 0.5 <= 0.5.
189     self deny: 1.0 < 0.0.
190     self deny: 0.5 < 0.499999999.
191 )
192
193 testGreaterThanOrEqual = (
194     self deny: 0.0 >= 1.0.
195     self deny: 0.499999999 >= 0.5.
196     self assert: 0.5 >= 0.5.
197     self assert: 1.0 >= 0.0.
198     self assert: 0.5 >= 0.499999999.
199 )
200
201 testNegative = (
202     self assert: -0.000000001 negative.
203     self assert: -1.0 negative.
204     self assert: -123123.000000001 negative.
205     self deny: 0.000000001 negative.
206     self deny: 1.0 negative.
207     self deny: 123123.000000001 negative.
208 )
209
210 testBetween = (
211     self assert: (1.0 between: 0.0 and: 2.0).
212     self assert: (0.000001 between: 0.0 and: 2.0).
213     self assert: (1.999999 between: 0.0 and: 2.0).
214     self deny: (0.0 between: 0.0 and: 2.0).
215     self deny: (2.0 between: 0.0 and: 2.0).
216 )
217
218 testToDo = (
219     | d |
220     d := 0.0.
221     0.0 to: 10.0 do: [:ii |
222         d := d + ii.
223     ].
224
225     self assert: 55.0 equals: d.
226
227     d := 0.0.
228     0.0 to: 10.1 do: [:ii |
229         d := d + ii.
230     ].
231

```

```

232     self assert: 55.0 equals: d.
233
234     d := 0.0.
235     0.1 to: 10.1 do: [:ii |
236         d := d + ii.
237     ].
238
239     self assert: 55.0 + 1.1 equals: d.
240 )
241
242 testDownToDo = (
243     | d |
244     d := 0.0.
245     10.0 downTo: 0.0 do: [:ii |
246         d := d + ii.
247     ].
248
249     self assert: 55.0 equals: d.
250
251     d := 0.0.
252     10.1 downTo: 0.0 do: [:ii |
253         d := d + ii.
254     ].
255
256     self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
257 )
258 )
259

```



```
1 "  
2  
3 $Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Double3Test = TestCase (  
27  
28     testAdd = (  
29         self assert: 1.0 equals: 0.5 + 0.5.  
30         self assert: 0.0 equals: 0.5 + -0.5.  
31  
32         self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.  
33         self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.  
34     )  
35  
36     testSubtract = (  
37         self assert: 0.0 equals: 0.5 - 0.5.  
38         self assert: 1.0 equals: 0.5 - -0.5.  
39  
40         self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.  
41         self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.  
42     )  
43  
44     testMultiply = (  
45         self assert: 4.0 equals: 2.0 * 2.0.  
46         self assert: -4.0 equals: 2.0 * -2.0.  
47  
48         self assert: 1.0 equals: 4.0 * 0.25.  
49         self assert: -1.0 equals: -4.0 * 0.25.  
50     )  
51  
52     testIntegerDivision = (  
53         self assert: 1 equals: (4/3) + (4/5)
```

```

54 )
55
56 testDoubleDivision = (
57     self assert: 32 // 15 equals: (4//3) + (4//5)
58 )
59
60 testModulo = (
61     self assert: 1.0 equals: 3.0 % 2.0.
62     self assert: 0.0 equals: 3.0 % 3.0.
63
64     self assert: -1.0 equals: -3.0 % 2.0.
65     self assert: -1.0 equals: -3.0 % -2.0.
66     self assert: 1.0 equals: 3.0 % -2.0.
67
68     self assert: 0.0 equals: 3.0 % 3.0.
69     self assert: 0.0 equals: -3.0 % -3.0.
70     self assert: 0.0 equals: 3.0 % -3.0.
71 )
72
73 testAbs = (
74     self assert: 1.0 equals: 1.0 abs.
75     self assert: 1.0 equals: -1.0 abs.
76
77     self assert: 9007199254740992.0 equals: 9007199254740992.0 abs.
78     self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.
79
80     self assert: 19007199254740992.0 equals: 19007199254740992.0 abs.
81     self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
82 )
83
84 testSqrt = (
85     self assert: 3.0 equals: 9.0 sqrt.
86     self assert: 16.0 equals: 256.0 sqrt.
87
88     self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
89 )
90
91 testNegated = (
92     self assert: 0.0 equals: 0.0 negated.
93     self assert: -1.0 equals: 1.0 negated.
94     self assert: 1.0 equals: -1.0 negated.
95
96     self assert: -9007199254740992.0 equals: 9007199254740992.0 negated.
97     self assert: 9007199254740992.0 equals: -9007199254740992.0 negated.
98
99     self assert: -19007199254740992.0 equals: 19007199254740992.0
negated.
100     self assert: 19007199254740992.0 equals: -19007199254740992.0
negated.
101 )
102
103 testAsString = (
104     self assert: '0.5' equals: (1//2) asString.
105     self assert: '0.5' equals: 0.5 asString.
106 )
107
108 testEquals = (
109     self assert: (1.0 = 1).
110 )
111
112 testRound = (

```

```

113     self assert: 1 equals: 1.0 round.
114     self assert: 1 equals: 1.4 round.
115     self assert: 1 equals: 1.4999 round.
116     self assert: 2 equals: 1.5 round.
117     self assert: 2 equals: 1.5000001 round.
118     self assert: 1 equals: (5//10) round.
119     self assert: 1 equals: (14//10) round.
120     self assert: 445 equals: (44534//100) round.
121 )
122
123 testAsInteger = (
124     self assert: 1 equals: 1.0 asInteger.
125     self assert: 1 equals: 1.1 asInteger.
126     self assert: 1 equals: 1.999 asInteger.
127
128     self assert: -1 equals: -1.0 asInteger.
129     self assert: -1 equals: -1.999 asInteger.
130 )
131
132 testSin = (
133     | pi |
134     pi := 3.141592653589.
135     self assert: 0.0 equals: 0.0 sin.
136     self assert: pi sin abs < 0.000000000001.
137     self assert: (pi // 2.0) sin > 0.9999999999.
138 )
139
140 testCos = (
141     | pi |
142     pi := 3.141592653589.
143     self assert: 1.0 equals: 0.0 cos.
144     self assert: (pi // 2.0) cos abs < 0.000000000001.
145     self assert: pi cos < -0.9999999999.
146 )
147
148 testInfinity = (
149     self assert: Double PositiveInfinity > 1.
150     self assert: Double PositiveInfinity equals: Double PositiveInfinity
151 + 1.
151     self assert: Double PositiveInfinity equals: Double PositiveInfinity
152 - 1.
152
153     self assert: Double PositiveInfinity > (999999 * 999999 * 999999 *
154 999999).
154 )
155
156 testFromString = (
157     self assert: 0.0 equals: (Double fromString: '0.0').
158     self assert: -1.1 equals: (Double fromString: '-1.1').
159
160     self assert: 3423.54656 equals: (Double fromString: '3423.54656').
161     self assert: -672.433244 equals: (Double fromString: '-672.433244').
162 )
163
164 testEqual = (
165     self assert: 0.0 = 0.0.
166     self assert: 1.0 = 1.0.
167     self assert: 0.0 = -0.0.
168     self assert: -0.0 = 0.0.
169 )
170

```

```

171 testLessThan = (
172     self assert: 0.0 < 1.0.
173     self assert: 0.499999999 < 0.5.
174     self deny: 1.0 < 0.0.
175     self deny: 0.5 < 0.499999999.
176 )
177
178 testGreaterThan = (
179     self deny: 0.0 > 1.0.
180     self deny: 0.499999999 > 0.5.
181     self assert: 1.0 > 0.0.
182     self assert: 0.5 > 0.499999999.
183 )
184
185 testLessThanOrEqual = (
186     self assert: 0.0 <= 1.0.
187     self assert: 0.499999999 <= 0.5.
188     self assert: 0.5 <= 0.5.
189     self deny: 1.0 < 0.0.
190     self deny: 0.5 < 0.499999999.
191 )
192
193 testGreaterThanOrEqual = (
194     self deny: 0.0 >= 1.0.
195     self deny: 0.499999999 >= 0.5.
196     self assert: 0.5 >= 0.5.
197     self assert: 1.0 >= 0.0.
198     self assert: 0.5 >= 0.499999999.
199 )
200
201 testNegative = (
202     self assert: -0.000000001 negative.
203     self assert: -1.0 negative.
204     self assert: -123123.000000001 negative.
205     self deny: 0.000000001 negative.
206     self deny: 1.0 negative.
207     self deny: 123123.000000001 negative.
208 )
209
210 testBetween = (
211     self assert: (1.0 between: 0.0 and: 2.0).
212     self assert: (0.000001 between: 0.0 and: 2.0).
213     self assert: (1.999999 between: 0.0 and: 2.0).
214     self deny: (0.0 between: 0.0 and: 2.0).
215     self deny: (2.0 between: 0.0 and: 2.0).
216 )
217
218 testToDo = (
219     | d |
220     d := 0.0.
221     0.0 to: 10.0 do: [:ii |
222         d := d + ii.
223     ].
224
225     self assert: 55.0 equals: d.
226
227     d := 0.0.
228     0.0 to: 10.1 do: [:ii |
229         d := d + ii.
230     ].
231

```

```

232     self assert: 55.0 equals: d.
233
234     d := 0.0.
235     0.1 to: 10.1 do: [:ii |
236         d := d + ii.
237     ].
238
239     self assert: 55.0 + 1.1 equals: d.
240 )
241
242 testDownToDo = (
243     | d |
244     d := 0.0.
245     10.0 downTo: 0.0 do: [:ii |
246         d := d + ii.
247     ].
248
249     self assert: 55.0 equals: d.
250
251     d := 0.0.
252     10.1 downTo: 0.0 do: [:ii |
253         d := d + ii.
254     ].
255
256     self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
257 )
258 )
259

```

```
1 "  
2  
3 $Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Double4Test = TestCase (  
27  
28     testAdd = (  
29         self assert: 1.0 equals: 0.5 + 0.5.  
30         self assert: 0.0 equals: 0.5 + -0.5.  
31  
32         self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.  
33         self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.  
34     )  
35  
36     testSubtract = (  
37         self assert: 0.0 equals: 0.5 - 0.5.  
38         self assert: 1.0 equals: 0.5 - -0.5.  
39  
40         self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.  
41         self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.  
42     )  
43  
44     testMultiply = (  
45         self assert: 4.0 equals: 2.0 * 2.0.  
46         self assert: -4.0 equals: 2.0 * -2.0.  
47  
48         self assert: 1.0 equals: 4.0 * 0.25.  
49         self assert: -1.0 equals: -4.0 * 0.25.  
50     )  
51  
52     testIntegerDivision = (  
53         self assert: 1 equals: (4/3) + (4/5)
```

```

54 )
55
56 testDoubleDivision = (
57     self assert: 32 // 15 equals: (4//3) + (4//5)
58 )
59
60 testModulo = (
61     self assert: 1.0 equals: 3.0 % 2.0.
62     self assert: 0.0 equals: 3.0 % 3.0.
63
64     self assert: -1.0 equals: -3.0 % 2.0.
65     self assert: -1.0 equals: -3.0 % -2.0.
66     self assert: 1.0 equals: 3.0 % -2.0.
67
68     self assert: 0.0 equals: 3.0 % 3.0.
69     self assert: 0.0 equals: -3.0 % -3.0.
70     self assert: 0.0 equals: 3.0 % -3.0.
71 )
72
73 testAbs = (
74     self assert: 1.0 equals: 1.0 abs.
75     self assert: 1.0 equals: -1.0 abs.
76
77     self assert: 9007199254740992.0 equals: 9007199254740992.0 abs.
78     self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.
79
80     self assert: 19007199254740992.0 equals: 19007199254740992.0 abs.
81     self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
82 )
83
84 testSqrt = (
85     self assert: 3.0 equals: 9.0 sqrt.
86     self assert: 16.0 equals: 256.0 sqrt.
87
88     self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
89 )
90
91 testNegated = (
92     self assert: 0.0 equals: 0.0 negated.
93     self assert: -1.0 equals: 1.0 negated.
94     self assert: 1.0 equals: -1.0 negated.
95
96     self assert: -9007199254740992.0 equals: 9007199254740992.0 negated.
97     self assert: 9007199254740992.0 equals: -9007199254740992.0 negated.
98
99     self assert: -19007199254740992.0 equals: 19007199254740992.0
negated.
100     self assert: 19007199254740992.0 equals: -19007199254740992.0
negated.
101 )
102
103 testAsString = (
104     self assert: '0.5' equals: (1//2) asString.
105     self assert: '0.5' equals: 0.5 asString.
106 )
107
108 testEquals = (
109     self assert: (1.0 = 1).
110 )
111
112 testRound = (

```

```

113     self assert: 1 equals: 1.0 round.
114     self assert: 1 equals: 1.4 round.
115     self assert: 1 equals: 1.4999 round.
116     self assert: 2 equals: 1.5 round.
117     self assert: 2 equals: 1.5000001 round.
118     self assert: 1 equals: (5//10) round.
119     self assert: 1 equals: (14//10) round.
120     self assert: 445 equals: (44534//100) round.
121 )
122
123 testAsInteger = (
124     self assert: 1 equals: 1.0 asInteger.
125     self assert: 1 equals: 1.1 asInteger.
126     self assert: 1 equals: 1.999 asInteger.
127
128     self assert: -1 equals: -1.0 asInteger.
129     self assert: -1 equals: -1.999 asInteger.
130 )
131
132 testSin = (
133     | pi |
134     pi := 3.141592653589.
135     self assert: 0.0 equals: 0.0 sin.
136     self assert: pi sin abs < 0.000000000001.
137     self assert: (pi // 2.0) sin > 0.9999999999.
138 )
139
140 testCos = (
141     | pi |
142     pi := 3.141592653589.
143     self assert: 1.0 equals: 0.0 cos.
144     self assert: (pi // 2.0) cos abs < 0.000000000001.
145     self assert: pi cos < -0.9999999999.
146 )
147
148 testInfinity = (
149     self assert: Double PositiveInfinity > 1.
150     self assert: Double PositiveInfinity equals: Double PositiveInfinity
151 + 1.
151     self assert: Double PositiveInfinity equals: Double PositiveInfinity
152 - 1.
152
153     self assert: Double PositiveInfinity > (999999 * 999999 * 999999 *
154 999999).
154 )
155
156 testFromString = (
157     self assert: 0.0 equals: (Double fromString: '0.0').
158     self assert: -1.1 equals: (Double fromString: '-1.1').
159
160     self assert: 3423.54656 equals: (Double fromString: '3423.54656').
161     self assert: -672.433244 equals: (Double fromString: '-672.433244').
162 )
163
164 testEqual = (
165     self assert: 0.0 = 0.0.
166     self assert: 1.0 = 1.0.
167     self assert: 0.0 = -0.0.
168     self assert: -0.0 = 0.0.
169 )
170

```



```

171 testLessThan = (
172     self assert: 0.0 < 1.0.
173     self assert: 0.499999999 < 0.5.
174     self deny: 1.0 < 0.0.
175     self deny: 0.5 < 0.499999999.
176 )
177
178 testGreaterThan = (
179     self deny: 0.0 > 1.0.
180     self deny: 0.499999999 > 0.5.
181     self assert: 1.0 > 0.0.
182     self assert: 0.5 > 0.499999999.
183 )
184
185 testLessThanOrEqual = (
186     self assert: 0.0 <= 1.0.
187     self assert: 0.499999999 <= 0.5.
188     self assert: 0.5 <= 0.5.
189     self deny: 1.0 < 0.0.
190     self deny: 0.5 < 0.499999999.
191 )
192
193 testGreaterThanOrEqual = (
194     self deny: 0.0 >= 1.0.
195     self deny: 0.499999999 >= 0.5.
196     self assert: 0.5 >= 0.5.
197     self assert: 1.0 >= 0.0.
198     self assert: 0.5 >= 0.499999999.
199 )
200
201 testNegative = (
202     self assert: -0.000000001 negative.
203     self assert: -1.0 negative.
204     self assert: -123123.000000001 negative.
205     self deny: 0.000000001 negative.
206     self deny: 1.0 negative.
207     self deny: 123123.000000001 negative.
208 )
209
210 testBetween = (
211     self assert: (1.0 between: 0.0 and: 2.0).
212     self assert: (0.000001 between: 0.0 and: 2.0).
213     self assert: (1.999999 between: 0.0 and: 2.0).
214     self deny: (0.0 between: 0.0 and: 2.0).
215     self deny: (2.0 between: 0.0 and: 2.0).
216 )
217
218 testToDo = (
219     | d |
220     d := 0.0.
221     0.0 to: 10.0 do: [:ii |
222         d := d + ii.
223     ].
224
225     self assert: 55.0 equals: d.
226
227     d := 0.0.
228     0.0 to: 10.1 do: [:ii |
229         d := d + ii.
230     ].
231

```

```

232     self assert: 55.0 equals: d.
233
234     d := 0.0.
235     0.1 to: 10.1 do: [:ii |
236         d := d + ii.
237     ].
238
239     self assert: 55.0 + 1.1 equals: d.
240 )
241
242 testDownToDo = (
243     | d |
244     d := 0.0.
245     10.0 downTo: 0.0 do: [:ii |
246         d := d + ii.
247     ].
248
249     self assert: 55.0 equals: d.
250
251     d := 0.0.
252     10.1 downTo: 0.0 do: [:ii |
253         d := d + ii.
254     ].
255
256     self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
257 )
258 )
259

```

```
1 "  
2  
3 $Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Double5Test = TestCase (  
27  
28     testAdd = (  
29         self assert: 1.0 equals: 0.5 + 0.5.  
30         self assert: 0.0 equals: 0.5 + -0.5.  
31  
32         self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.  
33         self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.  
34     )  
35  
36     testSubtract = (  
37         self assert: 0.0 equals: 0.5 - 0.5.  
38         self assert: 1.0 equals: 0.5 - -0.5.  
39  
40         self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.  
41         self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.  
42     )  
43  
44     testMultiply = (  
45         self assert: 4.0 equals: 2.0 * 2.0.  
46         self assert: -4.0 equals: 2.0 * -2.0.  
47  
48         self assert: 1.0 equals: 4.0 * 0.25.  
49         self assert: -1.0 equals: -4.0 * 0.25.  
50     )  
51  
52     testIntegerDivision = (  
53         self assert: 1 equals: (4/3) + (4/5)
```

```

54 )
55
56 testDoubleDivision = (
57     self assert: 32 // 15 equals: (4//3) + (4//5)
58 )
59
60 testModulo = (
61     self assert: 1.0 equals: 3.0 % 2.0.
62     self assert: 0.0 equals: 3.0 % 3.0.
63
64     self assert: -1.0 equals: -3.0 % 2.0.
65     self assert: -1.0 equals: -3.0 % -2.0.
66     self assert: 1.0 equals: 3.0 % -2.0.
67
68     self assert: 0.0 equals: 3.0 % 3.0.
69     self assert: 0.0 equals: -3.0 % -3.0.
70     self assert: 0.0 equals: 3.0 % -3.0.
71 )
72
73 testAbs = (
74     self assert: 1.0 equals: 1.0 abs.
75     self assert: 1.0 equals: -1.0 abs.
76
77     self assert: 9007199254740992.0 equals: 9007199254740992.0 abs.
78     self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.
79
80     self assert: 19007199254740992.0 equals: 19007199254740992.0 abs.
81     self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
82 )
83
84 testSqrt = (
85     self assert: 3.0 equals: 9.0 sqrt.
86     self assert: 16.0 equals: 256.0 sqrt.
87
88     self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
89 )
90
91 testNegated = (
92     self assert: 0.0 equals: 0.0 negated.
93     self assert: -1.0 equals: 1.0 negated.
94     self assert: 1.0 equals: -1.0 negated.
95
96     self assert: -9007199254740992.0 equals: 9007199254740992.0 negated.
97     self assert: 9007199254740992.0 equals: -9007199254740992.0 negated.
98
99     self assert: -19007199254740992.0 equals: 19007199254740992.0
negated.
100     self assert: 19007199254740992.0 equals: -19007199254740992.0
negated.
101 )
102
103 testAsString = (
104     self assert: '0.5' equals: (1//2) asString.
105     self assert: '0.5' equals: 0.5 asString.
106 )
107
108 testEquals = (
109     self assert: (1.0 = 1).
110 )
111
112 testRound = (

```

```

113     self assert: 1 equals: 1.0 round.
114     self assert: 1 equals: 1.4 round.
115     self assert: 1 equals: 1.4999 round.
116     self assert: 2 equals: 1.5 round.
117     self assert: 2 equals: 1.5000001 round.
118     self assert: 1 equals: (5//10) round.
119     self assert: 1 equals: (14//10) round.
120     self assert: 445 equals: (44534//100) round.
121 )
122
123 testAsInteger = (
124     self assert: 1 equals: 1.0 asInteger.
125     self assert: 1 equals: 1.1 asInteger.
126     self assert: 1 equals: 1.999 asInteger.
127
128     self assert: -1 equals: -1.0 asInteger.
129     self assert: -1 equals: -1.999 asInteger.
130 )
131
132 testSin = (
133     | pi |
134     pi := 3.141592653589.
135     self assert: 0.0 equals: 0.0 sin.
136     self assert: pi sin abs < 0.000000000001.
137     self assert: (pi // 2.0) sin > 0.9999999999.
138 )
139
140 testCos = (
141     | pi |
142     pi := 3.141592653589.
143     self assert: 1.0 equals: 0.0 cos.
144     self assert: (pi // 2.0) cos abs < 0.000000000001.
145     self assert: pi cos < -0.9999999999.
146 )
147
148 testInfinity = (
149     self assert: Double PositiveInfinity > 1.
150     self assert: Double PositiveInfinity equals: Double PositiveInfinity
151 + 1.
151     self assert: Double PositiveInfinity equals: Double PositiveInfinity
152 - 1.
152
153     self assert: Double PositiveInfinity > (999999 * 999999 * 999999 *
154 999999).
154 )
155
156 testFromString = (
157     self assert: 0.0 equals: (Double fromString: '0.0').
158     self assert: -1.1 equals: (Double fromString: '-1.1').
159
160     self assert: 3423.54656 equals: (Double fromString: '3423.54656').
161     self assert: -672.433244 equals: (Double fromString: '-672.433244').
162 )
163
164 testEqual = (
165     self assert: 0.0 = 0.0.
166     self assert: 1.0 = 1.0.
167     self assert: 0.0 = -0.0.
168     self assert: -0.0 = 0.0.
169 )
170

```

```

171 testLessThan = (
172     self assert: 0.0 < 1.0.
173     self assert: 0.499999999 < 0.5.
174     self deny: 1.0 < 0.0.
175     self deny: 0.5 < 0.499999999.
176 )
177
178 testGreaterThan = (
179     self deny: 0.0 > 1.0.
180     self deny: 0.499999999 > 0.5.
181     self assert: 1.0 > 0.0.
182     self assert: 0.5 > 0.499999999.
183 )
184
185 testLessThanOrEqual = (
186     self assert: 0.0 <= 1.0.
187     self assert: 0.499999999 <= 0.5.
188     self assert: 0.5 <= 0.5.
189     self deny: 1.0 < 0.0.
190     self deny: 0.5 < 0.499999999.
191 )
192
193 testGreaterThanOrEqual = (
194     self deny: 0.0 >= 1.0.
195     self deny: 0.499999999 >= 0.5.
196     self assert: 0.5 >= 0.5.
197     self assert: 1.0 >= 0.0.
198     self assert: 0.5 >= 0.499999999.
199 )
200
201 testNegative = (
202     self assert: -0.000000001 negative.
203     self assert: -1.0 negative.
204     self assert: -123123.000000001 negative.
205     self deny: 0.000000001 negative.
206     self deny: 1.0 negative.
207     self deny: 123123.000000001 negative.
208 )
209
210 testBetween = (
211     self assert: (1.0 between: 0.0 and: 2.0).
212     self assert: (0.000001 between: 0.0 and: 2.0).
213     self assert: (1.999999 between: 0.0 and: 2.0).
214     self deny: (0.0 between: 0.0 and: 2.0).
215     self deny: (2.0 between: 0.0 and: 2.0).
216 )
217
218 testToDo = (
219     | d |
220     d := 0.0.
221     0.0 to: 10.0 do: [:ii |
222         d := d + ii.
223     ].
224
225     self assert: 55.0 equals: d.
226
227     d := 0.0.
228     0.0 to: 10.1 do: [:ii |
229         d := d + ii.
230     ].
231

```

```

232     self assert: 55.0 equals: d.
233
234     d := 0.0.
235     0.1 to: 10.1 do: [:ii |
236         d := d + ii.
237     ].
238
239     self assert: 55.0 + 1.1 equals: d.
240 )
241
242 testDownToDo = (
243     | d |
244     d := 0.0.
245     10.0 downTo: 0.0 do: [:ii |
246         d := d + ii.
247     ].
248
249     self assert: 55.0 equals: d.
250
251     d := 0.0.
252     10.1 downTo: 0.0 do: [:ii |
253         d := d + ii.
254     ].
255
256     self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
257 )
258 )
259

```

```
1 "  
2  
3 $Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 DoubleTest = TestCase (  
27  
28     testAdd = (  
29         self assert: 1.0 equals: 0.5 + 0.5.  
30         self assert: 0.0 equals: 0.5 + -0.5.  
31  
32         self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.  
33         self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.  
34     )  
35  
36     testSubtract = (  
37         self assert: 0.0 equals: 0.5 - 0.5.  
38         self assert: 1.0 equals: 0.5 - -0.5.  
39  
40         self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.  
41         self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.  
42     )  
43  
44     testMultiply = (  
45         self assert: 4.0 equals: 2.0 * 2.0.  
46         self assert: -4.0 equals: 2.0 * -2.0.  
47  
48         self assert: 1.0 equals: 4.0 * 0.25.  
49         self assert: -1.0 equals: -4.0 * 0.25.  
50     )  
51  
52     testIntegerDivision = (  
53         self assert: 1 equals: (4/3) + (4/5)
```



```

54 )
55
56 testDoubleDivision = (
57     self assert: 32 // 15 equals: (4//3) + (4//5)
58 )
59
60 testModulo = (
61     self assert: 1.0 equals: 3.0 % 2.0.
62     self assert: 0.0 equals: 3.0 % 3.0.
63
64     self assert: -1.0 equals: -3.0 % 2.0.
65     self assert: -1.0 equals: -3.0 % -2.0.
66     self assert: 1.0 equals: 3.0 % -2.0.
67
68     self assert: 0.0 equals: 3.0 % 3.0.
69     self assert: 0.0 equals: -3.0 % -3.0.
70     self assert: 0.0 equals: 3.0 % -3.0.
71 )
72
73 testAbs = (
74     self assert: 1.0 equals: 1.0 abs.
75     self assert: 1.0 equals: -1.0 abs.
76
77     self assert: 9007199254740992.0 equals: 9007199254740992.0 abs.
78     self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.
79
80     self assert: 19007199254740992.0 equals: 19007199254740992.0 abs.
81     self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
82 )
83
84 testSqrt = (
85     self assert: 3.0 equals: 9.0 sqrt.
86     self assert: 16.0 equals: 256.0 sqrt.
87
88     self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
89 )
90
91 testNegated = (
92     self assert: 0.0 equals: 0.0 negated.
93     self assert: -1.0 equals: 1.0 negated.
94     self assert: 1.0 equals: -1.0 negated.
95
96     self assert: -9007199254740992.0 equals: 9007199254740992.0 negated.
97     self assert: 9007199254740992.0 equals: -9007199254740992.0 negated.
98
99     self assert: -19007199254740992.0 equals: 19007199254740992.0
negated.
100     self assert: 19007199254740992.0 equals: -19007199254740992.0
negated.
101 )
102
103 testAsString = (
104     self assert: '0.5' equals: (1//2) asString.
105     self assert: '0.5' equals: 0.5 asString.
106 )
107
108 testEquals = (
109     self assert: (1.0 = 1).
110 )
111
112 testRound = (

```

```

113     self assert: 1 equals: 1.0 round.
114     self assert: 1 equals: 1.4 round.
115     self assert: 1 equals: 1.4999 round.
116     self assert: 2 equals: 1.5 round.
117     self assert: 2 equals: 1.5000001 round.
118     self assert: 1 equals: (5//10) round.
119     self assert: 1 equals: (14//10) round.
120     self assert: 445 equals: (44534//100) round.
121 )
122
123 testAsInteger = (
124     self assert: 1 equals: 1.0 asInteger.
125     self assert: 1 equals: 1.1 asInteger.
126     self assert: 1 equals: 1.999 asInteger.
127
128     self assert: -1 equals: -1.0 asInteger.
129     self assert: -1 equals: -1.999 asInteger.
130 )
131
132 testSin = (
133     | pi |
134     pi := 3.141592653589.
135     self assert: 0.0 equals: 0.0 sin.
136     self assert: pi sin abs < 0.000000000001.
137     self assert: (pi // 2.0) sin > 0.9999999999.
138 )
139
140 testCos = (
141     | pi |
142     pi := 3.141592653589.
143     self assert: 1.0 equals: 0.0 cos.
144     self assert: (pi // 2.0) cos abs < 0.000000000001.
145     self assert: pi cos < -0.9999999999.
146 )
147
148 testInfinity = (
149     self assert: Double PositiveInfinity > 1.
150     self assert: Double PositiveInfinity equals: Double PositiveInfinity
151 + 1.
151     self assert: Double PositiveInfinity equals: Double PositiveInfinity
152 - 1.
152
153     self assert: Double PositiveInfinity > (999999 * 999999 * 999999 *
154 999999).
154 )
155
156 testFromString = (
157     self assert: 0.0 equals: (Double fromString: '0.0').
158     self assert: -1.1 equals: (Double fromString: '-1.1').
159
160     self assert: 3423.54656 equals: (Double fromString: '3423.54656').
161     self assert: -672.433244 equals: (Double fromString: '-672.433244').
162 )
163
164 testEqual = (
165     self assert: 0.0 = 0.0.
166     self assert: 1.0 = 1.0.
167     self assert: 0.0 = -0.0.
168     self assert: -0.0 = 0.0.
169 )
170

```

```

171 testLessThan = (
172     self assert: 0.0 < 1.0.
173     self assert: 0.499999999 < 0.5.
174     self deny: 1.0 < 0.0.
175     self deny: 0.5 < 0.499999999.
176 )
177
178 testGreaterThan = (
179     self deny: 0.0 > 1.0.
180     self deny: 0.499999999 > 0.5.
181     self assert: 1.0 > 0.0.
182     self assert: 0.5 > 0.499999999.
183 )
184
185 testLessThanOrEqual = (
186     self assert: 0.0 <= 1.0.
187     self assert: 0.499999999 <= 0.5.
188     self assert: 0.5 <= 0.5.
189     self deny: 1.0 < 0.0.
190     self deny: 0.5 < 0.499999999.
191 )
192
193 testGreaterThanOrEqual = (
194     self deny: 0.0 >= 1.0.
195     self deny: 0.499999999 >= 0.5.
196     self assert: 0.5 >= 0.5.
197     self assert: 1.0 >= 0.0.
198     self assert: 0.5 >= 0.499999999.
199 )
200
201 testNegative = (
202     self assert: -0.000000001 negative.
203     self assert: -1.0 negative.
204     self assert: -123123.000000001 negative.
205     self deny: 0.000000001 negative.
206     self deny: 1.0 negative.
207     self deny: 123123.000000001 negative.
208 )
209
210 testBetween = (
211     self assert: (1.0 between: 0.0 and: 2.0).
212     self assert: (0.000001 between: 0.0 and: 2.0).
213     self assert: (1.999999 between: 0.0 and: 2.0).
214     self deny: (0.0 between: 0.0 and: 2.0).
215     self deny: (2.0 between: 0.0 and: 2.0).
216 )
217
218 testToDo = (
219     | d |
220     d := 0.0.
221     0.0 to: 10.0 do: [:ii |
222         d := d + ii.
223     ].
224
225     self assert: 55.0 equals: d.
226
227     d := 0.0.
228     0.0 to: 10.1 do: [:ii |
229         d := d + ii.
230     ].
231

```

```

232     self assert: 55.0 equals: d.
233
234     d := 0.0.
235     0.1 to: 10.1 do: [:ii |
236         d := d + ii.
237     ].
238
239     self assert: 55.0 + 1.1 equals: d.
240 )
241
242 testDownToDo = (
243     | d |
244     d := 0.0.
245     10.0 downTo: 0.0 do: [:ii |
246         d := d + ii.
247     ].
248
249     self assert: 55.0 equals: d.
250
251     d := 0.0.
252     10.1 downTo: 0.0 do: [:ii |
253         d := d + ii.
254     ].
255
256     self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
257 )
258 )
259

```

```
1 "  
2  
3 $Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Empty2Test = TestCase (  
27  
28     "This is just an empty TestCase.  
29     It only tests the basic infrastructure"  
30  
31 )  
32
```

```
1 "  
2  
3 $Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Empty3Test = TestCase (  
27  
28     "This is just an empty TestCase.  
29     It only tests the basic infrastructure"  
30  
31 )  
32
```

```
1 "  
2  
3 $Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Empty4Test = TestCase (  
27  
28     "This is just an empty TestCase.  
29     It only tests the basic infrastructure"  
30  
31 )  
32
```

```
1 "  
2  
3 $Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Empty5Test = TestCase (  
27  
28     "This is just an empty TestCase.  
29     It only tests the basic infrastructure"  
30  
31 )  
32
```



```
1 "  
2  
3 $Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 EmptyTest = TestCase (  
27  
28     "This is just an empty TestCase.  
29     It only tests the basic infrastructure"  
30  
31 )  
32
```

```

1 Global2Test = TestCase (
2   | doesntKnow |
3   unknownGlobal: name = ( doesntKnow := name. ^ name )
4
5   testUnknownGlobalHandler = (
6     self assert: #foobar equals: foobar.      "should return the unknown
globals name"
7     self assert: #foobar equals: doesntKnow. "and should have set it in
the field"
8   )
9
10  testKnownGlobals = (
11    self assert: True   equals: true   class.
12    self assert: False  equals: false  class.
13    self assert: Nil    equals: nil    class.
14    self assert: System equals: system class.
15  )
16
17  escapingBlock = (
18    ^ [ EscapingBlockGlobal ]
19  )
20
21  testUnknownGlobalSemanticsInBlocks = (
22    self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
23    self assert: #NormalBlockGlobal is: doesntKnow.
24
25    self assert: #EscapingBlockGlobal is: self escapingBlock value.
26    self assert: #EscapingBlockGlobal is: doesntKnow.
27
28    self assert: #NestedBlockGlobal is: [
29      [ [ NestedBlockGlobal ] value ] value ] value.
30    self assert: #NestedBlockGlobal is: doesntKnow.
31  )
32 )
33

```

```

1 Global3Test = TestCase (
2   | doesntKnow |
3   unknownGlobal: name = ( doesntKnow := name. ^ name )
4
5   testUnknownGlobalHandler = (
6     self assert: #foobar equals: foobar.      "should return the unknown
globals name"
7     self assert: #foobar equals: doesntKnow. "and should have set it in
the field"
8   )
9
10  testKnownGlobals = (
11    self assert: True   equals: true   class.
12    self assert: False  equals: false  class.
13    self assert: Nil    equals: nil    class.
14    self assert: System equals: system class.
15  )
16
17  escapingBlock = (
18    ^ [ EscapingBlockGlobal ]
19  )
20
21  testUnknownGlobalSemanticsInBlocks = (
22    self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
23    self assert: #NormalBlockGlobal is: doesntKnow.
24
25    self assert: #EscapingBlockGlobal is: self escapingBlock value.
26    self assert: #EscapingBlockGlobal is: doesntKnow.
27
28    self assert: #NestedBlockGlobal is: [
29      [ [ NestedBlockGlobal ] value ] value ] value.
30    self assert: #NestedBlockGlobal is: doesntKnow.
31  )
32 )
33

```

```

1 Global4Test = TestCase (
2   | doesntKnow |
3   unknownGlobal: name = ( doesntKnow := name. ^ name )
4
5   testUnknownGlobalHandler = (
6     self assert: #foobar equals: foobar.      "should return the unknown
globals name"
7     self assert: #foobar equals: doesntKnow. "and should have set it in
the field"
8   )
9
10  testKnownGlobals = (
11    self assert: True   equals: true   class.
12    self assert: False  equals: false  class.
13    self assert: Nil    equals: nil    class.
14    self assert: System equals: system class.
15  )
16
17  escapingBlock = (
18    ^ [ EscapingBlockGlobal ]
19  )
20
21  testUnknownGlobalSemanticsInBlocks = (
22    self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
23    self assert: #NormalBlockGlobal is: doesntKnow.
24
25    self assert: #EscapingBlockGlobal is: self escapingBlock value.
26    self assert: #EscapingBlockGlobal is: doesntKnow.
27
28    self assert: #NestedBlockGlobal is: [
29      [ [ NestedBlockGlobal ] value ] value ] value.
30    self assert: #NestedBlockGlobal is: doesntKnow.
31  )
32 )
33

```

```

1 Global5Test = TestCase (
2   | doesntKnow |
3   unknownGlobal: name = ( doesntKnow := name. ^ name )
4
5   testUnknownGlobalHandler = (
6     self assert: #foobar equals: foobar.      "should return the unknown
globals name"
7     self assert: #foobar equals: doesntKnow. "and should have set it in
the field"
8   )
9
10  testKnownGlobals = (
11    self assert: True   equals: true   class.
12    self assert: False  equals: false  class.
13    self assert: Nil    equals: nil    class.
14    self assert: System equals: system class.
15  )
16
17  escapingBlock = (
18    ^ [ EscapingBlockGlobal ]
19  )
20
21  testUnknownGlobalSemanticsInBlocks = (
22    self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
23    self assert: #NormalBlockGlobal is: doesntKnow.
24
25    self assert: #EscapingBlockGlobal is: self escapingBlock value.
26    self assert: #EscapingBlockGlobal is: doesntKnow.
27
28    self assert: #NestedBlockGlobal is: [
29      [ [ NestedBlockGlobal ] value ] value ] value.
30    self assert: #NestedBlockGlobal is: doesntKnow.
31  )
32 )
33

```

```

1 GlobalTest = TestCase (
2   | doesntKnow |
3   unknownGlobal: name = ( doesntKnow := name. ^ name )
4
5   testUnknownGlobalHandler = (
6     self assert: #foobar equals: foobar.      "should return the unknown
globals name"
7     self assert: #foobar equals: doesntKnow. "and should have set it in
the field"
8   )
9
10  testKnownGlobals = (
11    self assert: True   equals: true   class.
12    self assert: False  equals: false  class.
13    self assert: Nil    equals: nil    class.
14    self assert: System equals: system class.
15  )
16
17  escapingBlock = (
18    ^ [ EscapingBlockGlobal ]
19  )
20
21  testUnknownGlobalSemanticsInBlocks = (
22    self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
23    self assert: #NormalBlockGlobal is: doesntKnow.
24
25    self assert: #EscapingBlockGlobal is: self escapingBlock value.
26    self assert: #EscapingBlockGlobal is: doesntKnow.
27
28    self assert: #NestedBlockGlobal is: [
29      [ [ NestedBlockGlobal ] value ] value ] value.
30    self assert: #NestedBlockGlobal is: doesntKnow.
31  )
32 )
33

```

```
1 "  
2  
3 $Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Hash2Test = TestCase (  
27   testHashtable = (  
28     | ht string array t |  
29  
30     ht := Hashtable new.  
31     self assert: ht isEmpty description: 'new ht needs to be empty'.  
32  
33     ht at: 'a' put: 'b'.  
34     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
35     self deny: ht isEmpty.  
36  
37     self assert: 1 equals: ht size.  
38  
39     ht at: 'c' put: 'd'.  
40     self assert: 2 equals: ht size.  
41  
42     ht at: 1 put: 2.  
43     t := Hashtable new.  
44     ht at: Hashtable put: t.  
45  
46     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
47     self assert: (ht containsValue: 'd') description: 'needs to contain  
"d"'.  
48     self assert: (ht containsValue: 2) description: 'needs to contain  
"2"'.  
49     self assert: (ht containsValue: t) description: 'needs to contain t'.  
50
```

```
51     self assert: (ht containsKey: Hashtable) description: 'needs to
contain Hashtable'.
52
53     ht clear.
54     self assert: ht isEmpty.
55     self assert: 0 equals: ht size.
56
57     self assert: nil equals: (ht get: 'a').
58 )
59 )
60
61
```



```
1 "  
2  
3 $Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Hash3Test = TestCase (  
27   testHashtable = (  
28     | ht string array t |  
29  
30     ht := Hashtable new.  
31     self assert: ht isEmpty description: 'new ht needs to be empty'.  
32  
33     ht at: 'a' put: 'b'.  
34     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
35     self deny: ht isEmpty.  
36  
37     self assert: 1 equals: ht size.  
38  
39     ht at: 'c' put: 'd'.  
40     self assert: 2 equals: ht size.  
41  
42     ht at: 1 put: 2.  
43     t := Hashtable new.  
44     ht at: Hashtable put: t.  
45  
46     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
47     self assert: (ht containsValue: 'd') description: 'needs to contain  
"d"'.  
48     self assert: (ht containsValue: 2) description: 'needs to contain  
"2"'.  
49     self assert: (ht containsValue: t) description: 'needs to contain t'.  
50
```

```
51     self assert: (ht containsKey: Hashtable) description: 'needs to
contain Hashtable'.
52
53     ht clear.
54     self assert: ht isEmpty.
55     self assert: 0 equals: ht size.
56
57     self assert: nil equals: (ht get: 'a').
58 )
59 )
60
61
```

```
1 "  
2  
3 $Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Hash4Test = TestCase (  
27  
28   testHashtable = (  
29     | ht string array t |  
30  
31     ht := Hashtable new.  
32     self assert: ht isEmpty description: 'new ht needs to be empty'.  
33  
34     ht at: 'a' put: 'b'.  
35     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
36     self deny: ht isEmpty.  
37  
38     self assert: 1 equals: ht size.  
39  
40     ht at: 'c' put: 'd'.  
41     self assert: 2 equals: ht size.  
42  
43     ht at: 1 put: 2.  
44     t := Hashtable new.  
45     ht at: Hashtable put: t.  
46  
47     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
48     self assert: (ht containsValue: 'd') description: 'needs to contain  
"d"'.  
49     self assert: (ht containsValue: 2) description: 'needs to contain  
"2"'.  
50     self assert: (ht containsValue: t) description: 'needs to contain t'.
```

```
51     self assert: (ht containsKey: Hashtable) description: 'needs to
contain Hashtable'.
52
53     ht clear.
54     self assert: ht isEmpty.
55     self assert: 0 equals: ht size.
56
57     self assert: nil equals: (ht get: 'a').
58 )
59 )
60
61
```

```
1 "  
2  
3 $Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Hash5Test = TestCase (  
27  
28   testHashtable = (  
29     | ht string array t |  
30  
31     ht := Hashtable new.  
32     self assert: ht isEmpty description: 'new ht needs to be empty'.  
33  
34     ht at: 'a' put: 'b'.  
35     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
36     self deny: ht isEmpty.  
37  
38     self assert: 1 equals: ht size.  
39  
40     ht at: 'c' put: 'd'.  
41     self assert: 2 equals: ht size.  
42  
43     ht at: 1 put: 2.  
44     t := Hashtable new.  
45     ht at: Hashtable put: t.  
46  
47     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
48     self assert: (ht containsValue: 'd') description: 'needs to contain  
"d"'.  
49     self assert: (ht containsValue: 2) description: 'needs to contain  
"2"'.  
50     self assert: (ht containsValue: t) description: 'needs to contain t'.
```

```
51     self assert: (ht containsKey: Hashtable) description: 'needs to
contain Hashtable'.
52
53     ht clear.
54     self assert: ht isEmpty.
55     self assert: 0 equals: ht size.
56
57     self assert: nil equals: (ht get: 'a').
58 )
59 )
60
61
```

```
1 "  
2  
3 $Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 HashTest = TestCase (  
27  
28   testHashtable = (  
29     | ht string array t |  
30  
31     ht := Hashtable new.  
32     self assert: ht isEmpty description: 'new ht needs to be empty'.  
33  
34     ht at: 'a' put: 'b'.  
35     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
36     self deny: ht isEmpty.  
37  
38     self assert: 1 equals: ht size.  
39  
40     ht at: 'c' put: 'd'.  
41     self assert: 2 equals: ht size.  
42  
43     ht at: 1 put: 2.  
44     t := Hashtable new.  
45     ht at: Hashtable put: t.  
46  
47     self assert: (ht containsValue: 'b') description: 'needs to contain  
"b"'.  
48     self assert: (ht containsValue: 'd') description: 'needs to contain  
"d"'.  
49     self assert: (ht containsValue: 2) description: 'needs to contain  
"2"'.  
50     self assert: (ht containsValue: t) description: 'needs to contain t'.
```

```
51     self assert: (ht containsKey: Hashtable) description: 'needs to
contain Hashtable'.
52
53     ht clear.
54     self assert: ht isEmpty.
55     self assert: 0 equals: ht size.
56
57     self assert: nil equals: (ht get: 'a').
58 )
59 )
60
61
```



```
1 "  
2  
3 $Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Integer2Test = TestCase (  
29  
30   testEqualityAndIdentity = (  
31     | a b |  
32     a := 42.  
33     b := 42.  
34  
35     self assert: a = b    description: 'Integers are equal based on their  
value'.  
36     self assert: a == b  description: 'Integers do not have pointer/  
reference equality. It is also supposed to be value equality'.  
37  
38     "Sometimes it can be hard to implement efficiently, but it SHOULD  
really  
39     be true for all values of integers."  
40     a := 1 << 30.  b := 1 << 30.  
41     self optional: #integerIdentity assert: a is: b.  
42  
43     a := 1 << 32.  b := 1 << 32.  
44     self optional: #integerIdentity assert: a is: b.  
45  
46     a := 1 << 60.  b := 1 << 60.  
47     self optional: #integerIdentity assert: a is: b.  
48   )  
49  
50   testClassAndValueRanges = (  

```

```

51     | i |
52     self assert: Integer equals: -42 class.
53     self assert: Integer equals: 0 class.
54     self assert: Integer equals: 23 class.
55     self assert: Integer equals: 1073741823 class.
56     self assert: Integer equals: 1073741824 class.
57
58     "Let's test for size behavior and corresponding class"
59     i := 1 << 30.
60     self assert: Integer equals: i class.
61     self assert: i > 0 description: 'should not overflow'.
62     self assert: '1073741824' equals: i asString.
63
64     i := 1 << 32.
65     self assert: Integer equals: i class.
66     self assert: i > 0 description: 'should not overflow'.
67     self assert: '4294967296' equals: i asString.
68
69     i := 1 << 60.
70     self assert: Integer equals: i class.
71     self assert: i > 0 description: 'should not overflow'.
72     self assert: '1152921504606846976' equals: i asString.
73
74     i := 1 << 70.
75     self assert: Integer equals: i class.
76     self assert: i > 0 description: 'should not overflow'.
77     self optional: #bigIntShifts assert: '1180591620717411303424' equals:
i asString.
78
79     i := -1 << 30.
80     self assert: Integer equals: i class.
81     self assert: i < 0 description: 'should not underflow'.
82     self assert: '-1073741824' equals: i asString.
83
84     i := -1 << 32.
85     self assert: Integer equals: i class.
86     self assert: i < 0 description: 'should not underflow'.
87     self assert: '-4294967296' equals: i asString.
88
89     i := -1 << 60.
90     self assert: Integer equals: i class.
91     self assert: i < 0 description: 'should not underflow'.
92     self assert: '-1152921504606846976' equals: i asString.
93
94     i := -1 << 70.
95     self assert: Integer equals: i class.
96     self assert: i < 0 description: 'should not underflow'.
97     self optional: #bigIntShifts assert: '-1180591620717411303424'
equals: i asString.
98 )
99
100 testStringConversion = (
101     self assert: '0' equals: ( 0 asString).
102     self assert: '1' equals: ( 1 asString).
103     self assert: '2' equals: ( 2 asString).
104     self assert: '-1' equals: (-1 asString).
105     self assert: '-2' equals: (-2 asString).
106
107     self assert: 42 equals: '42' asInteger.
108     self assert: -2 equals: '-2' asInteger.
109 )

```

```

110
111     testIntegerLiterals = (
112         "Make sure the parser reads literals correctly. So, check some basic
properties"
113         self assert: 2 / 2
equals: 1.
114         self assert: 50 + 50
equals: 100.
115         self assert: 92233720368 * 100000000 + 54775807 equals:
9223372036854775807.
116         self assert: 92233720368 * 100000000 + 54775807 * 100 equals:
922337203685477580700.
117         self assert: 50 - 100
equals: -50.
118         self assert: 21474 * -100000 - 83648
equals: -2147483648.
119         self assert: 92233720368 * 100000000 + 54775807 * -100 equals:
-922337203685477580700.
120     )
121
122     testFromString = (
123         self assert: 1 equals: (Integer
fromString: '1').
124         self assert: 100 equals: (Integer
fromString: '100').
125         self assert: 9223372036854775807 equals: (Integer fromString:
'9223372036854775807').
126         self assert: 922337203685477580700 equals: (Integer fromString:
'922337203685477580700').
127         self assert: -50 equals: (Integer
fromString: '-50').
128         self assert: -2147483648 equals: (Integer
fromString: '-2147483648').
129         self assert: -922337203685477580700 equals: (Integer fromString:
'-922337203685477580700').
130     )
131
132     testRangeBorders = (
133         self assert: '536870911' equals: 536870911 asString.
134         self assert: '536870912' equals: 536870912 asString.
135         self assert: '536870913' equals: 536870913 asString.
136         self assert: '1073741823' equals: 1073741823 asString.
137         self assert: '1073741824' equals: 1073741824 asString.
138         self assert: '1073741825' equals: 1073741825 asString.
139         self assert: '2147483647' equals: 2147483647 asString.
140         self assert: '-536870911' equals: -536870911 asString.
141         self assert: '-536870912' equals: -536870912 asString.
142         self assert: '-536870913' equals: -536870913 asString.
143         self assert: '-1073741823' equals: -1073741823 asString.
144         self assert: '-1073741824' equals: -1073741824 asString.
145         self assert: '-1073741825' equals: -1073741825 asString.
146         self assert: '-2147483647' equals: -2147483647 asString.
147         self assert: '-2147483648' equals: -2147483648 asString.
148     )
149
150     testComparisons = (
151         self assert: ( 9 = 9 ).
152         self deny: ( 1 = 2 ).
153         self deny: ( 0 < 0 ).
154         self assert: ( 1 < 2 ).
155         self deny: ( 2 < 1 ).

```

```

156     self assert: (-3 < 2).
157     self deny: ( 3 < -2).
158     self deny: ( 0 > 0).
159     self deny: ( 1 > 2).
160     self assert: ( 2 > 1).
161     self deny: (-3 > 2).
162     self assert: ( 3 > -2).
163     self assert: ( 4 >= 3).
164     self assert: ( 3 >= 3).
165     self deny: ( 2 >= 3).
166     self assert: ( 2 <= 4).
167     self assert: ( 3 <= 3).
168     self deny: ( 4 <= 3).
169 )
170
171 testAddition = (
172     self assert: 0 equals: ( 0+0).
173     self assert: 1 equals: ( 1+0).
174     self assert: 1 equals: ( 0+1).
175     self assert: 2 equals: ( 1+1).
176     self assert: 0 equals: (-1+1).
177     self assert: 1 equals: (-1+2).
178 )
179
180 testSubtraction = (
181     self assert: 1 equals: (1-0).
182     self assert: -1 equals: (0-1).
183     self assert: 1 equals: (2-1).
184 )
185
186 testMultiplication = (
187     self assert: 0 equals: ( 1* 0).
188     self assert: -1 equals: (-1* 1).
189     self assert: -25 equals: ( 5* -5).
190     self assert: 12 equals: (-3* -4).
191 )
192
193 testDivision = (
194     self assert: 1 equals: ( 1/ 1).
195     self assert: 1 equals: ( 3/ 2).
196     self assert: -2 equals: ( 4/ -2).
197     self assert: -2 equals: (-6/ 3).
198     self assert: 3 equals: (-12/ -4).
199 )
200
201 testDouble = (
202     self assert: 6 equals: ( 36// 6).
203     self assert: -5 equals: (-10// 2).
204     self assert: -4 equals: ( 20// -5).
205     self assert: 1 equals: ( -5// -5).
206 )
207
208 testModulo = (
209     self assert: 1 equals: ( 10 % 3).
210     self assert: -2 equals: ( 10 % -3).
211     self assert: 2 equals: (-10 % 3).
212     self assert: -1 equals: (-10 % -3).
213     self assert: 0 equals: ( 10 % 5).
214
215     self assert: 1 equals: ( 10 rem: 3).
216     self assert: 1 equals: ( 10 rem: -3).

```

```

217     self assert: -1 equals: (-10 rem: 3).
218     self assert: -1 equals: (-10 rem: -3).
219     self assert: 0 equals: (10 rem: 5).
220 )
221
222 testAbsoluteValue = (
223     self assert: 4 equals: -4 abs.
224     self assert: 4 equals: 4 abs.
225
226     self assert: 9223372036854775296 equals: -9223372036854775296 abs.
227     self assert: 9223372036854775296 equals: 9223372036854775296 abs.
228 )
229
230 testNegated = (
231     self assert: -23 equals: (23 negated).
232     self assert: 23 equals: (-23 negated).
233 )
234
235 testSquareRoot = (
236     self assert: 5 equals: (25 sqrt).
237     self assert: Integer equals: (25 sqrt class).
238 )
239
240 testAnd = (
241     self assert: 0 equals: (2 & 1).
242     self assert: 2 equals: (2 & 2).
243 )
244
245 testBitXor = (
246     self assert: 0 equals: (1 bitXor: 1).
247     self assert: 3 equals: (2 bitXor: 1).
248 )
249
250 testAs32BitUnsignedValue = (
251     self assert: 1 << 1 equals: (1 << 1) as32BitUnsignedValue.
252     self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
253     self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
254     self assert: 0 equals: (1 << 32) as32BitUnsignedValue.
255     self assert: 4294967295 equals: -1 as32BitUnsignedValue.
256     self assert: 512 equals: -9223372036854775296
as32BitUnsignedValue.
257     self assert: 4294966784 equals: 9223372036854775296
as32BitUnsignedValue.
258 )
259
260 testAs32BitSignedValue = (
261     self assert: 1 << 1 equals: (1 << 1) as32BitSignedValue.
262     self assert: 1 << 10 equals: (1 << 10) as32BitSignedValue.
263     self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
264     self assert: 0 equals: (1 << 32) as32BitSignedValue.
265
266     self assert: 512 equals: -9223372036854775296 as32BitSignedValue.
267     self assert: -512 equals: 9223372036854775296 as32BitSignedValue.
268 )
269
270 testAsDouble = (
271     self assert: 0.0 equals: 0 asDouble.
272     self assert: Double is: 0 asDouble class.
273
274     self assert: 2147483648.0 equals: 2147483648 asDouble.
275     self assert: Double is: 2147483648 asDouble class.

```

```

276
277     self assert: -2147483648.0 equals: -2147483648 asDouble.
278     self assert: Double      is:      -2147483648 asDouble class.
279 )
280
281 testUnsignedRightShift = (
282     self assert: 0 equals: 1 >>> 1.
283     self assert: 512 equals: 1024 >>> 1.
284     self assert: 127 equals: 1023 >>> 3.
285
286     "not sure whether we should really insist on this"
287     self optional: #toBeSpecified assert: 9223372036854775807 equals:
-1 >>> 1.
288     self optional: #toBeSpecified assert: 9223372036854775296 equals:
-1024 >>> 1.
289 )
290
291 testMin = (
292     "We need to test numbers that are 64bit or less, larger than 64bit,
293     positive, and negative"
294     | big small |
295     big := #(1 100 9223372036854775807 922337203685477580700
296             -50 -2147483648 922337203685477580700
-922337203685477580700
297             922337203685477580700).
298     small := #(0 52 9223372036854775296 922337203685477529600
299                -51 -2147483650 1
-922337203685477580701
300                -922337203685477580701).
301
302     big doIndexes: [:i |
303         self assert: (small at: i) equals: ((big at: i) min: (small at:
i)).
304         self assert: (small at: i) equals: ((small at: i) min: (big at:
i))] ]
305 )
306
307 testMax = (
308     "We need to test numbers that are 64bit or less, larger than 64bit,
309     positive, and negative"
310     | big small |
311     big := #(1 100 9223372036854775807 922337203685477580700
312             -50 -2147483648 922337203685477580700
-922337203685477580700
313             922337203685477580700).
314     small := #(0 52 9223372036854775296 922337203685477529600
315                -51 -2147483650 1
-922337203685477580701
316                -922337203685477580701).
317     big doIndexes: [:i |
318         self assert: (big at: i) equals: ((big at: i) max: (small at:
i)).
319         self assert: (big at: i) equals: ((small at: i) max: (big at:
i))] ]
320 )
321 )
322

```

```
1 "  
2  
3 $Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Integer3Test = TestCase (  
29  
30   testEqualityAndIdentity = (  
31     | a b |  
32     a := 42.  
33     b := 42.  
34  
35     self assert: a = b    description: 'Integers are equal based on their  
value'.  
36     self assert: a == b  description: 'Integers do not have pointer/  
reference equality. It is also supposed to be value equality'.  
37  
38     "Sometimes it can be hard to implement efficiently, but it SHOULD  
really  
39     be true for all values of integers."  
40     a := 1 << 30.  b := 1 << 30.  
41     self optional: #integerIdentity assert: a is: b.  
42  
43     a := 1 << 32.  b := 1 << 32.  
44     self optional: #integerIdentity assert: a is: b.  
45  
46     a := 1 << 60.  b := 1 << 60.  
47     self optional: #integerIdentity assert: a is: b.  
48   )  
49  
50   testClassAndValueRanges = (  

```

```

51     | i |
52     self assert: Integer equals: -42 class.
53     self assert: Integer equals: 0 class.
54     self assert: Integer equals: 23 class.
55     self assert: Integer equals: 1073741823 class.
56     self assert: Integer equals: 1073741824 class.
57
58     "Let's test for size behavior and corresponding class"
59     i := 1 << 30.
60     self assert: Integer equals: i class.
61     self assert: i > 0 description: 'should not overflow'.
62     self assert: '1073741824' equals: i asString.
63
64     i := 1 << 32.
65     self assert: Integer equals: i class.
66     self assert: i > 0 description: 'should not overflow'.
67     self assert: '4294967296' equals: i asString.
68
69     i := 1 << 60.
70     self assert: Integer equals: i class.
71     self assert: i > 0 description: 'should not overflow'.
72     self assert: '1152921504606846976' equals: i asString.
73
74     i := 1 << 70.
75     self assert: Integer equals: i class.
76     self assert: i > 0 description: 'should not overflow'.
77     self optional: #bigIntShifts assert: '1180591620717411303424' equals:
i asString.
78
79     i := -1 << 30.
80     self assert: Integer equals: i class.
81     self assert: i < 0 description: 'should not underflow'.
82     self assert: '-1073741824' equals: i asString.
83
84     i := -1 << 32.
85     self assert: Integer equals: i class.
86     self assert: i < 0 description: 'should not underflow'.
87     self assert: '-4294967296' equals: i asString.
88
89     i := -1 << 60.
90     self assert: Integer equals: i class.
91     self assert: i < 0 description: 'should not underflow'.
92     self assert: '-1152921504606846976' equals: i asString.
93
94     i := -1 << 70.
95     self assert: Integer equals: i class.
96     self assert: i < 0 description: 'should not underflow'.
97     self optional: #bigIntShifts assert: '-1180591620717411303424'
equals: i asString.
98 )
99
100 testStringConversion = (
101     self assert: '0' equals: ( 0 asString).
102     self assert: '1' equals: ( 1 asString).
103     self assert: '2' equals: ( 2 asString).
104     self assert: '-1' equals: (-1 asString).
105     self assert: '-2' equals: (-2 asString).
106
107     self assert: 42 equals: '42' asInteger.
108     self assert: -2 equals: '-2' asInteger.
109 )

```



```

110
111     testIntegerLiterals = (
112         "Make sure the parser reads literals correctly. So, check some basic
properties"
113         self assert: 2 / 2
equals: 1.
114         self assert: 50 + 50
equals: 100.
115         self assert: 92233720368 * 100000000 + 54775807 equals:
9223372036854775807.
116         self assert: 92233720368 * 100000000 + 54775807 * 100 equals:
922337203685477580700.
117         self assert: 50 - 100
equals: -50.
118         self assert: 21474 * -100000 - 83648
equals: -2147483648.
119         self assert: 92233720368 * 100000000 + 54775807 * -100 equals:
-922337203685477580700.
120     )
121
122     testFromString = (
123         self assert: 1 equals: (Integer
fromString: '1').
124         self assert: 100 equals: (Integer
fromString: '100').
125         self assert: 9223372036854775807 equals: (Integer fromString:
'9223372036854775807').
126         self assert: 922337203685477580700 equals: (Integer fromString:
'922337203685477580700').
127         self assert: -50 equals: (Integer
fromString: '-50').
128         self assert: -2147483648 equals: (Integer
fromString: '-2147483648').
129         self assert: -922337203685477580700 equals: (Integer fromString:
'-922337203685477580700').
130     )
131
132     testRangeBorders = (
133         self assert: '536870911' equals: 536870911 asString.
134         self assert: '536870912' equals: 536870912 asString.
135         self assert: '536870913' equals: 536870913 asString.
136         self assert: '1073741823' equals: 1073741823 asString.
137         self assert: '1073741824' equals: 1073741824 asString.
138         self assert: '1073741825' equals: 1073741825 asString.
139         self assert: '2147483647' equals: 2147483647 asString.
140         self assert: '-536870911' equals: -536870911 asString.
141         self assert: '-536870912' equals: -536870912 asString.
142         self assert: '-536870913' equals: -536870913 asString.
143         self assert: '-1073741823' equals: -1073741823 asString.
144         self assert: '-1073741824' equals: -1073741824 asString.
145         self assert: '-1073741825' equals: -1073741825 asString.
146         self assert: '-2147483647' equals: -2147483647 asString.
147         self assert: '-2147483648' equals: -2147483648 asString.
148     )
149
150     testComparisons = (
151         self assert: ( 9 = 9 ).
152         self deny: ( 1 = 2 ).
153         self deny: ( 0 < 0 ).
154         self assert: ( 1 < 2 ).
155         self deny: ( 2 < 1 ).

```

```

156     self assert: (-3 < 2).
157     self deny: ( 3 < -2).
158     self deny: ( 0 > 0).
159     self deny: ( 1 > 2).
160     self assert: ( 2 > 1).
161     self deny: (-3 > 2).
162     self assert: ( 3 > -2).
163     self assert: ( 4 >= 3).
164     self assert: ( 3 >= 3).
165     self deny: ( 2 >= 3).
166     self assert: ( 2 <= 4).
167     self assert: ( 3 <= 3).
168     self deny: ( 4 <= 3).
169 )
170
171 testAddition = (
172     self assert: 0 equals: ( 0+0).
173     self assert: 1 equals: ( 1+0).
174     self assert: 1 equals: ( 0+1).
175     self assert: 2 equals: ( 1+1).
176     self assert: 0 equals: (-1+1).
177     self assert: 1 equals: (-1+2).
178 )
179
180 testSubtraction = (
181     self assert: 1 equals: (1-0).
182     self assert: -1 equals: (0-1).
183     self assert: 1 equals: (2-1).
184 )
185
186 testMultiplication = (
187     self assert: 0 equals: ( 1* 0).
188     self assert: -1 equals: (-1* 1).
189     self assert: -25 equals: ( 5* -5).
190     self assert: 12 equals: (-3* -4).
191 )
192
193 testDivision = (
194     self assert: 1 equals: ( 1/ 1).
195     self assert: 1 equals: ( 3/ 2).
196     self assert: -2 equals: ( 4/ -2).
197     self assert: -2 equals: (-6/ 3).
198     self assert: 3 equals: (-12/ -4).
199 )
200
201 testDouble = (
202     self assert: 6 equals: ( 36// 6).
203     self assert: -5 equals: (-10// 2).
204     self assert: -4 equals: ( 20// -5).
205     self assert: 1 equals: ( -5// -5).
206 )
207
208 testModulo = (
209     self assert: 1 equals: ( 10 % 3).
210     self assert: -2 equals: ( 10 % -3).
211     self assert: 2 equals: (-10 % 3).
212     self assert: -1 equals: (-10 % -3).
213     self assert: 0 equals: ( 10 % 5).
214
215     self assert: 1 equals: ( 10 rem: 3).
216     self assert: 1 equals: ( 10 rem: -3).

```

```

217     self assert: -1 equals: (-10 rem: 3).
218     self assert: -1 equals: (-10 rem: -3).
219     self assert: 0 equals: (10 rem: 5).
220 )
221
222 testAbsoluteValue = (
223     self assert: 4 equals: -4 abs.
224     self assert: 4 equals: 4 abs.
225
226     self assert: 9223372036854775296 equals: -9223372036854775296 abs.
227     self assert: 9223372036854775296 equals: 9223372036854775296 abs.
228 )
229
230 testNegated = (
231     self assert: -23 equals: (23 negated).
232     self assert: 23 equals: (-23 negated).
233 )
234
235 testSquareRoot = (
236     self assert: 5 equals: (25 sqrt).
237     self assert: Integer equals: (25 sqrt class).
238 )
239
240 testAnd = (
241     self assert: 0 equals: (2 & 1).
242     self assert: 2 equals: (2 & 2).
243 )
244
245 testBitXor = (
246     self assert: 0 equals: (1 bitXor: 1).
247     self assert: 3 equals: (2 bitXor: 1).
248 )
249
250 testAs32BitUnsignedValue = (
251     self assert: 1 << 1 equals: (1 << 1) as32BitUnsignedValue.
252     self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
253     self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
254     self assert: 0 equals: (1 << 32) as32BitUnsignedValue.
255     self assert: 4294967295 equals: -1 as32BitUnsignedValue.
256     self assert: 512 equals: -9223372036854775296
as32BitUnsignedValue.
257     self assert: 4294966784 equals: 9223372036854775296
as32BitUnsignedValue.
258 )
259
260 testAs32BitSignedValue = (
261     self assert: 1 << 1 equals: (1 << 1) as32BitSignedValue.
262     self assert: 1 << 10 equals: (1 << 10) as32BitSignedValue.
263     self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
264     self assert: 0 equals: (1 << 32) as32BitSignedValue.
265
266     self assert: 512 equals: -9223372036854775296 as32BitSignedValue.
267     self assert: -512 equals: 9223372036854775296 as32BitSignedValue.
268 )
269
270 testAsDouble = (
271     self assert: 0.0 equals: 0 asDouble.
272     self assert: Double is: 0 asDouble class.
273
274     self assert: 2147483648.0 equals: 2147483648 asDouble.
275     self assert: Double is: 2147483648 asDouble class.

```

```

276
277     self assert: -2147483648.0 equals: -2147483648 asDouble.
278     self assert: Double      is:      -2147483648 asDouble class.
279 )
280
281 testUnsignedRightShift = (
282     self assert: 0 equals: 1 >>> 1.
283     self assert: 512 equals: 1024 >>> 1.
284     self assert: 127 equals: 1023 >>> 3.
285
286     "not sure whether we should really insist on this"
287     self optional: #toBeSpecified assert: 9223372036854775807 equals:
-1 >>> 1.
288     self optional: #toBeSpecified assert: 9223372036854775296 equals:
-1024 >>> 1.
289 )
290
291 testMin = (
292     "We need to test numbers that are 64bit or less, larger than 64bit,
293     positive, and negative"
294     | big small |
295     big := #(1 100 9223372036854775807 922337203685477580700
296             -50 -2147483648 922337203685477580700
-922337203685477580700
297             922337203685477580700).
298     small := #(0 52 9223372036854775296 922337203685477529600
299                -51 -2147483650 1
-922337203685477580701
300                -922337203685477580701).
301
302     big doIndexes: [:i |
303         self assert: (small at: i) equals: ((big at: i) min: (small at:
i)).
304         self assert: (small at: i) equals: ((small at: i) min: (big at:
i))] ]
305 )
306
307 testMax = (
308     "We need to test numbers that are 64bit or less, larger than 64bit,
309     positive, and negative"
310     | big small |
311     big := #(1 100 9223372036854775807 922337203685477580700
312             -50 -2147483648 922337203685477580700
-922337203685477580700
313             922337203685477580700).
314     small := #(0 52 9223372036854775296 922337203685477529600
315                -51 -2147483650 1
-922337203685477580701
316                -922337203685477580701).
317     big doIndexes: [:i |
318         self assert: (big at: i) equals: ((big at: i) max: (small at:
i)).
319         self assert: (big at: i) equals: ((small at: i) max: (big at:
i))] ]
320 )
321 )
322

```

```

1 "
2
3 $Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2007-2013 see AUTHORS file
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7 http://www.hpi.uni-potsdam.de/swa/
8
9 Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 Integer4Test = TestCase (
29
30   testEqualityAndIdentity = (
31     | a b |
32     a := 42.
33     b := 42.
34
35     self assert: a = b    description: 'Integers are equal based on their
value'.
36     self assert: a == b  description: 'Integers do not have pointer/
reference equality. It is also supposed to be value equality'.
37
38     "Sometimes it can be hard to implement efficiently, but it SHOULD
really
39     be true for all values of integers."
40     a := 1 << 30.  b := 1 << 30.
41     self optional: #integerIdentity assert: a is: b.
42
43     a := 1 << 32.  b := 1 << 32.
44     self optional: #integerIdentity assert: a is: b.
45
46     a := 1 << 60.  b := 1 << 60.
47     self optional: #integerIdentity assert: a is: b.
48   )
49
50   testClassAndValueRanges = (

```

```

51     | i |
52     self assert: Integer equals: -42 class.
53     self assert: Integer equals: 0 class.
54     self assert: Integer equals: 23 class.
55     self assert: Integer equals: 1073741823 class.
56     self assert: Integer equals: 1073741824 class.
57
58     "Let's test for size behavior and corresponding class"
59     i := 1 << 30.
60     self assert: Integer equals: i class.
61     self assert: i > 0 description: 'should not overflow'.
62     self assert: '1073741824' equals: i asString.
63
64     i := 1 << 32.
65     self assert: Integer equals: i class.
66     self assert: i > 0 description: 'should not overflow'.
67     self assert: '4294967296' equals: i asString.
68
69     i := 1 << 60.
70     self assert: Integer equals: i class.
71     self assert: i > 0 description: 'should not overflow'.
72     self assert: '1152921504606846976' equals: i asString.
73
74     i := 1 << 70.
75     self assert: Integer equals: i class.
76     self assert: i > 0 description: 'should not overflow'.
77     self optional: #bigIntShifts assert: '1180591620717411303424' equals:
i asString.
78
79     i := -1 << 30.
80     self assert: Integer equals: i class.
81     self assert: i < 0 description: 'should not underflow'.
82     self assert: '-1073741824' equals: i asString.
83
84     i := -1 << 32.
85     self assert: Integer equals: i class.
86     self assert: i < 0 description: 'should not underflow'.
87     self assert: '-4294967296' equals: i asString.
88
89     i := -1 << 60.
90     self assert: Integer equals: i class.
91     self assert: i < 0 description: 'should not underflow'.
92     self assert: '-1152921504606846976' equals: i asString.
93
94     i := -1 << 70.
95     self assert: Integer equals: i class.
96     self assert: i < 0 description: 'should not underflow'.
97     self optional: #bigIntShifts assert: '-1180591620717411303424'
equals: i asString.
98 )
99
100 testStringConversion = (
101     self assert: '0' equals: ( 0 asString).
102     self assert: '1' equals: ( 1 asString).
103     self assert: '2' equals: ( 2 asString).
104     self assert: '-1' equals: (-1 asString).
105     self assert: '-2' equals: (-2 asString).
106
107     self assert: 42 equals: '42' asInteger.
108     self assert: -2 equals: '-2' asInteger.
109 )

```

```

110
111     testIntegerLiterals = (
112         "Make sure the parser reads literals correctly. So, check some basic
properties"
113         self assert: 2 / 2
equals: 1.
114         self assert: 50 + 50
equals: 100.
115         self assert: 92233720368 * 100000000 + 54775807 equals:
9223372036854775807.
116         self assert: 92233720368 * 100000000 + 54775807 * 100 equals:
922337203685477580700.
117         self assert: 50 - 100
equals: -50.
118         self assert: 21474 * -100000 - 83648
equals: -2147483648.
119         self assert: 92233720368 * 100000000 + 54775807 * -100 equals:
-922337203685477580700.
120     )
121
122     testFromString = (
123         self assert: 1 equals: (Integer
fromString: '1').
124         self assert: 100 equals: (Integer
fromString: '100').
125         self assert: 9223372036854775807 equals: (Integer fromString:
'9223372036854775807').
126         self assert: 922337203685477580700 equals: (Integer fromString:
'922337203685477580700').
127         self assert: -50 equals: (Integer
fromString: '-50').
128         self assert: -2147483648 equals: (Integer
fromString: '-2147483648').
129         self assert: -922337203685477580700 equals: (Integer fromString:
'-922337203685477580700').
130     )
131
132     testRangeBorders = (
133         self assert: '536870911' equals: 536870911 asString.
134         self assert: '536870912' equals: 536870912 asString.
135         self assert: '536870913' equals: 536870913 asString.
136         self assert: '1073741823' equals: 1073741823 asString.
137         self assert: '1073741824' equals: 1073741824 asString.
138         self assert: '1073741825' equals: 1073741825 asString.
139         self assert: '2147483647' equals: 2147483647 asString.
140         self assert: '-536870911' equals: -536870911 asString.
141         self assert: '-536870912' equals: -536870912 asString.
142         self assert: '-536870913' equals: -536870913 asString.
143         self assert: '-1073741823' equals: -1073741823 asString.
144         self assert: '-1073741824' equals: -1073741824 asString.
145         self assert: '-1073741825' equals: -1073741825 asString.
146         self assert: '-2147483647' equals: -2147483647 asString.
147         self assert: '-2147483648' equals: -2147483648 asString.
148     )
149
150     testComparisons = (
151         self assert: ( 9 = 9 ).
152         self deny: ( 1 = 2 ).
153         self deny: ( 0 < 0 ).
154         self assert: ( 1 < 2 ).
155         self deny: ( 2 < 1 ).

```

```

156     self assert: (-3 < 2).
157     self deny: ( 3 < -2).
158     self deny: ( 0 > 0).
159     self deny: ( 1 > 2).
160     self assert: ( 2 > 1).
161     self deny: (-3 > 2).
162     self assert: ( 3 > -2).
163     self assert: ( 4 >= 3).
164     self assert: ( 3 >= 3).
165     self deny: ( 2 >= 3).
166     self assert: ( 2 <= 4).
167     self assert: ( 3 <= 3).
168     self deny: ( 4 <= 3).
169 )
170
171 testAddition = (
172     self assert: 0 equals: ( 0+0).
173     self assert: 1 equals: ( 1+0).
174     self assert: 1 equals: ( 0+1).
175     self assert: 2 equals: ( 1+1).
176     self assert: 0 equals: (-1+1).
177     self assert: 1 equals: (-1+2).
178 )
179
180 testSubtraction = (
181     self assert: 1 equals: (1-0).
182     self assert: -1 equals: (0-1).
183     self assert: 1 equals: (2-1).
184 )
185
186 testMultiplication = (
187     self assert: 0 equals: ( 1* 0).
188     self assert: -1 equals: (-1* 1).
189     self assert: -25 equals: ( 5* -5).
190     self assert: 12 equals: (-3* -4).
191 )
192
193 testDivision = (
194     self assert: 1 equals: ( 1/ 1).
195     self assert: 1 equals: ( 3/ 2).
196     self assert: -2 equals: ( 4/ -2).
197     self assert: -2 equals: (-6/ 3).
198     self assert: 3 equals: (-12/ -4).
199 )
200
201 testDouble = (
202     self assert: 6 equals: ( 36// 6).
203     self assert: -5 equals: (-10// 2).
204     self assert: -4 equals: ( 20// -5).
205     self assert: 1 equals: ( -5// -5).
206 )
207
208 testModulo = (
209     self assert: 1 equals: ( 10 % 3).
210     self assert: -2 equals: ( 10 % -3).
211     self assert: 2 equals: (-10 % 3).
212     self assert: -1 equals: (-10 % -3).
213     self assert: 0 equals: ( 10 % 5).
214
215     self assert: 1 equals: ( 10 rem: 3).
216     self assert: 1 equals: ( 10 rem: -3).

```



```

217     self assert: -1 equals: (-10 rem: 3).
218     self assert: -1 equals: (-10 rem: -3).
219     self assert: 0 equals: (10 rem: 5).
220 )
221
222 testAbsoluteValue = (
223     self assert: 4 equals: -4 abs.
224     self assert: 4 equals: 4 abs.
225
226     self assert: 9223372036854775296 equals: -9223372036854775296 abs.
227     self assert: 9223372036854775296 equals: 9223372036854775296 abs.
228 )
229
230 testNegated = (
231     self assert: -23 equals: (23 negated).
232     self assert: 23 equals: (-23 negated).
233 )
234
235 testSquareRoot = (
236     self assert: 5 equals: (25 sqrt).
237     self assert: Integer equals: (25 sqrt class).
238 )
239
240 testAnd = (
241     self assert: 0 equals: (2 & 1).
242     self assert: 2 equals: (2 & 2).
243 )
244
245 testBitXor = (
246     self assert: 0 equals: (1 bitXor: 1).
247     self assert: 3 equals: (2 bitXor: 1).
248 )
249
250 testAs32BitUnsignedValue = (
251     self assert: 1 << 1 equals: (1 << 1) as32BitUnsignedValue.
252     self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
253     self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
254     self assert: 0 equals: (1 << 32) as32BitUnsignedValue.
255     self assert: 4294967295 equals: -1 as32BitUnsignedValue.
256     self assert: 512 equals: -9223372036854775296
as32BitUnsignedValue.
257     self assert: 4294966784 equals: 9223372036854775296
as32BitUnsignedValue.
258 )
259
260 testAs32BitSignedValue = (
261     self assert: 1 << 1 equals: (1 << 1) as32BitSignedValue.
262     self assert: 1 << 10 equals: (1 << 10) as32BitSignedValue.
263     self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
264     self assert: 0 equals: (1 << 32) as32BitSignedValue.
265
266     self assert: 512 equals: -9223372036854775296 as32BitSignedValue.
267     self assert: -512 equals: 9223372036854775296 as32BitSignedValue.
268 )
269
270 testAsDouble = (
271     self assert: 0.0 equals: 0 asDouble.
272     self assert: Double is: 0 asDouble class.
273
274     self assert: 2147483648.0 equals: 2147483648 asDouble.
275     self assert: Double is: 2147483648 asDouble class.

```

```

276
277     self assert: -2147483648.0 equals: -2147483648 asDouble.
278     self assert: Double is: -2147483648 asDouble class.
279 )
280
281 testUnsignedRightShift = (
282     self assert: 0 equals: 1 >>> 1.
283     self assert: 512 equals: 1024 >>> 1.
284     self assert: 127 equals: 1023 >>> 3.
285
286     "not sure whether we should really insist on this"
287     self optional: #toBeSpecified assert: 9223372036854775807 equals:
-1 >>> 1.
288     self optional: #toBeSpecified assert: 9223372036854775296 equals:
-1024 >>> 1.
289 )
290
291 testMin = (
292     "We need to test numbers that are 64bit or less, larger than 64bit,
293     positive, and negative"
294     | big small |
295     big := #(1 100 9223372036854775807 922337203685477580700
296             -50 -2147483648 922337203685477580700
-922337203685477580700
297             922337203685477580700).
298     small := #(0 52 9223372036854775296 922337203685477529600
299               -51 -2147483650 1
-922337203685477580701
300               -922337203685477580701).
301
302     big doIndexes: [:i |
303         self assert: (small at: i) equals: ((big at: i) min: (small at:
i)).
304         self assert: (small at: i) equals: ((small at: i) min: (big at:
i))] ]
305 )
306
307 testMax = (
308     "We need to test numbers that are 64bit or less, larger than 64bit,
309     positive, and negative"
310     | big small |
311     big := #(1 100 9223372036854775807 922337203685477580700
312             -50 -2147483648 922337203685477580700
-922337203685477580700
313             922337203685477580700).
314     small := #(0 52 9223372036854775296 922337203685477529600
315               -51 -2147483650 1
-922337203685477580701
316               -922337203685477580701).
317     big doIndexes: [:i |
318         self assert: (big at: i) equals: ((big at: i) max: (small at:
i)).
319         self assert: (big at: i) equals: ((small at: i) max: (big at:
i))] ]
320 )
321 )
322

```

```

1 "
2
3 $Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2007-2013 see AUTHORS file
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7 http://www.hpi.uni-potsdam.de/swa/
8
9 Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 Integer5Test = TestCase (
29
30   testEqualityAndIdentity = (
31     | a b |
32     a := 42.
33     b := 42.
34
35     self assert: a = b    description: 'Integers are equal based on their
value'.
36     self assert: a == b  description: 'Integers do not have pointer/
reference equality. It is also supposed to be value equality'.
37
38     "Sometimes it can be hard to implement efficiently, but it SHOULD
really
39     be true for all values of integers."
40     a := 1 << 30.  b := 1 << 30.
41     self optional: #integerIdentity assert: a is: b.
42
43     a := 1 << 32.  b := 1 << 32.
44     self optional: #integerIdentity assert: a is: b.
45
46     a := 1 << 60.  b := 1 << 60.
47     self optional: #integerIdentity assert: a is: b.
48   )
49
50   testClassAndValueRanges = (

```

```

51     | i |
52     self assert: Integer equals: -42 class.
53     self assert: Integer equals: 0 class.
54     self assert: Integer equals: 23 class.
55     self assert: Integer equals: 1073741823 class.
56     self assert: Integer equals: 1073741824 class.
57
58     "Let's test for size behavior and corresponding class"
59     i := 1 << 30.
60     self assert: Integer equals: i class.
61     self assert: i > 0 description: 'should not overflow'.
62     self assert: '1073741824' equals: i asString.
63
64     i := 1 << 32.
65     self assert: Integer equals: i class.
66     self assert: i > 0 description: 'should not overflow'.
67     self assert: '4294967296' equals: i asString.
68
69     i := 1 << 60.
70     self assert: Integer equals: i class.
71     self assert: i > 0 description: 'should not overflow'.
72     self assert: '1152921504606846976' equals: i asString.
73
74     i := 1 << 70.
75     self assert: Integer equals: i class.
76     self assert: i > 0 description: 'should not overflow'.
77     self optional: #bigIntShifts assert: '1180591620717411303424' equals:
i asString.
78
79     i := -1 << 30.
80     self assert: Integer equals: i class.
81     self assert: i < 0 description: 'should not underflow'.
82     self assert: '-1073741824' equals: i asString.
83
84     i := -1 << 32.
85     self assert: Integer equals: i class.
86     self assert: i < 0 description: 'should not underflow'.
87     self assert: '-4294967296' equals: i asString.
88
89     i := -1 << 60.
90     self assert: Integer equals: i class.
91     self assert: i < 0 description: 'should not underflow'.
92     self assert: '-1152921504606846976' equals: i asString.
93
94     i := -1 << 70.
95     self assert: Integer equals: i class.
96     self assert: i < 0 description: 'should not underflow'.
97     self optional: #bigIntShifts assert: '-1180591620717411303424'
equals: i asString.
98 )
99
100 testStringConversion = (
101     self assert: '0' equals: ( 0 asString).
102     self assert: '1' equals: ( 1 asString).
103     self assert: '2' equals: ( 2 asString).
104     self assert: '-1' equals: (-1 asString).
105     self assert: '-2' equals: (-2 asString).
106
107     self assert: 42 equals: '42' asInteger.
108     self assert: -2 equals: '-2' asInteger.
109 )

```

```

110
111     testIntegerLiterals = (
112         "Make sure the parser reads literals correctly. So, check some basic
properties"
113         self assert: 2 / 2
equals: 1.
114         self assert: 50 + 50
equals: 100.
115         self assert: 92233720368 * 100000000 + 54775807 equals:
9223372036854775807.
116         self assert: 92233720368 * 100000000 + 54775807 * 100 equals:
922337203685477580700.
117         self assert: 50 - 100
equals: -50.
118         self assert: 21474 * -100000 - 83648
equals: -2147483648.
119         self assert: 92233720368 * 100000000 + 54775807 * -100 equals:
-922337203685477580700.
120     )
121
122     testFromString = (
123         self assert: 1 equals: (Integer
fromString: '1').
124         self assert: 100 equals: (Integer
fromString: '100').
125         self assert: 9223372036854775807 equals: (Integer fromString:
'9223372036854775807').
126         self assert: 922337203685477580700 equals: (Integer fromString:
'922337203685477580700').
127         self assert: -50 equals: (Integer
fromString: '-50').
128         self assert: -2147483648 equals: (Integer
fromString: '-2147483648').
129         self assert: -922337203685477580700 equals: (Integer fromString:
'-922337203685477580700').
130     )
131
132     testRangeBorders = (
133         self assert: '536870911' equals: 536870911 asString.
134         self assert: '536870912' equals: 536870912 asString.
135         self assert: '536870913' equals: 536870913 asString.
136         self assert: '1073741823' equals: 1073741823 asString.
137         self assert: '1073741824' equals: 1073741824 asString.
138         self assert: '1073741825' equals: 1073741825 asString.
139         self assert: '2147483647' equals: 2147483647 asString.
140         self assert: '-536870911' equals: -536870911 asString.
141         self assert: '-536870912' equals: -536870912 asString.
142         self assert: '-536870913' equals: -536870913 asString.
143         self assert: '-1073741823' equals: -1073741823 asString.
144         self assert: '-1073741824' equals: -1073741824 asString.
145         self assert: '-1073741825' equals: -1073741825 asString.
146         self assert: '-2147483647' equals: -2147483647 asString.
147         self assert: '-2147483648' equals: -2147483648 asString.
148     )
149
150     testComparisons = (
151         self assert: ( 9 = 9 ).
152         self deny: ( 1 = 2 ).
153         self deny: ( 0 < 0 ).
154         self assert: ( 1 < 2 ).
155         self deny: ( 2 < 1 ).

```

```

156     self assert: (-3 < 2).
157     self deny: ( 3 < -2).
158     self deny: ( 0 > 0).
159     self deny: ( 1 > 2).
160     self assert: ( 2 > 1).
161     self deny: (-3 > 2).
162     self assert: ( 3 > -2).
163     self assert: ( 4 >= 3).
164     self assert: ( 3 >= 3).
165     self deny: ( 2 >= 3).
166     self assert: ( 2 <= 4).
167     self assert: ( 3 <= 3).
168     self deny: ( 4 <= 3).
169 )
170
171 testAddition = (
172     self assert: 0 equals: ( 0+0).
173     self assert: 1 equals: ( 1+0).
174     self assert: 1 equals: ( 0+1).
175     self assert: 2 equals: ( 1+1).
176     self assert: 0 equals: (-1+1).
177     self assert: 1 equals: (-1+2).
178 )
179
180 testSubtraction = (
181     self assert: 1 equals: (1-0).
182     self assert: -1 equals: (0-1).
183     self assert: 1 equals: (2-1).
184 )
185
186 testMultiplication = (
187     self assert: 0 equals: ( 1* 0).
188     self assert: -1 equals: (-1* 1).
189     self assert: -25 equals: ( 5* -5).
190     self assert: 12 equals: (-3* -4).
191 )
192
193 testDivision = (
194     self assert: 1 equals: ( 1/ 1).
195     self assert: 1 equals: ( 3/ 2).
196     self assert: -2 equals: ( 4/ -2).
197     self assert: -2 equals: (-6/ 3).
198     self assert: 3 equals: (-12/ -4).
199 )
200
201 testDouble = (
202     self assert: 6 equals: ( 36// 6).
203     self assert: -5 equals: (-10// 2).
204     self assert: -4 equals: ( 20// -5).
205     self assert: 1 equals: ( -5// -5).
206 )
207
208 testModulo = (
209     self assert: 1 equals: ( 10 % 3).
210     self assert: -2 equals: ( 10 % -3).
211     self assert: 2 equals: (-10 % 3).
212     self assert: -1 equals: (-10 % -3).
213     self assert: 0 equals: ( 10 % 5).
214
215     self assert: 1 equals: ( 10 rem: 3).
216     self assert: 1 equals: ( 10 rem: -3).

```

```

217     self assert: -1 equals: (-10 rem: 3).
218     self assert: -1 equals: (-10 rem: -3).
219     self assert: 0 equals: (10 rem: 5).
220 )
221
222 testAbsoluteValue = (
223     self assert: 4 equals: -4 abs.
224     self assert: 4 equals: 4 abs.
225
226     self assert: 9223372036854775296 equals: -9223372036854775296 abs.
227     self assert: 9223372036854775296 equals: 9223372036854775296 abs.
228 )
229
230 testNegated = (
231     self assert: -23 equals: (23 negated).
232     self assert: 23 equals: (-23 negated).
233 )
234
235 testSquareRoot = (
236     self assert: 5 equals: (25 sqrt).
237     self assert: Integer equals: (25 sqrt class).
238 )
239
240 testAnd = (
241     self assert: 0 equals: (2 & 1).
242     self assert: 2 equals: (2 & 2).
243 )
244
245 testBitXor = (
246     self assert: 0 equals: (1 bitXor: 1).
247     self assert: 3 equals: (2 bitXor: 1).
248 )
249
250 testAs32BitUnsignedValue = (
251     self assert: 1 << 1 equals: (1 << 1) as32BitUnsignedValue.
252     self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
253     self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
254     self assert: 0 equals: (1 << 32) as32BitUnsignedValue.
255     self assert: 4294967295 equals: -1 as32BitUnsignedValue.
256     self assert: 512 equals: -9223372036854775296
as32BitUnsignedValue.
257     self assert: 4294966784 equals: 9223372036854775296
as32BitUnsignedValue.
258 )
259
260 testAs32BitSignedValue = (
261     self assert: 1 << 1 equals: (1 << 1) as32BitSignedValue.
262     self assert: 1 << 10 equals: (1 << 10) as32BitSignedValue.
263     self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
264     self assert: 0 equals: (1 << 32) as32BitSignedValue.
265
266     self assert: 512 equals: -9223372036854775296 as32BitSignedValue.
267     self assert: -512 equals: 9223372036854775296 as32BitSignedValue.
268 )
269
270 testAsDouble = (
271     self assert: 0.0 equals: 0 asDouble.
272     self assert: Double is: 0 asDouble class.
273
274     self assert: 2147483648.0 equals: 2147483648 asDouble.
275     self assert: Double is: 2147483648 asDouble class.

```

```

276
277     self assert: -2147483648.0 equals: -2147483648 asDouble.
278     self assert: Double is: -2147483648 asDouble class.
279 )
280
281 testUnsignedRightShift = (
282     self assert: 0 equals: 1 >>> 1.
283     self assert: 512 equals: 1024 >>> 1.
284     self assert: 127 equals: 1023 >>> 3.
285
286     "not sure whether we should really insist on this"
287     self optional: #toBeSpecified assert: 9223372036854775807 equals:
-1 >>> 1.
288     self optional: #toBeSpecified assert: 9223372036854775296 equals:
-1024 >>> 1.
289 )
290
291 testMin = (
292     "We need to test numbers that are 64bit or less, larger than 64bit,
293     positive, and negative"
294     | big small |
295     big := #(1 100 9223372036854775807 922337203685477580700
296             -50 -2147483648 922337203685477580700
-922337203685477580700
297             922337203685477580700).
298     small := #(0 52 9223372036854775296 922337203685477529600
299               -51 -2147483650 1
-922337203685477580701
300               -922337203685477580701).
301
302     big doIndexes: [:i |
303         self assert: (small at: i) equals: ((big at: i) min: (small at:
i)).
304         self assert: (small at: i) equals: ((small at: i) min: (big at:
i))] ]
305 )
306
307 testMax = (
308     "We need to test numbers that are 64bit or less, larger than 64bit,
309     positive, and negative"
310     | big small |
311     big := #(1 100 9223372036854775807 922337203685477580700
312             -50 -2147483648 922337203685477580700
-922337203685477580700
313             922337203685477580700).
314     small := #(0 52 9223372036854775296 922337203685477529600
315               -51 -2147483650 1
-922337203685477580701
316               -922337203685477580701).
317     big doIndexes: [:i |
318         self assert: (big at: i) equals: ((big at: i) max: (small at:
i)).
319         self assert: (big at: i) equals: ((small at: i) max: (big at:
i))] ]
320 )
321 )
322

```



```

1 "
2
3 $Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2007-2013 see AUTHORS file
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7 http://www.hpi.uni-potsdam.de/swa/
8
9 Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 IntegerTest = TestCase (
29
30   testEqualityAndIdentity = (
31     | a b |
32     a := 42.
33     b := 42.
34
35     self assert: a = b    description: 'Integers are equal based on their
value'.
36     self assert: a == b  description: 'Integers do not have pointer/
reference equality. It is also supposed to be value equality'.
37
38     "Sometimes it can be hard to implement efficiently, but it SHOULD
really
39     be true for all values of integers."
40     a := 1 << 30.  b := 1 << 30.
41     self optional: #integerIdentity assert: a is: b.
42
43     a := 1 << 32.  b := 1 << 32.
44     self optional: #integerIdentity assert: a is: b.
45
46     a := 1 << 60.  b := 1 << 60.
47     self optional: #integerIdentity assert: a is: b.
48   )
49
50   testClassAndValueRanges = (

```

```

51     | i |
52     self assert: Integer equals: -42 class.
53     self assert: Integer equals: 0 class.
54     self assert: Integer equals: 23 class.
55     self assert: Integer equals: 1073741823 class.
56     self assert: Integer equals: 1073741824 class.
57
58     "Let's test for size behavior and corresponding class"
59     i := 1 << 30.
60     self assert: Integer equals: i class.
61     self assert: i > 0 description: 'should not overflow'.
62     self assert: '1073741824' equals: i asString.
63
64     i := 1 << 32.
65     self assert: Integer equals: i class.
66     self assert: i > 0 description: 'should not overflow'.
67     self assert: '4294967296' equals: i asString.
68
69     i := 1 << 60.
70     self assert: Integer equals: i class.
71     self assert: i > 0 description: 'should not overflow'.
72     self assert: '1152921504606846976' equals: i asString.
73
74     i := 1 << 70.
75     self assert: Integer equals: i class.
76     self assert: i > 0 description: 'should not overflow'.
77     self optional: #bigIntShifts assert: '1180591620717411303424' equals:
i asString.
78
79     i := -1 << 30.
80     self assert: Integer equals: i class.
81     self assert: i < 0 description: 'should not underflow'.
82     self assert: '-1073741824' equals: i asString.
83
84     i := -1 << 32.
85     self assert: Integer equals: i class.
86     self assert: i < 0 description: 'should not underflow'.
87     self assert: '-4294967296' equals: i asString.
88
89     i := -1 << 60.
90     self assert: Integer equals: i class.
91     self assert: i < 0 description: 'should not underflow'.
92     self assert: '-1152921504606846976' equals: i asString.
93
94     i := -1 << 70.
95     self assert: Integer equals: i class.
96     self assert: i < 0 description: 'should not underflow'.
97     self optional: #bigIntShifts assert: '-1180591620717411303424'
equals: i asString.
98 )
99
100 testStringConversion = (
101     self assert: '0' equals: (0 asString).
102     self assert: '1' equals: (1 asString).
103     self assert: '2' equals: (2 asString).
104     self assert: '-1' equals: (-1 asString).
105     self assert: '-2' equals: (-2 asString).
106
107     self assert: 42 equals: '42' asInteger.
108     self assert: -2 equals: '-2' asInteger.
109 )

```

```

110
111     testIntegerLiterals = (
112         "Make sure the parser reads literals correctly. So, check some basic
properties"
113         self assert: 2 / 2
equals: 1.
114         self assert: 50 + 50
equals: 100.
115         self assert: 92233720368 * 100000000 + 54775807 equals:
9223372036854775807.
116         self assert: 92233720368 * 100000000 + 54775807 * 100 equals:
922337203685477580700.
117         self assert: 50 - 100
equals: -50.
118         self assert: 21474 * -100000 - 83648
equals: -2147483648.
119         self assert: 92233720368 * 100000000 + 54775807 * -100 equals:
-922337203685477580700.
120     )
121
122     testFromString = (
123         self assert: 1 equals: (Integer
fromString: '1').
124         self assert: 100 equals: (Integer
fromString: '100').
125         self assert: 9223372036854775807 equals: (Integer fromString:
'9223372036854775807').
126         self assert: 922337203685477580700 equals: (Integer fromString:
'922337203685477580700').
127         self assert: -50 equals: (Integer
fromString: '-50').
128         self assert: -2147483648 equals: (Integer
fromString: '-2147483648').
129         self assert: -922337203685477580700 equals: (Integer fromString:
'-922337203685477580700').
130     )
131
132     testRangeBorders = (
133         self assert: '536870911' equals: 536870911 asString.
134         self assert: '536870912' equals: 536870912 asString.
135         self assert: '536870913' equals: 536870913 asString.
136         self assert: '1073741823' equals: 1073741823 asString.
137         self assert: '1073741824' equals: 1073741824 asString.
138         self assert: '1073741825' equals: 1073741825 asString.
139         self assert: '2147483647' equals: 2147483647 asString.
140         self assert: '-536870911' equals: -536870911 asString.
141         self assert: '-536870912' equals: -536870912 asString.
142         self assert: '-536870913' equals: -536870913 asString.
143         self assert: '-1073741823' equals: -1073741823 asString.
144         self assert: '-1073741824' equals: -1073741824 asString.
145         self assert: '-1073741825' equals: -1073741825 asString.
146         self assert: '-2147483647' equals: -2147483647 asString.
147         self assert: '-2147483648' equals: -2147483648 asString.
148     )
149
150     testComparisons = (
151         self assert: ( 9 = 9 ).
152         self deny: ( 1 = 2 ).
153         self deny: ( 0 < 0 ).
154         self assert: ( 1 < 2 ).
155         self deny: ( 2 < 1 ).

```

```

156     self assert: (-3 < 2).
157     self deny: ( 3 < -2).
158     self deny: ( 0 > 0).
159     self deny: ( 1 > 2).
160     self assert: ( 2 > 1).
161     self deny: (-3 > 2).
162     self assert: ( 3 > -2).
163     self assert: ( 4 >= 3).
164     self assert: ( 3 >= 3).
165     self deny: ( 2 >= 3).
166     self assert: ( 2 <= 4).
167     self assert: ( 3 <= 3).
168     self deny: ( 4 <= 3).
169 )
170
171 testAddition = (
172     self assert: 0 equals: ( 0+0).
173     self assert: 1 equals: ( 1+0).
174     self assert: 1 equals: ( 0+1).
175     self assert: 2 equals: ( 1+1).
176     self assert: 0 equals: (-1+1).
177     self assert: 1 equals: (-1+2).
178 )
179
180 testSubtraction = (
181     self assert: 1 equals: (1-0).
182     self assert: -1 equals: (0-1).
183     self assert: 1 equals: (2-1).
184 )
185
186 testMultiplication = (
187     self assert: 0 equals: ( 1* 0).
188     self assert: -1 equals: (-1* 1).
189     self assert: -25 equals: ( 5* -5).
190     self assert: 12 equals: (-3* -4).
191 )
192
193 testDivision = (
194     self assert: 1 equals: ( 1/ 1).
195     self assert: 1 equals: ( 3/ 2).
196     self assert: -2 equals: ( 4/ -2).
197     self assert: -2 equals: (-6/ 3).
198     self assert: 3 equals: (-12/ -4).
199 )
200
201 testDouble = (
202     self assert: 6 equals: ( 36// 6).
203     self assert: -5 equals: (-10// 2).
204     self assert: -4 equals: ( 20// -5).
205     self assert: 1 equals: ( -5// -5).
206 )
207
208 testModulo = (
209     self assert: 1 equals: ( 10 % 3).
210     self assert: -2 equals: ( 10 % -3).
211     self assert: 2 equals: (-10 % 3).
212     self assert: -1 equals: (-10 % -3).
213     self assert: 0 equals: ( 10 % 5).
214
215     self assert: 1 equals: ( 10 rem: 3).
216     self assert: 1 equals: ( 10 rem: -3).

```

```

217     self assert: -1 equals: (-10 rem: 3).
218     self assert: -1 equals: (-10 rem: -3).
219     self assert: 0 equals: (10 rem: 5).
220 )
221
222 testAbsoluteValue = (
223     self assert: 4 equals: -4 abs.
224     self assert: 4 equals: 4 abs.
225
226     self assert: 9223372036854775296 equals: -9223372036854775296 abs.
227     self assert: 9223372036854775296 equals: 9223372036854775296 abs.
228 )
229
230 testNegated = (
231     self assert: -23 equals: (23 negated).
232     self assert: 23 equals: (-23 negated).
233 )
234
235 testSquareRoot = (
236     self assert: 5 equals: (25 sqrt).
237     self assert: Integer equals: (25 sqrt class).
238 )
239
240 testAnd = (
241     self assert: 0 equals: (2 & 1).
242     self assert: 2 equals: (2 & 2).
243 )
244
245 testBitXor = (
246     self assert: 0 equals: (1 bitXor: 1).
247     self assert: 3 equals: (2 bitXor: 1).
248 )
249
250 testAs32BitUnsignedValue = (
251     self assert: 1 << 1 equals: (1 << 1) as32BitUnsignedValue.
252     self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
253     self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
254     self assert: 0 equals: (1 << 32) as32BitUnsignedValue.
255     self assert: 4294967295 equals: -1 as32BitUnsignedValue.
256     self assert: 512 equals: -9223372036854775296
as32BitUnsignedValue.
257     self assert: 4294966784 equals: 9223372036854775296
as32BitUnsignedValue.
258 )
259
260 testAs32BitSignedValue = (
261     self assert: 1 << 1 equals: (1 << 1) as32BitSignedValue.
262     self assert: 1 << 10 equals: (1 << 10) as32BitSignedValue.
263     self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
264     self assert: 0 equals: (1 << 32) as32BitSignedValue.
265
266     self assert: 512 equals: -9223372036854775296 as32BitSignedValue.
267     self assert: -512 equals: 9223372036854775296 as32BitSignedValue.
268 )
269
270 testAsDouble = (
271     self assert: 0.0 equals: 0 asDouble.
272     self assert: Double is: 0 asDouble class.
273
274     self assert: 2147483648.0 equals: 2147483648 asDouble.
275     self assert: Double is: 2147483648 asDouble class.

```

```

276
277     self assert: -2147483648.0 equals: -2147483648 asDouble.
278     self assert: Double      is:      -2147483648 asDouble class.
279 )
280
281 testUnsignedRightShift = (
282     self assert: 0 equals: 1 >>> 1.
283     self assert: 512 equals: 1024 >>> 1.
284     self assert: 127 equals: 1023 >>> 3.
285
286     "not sure whether we should really insist on this"
287     self optional: #toBeSpecified assert: 9223372036854775807 equals:
-1 >>> 1.
288     self optional: #toBeSpecified assert: 9223372036854775296 equals:
-1024 >>> 1.
289 )
290
291 testMin = (
292     "We need to test numbers that are 64bit or less, larger than 64bit,
293     positive, and negative"
294     | big small |
295     big := #(1 100 9223372036854775807 922337203685477580700
296             -50 -2147483648 922337203685477580700
-922337203685477580700
297             922337203685477580700).
298     small := #(0 52 9223372036854775296 922337203685477529600
299               -51 -2147483650 1
-922337203685477580701
300               -922337203685477580701).
301
302     big doIndexes: [:i |
303         self assert: (small at: i) equals: ((big at: i) min: (small at:
i)).
304         self assert: (small at: i) equals: ((small at: i) min: (big at:
i))] ]
305 )
306
307 testMax = (
308     "We need to test numbers that are 64bit or less, larger than 64bit,
309     positive, and negative"
310     | big small |
311     big := #(1 100 9223372036854775807 922337203685477580700
312             -50 -2147483648 922337203685477580700
-922337203685477580700
313             922337203685477580700).
314     small := #(0 52 9223372036854775296 922337203685477529600
315               -51 -2147483650 1
-922337203685477580701
316               -922337203685477580701).
317     big doIndexes: [:i |
318         self assert: (big at: i) equals: ((big at: i) max: (small at:
i)).
319         self assert: (big at: i) equals: ((small at: i) max: (big at:
i))] ]
320 )
321 )
322

```

```
1 "  
2  
3 $Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "... something just a bit complicated that tests iteration with  
27 blocks, so that we might fail here rather than when the other tests  
28 start, in case things are broken."  
29  
30 Preliminary2Test = TestCase (  
31  
32     testBasicSanity = (  
33         | sum |  
34         sum := 0.  
35         1, 2, 3 do: [ :i |  
36             sum := sum + i.  
37             i<2 ifTrue: [ sum := sum*2 ].  
38             i>2 ifFalse: [ sum := sum*2 ] ].  
39         self assert: 15 equals: sum  
40     )  
41  
42 )  
43
```

```
1 "  
2  
3 $Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "... something just a bit complicated that tests iteration with  
27 blocks, so that we might fail here rather than when the other tests  
28 start, in case things are broken."  
29  
30 Preliminary3Test = TestCase (  
31  
32     testBasicSanity = (  
33         | sum |  
34         sum := 0.  
35         1, 2, 3 do: [ :i |  
36             sum := sum + i.  
37             i<2 ifTrue: [ sum := sum*2 ].  
38             i>2 ifFalse: [ sum := sum*2 ] ].  
39         self assert: 15 equals: sum  
40     )  
41  
42 )  
43
```



```
1 "  
2  
3 $Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "... something just a bit complicated that tests iteration with  
27 blocks, so that we might fail here rather than when the other tests  
28 start, in case things are broken."  
29  
30 Preliminary4Test = TestCase (  
31  
32     testBasicSanity = (  
33         | sum |  
34         sum := 0.  
35         1, 2, 3 do: [ :i |  
36             sum := sum + i.  
37             i<2 ifTrue: [ sum := sum*2 ].  
38             i>2 ifFalse: [ sum := sum*2 ] ].  
39         self assert: 15 equals: sum  
40     )  
41  
42 )  
43
```

```
1 "  
2  
3 $Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "... something just a bit complicated that tests iteration with  
27 blocks, so that we might fail here rather than when the other tests  
28 start, in case things are broken."  
29  
30 Preliminary5Test = TestCase (  
31  
32     testBasicSanity = (  
33         | sum |  
34         sum := 0.  
35         1, 2, 3 do: [ :i |  
36             sum := sum + i.  
37             i<2 ifTrue: [ sum := sum*2 ].  
38             i>2 ifFalse: [ sum := sum*2 ] ].  
39         self assert: 15 equals: sum  
40     )  
41 )  
42 )  
43
```

```
1 "  
2  
3 $Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "... something just a bit complicated that tests iteration with  
27 blocks, so that we might fail here rather than when the other tests  
28 start, in case things are broken."  
29  
30 PreliminaryTest = TestCase (  
31  
32     testBasicSanity = (  
33         | sum |  
34         sum := 0.  
35         1, 2, 3 do: [ :i |  
36             sum := sum + i.  
37             i<2 ifTrue: [ sum := sum*2 ].  
38             i>2 ifFalse: [ sum := sum*2 ] ].  
39         self assert: 15 equals: sum  
40     )  
41  
42 )  
43
```

```
1 "  
2  
3 $Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Reflection2Test = TestCase (  
29   testRespondsTo = (  
30     self assert: (Object new respondsTo: #isNil).  
31     self assert: (23 respondsTo: #isNil).  
32     self assert: (23 respondsTo: #+).  
33   )  
34  
35   testMethods = (  
36     "First method in Object should be #class."  
37     self assert: #class equals: (Object methods at: 1) signature.  
38     self assert: (Object hasMethod: #==).  
39   )  
40  
41   testPerform = (  
42     | o |  
43     self assert: Integer equals: (23 perform: #class).  
44     self assert: (23 perform: #between:and: withArguments: (Array with: 22  
with: 24)).  
45  
46     o := SuperTest new.  
47     self assert: #super equals: (o perform: #something inSuperclass:  
SuperTestSuperClass).  
48  
49     "Trying to see whether the stack in bytecode-based SOMs works properly"  
50     self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a]  
ifFalse: [#b]).  
51
```

```

52     self assert: 28 equals: 5 + (23 perform: #value).
53 )
54
55 testInstVarAtAndPut = (
56     | tmp |
57     "Testing #at: and #at:put:"
58     tmp := Pair withKey: 3 andValue: 42.
59
60     self assert: tmp key equals: (tmp instVarAt: 1).
61
62     tmp instVarAt: 1 put: #foo.
63     self assert: #foo equals: tmp key.
64 )
65
66 testName = (
67     self assert: #Object equals: Object name.
68     self assert: #'Object class' equals: Object class name.
69     self assert: #Integer equals: 1 class name.
70 )
71
72 testAsString = (
73     self assert: 'Object' equals: Object asString.
74     self assert: 'Object class' equals: Object class asString.
75     self assert: 'Integer' equals: 1 class asString.
76 )
77
78 testSelectors = (
79     | sels |
80     sels := Object selectors.
81     self assert: 32 equals: sels length.
82
83     sels contains: #=.
84     self assert: (Object hasMethod: #=).
85
86     sels contains: #value.
87     self assert: (Object hasMethod: #value).
88
89     sels contains: #notNil.
90     self assert: (Object hasMethod: #notNil).
91 )
92 )
93

```

```
1 "  
2  
3 $Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Reflection3Test = TestCase (  
29   testRespondsTo = (  
30     self assert: (Object new respondsTo: #isNil).  
31     self assert: (23 respondsTo: #isNil).  
32     self assert: (23 respondsTo: #+).  
33   )  
34  
35   testMethods = (  
36     "First method in Object should be #class."  
37     self assert: #class equals: (Object methods at: 1) signature.  
38     self assert: (Object hasMethod: #==).  
39   )  
40  
41   testPerform = (  
42     | o |  
43     self assert: Integer equals: (23 perform: #class).  
44     self assert: (23 perform: #between:and: withArguments: (Array with: 22  
with: 24)).  
45  
46     o := SuperTest new.  
47     self assert: #super equals: (o perform: #something inSuperclass:  
SuperTestSuperClass).  
48  
49     "Trying to see whether the stack in bytecode-based SOMs works properly"  
50     self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a]  
ifFalse: [#b]).  
51
```

```

52     self assert: 28 equals: 5 + (23 perform: #value).
53 )
54
55 testInstVarAtAndPut = (
56     | tmp |
57     "Testing #at: and #at:put:"
58     tmp := Pair withKey: 3 andValue: 42.
59
60     self assert: tmp key equals: (tmp instVarAt: 1).
61
62     tmp instVarAt: 1 put: #foo.
63     self assert: #foo equals: tmp key.
64 )
65
66 testName = (
67     self assert: #Object equals: Object name.
68     self assert: #'Object class' equals: Object class name.
69     self assert: #Integer equals: 1 class name.
70 )
71
72 testAsString = (
73     self assert: 'Object' equals: Object asString.
74     self assert: 'Object class' equals: Object class asString.
75     self assert: 'Integer' equals: 1 class asString.
76 )
77
78 testSelectors = (
79     | sels |
80     sels := Object selectors.
81     self assert: 32 equals: sels length.
82
83     sels contains: #=.
84     self assert: (Object hasMethod: #=).
85
86     sels contains: #value.
87     self assert: (Object hasMethod: #value).
88
89     sels contains: #notNil.
90     self assert: (Object hasMethod: #notNil).
91 )
92 )
93

```

```
1 "  
2  
3 $Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Reflection4Test = TestCase (  
29   testRespondsTo = (  
30     self assert: (Object new respondsTo: #isNil).  
31     self assert: (23 respondsTo: #isNil).  
32     self assert: (23 respondsTo: #+).  
33   )  
34  
35   testMethods = (  
36     "First method in Object should be #class."  
37     self assert: #class equals: (Object methods at: 1) signature.  
38     self assert: (Object hasMethod: #==).  
39   )  
40  
41   testPerform = (  
42     | o |  
43     self assert: Integer equals: (23 perform: #class).  
44     self assert: (23 perform: #between:and: withArguments: (Array with: 22  
with: 24)).  
45  
46     o := SuperTest new.  
47     self assert: #super equals: (o perform: #something inSuperclass:  
SuperTestSuperClass).  
48  
49     "Trying to see whether the stack in bytecode-based SOMs works properly"  
50     self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a]  
ifFalse: [#b]).  
51
```



```

52     self assert: 28 equals: 5 + (23 perform: #value).
53 )
54
55 testInstVarAtAndPut = (
56     | tmp |
57     "Testing #at: and #at:put:"
58     tmp := Pair withKey: 3 andValue: 42.
59
60     self assert: tmp key equals: (tmp instVarAt: 1).
61
62     tmp instVarAt: 1 put: #foo.
63     self assert: #foo equals: tmp key.
64 )
65
66 testName = (
67     self assert: #Object equals: Object name.
68     self assert: #'Object class' equals: Object class name.
69     self assert: #Integer equals: 1 class name.
70 )
71
72 testAsString = (
73     self assert: 'Object' equals: Object asString.
74     self assert: 'Object class' equals: Object class asString.
75     self assert: 'Integer' equals: 1 class asString.
76 )
77
78 testSelectors = (
79     | sels |
80     sels := Object selectors.
81     self assert: 32 equals: sels length.
82
83     sels contains: #=.
84     self assert: (Object hasMethod: #=).
85
86     sels contains: #value.
87     self assert: (Object hasMethod: #value).
88
89     sels contains: #notNil.
90     self assert: (Object hasMethod: #notNil).
91 )
92 )
93

```

```
1 "  
2  
3 $Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Reflection5Test = TestCase (  
29   testRespondsTo = (  
30     self assert: (Object new respondsTo: #isNil).  
31     self assert: (23 respondsTo: #isNil).  
32     self assert: (23 respondsTo: #+).  
33   )  
34  
35   testMethods = (  
36     "First method in Object should be #class."  
37     self assert: #class equals: (Object methods at: 1) signature.  
38     self assert: (Object hasMethod: #==).  
39   )  
40  
41   testPerform = (  
42     | o |  
43     self assert: Integer equals: (23 perform: #class).  
44     self assert: (23 perform: #between:and: withArguments: (Array with: 22  
with: 24)).  
45  
46     o := SuperTest new.  
47     self assert: #super equals: (o perform: #something inSuperclass:  
SuperTestSuperClass).  
48  
49     "Trying to see whether the stack in bytecode-based SOMs works properly"  
50     self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a]  
ifFalse: [#b]).  
51
```

```

52     self assert: 28 equals: 5 + (23 perform: #value).
53 )
54
55 testInstVarAtAndPut = (
56     | tmp |
57     "Testing #at: and #at:put:"
58     tmp := Pair withKey: 3 andValue: 42.
59
60     self assert: tmp key equals: (tmp instVarAt: 1).
61
62     tmp instVarAt: 1 put: #foo.
63     self assert: #foo equals: tmp key.
64 )
65
66 testName = (
67     self assert: #Object equals: Object name.
68     self assert: #'Object class' equals: Object class name.
69     self assert: #Integer equals: 1 class name.
70 )
71
72 testAsString = (
73     self assert: 'Object' equals: Object asString.
74     self assert: 'Object class' equals: Object class asString.
75     self assert: 'Integer' equals: 1 class asString.
76 )
77
78 testSelectors = (
79     | sels |
80     sels := Object selectors.
81     self assert: 32 equals: sels length.
82
83     sels contains: #=.
84     self assert: (Object hasMethod: #=).
85
86     sels contains: #value.
87     self assert: (Object hasMethod: #value).
88
89     sels contains: #notNil.
90     self assert: (Object hasMethod: #notNil).
91 )
92 )
93

```

```
1 "  
2  
3 $Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 ReflectionTest = TestCase (  
29   testRespondsTo = (  
30     self assert: (Object new respondsTo: #isNil).  
31     self assert: (23 respondsTo: #isNil).  
32     self assert: (23 respondsTo: #+).  
33   )  
34  
35   testMethods = (  
36     "First method in Object should be #class."  
37     self assert: #class equals: (Object methods at: 1) signature.  
38     self assert: (Object hasMethod: #==).  
39   )  
40  
41   testPerform = (  
42     | o |  
43     self assert: Integer equals: (23 perform: #class).  
44     self assert: (23 perform: #between:and: withArguments: (Array with: 22  
with: 24)).  
45  
46     o := SuperTest new.  
47     self assert: #super equals: (o perform: #something inSuperclass:  
SuperTestSuperClass).  
48  
49     "Trying to see whether the stack in bytecode-based SOMs works properly"  
50     self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a]  
ifFalse: [#b]).  
51
```

```

52     self assert: 28 equals: 5 + (23 perform: #value).
53 )
54
55 testInstVarAtAndPut = (
56     | tmp |
57     "Testing #at: and #at:put:"
58     tmp := Pair withKey: 3 andValue: 42.
59
60     self assert: tmp key equals: (tmp instVarAt: 1).
61
62     tmp instVarAt: 1 put: #foo.
63     self assert: #foo equals: tmp key.
64 )
65
66 testName = (
67     self assert: #Object equals: Object name.
68     self assert: #'Object class' equals: Object class name.
69     self assert: #Integer equals: 1 class name.
70 )
71
72 testAsString = (
73     self assert: 'Object' equals: Object asString.
74     self assert: 'Object class' equals: Object class asString.
75     self assert: 'Integer' equals: 1 class asString.
76 )
77
78 testSelectors = (
79     | sels |
80     sels := Object selectors.
81     self assert: 32 equals: sels length.
82
83     sels contains: #=.
84     self assert: (Object hasMethod: #=).
85
86     sels contains: #value.
87     self assert: (Object hasMethod: #value).
88
89     sels contains: #notNil.
90     self assert: (Object hasMethod: #notNil).
91 )
92 )
93

```

```
1 "  
2  
3 $Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SelfBlock2Test = TestCase (  
29  
30     testEscapedBlock = (  
31         self assert: 42 equals: self give42 value  
32     )  
33  
34     give42 = (  
35         ^[ self giveBlock value ]  
36     )  
37  
38     giveBlock = (  
39         ^self returnBlock value  
40     )  
41  
42     returnBlock = (  
43         ^[ self returnBlock2 value ]  
44     )  
45  
46     returnBlock2 = (  
47         ^[ 42 ]  
48     )  
49 )  
50
```

```
1 "  
2  
3 $Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SelfBlock3Test = TestCase (  
29  
30     testEscapedBlock = (  
31         self assert: 42 equals: self give42 value  
32     )  
33  
34     give42 = (  
35         ^[ self giveBlock value ]  
36     )  
37  
38     giveBlock = (  
39         ^self returnBlock value  
40     )  
41  
42     returnBlock = (  
43         ^[ self returnBlock2 value ]  
44     )  
45  
46     returnBlock2 = (  
47         ^[ 42 ]  
48     )  
49 )  
50
```

```
1 "  
2  
3 $Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SelfBlock4Test = TestCase (  
29  
30   testEscapedBlock = (  
31     self assert: 42 equals: self give42 value  
32   )  
33  
34   give42 = (  
35     ^[ self giveBlock value ]  
36   )  
37  
38   giveBlock = (  
39     ^self returnBlock value  
40   )  
41  
42   returnBlock = (  
43     ^[ self returnBlock2 value ]  
44   )  
45  
46   returnBlock2 = (  
47     ^[ 42 ]  
48   )  
49 )  
50
```



```
1 "  
2  
3 $Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SelfBlock5Test = TestCase (  
29  
30     testEscapedBlock = (  
31         self assert: 42 equals: self give42 value  
32     )  
33  
34     give42 = (  
35         ^[ self giveBlock value ]  
36     )  
37  
38     giveBlock = (  
39         ^self returnBlock value  
40     )  
41  
42     returnBlock = (  
43         ^[ self returnBlock2 value ]  
44     )  
45  
46     returnBlock2 = (  
47         ^[ 42 ]  
48     )  
49 )  
50
```

```
1 "  
2  
3 $Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SelfBlockTest = TestCase (  
29  
30   testEscapedBlock = (  
31     self assert: 42 equals: self give42 value  
32   )  
33  
34   give42 = (  
35     ^[ self giveBlock value ]  
36   )  
37  
38   giveBlock = (  
39     ^self returnBlock value  
40   )  
41  
42   returnBlock = (  
43     ^[ self returnBlock2 value ]  
44   )  
45  
46   returnBlock2 = (  
47     ^[ 42 ]  
48   )  
49 )  
50
```

```

1 Set2Test = TestCase (
2   testBasics = (
3     | a b t |
4     a := Set new.
5     b := Set new.
6
7     a add: #a.
8     b add: #b.
9
10    self deny: a = b.
11
12    t := Set new.
13    t add: #a.
14
15    self deny: a == t. "different objects"
16    self assert: a equals: t. "but with equal value"
17  )
18
19  testEquality = (
20    | a b |
21    a := Set new.
22    a addAll: #(1 2 3 4).
23
24    b := Set new.
25    b addAll: #(1 2 3 4).
26
27    self assert: a = b.
28    self deny: a == b.
29
30    a add: 5.
31
32    self deny: a = b.
33    b add: 5.
34
35    self assert: a = b.
36
37    b add: #foo.
38    self deny: a = b.
39    a add: #foo.
40    self assert: a = b.
41  )
42
43  testUnion = (
44    | a b u |
45    a := Set new.
46    b := Set new.
47    a addAll: #(1 2 3 4).
48    b addAll: #(1 2 3 4).
49
50    u := a union: b.
51    self assert: a = b.
52    self assert: u = a.
53    self assert: u = b.
54
55    self deny: a == b.
56    self deny: u == a.
57    self deny: u == b.
58

```

```

59     self assert: 4 equals: u size.
60
61     a add: #mm.
62     u := a union: b.
63     self assert: 5 equals: u size.
64
65     b add: #oo.
66     u := a union: b.
67     self assert: 6 equals: u size.
68
69     b add: #mm.
70     u := a union: b.
71     self assert: 6 equals: u size.
72 )
73
74 testIntersection = (
75     | a b i |
76     a := Set new.
77     b := Set new.
78     a addAll: #(43 64 730 667).
79     b addAll: #(43 64 730 667).
80
81     i := a intersection: b.
82     self assert: 4 equals: i size.
83
84     a do: [:e |
85         self assert: (i contains: e) ].
86
87     b := Set new.
88     b add: 64.
89     b add: 667.
90
91     i := a intersection: b.
92     self assert: 2 equals: i size.
93
94     b do: [:e |
95         self assert: (i contains: e) ].
96 )
97
98 testSetDifference = (
99     | a b d |
100    a := Set new.
101    b := Set new.
102    a addAll: #(43 64 730 667).
103    b addAll: #(43 64 730 667).
104
105    d := a - b.
106    self assert: d isEmpty.
107    self assert: 0 equals: d size.
108
109    b := Set new.
110    b add: 43.
111    b add: 667.
112    b add: 345345.
113
114    d := a - b.
115    self assert: 2 equals: d size.
116    self assert: (d contains: 64).
117    self assert: (d contains: 730).
118 )
119

```

```

120 testContains = (
121     | s |
122     s := Set new.
123
124     self deny: (s contains: #'333').
125     s add: #'333'.
126
127     self assert: (s contains: #'333').
128     s add: 333.
129     self assert: (s contains: #'333').
130     self assert: (s contains: 333).
131 )
132
133 testRemove = (
134     | s o |
135     o := Object new.
136     s := Set new.
137     s add: #sfsdf.
138     s add: 323.
139     s add: 545.
140     s add: self.
141     s add: o.
142
143     self assert: (s contains: o).
144     self assert: 5 equals: s size.
145
146     s remove: 323.
147     self assert: 4 equals: s size.
148     self deny: (s contains: 323).
149     s add: 65767.
150     self assert: 5 equals: s size.
151
152     s remove: o.
153     self assert: 4 equals: s size.
154     self deny: (s contains: o).
155     s remove: self.
156     s remove: #sfsdf.
157     s remove: 323.
158     s remove: 545.
159     s remove: 65767.
160
161     self assert: s isEmpty.
162 )
163
164 testFirst = (
165     | s |
166     s := Set new.
167     s addAll: #(233 545 665).
168
169     self assert: 233 equals: s first.
170
171     s remove: 233.
172     self assert: 545 equals: s first.
173
174     s remove: 545.
175     self assert: 665 equals: s first.
176 )
177
178 testCollect = (
179     | s r |
180     s := Set new.

```

```
181     s addAll: #(21 54642 6753 344 655).
182
183     r := s collect: [:e | e % 10 ].
184
185     self assert: 5 equals: r size.
186     self assert: 1 equals: r first.
187     self assert: 2 equals: (r at: 2).
188 )
189 )
190
```

```

1 Set3Test = TestCase (
2   testBasics = (
3     | a b t |
4     a := Set new.
5     b := Set new.
6
7     a add: #a.
8     b add: #b.
9
10    self deny: a = b.
11
12    t := Set new.
13    t add: #a.
14
15    self deny: a == t. "different objects"
16    self assert: a equals: t. "but with equal value"
17  )
18
19  testEquality = (
20    | a b |
21    a := Set new.
22    a addAll: #(1 2 3 4).
23
24    b := Set new.
25    b addAll: #(1 2 3 4).
26
27    self assert: a = b.
28    self deny: a == b.
29
30    a add: 5.
31
32    self deny: a = b.
33    b add: 5.
34
35    self assert: a = b.
36
37    b add: #foo.
38    self deny: a = b.
39    a add: #foo.
40    self assert: a = b.
41  )
42
43  testUnion = (
44    | a b u |
45    a := Set new.
46    b := Set new.
47    a addAll: #(1 2 3 4).
48    b addAll: #(1 2 3 4).
49
50    u := a union: b.
51    self assert: a = b.
52    self assert: u = a.
53    self assert: u = b.
54
55    self deny: a == b.
56    self deny: u == a.
57    self deny: u == b.
58

```

```

59     self assert: 4 equals: u size.
60
61     a add: #mm.
62     u := a union: b.
63     self assert: 5 equals: u size.
64
65     b add: #oo.
66     u := a union: b.
67     self assert: 6 equals: u size.
68
69     b add: #mm.
70     u := a union: b.
71     self assert: 6 equals: u size.
72 )
73
74 testIntersection = (
75     | a b i |
76     a := Set new.
77     b := Set new.
78     a addAll: #(43 64 730 667).
79     b addAll: #(43 64 730 667).
80
81     i := a intersection: b.
82     self assert: 4 equals: i size.
83
84     a do: [:e |
85         self assert: (i contains: e) ].
86
87     b := Set new.
88     b add: 64.
89     b add: 667.
90
91     i := a intersection: b.
92     self assert: 2 equals: i size.
93
94     b do: [:e |
95         self assert: (i contains: e) ].
96 )
97
98 testSetDifference = (
99     | a b d |
100    a := Set new.
101    b := Set new.
102    a addAll: #(43 64 730 667).
103    b addAll: #(43 64 730 667).
104
105    d := a - b.
106    self assert: d isEmpty.
107    self assert: 0 equals: d size.
108
109    b := Set new.
110    b add: 43.
111    b add: 667.
112    b add: 345345.
113
114    d := a - b.
115    self assert: 2 equals: d size.
116    self assert: (d contains: 64).
117    self assert: (d contains: 730).
118 )
119

```



```

120 testContains = (
121     | s |
122     s := Set new.
123
124     self deny: (s contains: #'333').
125     s add: #'333'.
126
127     self assert: (s contains: #'333').
128     s add: 333.
129     self assert: (s contains: #'333').
130     self assert: (s contains: 333).
131 )
132
133 testRemove = (
134     | s o |
135     o := Object new.
136     s := Set new.
137     s add: #sfsdf.
138     s add: 323.
139     s add: 545.
140     s add: self.
141     s add: o.
142
143     self assert: (s contains: o).
144     self assert: 5 equals: s size.
145
146     s remove: 323.
147     self assert: 4 equals: s size.
148     self deny: (s contains: 323).
149     s add: 65767.
150     self assert: 5 equals: s size.
151
152     s remove: o.
153     self assert: 4 equals: s size.
154     self deny: (s contains: o).
155     s remove: self.
156     s remove: #sfsdf.
157     s remove: 323.
158     s remove: 545.
159     s remove: 65767.
160
161     self assert: s isEmpty.
162 )
163
164 testFirst = (
165     | s |
166     s := Set new.
167     s addAll: #(233 545 665).
168
169     self assert: 233 equals: s first.
170
171     s remove: 233.
172     self assert: 545 equals: s first.
173
174     s remove: 545.
175     self assert: 665 equals: s first.
176 )
177
178 testCollect = (
179     | s r |
180     s := Set new.

```

```
181     s addAll: #(21 54642 6753 344 655).
182
183     r := s collect: [:e | e % 10 ].
184
185     self assert: 5 equals: r size.
186     self assert: 1 equals: r first.
187     self assert: 2 equals: (r at: 2).
188 )
189 )
190
```

```

1 Set4Test = TestCase (
2   testBasics = (
3     | a b t |
4     a := Set new.
5     b := Set new.
6
7     a add: #a.
8     b add: #b.
9
10    self deny: a = b.
11
12    t := Set new.
13    t add: #a.
14
15    self deny: a == t. "different objects"
16    self assert: a equals: t. "but with equal value"
17  )
18
19  testEquality = (
20    | a b |
21    a := Set new.
22    a addAll: #(1 2 3 4).
23
24    b := Set new.
25    b addAll: #(1 2 3 4).
26
27    self assert: a = b.
28    self deny: a == b.
29
30    a add: 5.
31
32    self deny: a = b.
33    b add: 5.
34
35    self assert: a = b.
36
37    b add: #foo.
38    self deny: a = b.
39    a add: #foo.
40    self assert: a = b.
41  )
42
43  testUnion = (
44    | a b u |
45    a := Set new.
46    b := Set new.
47    a addAll: #(1 2 3 4).
48    b addAll: #(1 2 3 4).
49
50    u := a union: b.
51    self assert: a = b.
52    self assert: u = a.
53    self assert: u = b.
54
55    self deny: a == b.
56    self deny: u == a.
57    self deny: u == b.
58

```

```

59     self assert: 4 equals: u size.
60
61     a add: #mm.
62     u := a union: b.
63     self assert: 5 equals: u size.
64
65     b add: #oo.
66     u := a union: b.
67     self assert: 6 equals: u size.
68
69     b add: #mm.
70     u := a union: b.
71     self assert: 6 equals: u size.
72 )
73
74 testIntersection = (
75     | a b i |
76     a := Set new.
77     b := Set new.
78     a addAll: #(43 64 730 667).
79     b addAll: #(43 64 730 667).
80
81     i := a intersection: b.
82     self assert: 4 equals: i size.
83
84     a do: [:e |
85         self assert: (i contains: e) ].
86
87     b := Set new.
88     b add: 64.
89     b add: 667.
90
91     i := a intersection: b.
92     self assert: 2 equals: i size.
93
94     b do: [:e |
95         self assert: (i contains: e) ].
96 )
97
98 testSetDifference = (
99     | a b d |
100    a := Set new.
101    b := Set new.
102    a addAll: #(43 64 730 667).
103    b addAll: #(43 64 730 667).
104
105    d := a - b.
106    self assert: d isEmpty.
107    self assert: 0 equals: d size.
108
109    b := Set new.
110    b add: 43.
111    b add: 667.
112    b add: 345345.
113
114    d := a - b.
115    self assert: 2 equals: d size.
116    self assert: (d contains: 64).
117    self assert: (d contains: 730).
118 )
119

```

```

120 testContains = (
121     | s |
122     s := Set new.
123
124     self deny: (s contains: #'333').
125     s add: #'333'.
126
127     self assert: (s contains: #'333').
128     s add: 333.
129     self assert: (s contains: #'333').
130     self assert: (s contains: 333).
131 )
132
133 testRemove = (
134     | s o |
135     o := Object new.
136     s := Set new.
137     s add: #sfsdf.
138     s add: 323.
139     s add: 545.
140     s add: self.
141     s add: o.
142
143     self assert: (s contains: o).
144     self assert: 5 equals: s size.
145
146     s remove: 323.
147     self assert: 4 equals: s size.
148     self deny: (s contains: 323).
149     s add: 65767.
150     self assert: 5 equals: s size.
151
152     s remove: o.
153     self assert: 4 equals: s size.
154     self deny: (s contains: o).
155     s remove: self.
156     s remove: #sfsdf.
157     s remove: 323.
158     s remove: 545.
159     s remove: 65767.
160
161     self assert: s isEmpty.
162 )
163
164 testFirst = (
165     | s |
166     s := Set new.
167     s addAll: #(233 545 665).
168
169     self assert: 233 equals: s first.
170
171     s remove: 233.
172     self assert: 545 equals: s first.
173
174     s remove: 545.
175     self assert: 665 equals: s first.
176 )
177
178 testCollect = (
179     | s r |
180     s := Set new.

```

```
181     s addAll: #(21 54642 6753 344 655).
182
183     r := s collect: [:e | e % 10 ].
184
185     self assert: 5 equals: r size.
186     self assert: 1 equals: r first.
187     self assert: 2 equals: (r at: 2).
188 )
189 )
190
```

```

1 Set5Test = TestCase (
2   testBasics = (
3     | a b t |
4     a := Set new.
5     b := Set new.
6
7     a add: #a.
8     b add: #b.
9
10    self deny: a = b.
11
12    t := Set new.
13    t add: #a.
14
15    self deny: a == t. "different objects"
16    self assert: a equals: t. "but with equal value"
17  )
18
19  testEquality = (
20    | a b |
21    a := Set new.
22    a addAll: #(1 2 3 4).
23
24    b := Set new.
25    b addAll: #(1 2 3 4).
26
27    self assert: a = b.
28    self deny: a == b.
29
30    a add: 5.
31
32    self deny: a = b.
33    b add: 5.
34
35    self assert: a = b.
36
37    b add: #foo.
38    self deny: a = b.
39    a add: #foo.
40    self assert: a = b.
41  )
42
43  testUnion = (
44    | a b u |
45    a := Set new.
46    b := Set new.
47    a addAll: #(1 2 3 4).
48    b addAll: #(1 2 3 4).
49
50    u := a union: b.
51    self assert: a = b.
52    self assert: u = a.
53    self assert: u = b.
54
55    self deny: a == b.
56    self deny: u == a.
57    self deny: u == b.
58

```

```

59     self assert: 4 equals: u size.
60
61     a add: #mm.
62     u := a union: b.
63     self assert: 5 equals: u size.
64
65     b add: #oo.
66     u := a union: b.
67     self assert: 6 equals: u size.
68
69     b add: #mm.
70     u := a union: b.
71     self assert: 6 equals: u size.
72 )
73
74 testIntersection = (
75     | a b i |
76     a := Set new.
77     b := Set new.
78     a addAll: #(43 64 730 667).
79     b addAll: #(43 64 730 667).
80
81     i := a intersection: b.
82     self assert: 4 equals: i size.
83
84     a do: [:e |
85         self assert: (i contains: e) ].
86
87     b := Set new.
88     b add: 64.
89     b add: 667.
90
91     i := a intersection: b.
92     self assert: 2 equals: i size.
93
94     b do: [:e |
95         self assert: (i contains: e) ].
96 )
97
98 testSetDifference = (
99     | a b d |
100    a := Set new.
101    b := Set new.
102    a addAll: #(43 64 730 667).
103    b addAll: #(43 64 730 667).
104
105    d := a - b.
106    self assert: d isEmpty.
107    self assert: 0 equals: d size.
108
109    b := Set new.
110    b add: 43.
111    b add: 667.
112    b add: 345345.
113
114    d := a - b.
115    self assert: 2 equals: d size.
116    self assert: (d contains: 64).
117    self assert: (d contains: 730).
118 )
119

```



```

120 testContains = (
121     | s |
122     s := Set new.
123
124     self deny: (s contains: #'333').
125     s add: #'333'.
126
127     self assert: (s contains: #'333').
128     s add: 333.
129     self assert: (s contains: #'333').
130     self assert: (s contains: 333).
131 )
132
133 testRemove = (
134     | s o |
135     o := Object new.
136     s := Set new.
137     s add: #sfsdf.
138     s add: 323.
139     s add: 545.
140     s add: self.
141     s add: o.
142
143     self assert: (s contains: o).
144     self assert: 5 equals: s size.
145
146     s remove: 323.
147     self assert: 4 equals: s size.
148     self deny: (s contains: 323).
149     s add: 65767.
150     self assert: 5 equals: s size.
151
152     s remove: o.
153     self assert: 4 equals: s size.
154     self deny: (s contains: o).
155     s remove: self.
156     s remove: #sfsdf.
157     s remove: 323.
158     s remove: 545.
159     s remove: 65767.
160
161     self assert: s isEmpty.
162 )
163
164 testFirst = (
165     | s |
166     s := Set new.
167     s addAll: #(233 545 665).
168
169     self assert: 233 equals: s first.
170
171     s remove: 233.
172     self assert: 545 equals: s first.
173
174     s remove: 545.
175     self assert: 665 equals: s first.
176 )
177
178 testCollect = (
179     | s r |
180     s := Set new.

```

```
181     s addAll: #(21 54642 6753 344 655).
182
183     r := s collect: [:e | e % 10 ].
184
185     self assert: 5 equals: r size.
186     self assert: 1 equals: r first.
187     self assert: 2 equals: (r at: 2).
188 )
189 )
190
```

```

1 SetTest = TestCase (
2   testBasics = (
3     | a b t |
4     a := Set new.
5     b := Set new.
6
7     a add: #a.
8     b add: #b.
9
10    self deny: a = b.
11
12    t := Set new.
13    t add: #a.
14
15    self deny: a == t. "different objects"
16    self assert: a equals: t. "but with equal value"
17  )
18
19  testEquality = (
20    | a b |
21    a := Set new.
22    a addAll: #(1 2 3 4).
23
24    b := Set new.
25    b addAll: #(1 2 3 4).
26
27    self assert: a = b.
28    self deny: a == b.
29
30    a add: 5.
31
32    self deny: a = b.
33    b add: 5.
34
35    self assert: a = b.
36
37    b add: #foo.
38    self deny: a = b.
39    a add: #foo.
40    self assert: a = b.
41  )
42
43  testUnion = (
44    | a b u |
45    a := Set new.
46    b := Set new.
47    a addAll: #(1 2 3 4).
48    b addAll: #(1 2 3 4).
49
50    u := a union: b.
51    self assert: a = b.
52    self assert: u = a.
53    self assert: u = b.
54
55    self deny: a == b.
56    self deny: u == a.
57    self deny: u == b.
58

```

```

59     self assert: 4 equals: u size.
60
61     a add: #mm.
62     u := a union: b.
63     self assert: 5 equals: u size.
64
65     b add: #oo.
66     u := a union: b.
67     self assert: 6 equals: u size.
68
69     b add: #mm.
70     u := a union: b.
71     self assert: 6 equals: u size.
72 )
73
74 testIntersection = (
75     | a b i |
76     a := Set new.
77     b := Set new.
78     a addAll: #(43 64 730 667).
79     b addAll: #(43 64 730 667).
80
81     i := a intersection: b.
82     self assert: 4 equals: i size.
83
84     a do: [:e |
85         self assert: (i contains: e) ].
86
87     b := Set new.
88     b add: 64.
89     b add: 667.
90
91     i := a intersection: b.
92     self assert: 2 equals: i size.
93
94     b do: [:e |
95         self assert: (i contains: e) ].
96 )
97
98 testSetDifference = (
99     | a b d |
100    a := Set new.
101    b := Set new.
102    a addAll: #(43 64 730 667).
103    b addAll: #(43 64 730 667).
104
105    d := a - b.
106    self assert: d isEmpty.
107    self assert: 0 equals: d size.
108
109    b := Set new.
110    b add: 43.
111    b add: 667.
112    b add: 345345.
113
114    d := a - b.
115    self assert: 2 equals: d size.
116    self assert: (d contains: 64).
117    self assert: (d contains: 730).
118 )
119

```

```

120 testContains = (
121     | s |
122     s := Set new.
123
124     self deny: (s contains: #'333').
125     s add: #'333'.
126
127     self assert: (s contains: #'333').
128     s add: 333.
129     self assert: (s contains: #'333').
130     self assert: (s contains: 333).
131 )
132
133 testRemove = (
134     | s o |
135     o := Object new.
136     s := Set new.
137     s add: #sfsdf.
138     s add: 323.
139     s add: 545.
140     s add: self.
141     s add: o.
142
143     self assert: (s contains: o).
144     self assert: 5 equals: s size.
145
146     s remove: 323.
147     self assert: 4 equals: s size.
148     self deny: (s contains: 323).
149     s add: 65767.
150     self assert: 5 equals: s size.
151
152     s remove: o.
153     self assert: 4 equals: s size.
154     self deny: (s contains: o).
155     s remove: self.
156     s remove: #sfsdf.
157     s remove: 323.
158     s remove: 545.
159     s remove: 65767.
160
161     self assert: s isEmpty.
162 )
163
164 testFirst = (
165     | s |
166     s := Set new.
167     s addAll: #(233 545 665).
168
169     self assert: 233 equals: s first.
170
171     s remove: 233.
172     self assert: 545 equals: s first.
173
174     s remove: 545.
175     self assert: 665 equals: s first.
176 )
177
178 testCollect = (
179     | s r |
180     s := Set new.

```

```
181     s addAll: #(21 54642 6753 344 655).
182
183     r := s collect: [:e | e % 10 ].
184
185     self assert: 5 equals: r size.
186     self assert: 1 equals: r first.
187     self assert: 2 equals: (r at: 2).
188 )
189 )
190
```

```
1 SpecialSelectors2Test = TestCase (
2   testMinusMinsPrefix = (
3     self assert: self --> 1 equals: 1.
4     self assert: self -- 1 equals: 1.
5   )
6
7   --> aValue = (
8     ^1
9   )
10
11   -- aValue = (
12     •ã
13   )
14 )
15
```

```
1 SpecialSelectors3Test = TestCase (
2   testMinusMinsPrefix = (
3     self assert: self --> 1 equals: 1.
4     self assert: self -- 1 equals: 1.
5   )
6
7   --> aValue = (
8     ^1
9   )
10
11   -- aValue = (
12     •ã
13   )
14 )
15
```



```
1 SpecialSelectors4Test = TestCase (
2   testMinusMinsPrefix = (
3     self assert: self --> 1 equals: 1.
4     self assert: self -- 1 equals: 1.
5   )
6
7   --> aValue = (
8     ^1
9   )
10
11   -- aValue = (
12     •ã
13   )
14 )
15
```

```
1 SpecialSelectors5Test = TestCase (
2   testMinusMinsPrefix = (
3     self assert: self --> 1 equals: 1.
4     self assert: self -- 1 equals: 1.
5   )
6
7   --> aValue = (
8     ^1
9   )
10
11   -- aValue = (
12     •ã
13   )
14 )
15
```

```
1 SpecialSelectorsTest = TestCase (
2   testMinusMinsPrefix = (
3     self assert: self --> 1 equals: 1.
4     self assert: self -- 1 equals: 1.
5   )
6
7   --> aValue = (
8     ^1
9   )
10
11   -- aValue = (
12     •ã
13   )
14 )
15
```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 String2Test = TestCase (  
24  
25   testEquality = (  
26     | str1 str2 |  
27     str1 := 'foo'.  
28     str2 := 'bar'.  
29  
30     self assert: str1 = str1.  
31     self assert: str1 = 'foo'.  
32     self assert: str1 = ('f' + 'oo').  
33     self deny:   str1 = str2.  
34     self assert: str2 = str2.  
35  
36     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').  
37     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.  
38     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.  
39     self assert: ('f' + 'o' + 'o') = #foo.  
40   )  
41  
42   testEqualEqual = (  
43     | str1 |  
44     str1 := 'foo'.  
45     self assert: str1 == str1.  
46     self deny:   str1 == str1 asSymbol.  
47     self deny:   str1 == #foo.  
48   )  
49  
50   testLength = (  
51     self assert: 1 equals: 't' length.  
52     self assert: 6 equals: ('foo' + 'bar') length.  
53   )
```

```

54
55 testCharAt = (
56     | str |
57     str := 'foobar'.
58     self assert: 'f' equals: (str charAt: 1).
59     self assert: 'o' equals: (str charAt: 2).
60     self assert: 'o' equals: (str charAt: 3).
61     self assert: 'b' equals: (str charAt: 4).
62     self assert: 'a' equals: (str charAt: 5).
63     self assert: 'r' equals: (str charAt: 6).
64 )
65
66 testStringLiteralLineBreak = (
67     | str |
68     "Some parsers get the literals and line boundaries wrong"
69     str := '
70 '.
71     self assert: '\n' equals: (str charAt: 1).
72     self assert: 1 equals: str length.
73 )
74
75 testPrimSubstringFrom = (
76     | str |
77     str := 'foobar'.
78     self assert: 'foo' equals: (str primSubstringFrom: 1 to: 3).
79     self assert: 'bar' equals: (str primSubstringFrom: 4 to: 6).
80     self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
81     self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
82 )
83
84 testSplit = (
85     | r |
86     r := 'aaaa' split: ','.
87     self assert: 1 equals: r length.
88     self assert: 'aaaa' equals: (r at: 1).
89
90     r := 'foo.bar' split: '.'.
91     self assert: 2 equals: r length.
92     self assert: 'foo' equals: (r at: 1).
93     self assert: 'bar' equals: (r at: 2).
94
95     r := 'foo..bar' split: '..'.
96     self assert: 3 equals: r length.
97     self assert: 'foo' equals: (r at: 1).
98     self assert: '' equals: (r at: 2).
99     self assert: 'bar' equals: (r at: 3).
100
101     r := 'foo..bar' split: '...'.
102     self assert: 2 equals: r length.
103     self assert: 'foo' equals: (r at: 1).
104     self assert: 'bar' equals: (r at: 2).
105
106     r := 'foo' split: 'bar'.
107     self assert: 1 equals: r length.
108     self assert: 'foo' equals: (r at: 1).
109
110     self assert: Array is: r class
111 )
112
113 testIndexOf = (
114     self assert: -1 equals: ('foo' indexOf: 'b').

```

```

115     self assert: 1 equals: ('foo' indexOf: 'f').
116     self assert: 2 equals: ('foo' indexOf: 'o').
117     self assert: 3 equals: ('foo' indexOf: 'o' startingAt: 3).
118
119     self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).
120
121     self assert: 2 equals: ('foo' indexOf: 'oo').
122 )
123
124 testBeginsWith = (
125     self deny: ('foo' beginsWith: 'oo').
126     self assert: ('foo' beginsWith: 'foo').
127 )
128
129 testEndsWith = (
130     self assert: ('foo' endsWith: 'foo').
131     self assert: ('foo' endsWith: 'oo').
132     self deny: ('f' endsWith: 'bar').
133     self deny: ('f' endsWith: 'foo').
134 )
135
136 testMultiLineString = (
137     "Test whether the parser will parse multi-line strings correctly."
138     self assert: '
139 1234567890
140 1234567890
141 1234567890
142 1234567890
143 1234567890' equals: '
144 1234567890
145 1234567890
146 1234567890
147 1234567890
148 1234567890'
149 )
150
151 testEscapeSequences = (
152     "Tests for escape sequences, not all of them are reliable represented
as
153     proper strings. So, we do a simple equality test, and check
substring or
154     length.
155
156     \t'    F " 6† & 7FW
157     \b'    & 6·7 6R 6† & 7FW
158     \n'    æWvÆ-æR 6† & 7FW
159     \r'    6 '&- vR &WGW&â 6† & 7FW
160     \f'    f÷&ÖfVVB 6† & 7FW
161     \'     6-ævÆR V÷FR 6† & 7FW
162     \\'    & 6·6Æ 6, 6† & 7FW
163     \0     zero byte character
164     "
165
166     self assert: '\t' equals: '\t'.
167     self assert: 1 equals: '\t' length.
168
169     self assert: '\b' equals: '\b'.
170     self assert: 1 equals: '\b' length.
171
172     self assert: '\n' equals: '\n'.
173     self assert: 1 equals: '\n' length.

```

```

174     self deny: ('\n' endsWith: 'n').
175
176     self assert: '\r' equals: '\r'.
177     self assert: 1 equals: '\n' length.
178     self deny: ('\r' endsWith: 'r').
179
180     self assert: '\f' equals: '\f'.
181     self assert: 1 equals: '\f' length.
182     self deny: ('\f' endsWith: 'f').
183
184     self assert: '\\' equals: '\\'.
185     self assert: 1 equals: '\\' length.
186
187     self assert: '\\\\' equals: '\\\\'.
188     self assert: 1 equals: '\\\\' length.
189
190     self assert: '\\0' equals: '\\0'.
191     self assert: 1 equals: '\\0' length.
192     self assert: 5 equals: '\\0rest' length.
193 )
194
195 testHash = (
196     | str |
197     "Hash should be identical for strings that are identical,
198     whether given literal or composed at runtime"
199     self assert: 'foobar' hashCode equals: 'foobar' hashCode.
200     self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode
201         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode.
202
203     str := 'foo' + 'bar'.
204     str := str + str.
205     self assert: 'foobarfoobar' hashCode equals: str hashCode.
206
207     str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw
erltnwrtln'.
208     self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw
erltnwrtln' hashCode
209         equals: str hashCode.
210 )
211
212 testWhiteSpace = (
213     self assert: ' ' isWhiteSpace.
214     self assert: '\t' isWhiteSpace.
215     self assert: '\t\n \n \n' isWhiteSpace.
216
217     self deny: '' isWhiteSpace.
218     self deny: '\t\n N \n \n' isWhiteSpace.
219     self deny: 'N' isWhiteSpace.
220     self deny: '3' isWhiteSpace.
221 )
222
223 testLetters = (
224     self assert: 'a' isLetters.
225     self assert: 'all' isLetters.
226
227     self deny: '' isLetters.
228     self deny: ' ' isLetters.
229     self deny: '3' isLetters.
230     self deny: '3333' isLetters.
231     self deny: 'aOo öéÉíä' isLetters.
232     self deny: 'aOolöéÉíä' isLetters.

```

```

233     )
234
235     testDigits = (
236         self assert: '0' isDigits.
237         self assert: '0123' isDigits.
238         self assert: '0123456789' isDigits.
239
240         self deny: '' isDigits.
241         self deny: ' ' isDigits.
242         self deny: 'S' isDigits.
243         self deny: '333 3' isDigits.
244         self deny: '66i77' isDigits.
245         self deny: '66e7' isDigits.
246         self deny: 'aOolöéÉíä' isDigits.
247     )
248
249     testAsInteger = (
250         self assert: 0 equals: '0' asInteger.
251         self assert: 100 equals: '100' asInteger.
252         self assert: 923 equals: '923' asInteger.
253
254         self assert: -0 equals: '-0' asInteger.
255         self assert: -100 equals: '-100' asInteger.
256         self assert: -923 equals: '-923' asInteger.
257
258         self assert: 123342353453453456456456 equals:
259         '123342353453453456456456' asInteger.
260     )
261

```



```

1  "
2  Copyright (c) 2001-2013 see AUTHORS file
3
4  Permission is hereby granted, free of charge, to any person obtaining a
copy
5  of this software and associated documentation files (the 'Software'), to
deal
6  in the Software without restriction, including without limitation the
rights
7  to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8  copies of the Software, and to permit persons to whom the Software is
9  furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 String3Test = TestCase (
24
25   testEquality = (
26     | str1 str2 |
27     str1 := 'foo'.
28     str2 := 'bar'.
29
30     self assert: str1 = str1.
31     self assert: str1 = 'foo'.
32     self assert: str1 = ('f' + 'oo').
33     self deny:   str1 = str2.
34     self assert: str2 = str2.
35
36     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
37     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
38     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
39     self assert: ('f' + 'o' + 'o') = #foo.
40   )
41
42   testEqualEqual = (
43     | str1 |
44     str1 := 'foo'.
45     self assert: str1 == str1.
46     self deny:   str1 == str1 asSymbol.
47     self deny:   str1 == #foo.
48   )
49
50   testLength = (
51     self assert: 1 equals: 't' length.
52     self assert: 6 equals: ('foo' + 'bar') length.
53   )

```

```

54
55 testCharAt = (
56     | str |
57     str := 'foobar'.
58     self assert: 'f' equals: (str charAt: 1).
59     self assert: 'o' equals: (str charAt: 2).
60     self assert: 'o' equals: (str charAt: 3).
61     self assert: 'b' equals: (str charAt: 4).
62     self assert: 'a' equals: (str charAt: 5).
63     self assert: 'r' equals: (str charAt: 6).
64 )
65
66 testStringLiteralLineBreak = (
67     | str |
68     "Some parsers get the literals and line boundaries wrong"
69     str := '
70 '.
71     self assert: '\n' equals: (str charAt: 1).
72     self assert: 1 equals: str length.
73 )
74
75 testPrimSubstringFrom = (
76     | str |
77     str := 'foobar'.
78     self assert: 'foo' equals: (str primSubstringFrom: 1 to: 3).
79     self assert: 'bar' equals: (str primSubstringFrom: 4 to: 6).
80     self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
81     self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
82 )
83
84 testSplit = (
85     | r |
86     r := 'aaaa' split: ','.
87     self assert: 1 equals: r length.
88     self assert: 'aaaa' equals: (r at: 1).
89
90     r := 'foo.bar' split: '.'.
91     self assert: 2 equals: r length.
92     self assert: 'foo' equals: (r at: 1).
93     self assert: 'bar' equals: (r at: 2).
94
95     r := 'foo..bar' split: '.'.
96     self assert: 3 equals: r length.
97     self assert: 'foo' equals: (r at: 1).
98     self assert: '' equals: (r at: 2).
99     self assert: 'bar' equals: (r at: 3).
100
101     r := 'foo..bar' split: '..'.
102     self assert: 2 equals: r length.
103     self assert: 'foo' equals: (r at: 1).
104     self assert: 'bar' equals: (r at: 2).
105
106     r := 'foo' split: 'bar'.
107     self assert: 1 equals: r length.
108     self assert: 'foo' equals: (r at: 1).
109
110     self assert: Array is: r class
111 )
112
113 testIndexOf = (
114     self assert: -1 equals: ('foo' indexOf: 'b').

```

```

115     self assert: 1 equals: ('foo' indexOf: 'f').
116     self assert: 2 equals: ('foo' indexOf: 'o').
117     self assert: 3 equals: ('foo' indexOf: 'o' startingAt: 3).
118
119     self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).
120
121     self assert: 2 equals: ('foo' indexOf: 'oo').
122 )
123
124 testBeginsWith = (
125     self deny: ('foo' beginsWith: 'oo').
126     self assert: ('foo' beginsWith: 'foo').
127 )
128
129 testEndsWith = (
130     self assert: ('foo' endsWith: 'foo').
131     self assert: ('foo' endsWith: 'oo').
132     self deny: ('f' endsWith: 'bar').
133     self deny: ('f' endsWith: 'foo').
134 )
135
136 testMultiLineString = (
137     "Test whether the parser will parse multi-line strings correctly."
138     self assert: '
139 1234567890
140 1234567890
141 1234567890
142 1234567890
143 1234567890' equals: '
144 1234567890
145 1234567890
146 1234567890
147 1234567890
148 1234567890'
149 )
150
151 testEscapeSequences = (
152     "Tests for escape sequences, not all of them are reliable represented
as
153     proper strings. So, we do a simple equality test, and check
substring or
154     length.
155
156     \t'    F " 6† & 7FW
157     \b'    & 6·7 6R 6† & 7FW
158     \n'    æWvÆ-æR 6† & 7FW
159     \r'    6 '&- vR &WGW&â 6† & 7FW
160     \f'    f÷&ÖfVVB 6† & 7FW
161     \'     6-ævÆR V÷FR 6† & 7FW
162     \\'    & 6·6Æ 6, 6† & 7FW
163     \0     zero byte character
164     "
165
166     self assert: '\t' equals: '\t'.
167     self assert: 1 equals: '\t' length.
168
169     self assert: '\b' equals: '\b'.
170     self assert: 1 equals: '\b' length.
171
172     self assert: '\n' equals: '\n'.
173     self assert: 1 equals: '\n' length.

```

```

174     self deny: ('\n' endsWith: 'n').
175
176     self assert: '\r' equals: '\r'.
177     self assert: 1 equals: '\n' length.
178     self deny: ('\r' endsWith: 'r').
179
180     self assert: '\f' equals: '\f'.
181     self assert: 1 equals: '\f' length.
182     self deny: ('\f' endsWith: 'f').
183
184     self assert: '\\' equals: '\\'.
185     self assert: 1 equals: '\\' length.
186
187     self assert: '\\\\' equals: '\\\\'.
188     self assert: 1 equals: '\\\\' length.
189
190     self assert: '\\0' equals: '\\0'.
191     self assert: 1 equals: '\\0' length.
192     self assert: 5 equals: '\\0rest' length.
193 )
194
195 testHash = (
196     | str |
197     "Hash should be identical for strings that are identical,
198     whether given literal or composed at runtime"
199     self assert: 'foobar' hashCode equals: 'foobar' hashCode.
200     self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode
201         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode.
202
203     str := 'foo' + 'bar'.
204     str := str + str.
205     self assert: 'foobarfoobar' hashCode equals: str hashCode.
206
207     str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw
erltnwrtln'.
208     self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw
erltnwrtln' hashCode
209         equals: str hashCode.
210 )
211
212 testWhiteSpace = (
213     self assert: ' ' isWhiteSpace.
214     self assert: '\t' isWhiteSpace.
215     self assert: '\t\n \n \n' isWhiteSpace.
216
217     self deny: '' isWhiteSpace.
218     self deny: '\t\n N \n \n' isWhiteSpace.
219     self deny: 'N' isWhiteSpace.
220     self deny: '3' isWhiteSpace.
221 )
222
223 testLetters = (
224     self assert: 'a' isLetters.
225     self assert: 'all' isLetters.
226
227     self deny: '' isLetters.
228     self deny: ' ' isLetters.
229     self deny: '3' isLetters.
230     self deny: '3333' isLetters.
231     self deny: 'aOo öéÉíä' isLetters.
232     self deny: 'aOolöéÉíä' isLetters.

```

```

233     )
234
235     testDigits = (
236         self assert: '0' isDigits.
237         self assert: '0123' isDigits.
238         self assert: '0123456789' isDigits.
239
240         self deny: '' isDigits.
241         self deny: ' ' isDigits.
242         self deny: 'S' isDigits.
243         self deny: '333 3' isDigits.
244         self deny: '66i77' isDigits.
245         self deny: '66e7' isDigits.
246         self deny: 'aOolöéÉíä' isDigits.
247     )
248
249     testAsInteger = (
250         self assert: 0 equals: '0' asInteger.
251         self assert: 100 equals: '100' asInteger.
252         self assert: 923 equals: '923' asInteger.
253
254         self assert: -0 equals: '-0' asInteger.
255         self assert: -100 equals: '-100' asInteger.
256         self assert: -923 equals: '-923' asInteger.
257
258         self assert: 123342353453453456456456 equals:
259         '123342353453453456456456' asInteger.
260     )
261

```

```

1  "
2  Copyright (c) 2001-2013 see AUTHORS file
3
4  Permission is hereby granted, free of charge, to any person obtaining a
copy
5  of this software and associated documentation files (the 'Software'), to
deal
6  in the Software without restriction, including without limitation the
rights
7  to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8  copies of the Software, and to permit persons to whom the Software is
9  furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 String4Test = TestCase (
24
25   testEquality = (
26     | str1 str2 |
27     str1 := 'foo'.
28     str2 := 'bar'.
29
30     self assert: str1 = str1.
31     self assert: str1 = 'foo'.
32     self assert: str1 = ('f' + 'oo').
33     self deny:   str1 = str2.
34     self assert: str2 = str2.
35
36     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
37     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
38     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
39     self assert: ('f' + 'o' + 'o') = #foo.
40   )
41
42   testEqualEqual = (
43     | str1 |
44     str1 := 'foo'.
45     self assert: str1 == str1.
46     self deny:   str1 == str1 asSymbol.
47     self deny:   str1 == #foo.
48   )
49
50   testLength = (
51     self assert: 1 equals: 't' length.
52     self assert: 6 equals: ('foo' + 'bar') length.
53   )

```

```

54
55 testCharAt = (
56     | str |
57     str := 'foobar'.
58     self assert: 'f' equals: (str charAt: 1).
59     self assert: 'o' equals: (str charAt: 2).
60     self assert: 'o' equals: (str charAt: 3).
61     self assert: 'b' equals: (str charAt: 4).
62     self assert: 'a' equals: (str charAt: 5).
63     self assert: 'r' equals: (str charAt: 6).
64 )
65
66 testStringLiteralLineBreak = (
67     | str |
68     "Some parsers get the literals and line boundaries wrong"
69     str := '
70 '.
71     self assert: '\n' equals: (str charAt: 1).
72     self assert: 1 equals: str length.
73 )
74
75 testPrimSubstringFrom = (
76     | str |
77     str := 'foobar'.
78     self assert: 'foo' equals: (str primSubstringFrom: 1 to: 3).
79     self assert: 'bar' equals: (str primSubstringFrom: 4 to: 6).
80     self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
81     self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
82 )
83
84 testSplit = (
85     | r |
86     r := 'aaaa' split: ','.
87     self assert: 1 equals: r length.
88     self assert: 'aaaa' equals: (r at: 1).
89
90     r := 'foo.bar' split: '.'.
91     self assert: 2 equals: r length.
92     self assert: 'foo' equals: (r at: 1).
93     self assert: 'bar' equals: (r at: 2).
94
95     r := 'foo..bar' split: '..'.
96     self assert: 3 equals: r length.
97     self assert: 'foo' equals: (r at: 1).
98     self assert: '' equals: (r at: 2).
99     self assert: 'bar' equals: (r at: 3).
100
101     r := 'foo..bar' split: '...'.
102     self assert: 2 equals: r length.
103     self assert: 'foo' equals: (r at: 1).
104     self assert: 'bar' equals: (r at: 2).
105
106     r := 'foo' split: 'bar'.
107     self assert: 1 equals: r length.
108     self assert: 'foo' equals: (r at: 1).
109
110     self assert: Array is: r class
111 )
112
113 testIndexOf = (
114     self assert: -1 equals: ('foo' indexOf: 'b').

```

```

115     self assert: 1 equals: ('foo' indexOf: 'f').
116     self assert: 2 equals: ('foo' indexOf: 'o').
117     self assert: 3 equals: ('foo' indexOf: 'o' startingAt: 3).
118
119     self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).
120
121     self assert: 2 equals: ('foo' indexOf: 'oo').
122 )
123
124 testBeginsWith = (
125     self deny: ('foo' beginsWith: 'oo').
126     self assert: ('foo' beginsWith: 'foo').
127 )
128
129 testEndsWith = (
130     self assert: ('foo' endsWith: 'foo').
131     self assert: ('foo' endsWith: 'oo').
132     self deny: ('f' endsWith: 'bar').
133     self deny: ('f' endsWith: 'foo').
134 )
135
136 testMultiLineString = (
137     "Test whether the parser will parse multi-line strings correctly."
138     self assert: '
139 1234567890
140 1234567890
141 1234567890
142 1234567890
143 1234567890' equals: '
144 1234567890
145 1234567890
146 1234567890
147 1234567890
148 1234567890'
149 )
150
151 testEscapeSequences = (
152     "Tests for escape sequences, not all of them are reliable represented
153 as
154 proper strings. So, we do a simple equality test, and check
155 substring or
156 length.
157
158 \t'    F " 6† & 7FW
159 \b'    & 6·7 6R 6† & 7FW
160 \n'    æWvÆ-æR 6† & 7FW
161 \r'    6 '&- vR &WGW&â 6† & 7FW
162 \f'    f÷&ÖfVVB 6† & 7FW
163 \'     6-ævÆR V÷FR 6† & 7FW
164 \\'    & 6·6Æ 6, 6† & 7FW
165 \0     zero byte character
166
167 self assert: '\t' equals: '\t'.
168 self assert: 1 equals: '\t' length.
169
170 self assert: '\b' equals: '\b'.
171 self assert: 1 equals: '\b' length.
172
173 self assert: '\n' equals: '\n'.
174 self assert: 1 equals: '\n' length.

```



```

174     self deny: ('\n' endsWith: 'n').
175
176     self assert: '\r' equals: '\r'.
177     self assert: 1 equals: '\n' length.
178     self deny: ('\r' endsWith: 'r').
179
180     self assert: '\f' equals: '\f'.
181     self assert: 1 equals: '\f' length.
182     self deny: ('\f' endsWith: 'f').
183
184     self assert: '\\' equals: '\\'.
185     self assert: 1 equals: '\\' length.
186
187     self assert: '\\\\' equals: '\\\\'.
188     self assert: 1 equals: '\\\\' length.
189
190     self assert: '\\0' equals: '\\0'.
191     self assert: 1 equals: '\\0' length.
192     self assert: 5 equals: '\\0rest' length.
193 )
194
195 testHash = (
196     | str |
197     "Hash should be identical for strings that are identical,
198     whether given literal or composed at runtime"
199     self assert: 'foobar' hashCode equals: 'foobar' hashCode.
200     self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode
201         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode.
202
203     str := 'foo' + 'bar'.
204     str := str + str.
205     self assert: 'foobarfoobar' hashCode equals: str hashCode.
206
207     str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw
erltnwrtln'.
208     self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw
erltnwrtln' hashCode
209         equals: str hashCode.
210 )
211
212 testWhiteSpace = (
213     self assert: ' ' isWhiteSpace.
214     self assert: '\t' isWhiteSpace.
215     self assert: '\t\n \n \n' isWhiteSpace.
216
217     self deny: '' isWhiteSpace.
218     self deny: '\t\n N \n \n' isWhiteSpace.
219     self deny: 'N' isWhiteSpace.
220     self deny: '3' isWhiteSpace.
221 )
222
223 testLetters = (
224     self assert: 'a' isLetters.
225     self assert: 'all' isLetters.
226
227     self deny: '' isLetters.
228     self deny: ' ' isLetters.
229     self deny: '3' isLetters.
230     self deny: '3333' isLetters.
231     self deny: 'aOo öéÉíä' isLetters.
232     self deny: 'aOolöéÉíä' isLetters.

```

```

233     )
234
235     testDigits = (
236         self assert: '0' isDigits.
237         self assert: '0123' isDigits.
238         self assert: '0123456789' isDigits.
239
240         self deny: '' isDigits.
241         self deny: ' ' isDigits.
242         self deny: 'S' isDigits.
243         self deny: '333 3' isDigits.
244         self deny: '66i77' isDigits.
245         self deny: '66e7' isDigits.
246         self deny: 'aOolöéÉíä' isDigits.
247     )
248
249     testAsInteger = (
250         self assert: 0 equals: '0' asInteger.
251         self assert: 100 equals: '100' asInteger.
252         self assert: 923 equals: '923' asInteger.
253
254         self assert: -0 equals: '-0' asInteger.
255         self assert: -100 equals: '-100' asInteger.
256         self assert: -923 equals: '-923' asInteger.
257
258         self assert: 123342353453453456456456 equals:
259         '123342353453453456456456' asInteger.
260     )
261

```

```

1 "
2 Copyright (c) 2001-2013 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 String5Test = TestCase (
24
25     testEquality = (
26         | str1 str2 |
27         str1 := 'foo'.
28         str2 := 'bar'.
29
30         self assert: str1 = str1.
31         self assert: str1 = 'foo'.
32         self assert: str1 = ('f' + 'oo').
33         self deny: str1 = str2.
34         self assert: str2 = str2.
35
36         self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
37         self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
38         self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
39         self assert: ('f' + 'o' + 'o') = #foo.
40     )
41
42     testEqualEqual = (
43         | str1 |
44         str1 := 'foo'.
45         self assert: str1 == str1.
46         self deny: str1 == str1 asSymbol.
47         self deny: str1 == #foo.
48     )
49
50     testLength = (
51         self assert: 1 equals: 't' length.
52         self assert: 6 equals: ('foo' + 'bar') length.
53     )

```

```

54
55 testCharAt = (
56     | str |
57     str := 'foobar'.
58     self assert: 'f' equals: (str charAt: 1).
59     self assert: 'o' equals: (str charAt: 2).
60     self assert: 'o' equals: (str charAt: 3).
61     self assert: 'b' equals: (str charAt: 4).
62     self assert: 'a' equals: (str charAt: 5).
63     self assert: 'r' equals: (str charAt: 6).
64 )
65
66 testStringLiteralLineBreak = (
67     | str |
68     "Some parsers get the literals and line boundaries wrong"
69     str := '
70 '.
71     self assert: '\n' equals: (str charAt: 1).
72     self assert: 1 equals: str length.
73 )
74
75 testPrimSubstringFrom = (
76     | str |
77     str := 'foobar'.
78     self assert: 'foo' equals: (str primSubstringFrom: 1 to: 3).
79     self assert: 'bar' equals: (str primSubstringFrom: 4 to: 6).
80     self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
81     self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
82 )
83
84 testSplit = (
85     | r |
86     r := 'aaaa' split: ','.
87     self assert: 1 equals: r length.
88     self assert: 'aaaa' equals: (r at: 1).
89
90     r := 'foo.bar' split: '.'.
91     self assert: 2 equals: r length.
92     self assert: 'foo' equals: (r at: 1).
93     self assert: 'bar' equals: (r at: 2).
94
95     r := 'foo..bar' split: '..'.
96     self assert: 3 equals: r length.
97     self assert: 'foo' equals: (r at: 1).
98     self assert: '' equals: (r at: 2).
99     self assert: 'bar' equals: (r at: 3).
100
101     r := 'foo..bar' split: '...'.
102     self assert: 2 equals: r length.
103     self assert: 'foo' equals: (r at: 1).
104     self assert: 'bar' equals: (r at: 2).
105
106     r := 'foo' split: 'bar'.
107     self assert: 1 equals: r length.
108     self assert: 'foo' equals: (r at: 1).
109
110     self assert: Array is: r class
111 )
112
113 testIndexOf = (
114     self assert: -1 equals: ('foo' indexOf: 'b').

```

```

115     self assert: 1 equals: ('foo' indexOf: 'f').
116     self assert: 2 equals: ('foo' indexOf: 'o').
117     self assert: 3 equals: ('foo' indexOf: 'o' startingAt: 3).
118
119     self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).
120
121     self assert: 2 equals: ('foo' indexOf: 'oo').
122 )
123
124 testBeginsWith = (
125     self deny: ('foo' beginsWith: 'oo').
126     self assert: ('foo' beginsWith: 'foo').
127 )
128
129 testEndsWith = (
130     self assert: ('foo' endsWith: 'foo').
131     self assert: ('foo' endsWith: 'oo').
132     self deny: ('f' endsWith: 'bar').
133     self deny: ('f' endsWith: 'foo').
134 )
135
136 testMultiLineString = (
137     "Test whether the parser will parse multi-line strings correctly."
138     self assert: '
139 1234567890
140 1234567890
141 1234567890
142 1234567890
143 1234567890' equals: '
144 1234567890
145 1234567890
146 1234567890
147 1234567890
148 1234567890'
149 )
150
151 testEscapeSequences = (
152     "Tests for escape sequences, not all of them are reliable represented
as
153     proper strings. So, we do a simple equality test, and check
substring or
154     length.
155
156     \t'    F " 6† & 7FW
157     \b'    & 6·7 6R 6† & 7FW
158     \n'    æWvÆ-æR 6† & 7FW
159     \r'    6 '&- vR &WGW&â 6† & 7FW
160     \f'    f÷&ÖfVVB 6† & 7FW
161     \'     6-ævÆR V÷FR 6† & 7FW
162     \\'    & 6·6Æ 6, 6† & 7FW
163     \0     zero byte character
164     "
165
166     self assert: '\t' equals: '\t'.
167     self assert: 1 equals: '\t' length.
168
169     self assert: '\b' equals: '\b'.
170     self assert: 1 equals: '\b' length.
171
172     self assert: '\n' equals: '\n'.
173     self assert: 1 equals: '\n' length.

```

```

174     self deny: ('\n' endsWith: 'n').
175
176     self assert: '\r' equals: '\r'.
177     self assert: 1 equals: '\n' length.
178     self deny: ('\r' endsWith: 'r').
179
180     self assert: '\f' equals: '\f'.
181     self assert: 1 equals: '\f' length.
182     self deny: ('\f' endsWith: 'f').
183
184     self assert: '\'' equals: '\'.
185     self assert: 1 equals: '\'' length.
186
187     self assert: '\\' equals: '\\'.
188     self assert: 1 equals: '\\' length.
189
190     self assert: '\0' equals: '\0'.
191     self assert: 1 equals: '\0' length.
192     self assert: 5 equals: '\0rest' length.
193 )
194
195 testHash = (
196     | str |
197     "Hash should be identical for strings that are identical,
198     whether given literal or composed at runtime"
199     self assert: 'foobar' hashCode equals: 'foobar' hashCode.
200     self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode
201         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode.
202
203     str := 'foo' + 'bar'.
204     str := str + str.
205     self assert: 'foobarfoobar' hashCode equals: str hashCode.
206
207     str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw
erltnwrtln'.
208     self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw
erltnwrtln' hashCode
209         equals: str hashCode.
210 )
211
212 testWhiteSpace = (
213     self assert: ' ' isWhiteSpace.
214     self assert: '\t' isWhiteSpace.
215     self assert: '\t\n \n \n' isWhiteSpace.
216
217     self deny: '' isWhiteSpace.
218     self deny: '\t\n N \n \n' isWhiteSpace.
219     self deny: 'N' isWhiteSpace.
220     self deny: '3' isWhiteSpace.
221 )
222
223 testLetters = (
224     self assert: 'a' isLetters.
225     self assert: 'all' isLetters.
226
227     self deny: '' isLetters.
228     self deny: ' ' isLetters.
229     self deny: '3' isLetters.
230     self deny: '3333' isLetters.
231     self deny: 'aOo öéÉíä' isLetters.
232     self deny: 'aOolöéÉíä' isLetters.

```

```

233     )
234
235     testDigits = (
236         self assert: '0' isDigits.
237         self assert: '0123' isDigits.
238         self assert: '0123456789' isDigits.
239
240         self deny: '' isDigits.
241         self deny: ' ' isDigits.
242         self deny: 'S' isDigits.
243         self deny: '333 3' isDigits.
244         self deny: '66i77' isDigits.
245         self deny: '66e7' isDigits.
246         self deny: 'aOolöéÉíä' isDigits.
247     )
248
249     testAsInteger = (
250         self assert: 0 equals: '0' asInteger.
251         self assert: 100 equals: '100' asInteger.
252         self assert: 923 equals: '923' asInteger.
253
254         self assert: -0 equals: '-0' asInteger.
255         self assert: -100 equals: '-100' asInteger.
256         self assert: -923 equals: '-923' asInteger.
257
258         self assert: 123342353453453456456456 equals:
259         '123342353453453456456456' asInteger.
260     )
261

```

```

1 "
2 Copyright (c) 2001-2013 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 StringTest = TestCase (
24
25     testEquality = (
26         | str1 str2 |
27         str1 := 'foo'.
28         str2 := 'bar'.
29
30         self assert: str1 = str1.
31         self assert: str1 = 'foo'.
32         self assert: str1 = ('f' + 'oo').
33         self deny: str1 = str2.
34         self assert: str2 = str2.
35
36         self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
37         self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
38         self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
39         self assert: ('f' + 'o' + 'o') = #foo.
40     )
41
42     testEqualEqual = (
43         | str1 |
44         str1 := 'foo'.
45         self assert: str1 == str1.
46         self deny: str1 == str1 asSymbol.
47         self deny: str1 == #foo.
48     )
49
50     testLength = (
51         self assert: 1 equals: 't' length.
52         self assert: 6 equals: ('foo' + 'bar') length.
53     )

```



```

54
55 testCharAt = (
56     | str |
57     str := 'foobar'.
58     self assert: 'f' equals: (str charAt: 1).
59     self assert: 'o' equals: (str charAt: 2).
60     self assert: 'o' equals: (str charAt: 3).
61     self assert: 'b' equals: (str charAt: 4).
62     self assert: 'a' equals: (str charAt: 5).
63     self assert: 'r' equals: (str charAt: 6).
64 )
65
66 testStringLiteralLineBreak = (
67     | str |
68     "Some parsers get the literals and line boundaries wrong"
69     str := '
70 '.
71     self assert: '\n' equals: (str charAt: 1).
72     self assert: 1 equals: str length.
73 )
74
75 testPrimSubstringFrom = (
76     | str |
77     str := 'foobar'.
78     self assert: 'foo' equals: (str primSubstringFrom: 1 to: 3).
79     self assert: 'bar' equals: (str primSubstringFrom: 4 to: 6).
80     self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
81     self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
82 )
83
84 testSplit = (
85     | r |
86     r := 'aaaa' split: ','.
87     self assert: 1 equals: r length.
88     self assert: 'aaaa' equals: (r at: 1).
89
90     r := 'foo.bar' split: '.'.
91     self assert: 2 equals: r length.
92     self assert: 'foo' equals: (r at: 1).
93     self assert: 'bar' equals: (r at: 2).
94
95     r := 'foo..bar' split: '..'.
96     self assert: 3 equals: r length.
97     self assert: 'foo' equals: (r at: 1).
98     self assert: '' equals: (r at: 2).
99     self assert: 'bar' equals: (r at: 3).
100
101     r := 'foo..bar' split: '...'.
102     self assert: 2 equals: r length.
103     self assert: 'foo' equals: (r at: 1).
104     self assert: 'bar' equals: (r at: 2).
105
106     r := 'foo' split: 'bar'.
107     self assert: 1 equals: r length.
108     self assert: 'foo' equals: (r at: 1).
109
110     self assert: Array is: r class
111 )
112
113 testIndexOf = (
114     self assert: -1 equals: ('foo' indexOf: 'b').

```

```

115     self assert: 1 equals: ('foo' indexOf: 'f').
116     self assert: 2 equals: ('foo' indexOf: 'o').
117     self assert: 3 equals: ('foo' indexOf: 'o' startingAt: 3).
118
119     self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).
120
121     self assert: 2 equals: ('foo' indexOf: 'oo').
122 )
123
124 testBeginsWith = (
125     self deny: ('foo' beginsWith: 'oo').
126     self assert: ('foo' beginsWith: 'foo').
127 )
128
129 testEndsWith = (
130     self assert: ('foo' endsWith: 'foo').
131     self assert: ('foo' endsWith: 'oo').
132     self deny: ('f' endsWith: 'bar').
133     self deny: ('f' endsWith: 'foo').
134 )
135
136 testMultiLineString = (
137     "Test whether the parser will parse multi-line strings correctly."
138     self assert: '
139 1234567890
140 1234567890
141 1234567890
142 1234567890
143 1234567890' equals: '
144 1234567890
145 1234567890
146 1234567890
147 1234567890
148 1234567890'
149 )
150
151 testEscapeSequences = (
152     "Tests for escape sequences, not all of them are reliable represented
as
153     proper strings. So, we do a simple equality test, and check
substring or
154     length.
155
156     \t'    F " 6† & 7FW
157     \b'    & 6·7 6R 6† & 7FW
158     \n'    æWvÆ-æR 6† & 7FW
159     \r'    6 '&- vR &WGW&â 6† & 7FW
160     \f'    f÷&ÖfVVB 6† & 7FW
161     \'     6-ævÆR V÷FR 6† & 7FW
162     \\'    & 6·6Æ 6, 6† & 7FW
163     \0     zero byte character
164     "
165
166     self assert: '\t' equals: '\t'.
167     self assert: 1 equals: '\t' length.
168
169     self assert: '\b' equals: '\b'.
170     self assert: 1 equals: '\b' length.
171
172     self assert: '\n' equals: '\n'.
173     self assert: 1 equals: '\n' length.

```

```

174     self deny: ('\n' endsWith: 'n').
175
176     self assert: '\r' equals: '\r'.
177     self assert: 1 equals: '\n' length.
178     self deny: ('\r' endsWith: 'r').
179
180     self assert: '\f' equals: '\f'.
181     self assert: 1 equals: '\f' length.
182     self deny: ('\f' endsWith: 'f').
183
184     self assert: '\\' equals: '\\'.
185     self assert: 1 equals: '\\' length.
186
187     self assert: '\\\\' equals: '\\\\'.
188     self assert: 1 equals: '\\\\' length.
189
190     self assert: '\\0' equals: '\\0'.
191     self assert: 1 equals: '\\0' length.
192     self assert: 5 equals: '\\0rest' length.
193 )
194
195 testHash = (
196     | str |
197     "Hash should be identical for strings that are identical,
198     whether given literal or composed at runtime"
199     self assert: 'foobar' hashCode equals: 'foobar' hashCode.
200     self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode
201         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode.
202
203     str := 'foo' + 'bar'.
204     str := str + str.
205     self assert: 'foobarfoobar' hashCode equals: str hashCode.
206
207     str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw
erltnwrtln'.
208     self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw
erltnwrtln' hashCode
209         equals: str hashCode.
210 )
211
212 testWhiteSpace = (
213     self assert: ' ' isWhiteSpace.
214     self assert: '\t' isWhiteSpace.
215     self assert: '\t\n \n \n' isWhiteSpace.
216
217     self deny: '' isWhiteSpace.
218     self deny: '\t\n N \n \n' isWhiteSpace.
219     self deny: 'N' isWhiteSpace.
220     self deny: '3' isWhiteSpace.
221 )
222
223 testLetters = (
224     self assert: 'a' isLetters.
225     self assert: 'all' isLetters.
226
227     self deny: '' isLetters.
228     self deny: ' ' isLetters.
229     self deny: '3' isLetters.
230     self deny: '3333' isLetters.
231     self deny: 'aOo öéÉíä' isLetters.
232     self deny: 'aOolöéÉíä' isLetters.

```

```

233     )
234
235     testDigits = (
236         self assert: '0' isDigits.
237         self assert: '0123' isDigits.
238         self assert: '0123456789' isDigits.
239
240         self deny: '' isDigits.
241         self deny: ' ' isDigits.
242         self deny: 'S' isDigits.
243         self deny: '333 3' isDigits.
244         self deny: '66i77' isDigits.
245         self deny: '66e7' isDigits.
246         self deny: 'aOolöéÉíä' isDigits.
247     )
248
249     testAsInteger = (
250         self assert: 0 equals: '0' asInteger.
251         self assert: 100 equals: '100' asInteger.
252         self assert: 923 equals: '923' asInteger.
253
254         self assert: -0 equals: '-0' asInteger.
255         self assert: -100 equals: '-100' asInteger.
256         self assert: -923 equals: '-923' asInteger.
257
258         self assert: 123342353453453456456456 equals:
259         '123342353453453456456456' asInteger.
260     )
261

```

```
1 "  
2  
3 Copyright (c) 2007-2018 see AUTHORS file  
4 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
5 http://www.hpi.uni-potsdam.de/swa/  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Super2Test = SuperTestSuperClass (  
27  
28   testSuper = (  
29     self assert: 42 equals: self give42.  
30     self assert: 42 equals: self blockGive42.  
31   )  
32  
33   yourself = (  
34     record := record + 1000.  
35     ^ self  
36   )  
37  
38   give42 = (  
39     ^super give42  
40   )  
41  
42   blockGive42 = (  
43     ^[ super give42 ] value  
44   )  
45  
46   something = (  
47     ^ #sub  
48   )  
49  
50   number = (  
51     ^ 10  
52   )  
53
```

```

54 + other = (
55     ^ 11
56 )
57
58 +++++ other = (
59     ^ 111
60 )
61
62 keyword: other = (
63     ^ 1111
64 )
65
66 testBasicUnary = (
67     self assert: 10 equals: self number.
68     self assert: 1 equals: super number.
69 )
70
71 testBasicBinary = (
72     self assert: 11 equals: self + 3.
73     self assert: 22 equals: super + 5.
74 )
75
76
77 testBasicBinaryNonStandardOperator = (
78     self assert: 111 equals: self +++++ 3.
79     self assert: 222 equals: super +++++ 5.
80 )
81
82 testBasicKeyword = (
83     self assert: 1111 equals: (self keyword: 3).
84     self assert: 2222 equals: (super keyword: 5).
85 )
86
87 testWithBinaryUnaryMessage = (
88     | val |
89     record := 0.
90     val := super number * super number.
91     self assert: 1 equals: val.
92 )
93
94 testWithBinaryUnaryUnaryMessage = (
95     | val |
96     record := 0.
97     super yourself yourself @ super yourself yourself.
98     self assert: 2002 equals: record.
99 )
100
101 testWithKeywordUnaryUnaryMessage = (
102     | val |
103     record := 0.
104     super key: super yourself yourself key: super yourself yourself.
105     self assert: 2002 equals: record.
106
107     record := 0.
108     self key: super yourself yourself key: super yourself yourself.
109     self assert: 2002 equals: record.
110 )
111
112 "Note: testing assigning self was moved to basic interpreter tests"
113
114 testGlobalSelfDoesNotShadowKeyword = (

```

```
115     | that |
116     that := self.
117     system global: #self put: 42.
118     that optional: #selfSuperBug assert: that is: self.
119
120     self assert: 42 equals: (system global: #self)
121 )
122
123 testGlobalSuperDoesNotShadowKeyword = (
124     | that |
125     that := super.
126     system global: #super put: 42.
127     that optional: #selfSuperBug assert: that is: super.
128
129     self assert: 42 equals: (system global: #super)
130 )
131 )
132
```

```
1 "  
2  
3 Copyright (c) 2007-2018 see AUTHORS file  
4 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
5 http://www.hpi.uni-potsdam.de/swa/  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Super3Test = SuperTestSuperClass (  
27  
28   testSuper = (  
29     self assert: 42 equals: self give42.  
30     self assert: 42 equals: self blockGive42.  
31   )  
32  
33   yourself = (  
34     record := record + 1000.  
35     ^ self  
36   )  
37  
38   give42 = (  
39     ^super give42  
40   )  
41  
42   blockGive42 = (  
43     ^[ super give42 ] value  
44   )  
45  
46   something = (  
47     ^ #sub  
48   )  
49  
50   number = (  
51     ^ 10  
52   )  
53
```



```

54 + other = (
55     ^ 11
56 )
57
58 +++++ other = (
59     ^ 111
60 )
61
62 keyword: other = (
63     ^ 1111
64 )
65
66 testBasicUnary = (
67     self assert: 10 equals: self number.
68     self assert: 1 equals: super number.
69 )
70
71 testBasicBinary = (
72     self assert: 11 equals: self + 3.
73     self assert: 22 equals: super + 5.
74 )
75
76
77 testBasicBinaryNonStandardOperator = (
78     self assert: 111 equals: self +++++ 3.
79     self assert: 222 equals: super +++++ 5.
80 )
81
82 testBasicKeyword = (
83     self assert: 1111 equals: (self keyword: 3).
84     self assert: 2222 equals: (super keyword: 5).
85 )
86
87 testWithBinaryUnaryMessage = (
88     | val |
89     record := 0.
90     val := super number * super number.
91     self assert: 1 equals: val.
92 )
93
94 testWithBinaryUnaryUnaryMessage = (
95     | val |
96     record := 0.
97     super yourself yourself @ super yourself yourself.
98     self assert: 2002 equals: record.
99 )
100
101 testWithKeywordUnaryUnaryMessage = (
102     | val |
103     record := 0.
104     super key: super yourself yourself key: super yourself yourself.
105     self assert: 2002 equals: record.
106
107     record := 0.
108     self key: super yourself yourself key: super yourself yourself.
109     self assert: 2002 equals: record.
110 )
111
112 "Note: testing assigning self was moved to basic interpreter tests"
113
114 testGlobalSelfDoesNotShadowKeyword = (

```

```
115     | that |
116     that := self.
117     system global: #self put: 42.
118     that optional: #selfSuperBug assert: that is: self.
119
120     self assert: 42 equals: (system global: #self)
121 )
122
123 testGlobalSuperDoesNotShadowKeyword = (
124     | that |
125     that := super.
126     system global: #super put: 42.
127     that optional: #selfSuperBug assert: that is: super.
128
129     self assert: 42 equals: (system global: #super)
130 )
131 )
132
```

```
1 "  
2  
3 Copyright (c) 2007-2018 see AUTHORS file  
4 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
5 http://www.hpi.uni-potsdam.de/swa/  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Super4Test = SuperTestSuperClass (  
27  
28   testSuper = (  
29     self assert: 42 equals: self give42.  
30     self assert: 42 equals: self blockGive42.  
31   )  
32  
33   yourself = (  
34     record := record + 1000.  
35     ^ self  
36   )  
37  
38   give42 = (  
39     ^super give42  
40   )  
41  
42   blockGive42 = (  
43     ^[ super give42 ] value  
44   )  
45  
46   something = (  
47     ^ #sub  
48   )  
49  
50   number = (  
51     ^ 10  
52   )  
53
```

```

54   + other = (
55       ^ 11
56   )
57
58   ++++ other = (
59       ^ 111
60   )
61
62   keyword: other = (
63       ^ 1111
64   )
65
66   testBasicUnary = (
67       self assert: 10 equals: self number.
68       self assert: 1 equals: super number.
69   )
70
71   testBasicBinary = (
72       self assert: 11 equals: self + 3.
73       self assert: 22 equals: super + 5.
74   )
75
76
77   testBasicBinaryNonStandardOperator = (
78       self assert: 111 equals: self ++++ 3.
79       self assert: 222 equals: super ++++ 5.
80   )
81
82   testBasicKeyword = (
83       self assert: 1111 equals: (self keyword: 3).
84       self assert: 2222 equals: (super keyword: 5).
85   )
86
87   testWithBinaryUnaryMessage = (
88       | val |
89       record := 0.
90       val := super number * super number.
91       self assert: 1 equals: val.
92   )
93
94   testWithBinaryUnaryUnaryMessage = (
95       | val |
96       record := 0.
97       super yourself yourself @ super yourself yourself.
98       self assert: 2002 equals: record.
99   )
100
101   testWithKeywordUnaryUnaryMessage = (
102       | val |
103       record := 0.
104       super key: super yourself yourself key: super yourself yourself.
105       self assert: 2002 equals: record.
106
107       record := 0.
108       self key: super yourself yourself key: super yourself yourself.
109       self assert: 2002 equals: record.
110   )
111
112   "Note: testing assigning self was moved to basic interpreter tests"
113
114   testGlobalSelfDoesNotShadowKeyword = (

```

```
115     | that |
116     that := self.
117     system global: #self put: 42.
118     that optional: #selfSuperBug assert: that is: self.
119
120     self assert: 42 equals: (system global: #self)
121 )
122
123 testGlobalSuperDoesNotShadowKeyword = (
124     | that |
125     that := super.
126     system global: #super put: 42.
127     that optional: #selfSuperBug assert: that is: super.
128
129     self assert: 42 equals: (system global: #super)
130 )
131 )
132
```

```
1 "  
2  
3 Copyright (c) 2007-2018 see AUTHORS file  
4 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
5 http://www.hpi.uni-potsdam.de/swa/  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Super5Test = SuperTestSuperClass (  
27  
28   testSuper = (  
29     self assert: 42 equals: self give42.  
30     self assert: 42 equals: self blockGive42.  
31   )  
32  
33   yourself = (  
34     record := record + 1000.  
35     ^ self  
36   )  
37  
38   give42 = (  
39     ^super give42  
40   )  
41  
42   blockGive42 = (  
43     ^[ super give42 ] value  
44   )  
45  
46   something = (  
47     ^ #sub  
48   )  
49  
50   number = (  
51     ^ 10  
52   )  
53
```

```

54 + other = (
55     ^ 11
56 )
57
58 +++++ other = (
59     ^ 111
60 )
61
62 keyword: other = (
63     ^ 1111
64 )
65
66 testBasicUnary = (
67     self assert: 10 equals: self number.
68     self assert: 1 equals: super number.
69 )
70
71 testBasicBinary = (
72     self assert: 11 equals: self + 3.
73     self assert: 22 equals: super + 5.
74 )
75
76
77 testBasicBinaryNonStandardOperator = (
78     self assert: 111 equals: self +++++ 3.
79     self assert: 222 equals: super +++++ 5.
80 )
81
82 testBasicKeyword = (
83     self assert: 1111 equals: (self keyword: 3).
84     self assert: 2222 equals: (super keyword: 5).
85 )
86
87 testWithBinaryUnaryMessage = (
88     | val |
89     record := 0.
90     val := super number * super number.
91     self assert: 1 equals: val.
92 )
93
94 testWithBinaryUnaryUnaryMessage = (
95     | val |
96     record := 0.
97     super yourself yourself @ super yourself yourself.
98     self assert: 2002 equals: record.
99 )
100
101 testWithKeywordUnaryUnaryMessage = (
102     | val |
103     record := 0.
104     super key: super yourself yourself key: super yourself yourself.
105     self assert: 2002 equals: record.
106
107     record := 0.
108     self key: super yourself yourself key: super yourself yourself.
109     self assert: 2002 equals: record.
110 )
111
112 "Note: testing assigning self was moved to basic interpreter tests"
113
114 testGlobalSelfDoesNotShadowKeyword = (

```

```
115     | that |
116     that := self.
117     system global: #self put: 42.
118     that optional: #selfSuperBug assert: that is: self.
119
120     self assert: 42 equals: (system global: #self)
121 )
122
123 testGlobalSuperDoesNotShadowKeyword = (
124     | that |
125     that := super.
126     system global: #super put: 42.
127     that optional: #selfSuperBug assert: that is: super.
128
129     self assert: 42 equals: (system global: #super)
130 )
131 )
132
```



```
1 "  
2  
3 Copyright (c) 2007-2018 see AUTHORS file  
4 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
5 http://www.hpi.uni-potsdam.de/swa/  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 SuperTest = SuperTestSuperClass (  
27  
28   testSuper = (  
29     self assert: 42 equals: self give42.  
30     self assert: 42 equals: self blockGive42.  
31   )  
32  
33   yourself = (  
34     record := record + 1000.  
35     ^ self  
36   )  
37  
38   give42 = (  
39     ^super give42  
40   )  
41  
42   blockGive42 = (  
43     ^[ super give42 ] value  
44   )  
45  
46   something = (  
47     ^ #sub  
48   )  
49  
50   number = (  
51     ^ 10  
52   )  
53
```

```

54 + other = (
55     ^ 11
56 )
57
58 +++++ other = (
59     ^ 111
60 )
61
62 keyword: other = (
63     ^ 1111
64 )
65
66 testBasicUnary = (
67     self assert: 10 equals: self number.
68     self assert: 1 equals: super number.
69 )
70
71 testBasicBinary = (
72     self assert: 11 equals: self + 3.
73     self assert: 22 equals: super + 5.
74 )
75
76
77 testBasicBinaryNonStandardOperator = (
78     self assert: 111 equals: self +++++ 3.
79     self assert: 222 equals: super +++++ 5.
80 )
81
82 testBasicKeyword = (
83     self assert: 1111 equals: (self keyword: 3).
84     self assert: 2222 equals: (super keyword: 5).
85 )
86
87 testWithBinaryUnaryMessage = (
88     | val |
89     record := 0.
90     val := super number * super number.
91     self assert: 1 equals: val.
92 )
93
94 testWithBinaryUnaryUnaryMessage = (
95     | val |
96     record := 0.
97     super yourself yourself @ super yourself yourself.
98     self assert: 2002 equals: record.
99 )
100
101 testWithKeywordUnaryUnaryMessage = (
102     | val |
103     record := 0.
104     super key: super yourself yourself key: super yourself yourself.
105     self assert: 2002 equals: record.
106
107     record := 0.
108     self key: super yourself yourself key: super yourself yourself.
109     self assert: 2002 equals: record.
110 )
111
112 "Note: testing assigning self was moved to basic interpreter tests"
113
114 testGlobalSelfDoesNotShadowKeyword = (

```

```
115     | that |
116     that := self.
117     system global: #self put: 42.
118     that optional: #selfSuperBug assert: that is: self.
119
120     self assert: 42 equals: (system global: #self)
121 )
122
123 testGlobalSuperDoesNotShadowKeyword = (
124     | that |
125     that := super.
126     system global: #super put: 42.
127     that optional: #selfSuperBug assert: that is: super.
128
129     self assert: 42 equals: (system global: #super)
130 )
131 )
132
```

```
1 "  
2  
3 $Id: SuperTestSuperClass.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SuperTestSuperClass = TestCase (  
29   | record |  
30  
31   yourself = (  
32     record := record + 1.  
33     ^ self  
34   )  
35  
36   give42 = (  
37     ^ 42  
38   )  
39  
40   something = (  
41     ^ #super  
42   )  
43  
44   number = (  
45     ^ 1  
46   )  
47  
48   + other = (  
49     ^ 22  
50   )  
51  
52   ++++ other = (  
53     ^ 222  
54   )
```

```
55
56     keyword: other = (
57         ^ 2222
58     )
59
60     key: a key: b = (
61         ^ self
62     )
63
64     @ o = ( ^ self )
65 )
66
67
```

```
1 "  
2  
3 $Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Symbol2Test = TestCase (  
29  
30   testConversion = (  
31     self assert: 'gunk' equals: 'gunk' asSymbol asString.  
32     self assert: 'oink' equals: #oink asString.  
33   )  
34  
35   testEquality = (  
36     self assert: #oink == #oink.  
37     self assert: #oink == 'oink' asSymbol.  
38     self assert: #oink = #oink.  
39     self assert: #oink = 'oink' asSymbol.  
40  
41     self deny: #foo = #fooo.  
42     self deny: #foo == #fooo.  
43  
44     self assert: #foo = 'foo'.  
45     self deny: #foo == 'fooo'.  
46     self deny: #foo == #foo asString.  
47   )  
48  
49   testSymbolIsString = (  
50     self assert: (#oink beginsWith: 'oink').  
51     self assert: 100 equals: #'100' asInteger.  
52     self assert: String equals: #foo class superclass  
53   )  
54 )
```



```
1 "  
2  
3 $Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Symbol3Test = TestCase (  
29  
30   testConversion = (  
31     self assert: 'gunk' equals: 'gunk' asSymbol asString.  
32     self assert: 'oink' equals: #oink asString.  
33   )  
34  
35   testEquality = (  
36     self assert: #oink == #oink.  
37     self assert: #oink == 'oink' asSymbol.  
38     self assert: #oink = #oink.  
39     self assert: #oink = 'oink' asSymbol.  
40  
41     self deny: #foo = #fooo.  
42     self deny: #foo == #fooo.  
43  
44     self assert: #foo = 'foo'.  
45     self deny: #foo == 'fooo'.  
46     self deny: #foo == #foo asString.  
47   )  
48  
49   testSymbolIsString = (  
50     self assert: (#oink beginsWith: 'oink').  
51     self assert: 100 equals: #'100' asInteger.  
52     self assert: String equals: #foo class superclass  
53   )  
54 )
```



```
1 "  
2  
3 $Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Symbol4Test = TestCase (  
29  
30   testConversion = (  
31     self assert: 'gunk' equals: 'gunk' asSymbol asString.  
32     self assert: 'oink' equals: #oink asString.  
33   )  
34  
35   testEquality = (  
36     self assert: #oink == #oink.  
37     self assert: #oink == 'oink' asSymbol.  
38     self assert: #oink = #oink.  
39     self assert: #oink = 'oink' asSymbol.  
40  
41     self deny: #foo = #fooo.  
42     self deny: #foo == #fooo.  
43  
44     self assert: #foo = 'foo'.  
45     self deny: #foo == 'fooo'.  
46     self deny: #foo == #foo asString.  
47   )  
48  
49   testSymbolIsString = (  
50     self assert: (#oink beginsWith: 'oink').  
51     self assert: 100 equals: #'100' asInteger.  
52     self assert: String equals: #foo class superclass  
53   )  
54 )
```



```
1 "  
2  
3 $Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Symbol5Test = TestCase (  
29  
30   testConversion = (  
31     self assert: 'gunk' equals: 'gunk' asSymbol asString.  
32     self assert: 'oink' equals: #oink asString.  
33   )  
34  
35   testEquality = (  
36     self assert: #oink == #oink.  
37     self assert: #oink == 'oink' asSymbol.  
38     self assert: #oink = #oink.  
39     self assert: #oink = 'oink' asSymbol.  
40  
41     self deny: #foo = #fooo.  
42     self deny: #foo == #fooo.  
43  
44     self assert: #foo = 'foo'.  
45     self deny: #foo == 'fooo'.  
46     self deny: #foo == #foo asString.  
47   )  
48  
49   testSymbolIsString = (  
50     self assert: (#oink beginsWith: 'oink').  
51     self assert: 100 equals: #'100' asInteger.  
52     self assert: String equals: #foo class superclass  
53   )  
54 )
```



```
1 "  
2  
3 $Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SymbolTest = TestCase (  
29  
30   testConversion = (  
31     self assert: 'gunk' equals: 'gunk' asSymbol asString.  
32     self assert: 'oink' equals: #oink asString.  
33   )  
34  
35   testEquality = (  
36     self assert: #oink == #oink.  
37     self assert: #oink == 'oink' asSymbol.  
38     self assert: #oink = #oink.  
39     self assert: #oink = 'oink' asSymbol.  
40  
41     self deny: #foo = #fooo.  
42     self deny: #foo == #fooo.  
43  
44     self assert: #foo = 'foo'.  
45     self deny: #foo == 'fooo'.  
46     self deny: #foo == #foo asString.  
47   )  
48  
49   testSymbolIsString = (  
50     self assert: (#oink beginsWith: 'oink').  
51     self assert: 100 equals: #'100' asInteger.  
52     self assert: String equals: #foo class superclass  
53   )  
54 )
```



```
1 System2Test = TestCase (
2
3   testFullGCsupport = (
4     "Test whether #fullGC is support. We expect the VM now to return true,
5       to indicate the a GC was done."
6     self optional: #fullGCWithEffect assert: system fullGC description:
7     '#fullGC is not supported or has not immediate effect.'
8   )
9   testTicks = (
10    | ticks |
11    ticks := system ticks.
12    self assert: ticks class equals: Integer.
13    self assert: ticks > 0 description: 'Should return the microseconds
14    since the start'
15  )
16 )
```



```
1 System3Test = TestCase (
2
3   testFullGCsupport = (
4     "Test whether #fullGC is support. We expect the VM now to return true,
5       to indicate the a GC was done."
6     self optional: #fullGCWithEffect assert: system fullGC description:
7     '#fullGC is not supported or has not immediate effect.'
8   )
9   testTicks = (
10    | ticks |
11    ticks := system ticks.
12    self assert: ticks class equals: Integer.
13    self assert: ticks > 0 description: 'Should return the microseconds
14    since the start'
15  )
16 )
```

```
1 System4Test = TestCase (
2
3   testFullGCsupport = (
4     "Test whether #fullGC is support. We expect the VM now to return true,
5       to indicate the a GC was done."
6     self optional: #fullGCWithEffect assert: system fullGC description:
7     '#fullGC is not supported or has not immediate effect.'
8   )
9   testTicks = (
10    | ticks |
11    ticks := system ticks.
12    self assert: ticks class equals: Integer.
13    self assert: ticks > 0 description: 'Should return the microseconds
14    since the start'
15  )
16 )
```

```
1 System5Test = TestCase (
2
3   testFullGCsupport = (
4     "Test whether #fullGC is support. We expect the VM now to return true,
5       to indicate the a GC was done."
6     self optional: #fullGCWithEffect assert: system fullGC description:
7     '#fullGC is not supported or has not immediate effect.'
8   )
9   testTicks = (
10    | ticks |
11    ticks := system ticks.
12    self assert: ticks class equals: Integer.
13    self assert: ticks > 0 description: 'Should return the microseconds
since the start'
14  )
15 )
16
```

```
1 SystemTest = TestCase (
2
3   testFullGCsupport = (
4     "Test whether #fullGC is support. We expect the VM now to return true,
5     to indicate the a GC was done."
6     self optional: #fullGCWithEffect assert: system fullGC description:
7     '#fullGC is not supported or has not immediate effect.'
8   )
9   testTicks = (
10    | ticks |
11    ticks := system ticks.
12    self assert: ticks class equals: Integer.
13    self assert: ticks > 0 description: 'Should return the microseconds
since the start'
14  )
15 )
16
```

```
1 "  
2  
3 $Id: TestHarness.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Test = TestCommon (  
27     tests = ( "Now ordered by alphabetical order to improve  
maintainability"  
28         ^ EmptyTest,  
29           SpecialSelectorsTest,  
30           SpecialSelectors2Test,  
31           SpecialSelectors3Test,  
32           SpecialSelectors4Test,  
33           SpecialSelectors5Test,  
34           ArrayTest,  
35           Array2Test,  
36           Array3Test,  
37           Array4Test,  
38           Array5Test,  
39           BlockTest,  
40           Block2Test,  
41           Block3Test,  
42           Block4Test,  
43           Block5Test,  
44           BooleanTest,  
45           Boolean2Test,  
46           Boolean3Test,  
47           Boolean4Test,  
48           Boolean5Test,  
49           ClassLoadingTest,  
50           ClassLoading2Test,  
51           ClassLoading3Test,  
52           ClassLoading4Test,
```

53 ClassLoading5Test,
54 ClassStructureTest,
55 ClassStructure2Test,
56 ClassStructure3Test,
57 ClassStructure4Test,
58 ClassStructure5Test,
59 ClosureTest,
60 Closure2Test,
61 Closure3Test,
62 Closure4Test,
63 Closure5Test,
64 CoercionTest,
65 Coercion2Test,
66 Coercion3Test,
67 Coercion4Test,
68 Coercion5Test,
69 CompilerReturnTest,
70 CompilerReturn2Test,
71 CompilerReturn3Test,
72 CompilerReturn4Test,
73 CompilerReturn5Test,
74 DictionaryTest,
75 Dictionary2Test,
76 Dictionary3Test,
77 Dictionary4Test,
78 Dictionary5Test,
79 DoesNotUnderstandTest,
80 DoesNotUnderstand2Test,
81 DoesNotUnderstand3Test,
82 DoesNotUnderstand4Test,
83 DoesNotUnderstand5Test,
84 DoubleTest,
85 Double2Test,
86 Double3Test,
87 Double4Test,
88 Double5Test,
89 GlobalTest,
90 Global2Test,
91 Global3Test,
92 Global4Test,
93 Global5Test,
94 HashTest,
95 Hash2Test,
96 Hash3Test,
97 Hash4Test,
98 Hash5Test,
99 IntegerTest,
100 Integer2Test,
101 Integer3Test,
102 Integer4Test,
103 Integer5Test,
104 PreliminaryTest,
105 Preliminary2Test,
106 Preliminary3Test,
107 Preliminary4Test,
108 Preliminary5Test,
109 ReflectionTest,
110 Reflection2Test,
111 Reflection3Test,
112 Reflection4Test,
113 Reflection5Test,

```
114         SelfBlockTest,
115         SelfBlock2Test,
116         SelfBlock3Test,
117         SelfBlock4Test,
118         SelfBlock5Test,
119         SetTest,
120         Set2Test,
121         Set3Test,
122         Set4Test,
123         Set5Test,
124         StringTest,
125         String2Test,
126         String3Test,
127         String4Test,
128         String5Test,
129         SuperTest,
130         Super2Test,
131         Super3Test,
132         Super4Test,
133         Super5Test,
134         SymbolTest,
135         Symbol2Test,
136         Symbol3Test,
137         Symbol4Test,
138         Symbol5Test,
139         SystemTest,
140         System2Test,
141         System3Test,
142         System4Test,
143         System5Test,
144         VectorTest,
145         Vector2Test,
146         Vector3Test,
147         Vector4Test,
148         Vector5Test
149     )
150
151     numExecs = (
152     ^ 5
153     )
154 )
155
```

```

1 TestCase = (
2   | testSelector runner failed |
3
4   selector      = ( ^ testSelector )
5   selector: aSym = ( testSelector := aSym )
6
7   "asserting"
8   assert: aBoolean = (
9     runner countAssert.
10    aBoolean ifFalse: [
11      self signalFailure: 'Assertion failed' ] )
12
13   assert: aBoolean description: aStringOrBlock = (
14     runner countAssert.
15     aBoolean ifFalse: [
16       self signalFailure: aStringOrBlock value ] )
17
18   assert: expected equals: actual = (
19     "test value equality"
20     self assert: (expected = actual)
21     description: [self comparingStringBetween: expected and:
actual]
22   )
23
24   assert: expected is: actual = (
25     "test reference equality"
26     self assert: (expected == actual)
27     description: [self comparingStringBetween: expected and:
actual]
28   )
29
30   optional: aSymbol assert: aBoolean = (
31     runner countAssert.
32     aBoolean ifFalse: [
33       self signalUnsupported: aSymbol description: nil ] )
34
35   optional: aSymbol assert: expected equals: actual = (
36     self optional: aSymbol
37     assert: (expected = actual)
38     description: [self comparingStringBetween: expected and:
actual]
39   )
40
41   optional: aSymbol assert: expected is: actual = (
42     self optional: aSymbol
43     assert: (expected == actual)
44     description: [self comparingStringBetween: expected and:
actual]
45   )
46
47   optional: aSymbol assert: aBoolean description: aStringOrBlock = (
48     runner countAssert.
49     aBoolean ifFalse: [
50       self signalUnsupported: aSymbol description: aStringOrBlock
value ] )
51
52   deny: aBoolean = (
53     self assert: aBoolean not

```



```

54     )
55
56     deny: aBooleanOrBlock description: aString = (
57         self assert: aBooleanOrBlock value not description: aString
58     )
59
60     optional: aSymbol deny: aBoolean = (
61         self optional: aSymbol assert: aBoolean not
62     )
63
64     optional: aSymbol deny: aBooleanOrBlock description: aString = (
65         self optional: aSymbol assert: aBooleanOrBlock value not
description: aString
66     )
67
68     signalFailure: aString = (
69         failed := true.
70         runner fail: self class name + '>>#' + testSelector
71             because: aString.
72     )
73
74     signalUnsupported: aSymbol description: aDescription = (
75         runner unsupported: aSymbol
76             test: self class name + '>>#' + testSelector
77             because: aDescription.
78     )
79
80     comparingStringBetween: expected and: actual = (
81         ^ 'Expected ' + expected asString +
82         ' but was ' + actual asString + '.'
83     )
84
85     "running"
86     run: aRunner = (
87         runner := aRunner.
88         failed := false.
89
90         self setUp.
91         self performTest.
92         self tearDown.
93
94         failed ifFalse: [
95             runner passed: self class name + '>>#' + testSelector
96         ].
97     )
98
99     setUp = ()
100    tearDown = ()
101
102    performTest = ( self perform: testSelector )
103
104    ----
105
106    for: aSelector = (
107        | case |
108        case := self new.
109        case selector: aSelector.
110        ^ case
111    )
112
113    tests = (

```

```
114         | tests |
115         tests := Vector new: self methods length.
116         self methods do: [:m |
117             (m signature beginsWith: #test) ifTrue: [
118                 tests append: (self for: m signature).
119             ].
120         ].
121
122         ^ tests
123     )
124 )
125
```

```
1 "  
2  
3 $Id: TestHarness.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 TestCommon = Benchmark (  
27     | failOnUnsupportedOptionals |  
28  
29     tests = ( "Now ordered by alphabetical order to improve  
maintainability"  
30         self error: 'Implement in subclass'  
31     )  
32  
33     numExecs = (  
34         self error: 'Implement in subclass and return the number of times a  
test is executed'  
35     )  
36  
37     oneTimeSetup = (  
38         "Load all Tests. We don't really want to benchmark the parser."  
39         self tests  
40     )  
41  
42     runAllSuites = (  
43         | totalTestNum successfulTestNum unsupportedTestNum  
totalAssertionNum arr |  
44         totalTestNum := 0.  
45         unsupportedTestNum := 0.  
46         successfulTestNum := 0.  
47         totalAssertionNum := 0.  
48  
49         self tests do: [ :test |  
50             | runner |  
51             runner := TestRunner new.
```

```

52         runner initializeOn: test.
53         runner runAllTests.
54
55         totalTestNum      := totalTestNum + runner expectedPasses.
56         unsupportedTestNum := unsupportedTestNum + runner
actualUnsupported.
57         successfulTestNum := successfulTestNum + runner actualPasses.
58         totalAssertionNum := totalAssertionNum + runner numAsserts.
59     ].
60
61     arr := Array new: 4.
62     arr at: 1 put: totalTestNum.
63     arr at: 2 put: unsupportedTestNum.
64     arr at: 3 put: successfulTestNum.
65     arr at: 4 put: totalAssertionNum.
66     ^ arr
67 )
68
69 runOneSuite: name = (
70     | testName runner |
71     testName := name.
72     (testName endsWith: 'Test') ifFalse: [
73         testName := testName + 'Test'].
74
75     runner := TestRunner new.
76     runner initializeOn: (system resolve: testName asSymbol).
77     runner run.
78     runner hasFailures ifTrue: [system exit: 1]
79 )
80
81 benchmark = (
82     failOnUnsupportedOptionals := false.
83     ^ self runAllSuites
84 )
85
86 verifyResult: result = (
87     "result do: [:e | e println ]."
88     ^ (result at: 1) = (190 * self numExecs) and: [
89         (result at: 2) = (0 * self numExecs) and: [
90             (result at: 3) = (189 * self numExecs) and: [
91                 (result at: 4) = (992 * self numExecs)
92             ] ] ]
93 )
94 )
95

```

```
1 "  
2 Copyright (c) 2021 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 TestGC = TestGCCCommon (  
24     tests = ( "Now ordered by alphabetical order to improve  
maintainability"  
25         ^ EmptyTest,  
26           SpecialSelectorsTest,  
27           SpecialSelectors2Test,  
28           SpecialSelectors3Test,  
29           SpecialSelectors4Test,  
30           SpecialSelectors5Test,  
31           ArrayTest,  
32           Array2Test,  
33           Array3Test,  
34           Array4Test,  
35           Array5Test,  
36           BlockTest,  
37           Block2Test,  
38           Block3Test,  
39           Block4Test,  
40           Block5Test,  
41           BooleanTest,  
42           Boolean2Test,  
43           Boolean3Test,  
44           Boolean4Test,  
45           Boolean5Test,  
46           ClassLoadingTest,  
47           ClassLoading2Test,  
48           ClassLoading3Test,  
49           ClassLoading4Test,  
50           ClassLoading5Test,  
51           ClassStructureTest,  
52           ClassStructure2Test,
```

53 ClassStructure3Test,
54 ClassStructure4Test,
55 ClassStructure5Test,
56 ClosureTest,
57 Closure2Test,
58 Closure3Test,
59 Closure4Test,
60 Closure5Test,
61 CoercionTest,
62 Coercion2Test,
63 Coercion3Test,
64 Coercion4Test,
65 Coercion5Test,
66 CompilerReturnTest,
67 CompilerReturn2Test,
68 CompilerReturn3Test,
69 CompilerReturn4Test,
70 CompilerReturn5Test,
71 DictionaryTest,
72 Dictionary2Test,
73 Dictionary3Test,
74 Dictionary4Test,
75 Dictionary5Test,
76 DoesNotUnderstandTest,
77 DoesNotUnderstand2Test,
78 DoesNotUnderstand3Test,
79 DoesNotUnderstand4Test,
80 DoesNotUnderstand5Test,
81 DoubleTest,
82 Double2Test,
83 Double3Test,
84 Double4Test,
85 Double5Test,
86 GlobalTest,
87 Global2Test,
88 Global3Test,
89 Global4Test,
90 Global5Test,
91 HashTest,
92 Hash2Test,
93 Hash3Test,
94 Hash4Test,
95 Hash5Test,
96 IntegerTest,
97 Integer2Test,
98 Integer3Test,
99 Integer4Test,
100 Integer5Test,
101 PreliminaryTest,
102 Preliminary2Test,
103 Preliminary3Test,
104 Preliminary4Test,
105 Preliminary5Test,
106 ReflectionTest,
107 Reflection2Test,
108 Reflection3Test,
109 Reflection4Test,
110 Reflection5Test,
111 SelfBlockTest,
112 SelfBlock2Test,
113 SelfBlock3Test,

```
114         SelfBlock4Test,
115         SelfBlock5Test,
116         SetTest,
117         Set2Test,
118         Set3Test,
119         Set4Test,
120         Set5Test,
121         StringTest,
122         String2Test,
123         String3Test,
124         String4Test,
125         String5Test,
126         SuperTest,
127         Super2Test,
128         Super3Test,
129         Super4Test,
130         Super5Test,
131         SymbolTest,
132         Symbol2Test,
133         Symbol3Test,
134         Symbol4Test,
135         Symbol5Test,
136         SystemTest,
137         System2Test,
138         System3Test,
139         System4Test,
140         System5Test,
141         VectorTest,
142         Vector2Test,
143         Vector3Test,
144         Vector4Test,
145         Vector5Test
146     )
147
148     benchmark = (
149         ^ system fullGC
150     )
151 )
152
```

```
1 "  
2 Copyright (c) 2021 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 TestGCCCommon = Benchmark (  
24   | failOnUnsupportedOptionals |  
25  
26   tests = ( "Now ordered by alphabetical order to improve  
maintainability"  
27     self error: 'Implement in subclass'  
28   )  
29  
30   oneTimeSetup = (  
31     "Load all Tests. We don't really want to benchmark the parser."  
32     self tests  
33   )  
34  
35   runAllSuites = (  
36     | totalTestNum successfulTestNum unsupportedTestNum  
totalAssertionNum arr |  
37     totalTestNum := 0.  
38     unsupportedTestNum := 0.  
39     successfulTestNum := 0.  
40     totalAssertionNum := 0.  
41  
42     self tests do: [ :test |  
43       | runner |  
44       runner := TestRunner new.  
45       runner initializeOn: test.  
46       runner runAllTests.  
47  
48       totalTestNum      := totalTestNum + runner expectedPasses.  
49       unsupportedTestNum := unsupportedTestNum + runner  
actualUnsupported.  
50       successfulTestNum := successfulTestNum + runner actualPasses.  
51       totalAssertionNum := totalAssertionNum + runner numAsserts.
```



```

52     ].
53
54     arr := Array new: 4.
55     arr at: 1 put: totalTestNum.
56     arr at: 2 put: unsupportedTestNum.
57     arr at: 3 put: successfulTestNum.
58     arr at: 4 put: totalAssertionNum.
59     ^ arr
60 )
61
62 runOneSuite: name = (
63     | testName runner |
64     testName := name.
65     (testName endsWith: 'Test') ifFalse: [
66         testName := testName + 'Test'].
67
68     runner := TestRunner new.
69     runner initializeOn: (system resolve: testName asSymbol).
70     runner run.
71     runner hasFailures ifTrue: [system exit: 1]
72 )
73
74 benchmark = (
75     ^ system fullGC
76 )
77
78 verifyResult: result = (
79     "We expect the GC to actually promise us that it did work"
80     ^ result
81 )
82 )
83

```

```

1 TestRunner = (
2   | suite passes unsupported failures numAsserts |
3
4   initializeOn: aSuite = (
5     suite := aSuite.
6
7     passes      := Vector new.
8     unsupported  := Vector new.
9     failures     := Vector new.
10
11     numAsserts := 0.
12   )
13
14   hasUnsupported = ( ^ unsupported size > 0 )
15   hasFailures    = ( ^ failures size > 0 )
16
17   actualUnsupported = ( ^ unsupported size )
18   expectedPasses   = ( ^ suite tests size )
19   actualPasses     = ( ^ passes size )
20
21   run = (
22     self reportPreRun.
23     self runAllTests.
24     self reportPostRun.
25     self overviewReport.
26   )
27
28   countAssert = (
29     numAsserts := numAsserts + 1.
30   )
31
32   numAsserts = (
33     ^ numAsserts
34   )
35
36   reportPreRun = (
37     ('TestSuite ' + suite name + ':') println.
38     ('Tests: ' + suite tests size asString) println.
39   )
40
41   reportPostRun = (
42     self hasUnsupported ifTrue: [
43       ('Unsupported optional: ' + unsupported size asString) println
44     ].
45     self hasFailures ifTrue: [
46       ('Failures: ' + failures size asString) println
47     ].
48   )
49
50   runAllTests = (
51     suite tests do: [ :each |
52       each run: self ].
53   )
54
55   overviewReport = (
56     ('Tests passed: ' + passes size asString) println.
57
58     (self hasFailures or: [self hasUnsupported]) ifTrue: [

```

```

59         '-----' println ].
60
61     self hasUnsupported ifTrue: [
62         | lastCategory |
63         ('Unsupported optional features: ' + unsupported size asString)
println.
64         unsupported do: [:each |
65             | cat |
66             cat := each at: 1.
67             cat == lastCategory ifFalse: [
68                 lastCategory := cat.
69                 ('\t' + cat) println ].
70             ('\t\t' + (each at: 2) asString) println.
71             ('\t\t\t' + (each at: 3) value asString) println ].
72     ].
73
74     self hasFailures ifTrue: [
75         ('Failures: ' + failures size asString) println.
76         failures do: [:each |
77             (' ' + each key asString) println.
78             (' ' + each value asString) println ].
79     ].
80 )
81
82 fail: aSignature because: aReason = (
83     | pair |
84     pair := Pair withKey: aSignature andValue: aReason.
85     failures append: pair.
86 )
87
88 unsupported: aSymbol test: aSignature because: aReason = (
89     | array |
90     array := Array with: aSymbol with: aSignature with: aReason.
91     unsupported append: array.
92 )
93
94 passed: aSignature = (
95     passes append: aSignature
96 )
97 )
98

```

```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Vector2Test = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Vector new.  
33     a append: 'hello'.  
34     a append: #world.  
35     a append: 23.  
36   )  
37  
38   testSize = (  
39     self assert: 3 equals: a size.  
40   )  
41  
42   testAt = (  
43     self assert: #world equals: (a at: 2).  
44     self assert:      23 equals: (a at: 3).  
45   )  
46  
47   testFirst = (  
48     | v |  
49     self assert: 'hello' equals: a first.  
50  
51     v := Vector new.  
52     1 to: 10 do: [:i |  
53       v append: i.
```

```

54         self assert: 1 equals: v first ].
55
56     1 to: 10 do: [:i |
57         self assert: 1 equals: v first.
58         v removeFirst ]
59 )
60
61 testLast = (
62     | v |
63     self assert: 23 equals: a last.
64
65     v := Vector new.
66     1 to: 10 do: [:i |
67         v append: i.
68         self assert: i equals: v last ].
69
70     10 downTo: 1 do: [:i |
71         self assert: i equals: v last.
72         v remove ]
73 )
74
75 testContains = (
76     self assert: (a contains: 23).
77     self deny: (a contains: #nono).
78 )
79
80 testAppendAndRemoveFirst = (
81     | v |
82     v := Vector new: 10.
83     1 to: 100 do: [:i |
84         v append: i ].
85
86     "160 is implementation dependent, just here for orientation"
87     self assert: 160 equals: v capacity.
88     self assert: 100 equals: v size.
89
90     1 to: 100 do: [:i |
91         v removeFirst ].
92     1 to: 100 do: [:i |
93         v append: i ].
94
95     self assert: 320 equals: v capacity.
96     self assert: 100 equals: v size.
97 )
98
99 testIndexOf = (
100     | v |
101     v := Vector new: 3.
102     1 to: 17 do: [:i |
103         v append: i ].
104
105     self assert: -1 equals: (v indexOf: nil).
106     self assert: 1 equals: (v indexOf: 1).
107
108     self assert: 13 equals: (v indexOf: 13).
109     v at: 13 put: #test.
110
111     self assert: 13 equals: (v indexOf: #test).
112
113     v removeFirst.
114     self assert: -1 equals: (v indexOf: 1).

```

```

115
116     1 to: 12 do: [:i |
117         v removeFirst ].
118     self assert: -1 equals: (v indexOf: #test).
119 )
120
121 testAppendAll = (
122     | v c i |
123     v := Vector new: 2.
124     v append: 1.
125     v append: 2.
126     v append: 3.
127
128     c := Array with: 4 with: 5 with: 6.
129
130     v appendAll: c.
131     i := 1.
132
133     v do: [:e |
134         self assert: i equals: e.
135         i := i + 1 ]
136 )
137
138 testAsArray = (
139     | v arr |
140     v := Vector new.
141     self assert: 0 equals: v asArray length.
142
143     v append: 1.
144     v append: 2.
145
146     arr := v asArray.
147     self assert: 2 equals: arr length.
148     self assert: 1 equals: (arr at: 1).
149     self assert: 2 equals: (arr at: 2).
150 )
151
152 testAsSet = (
153     | v set |
154     v := Vector new.
155     v append: 1.
156     v append: 2.
157     v append: 3.
158     v append: 4.
159     self assert: 4 equals: v size.
160
161     set := v asSet.
162     self assert: 4 equals: set size.
163
164     v append: 1.
165     v append: 1.
166     v append: 1.
167
168     self assert: 4 + 3 equals: v size.
169
170     set := v asSet.
171     self assert: 4 equals: set size.
172 )
173
174 testIsEmpty = (
175     | v |

```

```

176     v := Vector new.
177     self assert: v isEmpty.
178
179     v append: 1.
180     self deny: v isEmpty.
181
182     v removeFirst.
183     self assert: v isEmpty.
184
185     v append: #ee.
186     self deny: v isEmpty.
187
188     v removeFirst.
189     self assert: v isEmpty.
190 )
191
192 testRemoveObj = (
193     | v |
194     v := Vector new.
195     v append: #a.
196     v append: #b.
197     v append: #c.
198     v append: #d.
199     v append: #e.
200     v append: #f.
201     v append: #g.
202     v append: #h.
203
204     self assert: 8 equals: v size.
205
206     self deny: (v remove: #aa).
207     self assert: (v remove: #e).
208     self assert: 7 equals: v size.
209 )
210
211 testAppendComma = (
212     | v |
213     v := Vector new.
214     v, #a.
215     v, #b.
216
217     self assert: 2 equals: v size.
218     self assert: (v contains: #a).
219     self assert: (v contains: #b).
220 )
221
222 testDoIndexes = (
223     | i v |
224     v := Vector new.
225     v doIndexes: [:j |
226         self assert: false ].
227
228     v appendAll: #(1 2 3 4 5).
229     i := 1.
230     v doIndexes: [:j |
231         self assert: i equals: j.
232         i := i + 1.
233     ].
234     self assert: 6 equals: i.
235 )
236

```

```
237 testDo = (  
238   | i v |  
239   v := Vector new.  
240   v appendAll: #(1 2 3 4 5).  
241   i := 1.  
242   v do: [:v |  
243     self assert: i equals: v.  
244     i := i + 1.  
245   ].  
246   self assert: 6 equals: i.  
247 )  
248 )  
249
```



```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Vector3Test = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Vector new.  
33     a append: 'hello'.  
34     a append: #world.  
35     a append: 23.  
36   )  
37  
38   testSize = (  
39     self assert: 3 equals: a size.  
40   )  
41  
42   testAt = (  
43     self assert: #world equals: (a at: 2).  
44     self assert:      23 equals: (a at: 3).  
45   )  
46  
47   testFirst = (  
48     | v |  
49     self assert: 'hello' equals: a first.  
50  
51     v := Vector new.  
52     1 to: 10 do: [:i |  
53       v append: i.
```

```

54         self assert: 1 equals: v first ].
55
56     1 to: 10 do: [:i |
57         self assert: 1 equals: v first.
58         v removeFirst ]
59 )
60
61 testLast = (
62     | v |
63     self assert: 23 equals: a last.
64
65     v := Vector new.
66     1 to: 10 do: [:i |
67         v append: i.
68         self assert: i equals: v last ].
69
70     10 downTo: 1 do: [:i |
71         self assert: i equals: v last.
72         v remove ]
73 )
74
75 testContains = (
76     self assert: (a contains: 23).
77     self deny: (a contains: #nono).
78 )
79
80 testAppendAndRemoveFirst = (
81     | v |
82     v := Vector new: 10.
83     1 to: 100 do: [:i |
84         v append: i ].
85
86     "160 is implementation dependent, just here for orientation"
87     self assert: 160 equals: v capacity.
88     self assert: 100 equals: v size.
89
90     1 to: 100 do: [:i |
91         v removeFirst ].
92     1 to: 100 do: [:i |
93         v append: i ].
94
95     self assert: 320 equals: v capacity.
96     self assert: 100 equals: v size.
97 )
98
99 testIndexOf = (
100     | v |
101     v := Vector new: 3.
102     1 to: 17 do: [:i |
103         v append: i ].
104
105     self assert: -1 equals: (v indexOf: nil).
106     self assert: 1 equals: (v indexOf: 1).
107
108     self assert: 13 equals: (v indexOf: 13).
109     v at: 13 put: #test.
110
111     self assert: 13 equals: (v indexOf: #test).
112
113     v removeFirst.
114     self assert: -1 equals: (v indexOf: 1).

```

```

115
116     1 to: 12 do: [:i |
117         v removeFirst ].
118     self assert: -1 equals: (v indexOf: #test).
119 )
120
121 testAppendAll = (
122     | v c i |
123     v := Vector new: 2.
124     v append: 1.
125     v append: 2.
126     v append: 3.
127
128     c := Array with: 4 with: 5 with: 6.
129
130     v appendAll: c.
131     i := 1.
132
133     v do: [:e |
134         self assert: i equals: e.
135         i := i + 1 ]
136 )
137
138 testAsArray = (
139     | v arr |
140     v := Vector new.
141     self assert: 0 equals: v asArray length.
142
143     v append: 1.
144     v append: 2.
145
146     arr := v asArray.
147     self assert: 2 equals: arr length.
148     self assert: 1 equals: (arr at: 1).
149     self assert: 2 equals: (arr at: 2).
150 )
151
152 testAsSet = (
153     | v set |
154     v := Vector new.
155     v append: 1.
156     v append: 2.
157     v append: 3.
158     v append: 4.
159     self assert: 4 equals: v size.
160
161     set := v asSet.
162     self assert: 4 equals: set size.
163
164     v append: 1.
165     v append: 1.
166     v append: 1.
167
168     self assert: 4 + 3 equals: v size.
169
170     set := v asSet.
171     self assert: 4 equals: set size.
172 )
173
174 testIsEmpty = (
175     | v |

```

```

176     v := Vector new.
177     self assert: v isEmpty.
178
179     v append: 1.
180     self deny: v isEmpty.
181
182     v removeFirst.
183     self assert: v isEmpty.
184
185     v append: #ee.
186     self deny: v isEmpty.
187
188     v removeFirst.
189     self assert: v isEmpty.
190 )
191
192 testRemoveObj = (
193     | v |
194     v := Vector new.
195     v append: #a.
196     v append: #b.
197     v append: #c.
198     v append: #d.
199     v append: #e.
200     v append: #f.
201     v append: #g.
202     v append: #h.
203
204     self assert: 8 equals: v size.
205
206     self deny: (v remove: #aa).
207     self assert: (v remove: #e).
208     self assert: 7 equals: v size.
209 )
210
211 testAppendComma = (
212     | v |
213     v := Vector new.
214     v, #a.
215     v, #b.
216
217     self assert: 2 equals: v size.
218     self assert: (v contains: #a).
219     self assert: (v contains: #b).
220 )
221
222 testDoIndexes = (
223     | i v |
224     v := Vector new.
225     v doIndexes: [:j |
226         self assert: false ].
227
228     v appendAll: #(1 2 3 4 5).
229     i := 1.
230     v doIndexes: [:j |
231         self assert: i equals: j.
232         i := i + 1.
233     ].
234     self assert: 6 equals: i.
235 )
236

```

```
237 testDo = (  
238   | i v |  
239   v := Vector new.  
240   v appendAll: #(1 2 3 4 5).  
241   i := 1.  
242   v do: [:v |  
243     self assert: i equals: v.  
244     i := i + 1.  
245   ].  
246   self assert: 6 equals: i.  
247 )  
248 )  
249
```

```

1 "
2
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2007-2013 see AUTHORS file
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7 http://www.hpi.uni-potsdam.de/swa/
8
9 Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 Vector4Test = TestCase (
29   | a |
30
31   setUp = (
32     a := Vector new.
33     a append: 'hello'.
34     a append: #world.
35     a append: 23.
36   )
37
38   testSize = (
39     self assert: 3 equals: a size.
40   )
41
42   testAt = (
43     self assert: #world equals: (a at: 2).
44     self assert:      23 equals: (a at: 3).
45   )
46
47   testFirst = (
48     | v |
49     self assert: 'hello' equals: a first.
50
51     v := Vector new.
52     1 to: 10 do: [:i |
53       v append: i.

```

```

54         self assert: 1 equals: v first ].
55
56     1 to: 10 do: [:i |
57         self assert: 1 equals: v first.
58         v removeFirst ]
59 )
60
61 testLast = (
62     | v |
63     self assert: 23 equals: a last.
64
65     v := Vector new.
66     1 to: 10 do: [:i |
67         v append: i.
68         self assert: i equals: v last ].
69
70     10 downTo: 1 do: [:i |
71         self assert: i equals: v last.
72         v remove ]
73 )
74
75 testContains = (
76     self assert: (a contains: 23).
77     self deny: (a contains: #nono).
78 )
79
80 testAppendAndRemoveFirst = (
81     | v |
82     v := Vector new: 10.
83     1 to: 100 do: [:i |
84         v append: i ].
85
86     "160 is implementation dependent, just here for orientation"
87     self assert: 160 equals: v capacity.
88     self assert: 100 equals: v size.
89
90     1 to: 100 do: [:i |
91         v removeFirst ].
92     1 to: 100 do: [:i |
93         v append: i ].
94
95     self assert: 320 equals: v capacity.
96     self assert: 100 equals: v size.
97 )
98
99 testIndexOf = (
100     | v |
101     v := Vector new: 3.
102     1 to: 17 do: [:i |
103         v append: i ].
104
105     self assert: -1 equals: (v indexOf: nil).
106     self assert: 1 equals: (v indexOf: 1).
107
108     self assert: 13 equals: (v indexOf: 13).
109     v at: 13 put: #test.
110
111     self assert: 13 equals: (v indexOf: #test).
112
113     v removeFirst.
114     self assert: -1 equals: (v indexOf: 1).

```

```

115
116     1 to: 12 do: [:i |
117         v removeFirst ].
118     self assert: -1 equals: (v indexOf: #test).
119 )
120
121 testAppendAll = (
122     | v c i |
123     v := Vector new: 2.
124     v append: 1.
125     v append: 2.
126     v append: 3.
127
128     c := Array with: 4 with: 5 with: 6.
129
130     v appendAll: c.
131     i := 1.
132
133     v do: [:e |
134         self assert: i equals: e.
135         i := i + 1 ]
136 )
137
138 testAsArray = (
139     | v arr |
140     v := Vector new.
141     self assert: 0 equals: v asArray length.
142
143     v append: 1.
144     v append: 2.
145
146     arr := v asArray.
147     self assert: 2 equals: arr length.
148     self assert: 1 equals: (arr at: 1).
149     self assert: 2 equals: (arr at: 2).
150 )
151
152 testAsSet = (
153     | v set |
154     v := Vector new.
155     v append: 1.
156     v append: 2.
157     v append: 3.
158     v append: 4.
159     self assert: 4 equals: v size.
160
161     set := v asSet.
162     self assert: 4 equals: set size.
163
164     v append: 1.
165     v append: 1.
166     v append: 1.
167
168     self assert: 4 + 3 equals: v size.
169
170     set := v asSet.
171     self assert: 4 equals: set size.
172 )
173
174 testIsEmpty = (
175     | v |

```



```

176     v := Vector new.
177     self assert: v isEmpty.
178
179     v append: 1.
180     self deny: v isEmpty.
181
182     v removeFirst.
183     self assert: v isEmpty.
184
185     v append: #ee.
186     self deny: v isEmpty.
187
188     v removeFirst.
189     self assert: v isEmpty.
190 )
191
192 testRemoveObj = (
193     | v |
194     v := Vector new.
195     v append: #a.
196     v append: #b.
197     v append: #c.
198     v append: #d.
199     v append: #e.
200     v append: #f.
201     v append: #g.
202     v append: #h.
203
204     self assert: 8 equals: v size.
205
206     self deny: (v remove: #aa).
207     self assert: (v remove: #e).
208     self assert: 7 equals: v size.
209 )
210
211 testAppendComma = (
212     | v |
213     v := Vector new.
214     v, #a.
215     v, #b.
216
217     self assert: 2 equals: v size.
218     self assert: (v contains: #a).
219     self assert: (v contains: #b).
220 )
221
222 testDoIndexes = (
223     | i v |
224     v := Vector new.
225     v doIndexes: [:j |
226         self assert: false ].
227
228     v appendAll: #(1 2 3 4 5).
229     i := 1.
230     v doIndexes: [:j |
231         self assert: i equals: j.
232         i := i + 1.
233     ].
234     self assert: 6 equals: i.
235 )
236

```

```
237 testDo = (  
238   | i v |  
239   v := Vector new.  
240   v appendAll: #(1 2 3 4 5).  
241   i := 1.  
242   v do: [:v |  
243     self assert: i equals: v.  
244     i := i + 1.  
245   ].  
246   self assert: 6 equals: i.  
247 )  
248 )  
249
```

```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 Vector5Test = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Vector new.  
33     a append: 'hello'.  
34     a append: #world.  
35     a append: 23.  
36   )  
37  
38   testSize = (  
39     self assert: 3 equals: a size.  
40   )  
41  
42   testAt = (  
43     self assert: #world equals: (a at: 2).  
44     self assert:      23 equals: (a at: 3).  
45   )  
46  
47   testFirst = (  
48     | v |  
49     self assert: 'hello' equals: a first.  
50  
51     v := Vector new.  
52     1 to: 10 do: [:i |  
53       v append: i.
```

```

54         self assert: 1 equals: v first ].
55
56     1 to: 10 do: [:i |
57         self assert: 1 equals: v first.
58         v removeFirst ]
59 )
60
61 testLast = (
62     | v |
63     self assert: 23 equals: a last.
64
65     v := Vector new.
66     1 to: 10 do: [:i |
67         v append: i.
68         self assert: i equals: v last ].
69
70     10 downTo: 1 do: [:i |
71         self assert: i equals: v last.
72         v remove ]
73 )
74
75 testContains = (
76     self assert: (a contains: 23).
77     self deny: (a contains: #nono).
78 )
79
80 testAppendAndRemoveFirst = (
81     | v |
82     v := Vector new: 10.
83     1 to: 100 do: [:i |
84         v append: i ].
85
86     "160 is implementation dependent, just here for orientation"
87     self assert: 160 equals: v capacity.
88     self assert: 100 equals: v size.
89
90     1 to: 100 do: [:i |
91         v removeFirst ].
92     1 to: 100 do: [:i |
93         v append: i ].
94
95     self assert: 320 equals: v capacity.
96     self assert: 100 equals: v size.
97 )
98
99 testIndexOf = (
100     | v |
101     v := Vector new: 3.
102     1 to: 17 do: [:i |
103         v append: i ].
104
105     self assert: -1 equals: (v indexOf: nil).
106     self assert: 1 equals: (v indexOf: 1).
107
108     self assert: 13 equals: (v indexOf: 13).
109     v at: 13 put: #test.
110
111     self assert: 13 equals: (v indexOf: #test).
112
113     v removeFirst.
114     self assert: -1 equals: (v indexOf: 1).

```

```

115
116     1 to: 12 do: [:i |
117         v removeFirst ].
118     self assert: -1 equals: (v indexOf: #test).
119 )
120
121 testAppendAll = (
122     | v c i |
123     v := Vector new: 2.
124     v append: 1.
125     v append: 2.
126     v append: 3.
127
128     c := Array with: 4 with: 5 with: 6.
129
130     v appendAll: c.
131     i := 1.
132
133     v do: [:e |
134         self assert: i equals: e.
135         i := i + 1 ]
136 )
137
138 testAsArray = (
139     | v arr |
140     v := Vector new.
141     self assert: 0 equals: v asArray length.
142
143     v append: 1.
144     v append: 2.
145
146     arr := v asArray.
147     self assert: 2 equals: arr length.
148     self assert: 1 equals: (arr at: 1).
149     self assert: 2 equals: (arr at: 2).
150 )
151
152 testAsSet = (
153     | v set |
154     v := Vector new.
155     v append: 1.
156     v append: 2.
157     v append: 3.
158     v append: 4.
159     self assert: 4 equals: v size.
160
161     set := v asSet.
162     self assert: 4 equals: set size.
163
164     v append: 1.
165     v append: 1.
166     v append: 1.
167
168     self assert: 4 + 3 equals: v size.
169
170     set := v asSet.
171     self assert: 4 equals: set size.
172 )
173
174 testIsEmpty = (
175     | v |

```

```

176     v := Vector new.
177     self assert: v isEmpty.
178
179     v append: 1.
180     self deny: v isEmpty.
181
182     v removeFirst.
183     self assert: v isEmpty.
184
185     v append: #ee.
186     self deny: v isEmpty.
187
188     v removeFirst.
189     self assert: v isEmpty.
190 )
191
192 testRemoveObj = (
193     | v |
194     v := Vector new.
195     v append: #a.
196     v append: #b.
197     v append: #c.
198     v append: #d.
199     v append: #e.
200     v append: #f.
201     v append: #g.
202     v append: #h.
203
204     self assert: 8 equals: v size.
205
206     self deny: (v remove: #aa).
207     self assert: (v remove: #e).
208     self assert: 7 equals: v size.
209 )
210
211 testAppendComma = (
212     | v |
213     v := Vector new.
214     v, #a.
215     v, #b.
216
217     self assert: 2 equals: v size.
218     self assert: (v contains: #a).
219     self assert: (v contains: #b).
220 )
221
222 testDoIndexes = (
223     | i v |
224     v := Vector new.
225     v doIndexes: [:j |
226         self assert: false ].
227
228     v appendAll: #(1 2 3 4 5).
229     i := 1.
230     v doIndexes: [:j |
231         self assert: i equals: j.
232         i := i + 1.
233     ].
234     self assert: 6 equals: i.
235 )
236

```

```
237 testDo = (  
238   | i v |  
239   v := Vector new.  
240   v appendAll: #(1 2 3 4 5).  
241   i := 1.  
242   v do: [:v |  
243     self assert: i equals: v.  
244     i := i + 1.  
245   ].  
246   self assert: 6 equals: i.  
247 )  
248 )  
249
```

```

1 "
2
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2007-2013 see AUTHORS file
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7 http://www.hpi.uni-potsdam.de/swa/
8
9 Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 VectorTest = TestCase (
29   | a |
30
31   setUp = (
32     a := Vector new.
33     a append: 'hello'.
34     a append: #world.
35     a append: 23.
36   )
37
38   testSize = (
39     self assert: 3 equals: a size.
40   )
41
42   testAt = (
43     self assert: #world equals: (a at: 2).
44     self assert:      23 equals: (a at: 3).
45   )
46
47   testFirst = (
48     | v |
49     self assert: 'hello' equals: a first.
50
51     v := Vector new.
52     1 to: 10 do: [:i |
53       v append: i.

```



```

54         self assert: 1 equals: v first ].
55
56     1 to: 10 do: [:i |
57         self assert: 1 equals: v first.
58         v removeFirst ]
59 )
60
61 testLast = (
62     | v |
63     self assert: 23 equals: a last.
64
65     v := Vector new.
66     1 to: 10 do: [:i |
67         v append: i.
68         self assert: i equals: v last ].
69
70     10 downTo: 1 do: [:i |
71         self assert: i equals: v last.
72         v remove ]
73 )
74
75 testContains = (
76     self assert: (a contains: 23).
77     self deny: (a contains: #nono).
78 )
79
80 testAppendAndRemoveFirst = (
81     | v |
82     v := Vector new: 10.
83     1 to: 100 do: [:i |
84         v append: i ].
85
86     "160 is implementation dependent, just here for orientation"
87     self assert: 160 equals: v capacity.
88     self assert: 100 equals: v size.
89
90     1 to: 100 do: [:i |
91         v removeFirst ].
92     1 to: 100 do: [:i |
93         v append: i ].
94
95     self assert: 320 equals: v capacity.
96     self assert: 100 equals: v size.
97 )
98
99 testIndexOf = (
100     | v |
101     v := Vector new: 3.
102     1 to: 17 do: [:i |
103         v append: i ].
104
105     self assert: -1 equals: (v indexOf: nil).
106     self assert: 1 equals: (v indexOf: 1).
107
108     self assert: 13 equals: (v indexOf: 13).
109     v at: 13 put: #test.
110
111     self assert: 13 equals: (v indexOf: #test).
112
113     v removeFirst.
114     self assert: -1 equals: (v indexOf: 1).

```

```

115
116     1 to: 12 do: [:i |
117         v removeFirst ].
118     self assert: -1 equals: (v indexOf: #test).
119 )
120
121 testAppendAll = (
122     | v c i |
123     v := Vector new: 2.
124     v append: 1.
125     v append: 2.
126     v append: 3.
127
128     c := Array with: 4 with: 5 with: 6.
129
130     v appendAll: c.
131     i := 1.
132
133     v do: [:e |
134         self assert: i equals: e.
135         i := i + 1 ]
136 )
137
138 testAsArray = (
139     | v arr |
140     v := Vector new.
141     self assert: 0 equals: v asArray length.
142
143     v append: 1.
144     v append: 2.
145
146     arr := v asArray.
147     self assert: 2 equals: arr length.
148     self assert: 1 equals: (arr at: 1).
149     self assert: 2 equals: (arr at: 2).
150 )
151
152 testAsSet = (
153     | v set |
154     v := Vector new.
155     v append: 1.
156     v append: 2.
157     v append: 3.
158     v append: 4.
159     self assert: 4 equals: v size.
160
161     set := v asSet.
162     self assert: 4 equals: set size.
163
164     v append: 1.
165     v append: 1.
166     v append: 1.
167
168     self assert: 4 + 3 equals: v size.
169
170     set := v asSet.
171     self assert: 4 equals: set size.
172 )
173
174 testIsEmpty = (
175     | v |

```

```

176     v := Vector new.
177     self assert: v isEmpty.
178
179     v append: 1.
180     self deny: v isEmpty.
181
182     v removeFirst.
183     self assert: v isEmpty.
184
185     v append: #ee.
186     self deny: v isEmpty.
187
188     v removeFirst.
189     self assert: v isEmpty.
190 )
191
192 testRemoveObj = (
193     | v |
194     v := Vector new.
195     v append: #a.
196     v append: #b.
197     v append: #c.
198     v append: #d.
199     v append: #e.
200     v append: #f.
201     v append: #g.
202     v append: #h.
203
204     self assert: 8 equals: v size.
205
206     self deny: (v remove: #aa).
207     self assert: (v remove: #e).
208     self assert: 7 equals: v size.
209 )
210
211 testAppendComma = (
212     | v |
213     v := Vector new.
214     v, #a.
215     v, #b.
216
217     self assert: 2 equals: v size.
218     self assert: (v contains: #a).
219     self assert: (v contains: #b).
220 )
221
222 testDoIndexes = (
223     | i v |
224     v := Vector new.
225     v doIndexes: [:j |
226         self assert: false ].
227
228     v appendAll: #(1 2 3 4 5).
229     i := 1.
230     v doIndexes: [:j |
231         self assert: i equals: j.
232         i := i + 1.
233     ].
234     self assert: 6 equals: i.
235 )
236

```

```
237 testDo = (  
238   | i v |  
239   v := Vector new.  
240   v appendAll: #(1 2 3 4 5).  
241   i := 1.  
242   v do: [:v |  
243     self assert: i equals: v.  
244     i := i + 1.  
245   ].  
246   self assert: 6 equals: i.  
247 )  
248 )  
249
```

```

1 #!/bin/bash
2 UNAME="$(uname -s)"
3
4 echo "Currently, we can only do up to 10 classes, because otherwise, we
have too many literals in a single method"
5 for i in $(seq 1 100)
6 do
7   if [ $((($i % 20)) -eq "0" ]
8   then
9     echo ". $i"
10  else
11    echo -n "."
12  fi
13
14  TESTS=("Array" "Block" "Boolean" "ClassLoading" "ClassStructure"
"Closure" "Coercion"
15        "CompilerReturn" "Dictionary" "DoesNotUnderstand" "Double"
"Empty" "Global"
16        "Hash" "Integer" "Preliminary" "Reflection" "SelfBlock" "Set"
"SpecialSelectors"
17        "String"
18        "Super" "Symbol" "System" "Vector")
19
20  for name in ${TESTS[@]}
21  do
22    cp "${name}Test.som" "${name}${i}Test.som"
23    if [ "${UNAME}" = "Darwin" ]; then
24      sed -i '' "s/${name}Test =/${name}${i}Test =/g" "${name}${i}Test.som"
25    else
26      sed -i'' "s/${name}Test =/${name}${i}Test =/g" "${name}${i}Test.som"
27    fi
28  done
29
30  # Create TestGC${i}.som
31  TEST_GC_FILE="TestGC${i}.som"
32  TEST_GC_CLASS="TestGC${i}"
33
34  echo "${TEST_GC_CLASS} = TestGCCCommon ( " > "${TEST_GC_FILE}"
35  echo "    tests = ( " >> "${TEST_GC_FILE}"
36  echo "        | v | " >> "${TEST_GC_FILE}"
37  echo "        v := Vector new." >> "${TEST_GC_FILE}"
38  for j in $(seq 1 "$i")
39  do
40    echo "        self tests${j}: v." >> "${TEST_GC_FILE}"
41  done
42  echo "        ^ v" >> "${TEST_GC_FILE}"
43  echo "    )" >> "${TEST_GC_FILE}"
44  echo " " >> "${TEST_GC_FILE}"
45
46  for j in $(seq 1 "$i")
47  do
48    echo "    tests${j}: v = ( " >> "${TEST_GC_FILE}"
49
50    for name in ${TESTS[@]}
51    do
52      echo "        v append: ${name}${j}Test." >> "${TEST_GC_FILE}"
53    done
54    echo "    )" >> "${TEST_GC_FILE}"

```

```

55     echo "" >> "${TEST_GC_FILE}"
56 done
57 echo ")" >> "${TEST_GC_FILE}"
58
59 # Create Test${i}.som
60 TEST_FILE="Test${i}.som"
61 TEST_CLASS="Test${i}"
62
63 echo "${TEST_CLASS} = TestCommon (" > "${TEST_FILE}"
64 echo "    numExecs = (" >> "${TEST_FILE}"
65 echo "        ^ $i" >> "${TEST_FILE}"
66 echo "    )" >> "${TEST_FILE}"
67 echo "" >> "${TEST_FILE}"
68 echo "    tests = (" >> "${TEST_FILE}"
69 echo "        | v |" >> "${TEST_FILE}"
70 echo "        v := Vector new." >> "${TEST_FILE}"
71 for j in $(seq 1 "$i")
72 do
73     echo "        self tests${j}: v." >> "${TEST_FILE}"
74 done
75 echo "        ^ v" >> "${TEST_FILE}"
76 echo "    )" >> "${TEST_FILE}"
77 echo "" >> "${TEST_FILE}"
78
79 for j in $(seq 1 "$i")
80 do
81     echo "    tests${j}: v = (" >> "${TEST_FILE}"
82
83     for name in ${TESTS[@]}
84     do
85         echo "        v append: ${name}${j}Test." >> "${TEST_FILE}"
86     done
87     echo "    )" >> "${TEST_FILE}"
88     echo "" >> "${TEST_FILE}"
89 done
90 echo ")" >> "${TEST_FILE}"
91
92 done
93

```

```
1 "  
2  
3 $Id: Towers.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "Mmm... Hanoi..."  
27  
28 Towers = Benchmark (  
29  
30   | piles movesdone |  
31  
32   pushDisk: disk onPile: pile = (  
33     | top |  
34  
35     top := piles at: pile.  
36     (top isNil not) && [ disk size >= top size ]  
37     ifTrue: [ self error: 'Cannot put a big disk on a smaller  
one' ].  
38  
39     disk next: top.  
40     piles at: pile put: disk.  
41   )  
42  
43   popDiskFrom: pile = (  
44     | top |  
45  
46     top := piles at: pile.  
47     top isNil  
48     ifTrue: [  
49       self error: 'Attempting to remove a disk from an empty  
pile' ].  
50  
51     piles at: pile put: top next.  
52     top next: nil.
```

```

53         ^top
54     )
55
56     moveTopDiskFrom: fromPile to: toPile = (
57         self pushDisk: (self popDiskFrom: fromPile) onPile: toPile.
58         movesdone := movesdone + 1.
59     )
60
61     buildTowerAt: pile disks: disks = (
62         disks downTo: 0 do: [ :i |
63             self pushDisk: (TowersDisk new: i) onPile: pile ]
64     )
65
66     move: disks disksFrom: fromPile to: toPile = (
67         disks = 1
68             ifTrue: [ self moveTopDiskFrom: fromPile to: toPile ]
69             ifFalse: [ | otherPile |
70                 otherPile := (6 - fromPile) - toPile.
71                 self move: disks - 1 disksFrom: fromPile to: otherPile.
72                 self moveTopDiskFrom: fromPile to: toPile.
73                 self move: disks - 1 disksFrom: otherPile to: toPile. ]
74     )
75
76     benchmark = (
77         piles := Array new: 4.
78         self buildTowerAt: 1 disks: 13.
79         movesdone := 0.
80         self move: 13 disksFrom: 1 to: 2.
81         ^ movesdone
82     )
83
84     verifyResult: result = (
85         ^ self assert: 8191 equals: result
86     )
87
88 )
89

```



```
1 "  
2  
3 $Id: TowersDisk.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 TowersDisk = (  
27  
28   | size next |  
29  
30   size          = ( ^size          )  
31   size: value = ( size := value )  
32   next         = ( ^next          )  
33   next: value = ( next := value )  
34  
35   -----  
36  
37   new: value = ( ^super new size: value )  
38  
39 )  
40
```

```

1 "
2
3 $Id: TreeNode.som 31 2009-07-31 12:25:18Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 TreeNode = (
27     | left right value |
28     value      = ( ^value      )
29     value: v = ( value := v )
30
31     check = (
32         ^(left isNil || [ (left value < value) && left check ]) &&
33         (right isNil || [ (right value >= value) && right check ])
34     )
35
36     insert: n = (
37         (n < value)
38         ifTrue: [
39             left isNil
40             ifTrue: [ left := TreeNode new: n ]
41             ifFalse: [ left insert: n ] ]
42         ifFalse: [
43             right isNil
44             ifTrue: [ right := TreeNode new: n ]
45             ifFalse: [ right insert: n ] ].
46     )
47
48     new: value = ( ^super new value: value )
49
50 -----
51
52 new: value = ( ^super new value: value )
53
54 )

```



```
1 "  
2  
3 $Id: TreeSort.som 31 2009-07-31 12:25:18Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 TreeSort = Sort (  
27  
28     sort: array = (  
29         | tree |  
30         array doIndexes: [ :i |  
31             (i = 1)  
32                 ifTrue: [ tree := TreeNode new: (array at: i) ]  
33                 ifFalse: [ tree insert: (array at: i) ] ].  
34         ^ tree  
35     )  
36  
37     verifyResult: tree = (  
38         tree check ifFalse: [ self error: 'Invalid result, tree not  
sorted' ].  
39         ^ true.  
40     )  
41  
42     dataSize = ( ^ 1000 )  
43 )  
44
```

```
1 "  
2  
3 $Id: Echo.som 226 2008-04-21 12:45:01Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Echo = (  
27  
28     run: args = (  
29         args from: 2 to: args length do: [ :arg | arg print. ' ' print ].  
30         '' println.  
31     )  
32  
33 )  
34
```

Examples/Hello.som

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 Hello = (  
24  
25     "The 'run' method is called when initializing the system"  
26     run = ('Hello, World from SOM' println )  
27  
28 )  
29
```

```
1 "  
2  
3 $Id: Apple.som 191 2008-04-10 18:15:47Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Apple = Element ()  
27
```

```
1 "  
2  
3 $Id: Board.som 426 2008-05-22 08:22:07Z stefan.marr $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Board = (  
27   | view width height board |  
28  
29   width = ( ^width )  
30   width: val = ( width := val. )  
31  
32   height = ( ^height )  
33   height: val = ( height := val )  
34  
35   board = ( ^board )  
36   board: val = ( board := val )  
37  
38   view = ( ^view )  
39  
40   addApple = (  
41     | added x y newApple |  
42     added := false.  
43  
44     [ added ] whileFalse: [  
45       x := 1 atRandom % width. "$x = rand(0, $this->width - 1);"  
46       y := 1 atRandom % height.  
47  
48       x := x + 1.  
49       y := y + 1.  
50  
51       (self board at: x) isNil ifTrue: [  
52         self board at: x put: (Array new: height).  
53       ]
```



```

54
55     ((self board at: x) at: y) isNil ifTrue: [
56         newApple := Apple newWithX: x Y: y.
57         (self board at: x) at: y put: newApple.
58         added := true.
59         view isNil ifFalse: [
60             view addApple: newApple.
61         ].
62     ]
63 ]
64 )
65
66 view: value = (
67     view := value.
68     value board: board.
69     value updateCompletely.
70 )
71
72 isAppleAtX: x Y: y = (
73     ((board at: x) isNil) ifFalse: [
74         ((board at: x) at: y) isNil ifFalse: [
75             ^((board at: x) at: y) class == Apple
76         ]
77     ].
78     ^false
79 )
80 •
81 isSnakeAtX: x Y: y = (
82     ((board at: x) isNil) ifFalse: [
83         ((board at: x) at: y) isNil ifFalse: [
84             ^((board at: x) at: y) class == SnakeElement
85         ]
86     ].
87     ^false
88 )
89 •
90 remove: element = (
91     (self board at: (element x)) at: (element y) put: nil.
92     self view remove: element
93 )
94 •
95 add: element = (
96     (self board at: element x) isNil ifTrue: [
97         self board at: (element x) put: (Array new: height).
98     ].
99     (self board at: (element x)) at: (element y) put: element.
100     view add: element
101 )
102
103 ----
104 ™
105 newWithWidth: width height: height numberOfApples: numberOfApples = (
106     | newBoard |
107     newBoard := Board new.
108     newBoard board: (Array new: width).
109     newBoard width: width.
110     newBoard height: height.
111 ™
112     [numberOfApples >= 0] whileTrue: [
113         newBoard addApple.
114         numberOfApples := numberOfApples - 1.

```

```
115         ].
116         ^newBoard
117     )
118 )
119
```

```
1 "  
2  
3 Copyright (c) 2001-2013 see AUTHORS file  
4  
5 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
6 of this software and associated documentation files (the 'Software'), to  
deal  
7 in the Software without restriction, including without limitation the  
rights  
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
9 copies of the Software, and to permit persons to whom the Software is  
10 furnished to do so, subject to the following conditions:  
11  
12 The above copyright notice and this permission notice shall be included in  
13 all copies or substantial portions of the Software.  
14  
15 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
21 THE SOFTWARE.  
22 "  
23  
24 BoardView = (  
25   | board width height |  
26   •  
27   board: value = ( board := value )  
28   width: value = ( width := value )  
29   height: value = ( height := value )  
30  
31   updateCompletely = (  
32     board do: [ :y |  
33       y isNil ifFalse: [  
34         y do: [ :apple |  
35           apple isNil ifFalse: [  
36             Terminal cursorToX: (apple x + 1) Y: (apple y + 1).  
37             Terminal put: 'o'  
38           ]  
39         ]  
40       ]  
41     ]  
42   )  
43   •  
44   remove: snakeElement = (  
45     Terminal cursorToX: snakeElement x + 1 Y: snakeElement y + 1.  
46     Terminal put: ' '  
47   )  
48   •  
49   add: snakeElement = (  
50     Terminal cursorToX: snakeElement x + 1 Y: snakeElement y + 1.  
51     Terminal put: '#'  
52   )  
53   •  
54   addApple: apple = (  

```

```

55     Terminal cursorToX: apple x + 1 Y: apple y + 1.
56     Terminal put: 'o'
57 )
58 •
59 drawBoarder = (
60     Terminal cursorToX: 1 Y: 1.
61     Terminal put: '/'.
62     width timesRepeat: [ Terminal put: '-' ].
63     Terminal put: '\\'.
64
65     1 to: height do: [ :i |
66         Terminal cursorToX: 1 Y: i + 1.
67         Terminal put: '|'.
68         Terminal cursorToX: (width + 2) Y: i + 1.
69         Terminal put: '|'
70     ].
71
72     Terminal cursorToX: 1 Y: height + 2.
73     Terminal put: '\\'.
74     width timesRepeat: [ Terminal put: '-' ].
75     Terminal put: '/'.
76 )
77
78 ----
79
80 new: board = (
81     | newBoardView |
82     newBoardView := BoardView new.
83     board view: newBoardView.
84     newBoardView width: board width.
85     newBoardView height: board height.
86     ^newBoardView
87 )
88 )
89

```

```

1 #include <string.h>
2
3 // #include "../misc/defs.h"
4 // #include "../vobjects/VMOobject.h"
5 #include "../vobjects/PrimitiveRoutine.h"
6
7 #include "Terminal.h"
8
9 #include "../primitivesCore/Routine.h"
10 #include "../primitivesCore/PrimitiveContainer.h"
11 #include "../primitivesCore/PrimitiveLoader.h"
12
13 static PrimitiveLoader* loader = nullptr;
14 // map<pString, PrimitiveContainer*> primitiveObjects;
15 // "Constructor"
16 static bool initialized = false;
17 extern "C" void setup() {
18     if (!loader) {
19         // Initialize loader
20         loader = new PrimitiveLoader();
21         loader->AddPrimitiveObject("Terminal",
22             static_cast<PrimitiveContainer*>(new Terminal()));
23     }
24
25 extern "C" bool supportsClass(const char* name) {
26     // if (!loader) setup();
27     return loader->SupportsClass(name);
28 }
29
30
31 extern "C" void tearDown() {
32     // primitiveClasses.clear();
33     if (loader) delete loader;
34     // if (primitiveObjects) delete primitiveObjects;
35 }
36
37 extern "C" PrimitiveRoutine* create(const pString& cname, const pString&
fname, bool isPrimitive) {
38
39 #ifdef __DEBUG
40     cout << "Loading PrimitiveContainer: " << cname << "::" << fname <<
endl;
41 #endif
42     // if (!loader) setup();
43
44     return loader->GetPrimitiveRoutine(cname, fname, isPrimitive);
45 }
46
47

```

```
1 "  
2  
3 $Id: Element.som 191 2008-04-10 18:15:47Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Element = (  
27   | x y |  
28  
29   x = ( ^x )  
30   x: val = ( x := val )  
31  
32   y = ( ^y )  
33   y: val = ( y := val )  
34  
35   ----  
36   newWithX: x Y: y = (  
37     | newElement |  
38     newElement := self new.  
39     newElement x: x.  
40     newElement y: y.  
41     ^newElement.  
42   )  
43 )  
44
```

```
1 "  
2  
3 $Id: Main.som 191 2008-04-10 18:15:47Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Main = (  
27   run = (  
28     | board view chr snake dir continue |  
29     Terminal init.  
30     Terminal clear.  
31     dir := Terminal KEY_UP.  
32  
33     board := Board newWithWidth: 10 height: 10 numberOfApples: 5.  
34     view := BoardView new: board.  
35     view drawBoarder.  
36  
37     snake := Snake newWithX: 5 Y: 5 andBoard: board.  
38     continue := true.  
39     [ continue ] whileTrue: [  
40       chr := Terminal get.  
41       ((Terminal KEY_UP = chr)  
42         || ((Terminal KEY_DOWN) = chr)  
43         || ((Terminal KEY_LEFT) = chr)  
44         || ((Terminal KEY_RIGHT) = chr)) ifTrue: [ dir := chr ].  
45       Terminal sleepFor: 250.  
46  
47       "Terminal cursorToX: 15 Y: 15."  
48       (Terminal KEY_UP = dir) ifTrue: [ continue := snake moveUp ].  
49       (Terminal KEY_DOWN = dir) ifTrue: [ continue := snake moveDown ].  
50       (Terminal KEY_LEFT = dir) ifTrue: [ continue := snake moveLeft ].  
51       (Terminal KEY_RIGHT = dir) ifTrue: [ continue := snake moveRight ].  
52       '' println.  
53     ].  
54
```

```

55     'GAME OVER' println.
56     Terminal uninit.
57 )
58 )
59
60 "
61
62 while (true) {
63 'F' Ò C°
64 -v†-ÆR ,F' â ' °
65 ™$key = Terminal::get(0);
66 ™if (in_array($key, array(Terminal::KEY_UP, Terminal::KEY_DOWN,
Terminal::KEY_LEFT, Terminal::KEY_RIGHT))) {
67 ™'FF-" Ò F¶W"°
68 ™}
69 ™usleep(100000);
70 ™$i--;
71 -Ð
72 ™
73 •
74 --b , G&W7VÇB' °
75 ™Terminal::cursorTo(5, 15);
76 ™Terminal::put('GAME OVER');
77 ™sleep(5);
78 ™return;
79 -Ð
80 }
81
82 Terminal::clear();
83 "
84

```


Examples/Snake/Makefile

```

1  #!/usr/bin/env make
2
3  TARGET™=Terminal
4
5
6  ##### ONLY CHANGE STUFF BELOW IF YOU REALLY HAVE
#####
7
8
9  CC™`Ör²º
10 CFLAGS™=-Wno-endif-labels -O2 $(DBG_FLAGS) $(INCLUDES)
11 LDFLAGS™=-enable-auto-import $(LIBRARIES)
12
13 INSTALL™=install
14
15 ##### global stuff -- overridden by ../Makefile
16
17 ROOT_DIR`óÖ B... tB'òââðââ
18 BUILD_DIR  ?= $(ROOT_DIR)/build
19 SRC_DIR™?= $(ROOT_DIR)/src
20
21 ST_DIR™?= $(ROOT_DIR)/Smalltalk
22 EX_DIR™?= $(ROOT_DIR)/Examples
23 TEST_DIR`óÖ B...$ôöEôD•"'öFW7E7V-FP
24
25 ##### include path
26
27 INCLUDES™=-I$(SRC_DIR)
28 LIBRARIES™=-L$(ROOT_DIR)
29
30 ifneq ($(OS),)
31 # only Windows has OS predefined.
32 "Ä"%9"òÖÆÖ ÖÂâðâââ5 4ôÖ ÖÂâðâââö &-Ö-F-ft6÷&P
33 endif
34
35 all: $(TARGET).csp
36
37 debug : DBG_FLAGS=-DDEBUG -g
38 debug: all
39
40 profiling : DBG_FLAGS=-g -pg
41 profiling : LDFLAGS+=-pg
42 profiling: all
43
44 $(TARGET).csp:
45 'B„42' B„4dÄ u2' B„ÄDdÄ u2' x6† &VB Ör Ä
46 ™$(PWD)/$(TARGET).cpp -o $(TARGET).csp $(LIBS)
47 •
48 clean:
49 -&Ö Ö&b B...D $tUB'æ77
50
51

```

```
1 "  
2  
3 $Id: Snake.som 426 2008-05-22 08:22:07Z stefan.marr $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Snake = (  
27   | head tail board |  
28  
29   head: val = ( head := val )  
30   tail: val = ( tail := val )  
31   board: val = ( board := val )  
32  
33   moveLeft = (  
34     | newPos |  
35     newPos := SnakeElement newWithX: head x Y: head y.  
36     newPos x: (self overflow: (newPos x - 1) max: board width).  
37     ^self move: newPos  
38   )  
39  
40   moveRight = (  
41     | newPos |  
42     newPos := SnakeElement newWithX: head x Y: head y.  
43     newPos x: (self overflow: (newPos x + 1) max: board width).  
44     ^self move: newPos  
45   )  
46  
47   moveUp = (  
48     | newPos |  
49     newPos := SnakeElement newWithX: head x Y: head y.  
50     newPos y: (self overflow: (newPos y - 1) max: board height).  
51     ^self move: newPos  
52   )  
53
```

```

54     moveDown = (
55         | newPos |
56         newPos := SnakeElement newWithX: head x Y: head y.
57         newPos y: (self overflow: (newPos y + 1) max: board height).
58         ^self move: newPos
59     )
60
61     move: newPos = (
62         newPos next: head.
63         head prev: newPos.
64         head := newPos.
65
66         (board isAppleAtX: (newPos x) Y: (newPos y)) ifTrue: [
67             board addApple
68         ] ifFalse: [
69
70             (board isSnakeAtX: newPos x Y: newPos y) ifTrue: [
71                 ^false
72             ].
73             board remove: tail.
74             tail := tail prev.
75             tail next: nil.
76         ].
77         board add: newPos.
78
79         ^true
80     )
81 •
82     overflow: val max: max = (
83         (val < 1) ifTrue: [ val := max + val ].
84         (val > max) ifTrue: [ val := val - max ].
85         ^val
86     )
87
88     ----
89     newWithX: x Y: y andBoard: board = (
90         | newSnake head |
91         newSnake := Snake new.
92         head := SnakeElement newWithX: x Y: y.
93
94         newSnake head: head.
95         newSnake tail: head.
96         newSnake board: board.
97         board add: head.
98         ^newSnake
99     )
100 )
101

```

```
1 "  
2  
3 $Id: SnakeElement.som 191 2008-04-10 18:15:47Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 SnakeElement = Element (  
27   | next prev |  
28   next = ( ^next )  
29   next: val = ( next := val )  
30  
31   prev = ( ^prev )  
32   prev: val = ( prev := val )  
33 )  
34
```

Examples/Snake/Terminal.c

```
1 /*
2 Copyright (c) 2001-2008 see AUTHORS file
3 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
4 http://www.hpi.uni-potsdam.de/swa/
5
6 Permission is hereby granted, free of charge, to any person obtaining a
copy
7 of this software and associated documentation files (the "Software"), to
deal
8 in the Software without restriction, including without limitation the
rights
9 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 copies of the Software, and to permit persons to whom the Software is
11 furnished to do so, subject to the following conditions:
12
13 The above copyright notice and this permission notice shall be included in
14 all copies or substantial portions of the Software.
15
16 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
19 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
21 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
22 THE SOFTWARE.
23 */
24
25 #include <string.h>
26 #include <stdlib.h>
27 #include <stdio.h>
28 #include <stdbool.h>
29 #include <stddef.h>
30
31
32 /*****
33 #pragma mark * Included Headers *
34 *****/
35
36
37 #include <vmobjects/VMFrame.h>
38 #include <vmobjects/VMString.h>
39 #include <vm/Universe.h>
40
41
42 #include <termios.h>
43 #include <fcntl.h>
44
45 /*****
46 #pragma mark * Primitive Foreward Declaration *
47 *****/
48
49 extern "C" {
50
51 void _Terminal_sleepFor_(pVMObject object, pVMFrame frame);
52 void _Terminal_getChar(pVMObject object, pVMFrame frame);
53 void _Terminal_uninit(pVMObject object, pVMFrame frame);
```

```

54 void _Terminal_init(pVMObject object, pVMFrame frame);
55
56 }
57
58 /*****/
59 #pragma mark * Internal functions and init. *
60 /*****/
61
62 /*** Lib initialization ***/
63 #ifdef __GNUC__
64 void init(void) __attribute__((constructor));
65 void fini(void) __attribute__((destructor));
66 #else
67 void _init(void);
68 void _fini(void);
69 #pragma init _init
70 #pragma fini _fini
71 #endif
72
73 #ifdef __GNUC__
74 void init(void)
75 #else
76 void _init(void)
77 #endif
78 {
79     'òò 6 ÅÅ -æ-B gVæ6-öç2à
80     •
81 }
82
83
84 #ifdef __GNUC__
85 void fini(void)
86 #else
87 void _fini(void)
88 #endif
89 {
90     •
91 }
92
93 // Classes supported by this lib.
94 static char *supported_classes[] = {
95     "Terminal",
96     nullptr
97 };
98
99
100 /*****/
101 #pragma mark * Exported functions starting here *
102 /*****/
103
104 extern "C" {
105
106 // returns, whether this lib is responsible for a specific class
107 bool™supports_class(const char* name) {
108     •
109     -6† " ç|-FW#×7W ÷'FVEö6Æ 76W3°
110     -v†-ÆR,|-FW"•
111     ™if (strcmp(name,*iter++)==0)
112     ™-&WGW&â G'VS°
113     -&WGW&â f Ç6S°
114     •

```

```

115 }
116
117 bool initialized = false;
118
119
120 /**
121  * init_csp()
122  *
123  * the library initializer. It is not equal to the init/fini-pair, as for
124  * them,
125  * the init is executed upon every loading of the shared library.
126  *
127  * All work that needs to be done before the actual primitives are
128  * assigned
129  * should be called from this function.
130  */
131 void init_csp(void) {
132     --æ-F- Æ-|VB Ò f Ç6S°
133 }
134
135 ptrdiff_t terminalStream;
136 struct termios old_tty;
137
138 void init_the_terminal() {
139     •
140     --b †-æ-F- Æ-|VB•
141     ™return;
142     •
143     -7G'V7B FW&Ö-÷2 GG"°
144     •
145     'òð W' &R FW&Ö-æ Â 6WGF-æw2 æB 6† ævR Fð æöâÖ6 æöæ-6 Â ÖÖFR f÷" 6† "Ð
146     wise input
147     -F6vWF GG"f Â fÖÆ÷GG'"°
148     -GG' Ò ÖÆ÷GG'"°
149     -GG'æ5öÆfÆ r Ò GG'æ5öÆfÆ r b â„T4„ð Â T4„ô² Â "4 äôâ"°
150     -GG'æ5ö65µeD"ÔUÔ Ò °
151     -F76WF GG"f Â D54 äörÂ gGG'"°
152     •
153     -FW&Ö-æ Å7G&V Ò Ò ÷ Vâ,"öFWb÷GG'"Â öö$DôäÅ' Â ööäöä$Äô4²"°
154     --b †FW&Ö-æ Å7G&V Ò Â ' °
155     ™Universe_error_exit("Could not open /dev/tty for read\n");
156     -Ð
157     --æ-F- Æ-|VB Ò G'VS°
158 }
159
160
161
162 /*****
163  * Primitive Implementatition here
164  */
165
166
167
168 void Terminal_getChar(pVMObject object, pVMFrame frame) {
169     init_the_terminal();
170     char chr;
171     char result[2];
172     pString str = nullptr;

```

```

173  VMObject* vmStr = nullptr;
174
175  pVMObject self __attribute__((unused)) = SEND(frame, pop);
176
177  if (read(terminalStream, &chr, sizeof(chr)) > 0) {
178  '   &W7VÇE³ Ò Ò 6‡#°
179  '   &W7VÇE³ Ò Ò °
180  '
181  '   7G" Ò 7G&-æuöæWr‡&W7VÇB""°
182  '   fÕ7G" Ò ‡ dÕö&|V7B•dÕ7G&-æuöæWu÷v-F,‡7G""°
183  '   4TäB‡g& ÖRÂ W6,Â fÕ7G""°
184  '   } else {
185  '   4TäB‡g& ÖRÂ W6,Â æ-Åöö&|V7B""°
186  '   }
187  }
188
189  void Terminal_uninit(pVMObject object, pVMFrame frame) {
190  -7G'V7B FW&Ö-÷2 GG""°
191
192  -6Æ÷6R‡FW&Ö-æ Å7G&V Ò""°
193  -F76WF GG"f Å D54 äörÂ gGG'""°
194  -F76WF GG"f Å D54 äörÂ föÆ÷GG'""°
195  }
196
197  void Terminal_init(pVMObject object, pVMFrame frame) {
198  }
199
200  void Terminal_sleepFor_(pVMObject object, pVMFrame frame) {
201      pVMInteger miliSeconds = (pVMInteger)SEND(frame, pop);
202      int64_t sec = (int64_t)SEND(miliSeconds, get_embedded_integer) * 1000;
203      sync();
204      usleep(sec);
205  }
206
207  }
208
209  /*****
210  /*****
211  /*****
212  #pragma mark * EOF *
213  /*****
214  /*****
215  /*****
216
217

```



```

1  /*
2  * $Id: Terminal.c 426 2008-05-22 08:22:07Z stefan.marr $
3  *
4  Copyright (c) 2001-2008 see AUTHORS file
5  Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
6  http://www.hpi.uni-potsdam.de/swa/
7
8  Permission is hereby granted, free of charge, to any person obtaining a
copy
9  of this software and associated documentation files (the "Software"), to
deal
10 in the Software without restriction, including without limitation the
rights
11 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
12 copies of the Software, and to permit persons to whom the Software is
13 furnished to do so, subject to the following conditions:
14
15 The above copyright notice and this permission notice shall be included in
16 all copies or substantial portions of the Software.
17
18 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
19 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
20 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
21 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
22 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
23 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
24 THE SOFTWARE.
25  */
26
27 #include <string.h>
28 #include <stdlib.h>
29 #include <stdio.h>
30 #include <stdbool.h>
31
32
33 /*****
34 #pragma mark * Included Headers
35 *****/
36
37
38 #include <vmobjects/VMOBJECT.h>
39 #include <vmobjects/VMFrame.h>
40 #include <vmobjects/VMString.h>
41 #include <vm/Universe.h>
42
43 #include <primitivesCore/Routine.h>
44
45 #include <termios.h>
46 #include <fcntl.h>
47
48 #include "Terminal.h"
49
50 /*****
51 #pragma mark * Primitive Foreward Declaration
52 *****/
53

```

```

54
55 /*****/
56 /*****/
57 /*****/
58 #pragma mark * Primitive Implementatition here *
59 /*****/
60 /*****/
61 /*****/
62
63
64 struct termios old_tty;
65 Terminal::Terminal() : PrimitiveContainer() {
66     terminalStream = 0;
67     SetPrimitive("getChar", new (GetHeap<HEAP_CLS>())
Routine<Terminal>(this, &Terminal::getChar));
68     SetPrimitive("uninit", new (GetHeap<HEAP_CLS>())
Routine<Terminal>(this, &Terminal::uninit));
69     SetPrimitive("init", new (GetHeap<HEAP_CLS>())
Routine<Terminal>(this, &Terminal::init));
70     SetPrimitive("sleepFor_", new (GetHeap<HEAP_CLS>())
Routine<Terminal>(this, &Terminal::sleepFor_));
71 }
72
73 void Terminal::getChar(VMObject* object, VMFrame* frame) {
74     char chr;
75     char result[2];
76     pString str = nullptr;
77     VMObject* vmStr = nullptr;
78     frame->Pop();
79     //VMObject self __attribute__((unused)) = SEND(frame, pop);
80
81     if (read(terminalStream, &chr, sizeof(chr)) > 0) {
82 ' &W7VÇE³ Ò Ò 6†#°
83 ' &W7VÇE³ Ò Ò °
84 '
85 ' 7G" Ò 7G&-ær†&W7VÇB"°
86         vmStr = (VMObject*)GetUniverse()->NewString(str);
87         frame->Push(vmStr);
88     } else {
89         frame->Push(Globals::NilObject());
90     }
91 }
92
93 void Terminal::uninit(VMObject* object, VMFrame* frame) {
94 -6Æ÷6R†FW&Ö-æ Å7G&V Ò"°
95 -F76WF GG"f Å D54 äörÂ föÆ÷GG'"°
96 }
97
98 void Terminal::init(VMObject* object, VMFrame* frame) {
99 -7G'V7B FW&Ö-÷2 GG"°
100 •
101 'òò W' &R FW&Ö-æ Å 6WGF-æw2 æB 6† ævR Fð æöâÖ6 æöæ-6 Å ÖöFR f÷" 6† "Ð
wise input
102 -F6vWF GG"f Å föÆ÷GG'"°
103 -GG' Ò öÆ÷GG"°
104 -GG'æ5öÆfÆ r Ò GG'æ5öÆfÆ r b â„T4„ð Å T4„ô² Å "4 äôâ"°
105 -GG'æ5ö65µeD"ÔUÒ Ò °
106 -F76WF GG"f Å D54 äörÂ gGG'"°
107 •
108 -FW&Ö-æ Å7G&V Ò Ò ÷ Vâ,"öFWb÷GG'"Å öö$DôâÅ' Å öôäôä$Äô4²"°
109 --b †FW&Ö-æ Å7G&V Ò Å ' °

```

```

110         GetUniverse()->ErrorExit("Could not open /dev/tty for read\n");
111     →
112 }
113
114 void Terminal::sleepFor_(VMObject* object, VMFrame* frame) {
115     VMInteger* miliSeconds = (VMInteger*)frame->Pop();
116     int64_t sec = (int64_t)miliSeconds->GetEmbeddedInteger() * 1000;
117     sync();
118     usleep(sec);
119 }
120
121 /*****/
122 /*****/
123 /*****/
124 #pragma mark * EOF *
125 /*****/
126 /*****/
127 /*****/
128
129

```

```
1 #pragma once
2
3 #include <primitivesCore/PrimitiveContainer.h>
4
5 class Terminal : public PrimitiveContainer {
6 public:
7     Terminal();
8     void sleepFor_(VMObject* object, VMFrame* frame);
9     void getChar(VMObject* object, VMFrame* frame);
10    void uninit(VMObject* object, VMFrame* frame);
11    void init(VMObject* object, VMFrame* frame);
12 private:
13     int terminalStream;
14 };
15
```

```
1 "  
2  
3 $Id: Terminal.som 191 2008-04-10 18:15:47Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Terminal = (  
27  
28     ----  
29     KEY_UP = (^#key_up)  
30     KEY_DOWN = (^#key_down)  
31     KEY_LEFT = (^#key_left)  
32     KEY_RIGHT = (^#key_right)  
33     •  
34     init = primitive  
35  
36     uninit = primitive  
37  
38     cursorToX: x Y: y = (  
39         '[' print.  
40         y print.  
41         ';' print.  
42         x print.  
43         'H' print.  
44     )  
45  
46     clear = (  
47         '[2J' print.  
48     )  
49  
50     put: str = (  
51         str print  
52     )  
53  
54     getChar = primitive
```

```

55
56  get = (
57      | chr result |
58      result := ''.
59      chr := self getChar.
60
61      [ chr = nil ] whileFalse: [
62          result := result + chr.
63          chr := self getChar
64      ].
65
66      ^self recognizeKeys: result.
67  )
68
69  recognizeKeys: chrs = (
70      (chrs = ' [A'] ifTrue: [
71          ^self KEY_UP.
72      ].
73
74      (chrs = ' [B'] ifTrue: [
75          ^self KEY_DOWN.
76      ].
77
78      (chrs = ' [C'] ifTrue: [
79          ^self KEY_RIGHT.
80      ].
81
82      (chrs = ' [D'] ifTrue: [
83          ^self KEY_LEFT.
84      ].
85
86      (chrs = '') ifTrue: [
87          ^nil.
88      ].
89
90      ^chrs
91  )
92
93  sleepFor: sec = primitive
94  )
95

```

LICENSE

1 Copyright (c) 2009 Michael Haupt, michael.haupt@hpi.uni-potsdam.de
2 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
3 <http://www.hpi.uni-potsdam.de/swa/>
4
5 Permission is hereby granted, free of charge, to any person obtaining a
copy
6 of this software and associated documentation files (the "Software"), to
deal
7 in the Software without restriction, including without limitation the
rights
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in
13 all copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
21 THE SOFTWARE.
22

README.md

```
1 SOM - Simple Object Machine
2 =====
3
4 SOM is a minimal Smalltalk dialect used to teach VM construction at the
[Hasso
5 Plattner Institute][SOM]. It was originally built at the University of
Århus
6 (Denmark) where it was also used for teaching.
7
8 Currently, implementations exist for Java (SOM), C (CSOM), C++ (SOM++), and
9 Squeak/Pharo Smalltalk (AweSOM).
10
11 ### Fork implemented in Go
12
13 ### SOM
14
15 A simple SOM Hello World looks like:
16
17 ```Smalltalk
18 Hello = (
19   run = (
20     'Hello World!' println.
21   )
22 )
23 ```
24
25 This repository contains the standard library of SOM, without an actual SOM
26 implementation. Please see the [main project page][SOMst] for links to the
27 VM implementation.
28
29 With CSOM, the given example could be executed for instance like:
30   `./CSOM -cp Smalltalk Hello.som`
31
32 AweSOM can be asked to directly evaluate a given string, for instance like:
33   `SOMUniverse new eval: '''Hello World!''' println'.'.`
34
35 A version of AweSOM is available for Pharo via:
36 ```Smalltalk
37 Gofer it
38   url: 'http://ss3.gemstone.com/ss/AweSOM';
39   package: 'ConfigurationOfAweSOM';
40   load.
41 (Smalltalk at: #ConfigurationOfAweSOM) loadDevelopment
42 ```
43
44 To install it into a recent Squeak, use the following expression:
45 ```Smalltalk
46 Installer ss3
47   project: 'AweSOM';
48   install: 'ConfigurationOfAweSOM'.
49 (Smalltalk at: #ConfigurationOfAweSOM) perform: #loadDevelopment
50 ```
51
52 Information on previous authors are included in the AUTHORS file. This
code is
53 distributed under the MIT License. Please see the LICENSE file for details.
54
55
```



```
56 Build Status
57 -----
58
59 Thanks to Travis CI, all commits of this repository are tested.
60 The current build status is: [![Build Status](https://travis-ci.org/SOM-st/
SOM.png?branch=master)](https://travis-ci.org/SOM-st/SOM/)
61
62 The build status of the SOM implementations is as follows:
63
64 * CSOM: [![CSOM Build Status](https://travis-ci.org/SOM-st/CSOM.png?
branch=master)](https://travis-ci.org/SOM-st/CSOM/)
65 * SOM (Java): [![SOM Java Build Status](https://travis-ci.org/SOM-st/som-
java.png?branch=master)](https://travis-ci.org/SOM-st/som-java/)
66 * PySOM: [![PySOM Build Status](https://travis-ci.org/SOM-st/PySOM.png?
branch=master)](https://travis-ci.org/SOM-st/PySOM)
67 * RPySOM: [![RPySOM Build Status](https://travis-ci.org/SOM-st/RPySOM.png?
branch=master)](https://travis-ci.org/SOM-st/RPySOM)
68 * TruffleSOM: [![TruffleSOM Build Status](https://travis-ci.org/SOM-st/
TruffleSOM.png?branch=master)](https://travis-ci.org/SOM-st/TruffleSOM)
69
70
71 [SOM]: http://www.hpi.uni-potsdam.de/hirschfeld/projects/som/
72 [SOMst]: https://travis-ci.org/SOM-st/
73
```

```

1 "
2
3 $Id: Array.som 29 2009-07-31 11:28:44Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 Array = (
27
28     "Accessing"
29     at: index          = primitive
30     at: index put: value = primitive
31     length              = primitive
32     putAll: block       = ( self doIndexes: [ :i |
33                             self at: i put: block value ] )
34     first = ( ^ self at: 1 )
35     last  = ( ^ self at: self length )
36
37
38     "Iterating"
39     do: block          = ( self doIndexes: [ :i |
40                             block value: (self at: i) ] )
41     doIndexes: block   = ( 1 to: self length do: [:i |
42                             block value: i. ] )
43
44     from: start to: end do: block = (
45         start to: end do: [:i | block value: (self at: i) ] )
46
47     "Copying (inclusively)"
48     copyFrom: start to: end = (
49         | result i |
50
51         result := Array new: end - start + 1.
52         i := 1.
53         self from: start to: end do: [ :e |

```

```

54         result at: i put: e.
55         i := i + 1 ].
56
57     ^result
58 )
59
60     copyFrom: start = ( ^self copyFrom: start to: self length )
61
62     replaceFrom: start to: stop with: replacement startingAt: repStart = (
63         "This destructively replaces elements from start to stop in the
64         receiver starting at index, repStart, in the sequenceable
collection,
65         replacementCollection. Answer the receiver. No range checks are
66         performed."
67         | index repOff |
68         repOff := repStart - start.
69         index := start - 1.
70         [(index := index + 1) <= stop]
71         whileTrue: [self at: index put: (replacement at: repOff +
index)]
72     )
73
74     copy = (^self copyFrom: 1)
75
76     prependedWith: val = (
77         | result |
78         result := Array new: self length + 1.
79         self doIndexes: [:i |
80             result at: i + 1 put: (self at: i)].
81         result at: 1 put: val.
82         ^ result
83     )
84
85     extendedWith: val = (
86         | result |
87         result := Array new: self length + 1.
88         self doIndexes: [:i |
89             result at: i put: (self at: i)].
90         result at: result length put: val.
91         ^ result
92     )
93
94     "Numerical"
95     sum      = ( ^self inject: 0 into: [ :sub :elem | sub + elem ] )
96     average = ( ^self sum / self length )
97
98     join: joiner = (
99         | result first |
100         first := true.
101
102         self do: [:e |
103             first
104                 ifTrue: [
105                     result := e.
106                     first := false]
107                 ifFalse: [
108                     result := result + joiner + e] ].
109         ^ result
110     )
111
112     "Containment check"

```

```

113 contains: element = ( self do: [ :e | e = element ifTrue: [ ^true ] ].
114                               ^false )
115 indexOf: element = (
116     self doIndexes: [ :i | (self at: i) = element ifTrue: [ ^ i ] ].
117     ^ nil
118 )
119
120 lastIndexOf: element = (
121     self length downTo: 1 do: [: i | (self at: i) = element ifTrue: [ ^
122 i ] ].
123     ^ nil
124 )
125 "Collection"
126 collect: aBlock = (
127     | result |
128     result := Array new: self length.
129     self doIndexes: [ :i | result at: i put: (aBlock value: (self at:
130 i)) ].
131     ^result
132 )
133 inject: sub into: aBlock = ( | next |
134     next := sub.
135     self do: [ :e | next := aBlock value: next with: e ].
136     ^next
137 )
138
139 reject: aBlock = (
140     ^ self select: [:element | (aBlock value: element) == false ]
141 )
142
143 select: aBlock = (
144     "TODO: fix the hard reference to Vector..."
145     | newCollection |
146     newCollection := Vector new: self length.
147     self do: [:each | (aBlock value: each)
148         ifTrue: [newCollection append: each]].
149     ^ newCollection
150 )
151
152 union: aCollection = (
153     | new |
154     new := Set new.
155     new addAll: self.
156     new addAll: aCollection.
157     ^ new
158 )
159
160 -----
161
162 "Allocation"
163 new = ( ^self new: 0 )
164 new: length = primitive
165 new: length withAll: block = ( ^((self new: length) putAll: block) )
166
167 "Convenience"
168 with: a = (
169     | arr |
170     arr := self new: 1.
171     arr at: 1 put: a.

```

```
172         ^ arr
173     )
174
175     with: a with: b                = (
176         | arr |
177         arr := self new: 2.
178         arr at: 1 put: a.
179         arr at: 2 put: b.
180         ^ arr
181     )
182
183     with: a with: b with: c        = (
184         | arr |
185         arr := self new: 3.
186         arr at: 1 put: a.
187         arr at: 2 put: b.
188         arr at: 3 put: c.
189         ^ arr
190     )
191 )
192
```

```
1 "  
2  
3 $Id: Block.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Block = (  
27  
28     "For the creation of Block instances, see Universe_new_block()."  
29  
30     "Evaluation"  
31     value = primitive  
32  
33     "Looping"  
34     whileFalse: block = (  
35         [ self value not ] whileTrue: block  
36     )  
37  
38     whileTrue: block = (  
39         self value ifFalse: [ ^nil ].  
40         block value.  
41         self restart  
42     )  
43  
44     "Restarting"  
45     restart = primitive  
46  
47 )  
48
```

```
1 "  
2  
3 $Id: Block1.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Block1 = Block (  
27  
28     "For the creation of Block instances, see Universe_new_block()."  
29  
30     "Evaluating"  
31     value = primitive  
32  
33 )  
34
```

```
1 "  
2  
3 $Id: Block2.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Block2 = Block (  
27  
28     "For the creation of Block instances, see Universe_new_block()."  
29  
30     "Evaluating"  
31     value          = ( self value: nil )  
32     value: argument = primitive  
33  
34 )  
35
```



```
1 "  
2  
3 $Id: Block3.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Block3 = Block (  
27  
28     "For the creation of Block instances, see Universe_new_block()."  
29  
30     "Evaluating"  
31     value                = ( self value: nil with: nil )  
32     value: arg           = ( self value: arg with: nil )  
33     value: arg1 with: arg2 = primitive  
34  
35 )  
36
```

```
1 "  
2  
3 $Id: Boolean.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Boolean = (  
27  
28     "Conditional evaluation"  
29     ifTrue: trueBlock ifFalse: falseBlock = (  
30         self ifTrue: [ ^trueBlock value ].  
31         self ifFalse: [ ^falseBlock value ].  
32     )  
33  
34     "Logical operations"  
35     || boolean = ( ^self or: boolean )  
36     && boolean = ( ^self and: boolean )  
37  
38 )  
39  
40
```

```
1 "  
2  
3 $Id: Class.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Class = (  
27  
28     "Accessing"  
29     name      = primitive  
30  
31     "Converting"  
32     asString = ( ^self name asString )  
33  
34     "Allocation"  
35     new = primitive  
36  
37     "Meta Information"  
38     superclass = primitive  
39     fields      = primitive  
40     methods     = primitive  
41     selectors   = ( ^self methods collect: [:inv | inv signature ] )  
42  
43     hasMethod: aSymbol = (  
44         self methods do: [ :m |  
45             m signature = aSymbol ifTrue: [ ^true ] ].  
46         ^false  
47     )  
48  
49 )  
50
```

```
1 "  
2  
3 $Id: Dictionary.som 29 2009-07-31 11:28:44Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Dictionary = (  
27     | pairs |  
28  
29     at: aKey put: aValue = (  
30         (self containsKey: aKey)  
31         ifTrue: [ (self pairAt: aKey) value: aValue ]  
32         ifFalse: [ pairs add: (Pair withKey: aKey andValue: aValue) ]  
33     )  
34  
35     at: aKey = (  
36         pairs do: [ :p | p key = aKey ifTrue: [ ^p value ] ].  
37         ^nil  
38     )  
39  
40     containsKey: aKey = (  
41         pairs do: [ :p | p key = aKey ifTrue: [ ^true ] ].  
42         ^false  
43     )  
44  
45     keys    = ( ^pairs collect: [ :p | p key ] )  
46     values = ( ^pairs collect: [ :p | p value ] )  
47  
48     "Iteration"  
49     do: block = ( pairs do: block )  
50  
51     "Private"  
52     pairs: aSet = ( pairs := aSet )  
53     pairAt: aKey = (  
54
```

```

55         pairs do: [ :p | p key = aKey ifTrue: [ ^p ] ].
56         ^nil
57     )
58
59     "Printing"
60     print = ( '{' print. pairs do: [ :p | p print ]. '}' print )
61     println = ( self print. '' println )
62
63     ----
64
65     new = (
66         | newDictionary |
67         newDictionary := super new.
68         newDictionary pairs: Set new.
69         ^newDictionary
70     )
71
72 )
73

```

```
1 "  
2  
3 $Id: Double.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Double = (  
27  
28     "Arithmetic"  
29     + argument = primitive  
30     - argument = primitive  
31     * argument = primitive  
32     // argument = primitive  
33     % argument = primitive  
34     abs        = ( ^(self < 0.0) ifTrue: (0.0 - self) ifFalse: self )  
35     sqrt       = primitive  
36     negated    = ( ^0.0 - self )  
37     round      = primitive  
38     asInteger  = primitive  
39     cos        = primitive  
40     sin        = primitive  
41  
42     raisedTo: exponent = (  
43         "Raise the receiver to the given exponent.  
44         Currently only positive integer exponents  
45         are fully supported."  
46         | output |  
47         output := 1.0.  
48         exponent asInteger  
49         timesRepeat: [ output := output * self ].  
50         ^ output  
51     )  
52  
53     "Comparing"  
54     = argument = primitive
```

```

55 < argument = primitive
56 > argument = ( ^(self >= argument) and: [ self <> argument ] )
57 >= argument = ( ^(self < argument) not )
58 <= argument = ( ^(self < argument) or: [ self = argument ] )
59 negative = ( ^self < 0.0 )
60 between: a and: b = ( ^(self > a) and: [ self < b ] )
61
62 "Converting"
63 asString = primitive
64
65 "Iterating"
66 to: limit do: block = (
67     | i |
68     i := self.
69     [ i <= limit ] whileTrue: [ block value: i. i := i + 1.0 ]
70 )
71
72 downTo: limit do: block = (
73     | i |
74     i := self.
75     [ i >= limit ] whileTrue: [ block value: i. i := i - 1.0 ]
76 )
77
78 ----
79
80 PositiveInfinity = primitive
81
82 "Convert string into Double. In case of any errors, the result is NaN"
83 fromString: aString = primitive
84 )
85

```

```
1 "  
2  
3 $Id: False.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 False = Boolean (  
27  
28     "Converting"  
29     asString      = ( ^'false' )  
30  
31     "Conditional evaluation"  
32     ifTrue: block = ( ^nil )  
33     ifFalse: block = ( ^block value )  
34  
35     "Logical operations"  
36     not      = ( ^true )  
37     or: block = ( ^block value )  
38     and: block = ( ^false )  
39  
40 )  
41
```



```
1 "  
2  
3 $Id: HashEntry.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "  
27 This class is not meant for direct use - it's an internal datastructure  
28 for Hashtable  
29 "  
30  
31 HashEntry = (  
32  
33     | key value next hash |  
34  
35     key      = ( ^key )  
36     value    = ( ^value )  
37     next     = ( ^next )  
38     hash     = ( ^hash )  
39  
40     key: k    = ( key := k )  
41     value: v  = ( value := v )  
42     next: n   = ( next := n )  
43     hash: h   = ( hash := h )  
44  
45     setKey: key value: value = (  
46         key = self key  
47         ifTrue: [ self value: value. ^false. ]  
48         ifFalse: [  
49             next isNil  
50                 ifTrue: [  
51                     self next: (HashEntry newKey: key value: value next:  
nil).  
52                     ^true. ]  
53                 ifFalse: [  

```

```

54             ^(self next setKey: key value: value) ] ].
55         )
56
57     getValue: key = (
58         key = self key ifTrue: [ ^value ].
59         next isNil ifTrue: [ ^nil ].
60         ^next getValue: key.
61     )
62
63     containsKey: key = (
64         key = self key ifTrue: [ ^true ].
65         next isNil ifTrue: [ ^false ].
66         ^next containsKey: key.
67     )
68
69     containsValue: value = (
70         value = self value ifTrue: [ ^true ].
71         next isNil ifTrue: [ ^false ].
72         ^next containsValue: value.
73     )
74
75     keys = (
76         next isNil
77             ifTrue: [ ^Vector new append: key ]
78             ifFalse: [ ^(next keys), key ]
79     )
80
81     values = (
82         next isNil
83             ifTrue: [ ^Vector new append: value ]
84             ifFalse: [ ^(next values), value ]
85     )
86
87     ----
88
89     newKey: k value: v next: n = (
90         | newEntry |
91         newEntry := super new.
92         newEntry key: k.
93         newEntry value: v.
94         newEntry next: n.
95         newEntry hash: (k hashCode).
96         ^newEntry
97     )
98
99 )
100

```

```
1 "  
2  
3 $Id: Hashtable.som 29 2009-07-31 11:28:44Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Hashtable = (  
27     | table count |  
28     "Testing"  
29     containsKey: key = ( | idx e |  
30         idx := self indexForKey: key.  
31         e := table at: idx.  
32         e isNil ifFalse: [ e keys do: [ :k | k = key ifTrue:  
33             [ ^true ] ] ].  
34         ^false.  
35     )  
36     containsValue: val = (  
37         table do: [ :ent |  
38             ent isNil ifFalse: [  
39                 ent values do: [ :v | v = val ifTrue: [ ^true ] ] ].  
40             ^false.  
41         )  
42     )  
43     isEmpty = ( ^count = 0 )  
44     size = ( ^count )  
45     "Accessing"  
46     get: key = ( | idx e |  
47         idx := self indexForKey: key.  
48         e := table at: idx.  
49         e isNil ifTrue: [ ^nil ].
```

```

53         ^e getValue: key.
54     )
55
56     at: key put: value = ( | idx |
57         idx := self indexForKey: key.
58         (table at: idx) isNil
59             ifTrue: [
60                 table at: idx put:
61                     (HashEntry newKey: key value: value next: nil).
62                 count := count + 1 ]
63             ifFalse: [
64                 ((table at: idx) setKey: key value: value)
65                 ifTrue: [ count := count + 1 ] ].
66         "TODO: enlarge table, rehash if too large"
67     )
68
69     "TODO: some way to delete keys'd be nice..."
70
71     "Enumerate"
72     keys = ( | vec |
73         vec := Vector new.
74         table do: [ :ent |
75             ent isNil ifFalse: [ ent keys do: [ :k | vec append: k ] ] ].
76         ^vec.
77     )
78
79     values = ( | vec |
80         vec := Vector new.
81         table do: [ :ent |
82             ent isNil ifFalse: [ ent values do: [ :v | vec append:
v ] ] ].
83         ^vec.
84     )
85
86     "Clearing"
87     clear = ( table := Array new: 11.
88         count := 0 )
89
90     "Private"
91     indexForKey: aKey = ( ^(aKey hashCode % table length) abs + 1 )
92
93     -----
94
95     "Allocation"
96     new = ( | ht |
97         ht := super new.
98         ht clear.
99         ^ht.
100     )
101
102 )
103

```

```
1 "  
2  
3 $Id: Integer.som 29 2009-07-31 11:28:44Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Integer = (  
27     "Arithmetic"  
28     + argument = primitive  
29     - argument = primitive  
30     * argument = primitive  
31     / argument = primitive  
32     // argument = primitive  
33     % argument = primitive "modulo with sign of divisor"  
34     rem: argument=primitive "modulo with sign of dividend"  
35     & argument = primitive  
36     << argument = primitive  
37     >>> argument= primitive  
38     bitXor: argument = primitive  
39     abs      = ( ^(self < 0) ifTrue: (0 - self) ifFalse: self )  
40     sqrt     = primitive  
41     negated  = ( ^0 - self )  
42  
43     raisedTo: exponent = (  
44         "Raise the receiver to the given exponent.  
45         Currently only positive integer exponents  
46         are fully supported."  
47         | output |  
48         output := 1.  
49         exponent asInteger  
50         timesRepeat: [ output := output * self ].  
51         ^ output  
52     )  
53 )
```

```

54
55     "Random numbers"
56     atRandom = primitive
57
58     "Comparing"
59     = argument = primitive
60     == argument = ( ^self = argument )
61     ~= argument = ( ^(self = argument) not )
62     < argument = primitive
63     > argument = ( ^(self >= argument) and: [ self <> argument ] )
64     >= argument = ( ^(self < argument) not )
65     <= argument = ( ^(self < argument) or: [ self = argument ] )
66     negative = ( ^self < 0 )
67     between: a and: b = ( ^(self > a) and: [ self < b ] )
68
69     "Converting"
70     asString = primitive
71     as32BitSignedValue = primitive " returns an int, with the value
that a signed 32-bit integer would have"
72     as32BitUnsignedValue = primitive " returns an int, with the value
that a unsigned 32-bit integer would have"
73     asDouble = primitive
74     asInteger = ( ^self )
75
76     hashCode = ( ^self )
77
78     "Iterating"
79     to: limit do: block = (
80         self to: limit by: 1 do: block
81     )
82
83     to: limit by: step do: block = (
84         | i |
85         i := self.
86         [ i <= limit ] whileTrue: [ block value: i. i := i + step ]
87     )
88
89     downTo: limit do: block = (
90         self downTo: limit by: 1 do: block
91     )
92
93     downTo: limit by: step do: block = (
94         | i |
95         i := self.
96         [ i >= limit ] whileTrue: [ block value: i. i := i - step ]
97     )
98
99     "More Iterations"
100    timesRepeat: block = (
101        1 to: self do: [ :i | block value ]
102    )
103
104    "Range Creation"
105    to: upper = (
106        | range |
107        range := Array new: upper - self + 1.
108        self to: upper do: [ :i | range at: i put: i ].
109        ^range
110    )
111
112    max: otherInt = (

```

```
113         (self < otherInt) ifTrue: [^otherInt] ifFalse: [^self].
114     )
115
116     min: otherInt = (
117         (self > otherInt) ifTrue: [^otherInt] ifFalse: [^self].
118     )
119
120
121     ----
122
123     fromString: aString = primitive
124
125 )
126
```

```
1 "  
2  
3 $Id: Metaclass.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Metaclass = Class ( )  
27
```



```
1 "  
2  
3 $Id: Method.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Method = (  
27  
28     "Meta Information"  
29     signature = primitive  
30     holder     = primitive  
31  
32     "Printing"  
33     asString = ( ^self holder asString + '>>' + self signature asString )  
34  
35     invokeOn: obj with: args = primitive  
36  
37 )  
38
```

```
1 "  
2  
3 $Id: Nil.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Nil = (  
27  
28     "Converting"  
29     asString = ( ^'nil' )  
30  
31     "Comparing"  
32     isNil      = ( ^true )  
33     notNil     = ( ^false )  
34  
35     "Convenience"  
36     ifNil: aBlock = (^aBlock value)  
37     ifNotNil: aBlock = (^self)  
38     ifNil: goBlock ifNotNil: noGoBlock = (^goBlock value)  
39  
40 )  
41
```

Smalltalk/Object.som

```
1 "  
2  
3 $Id: Object.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Object = nil (  
27     class      = primitive  
28     objectSize = primitive "size in bytes"  
29  
30     "Comparing"  
31  
32     " If you override =, you MUST override hashCode as well. The rule  
33     obj1 = obj2 => obj1 hashCode = obj2 hashCode  
34     must be valid for all objects, or Hashtable will not work"  
35     = other      = ( ^self == other )  
36     <> argument = ( ^(self = argument) not )  
37     == other    = primitive  
38     ~= other    = ( ^ (self == other) not )  
39     isNil       = ( ^false )  
40     notNil      = ( ^true )  
41  
42     "Converting"  
43     asString    = ( ^'instance of ' + (self class) )  
44     , element   = ( ^(Vector new append: self) append: element )  
45     hashCode    = primitive  
46  
47     "Evaluating"  
48     value      = ( ^self )  
49  
50     "Convenience"  
51     ifNil: aBlock = (^self)  
52     ifNotNil: aBlock = (^aBlock value)  
53     ifNil: noGoBlock ifNotNil: goBlock = (^goBlock value)
```

```

54
55     "Printing"
56     print      = ( self asString print )
57     println    = ( self print. system printNewline )
58
59     "Debugging"
60     inspect    = primitive
61     halt       = primitive
62
63     "Error handling"
64     error: string = ( '' println. ('ERROR: ' + string) println. system
exit: 1 )
65
66     "Abstract method support"
67     subclassResponsibility = (
68         self error: 'This method is abstract and should be overridden'
69     )
70
71     "Error recovering"
72     doesNotUnderstand: selector arguments: arguments = (
73         self error:
74             ('Method ' + selector + ' not found in class ' + self class
name)
75     )
76
77     escapedBlock: block = (
78         self error: 'Block has escaped and cannot be executed'
79     )
80
81     unknownGlobal: name = ( ^system resolve: name )
82
83     "Reflection"
84     respondsTo: aSymbol = (
85         (self class hasMethod: aSymbol)
86         ifTrue: [ ^true ]
87         ifFalse: [ | cls |
88             cls := self class superclass.
89             [ cls isNil ] whileFalse: [
90                 (cls hasMethod: aSymbol)
91                 ifTrue: [ ^true ]
92                 ifFalse: [ cls := cls superclass ] ].
93             ^ false ]
94     )
95
96     perform: aSymbol = primitive
97     perform: aSymbol withArguments: args = primitive
98
99     perform: aSymbol inSuperclass: cls = primitive
100    perform: aSymbol withArguments: args inSuperclass: cls = primitive
101
102    instVarAt: idx          = primitive
103    instVarAt: idx put: obj = primitive
104    instVarNamed: sym       = primitive
105
106 )
107

```

```
1 "  
2  
3 $Id: Pair.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Pair = (  
27     | key value |  
28     key = ( ^key )  
29     value = ( ^value )  
30     "Private"  
31     key: aKey = ( key := aKey )  
32     value: aValue = ( value := aValue )  
33     "Printing"  
34     print = ( '[' print. key print. '=>' print. value print. ']' print )  
35     println = ( self print. ' ' println )  
36     ----  
37     withKey: aKey andValue: aValue = (  
38         | newPair |  
39         newPair := super new.  
40         newPair key: aKey.  
41         newPair value: aValue.  
42         ^newPair  
43     )  
44 )  
45  
46 )  
47  
48 )  
49  
50  
51 )  
52
```

```
1 "  
2  
3 $Id: Primitive.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Primitive = (  
27  
28     signature = primitive  
29     holder    = primitive  
30  
31     invokeOn: obj with: args = primitive  
32  
33 )  
34
```

```
1 "  
2  
3 $Id: Set.som 29 2009-07-31 11:28:44Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Set = (  
27     | items |  
28  
29     = otherSet = (  
30         self size = otherSet size ifFalse: [^ false ].  
31  
32         self do: [:item | (otherSet contains: item) ifFalse: [^ false]. ].  
33  
34         ^ true.  
35     )  
36  
37     add: anObject = (  
38         (self contains: anObject)  
39         ifFalse: [ items append: anObject ]  
40     )  
41  
42     addAll: aCollection = (  
43         aCollection do: [:each |  
44             self add: each]  
45         )  
46     )  
47  
48     union: aCollection = (  
49         | new |  
50         new := Set new.  
51         new addAll: self.  
52         new addAll: aCollection.  
53         ^ new
```

```

54     )
55
56     intersection: aCollection = (
57         | new |
58         new := Set new.
59         self do: [:it |
60             (aCollection contains: it) ifTrue: [ new add: it ].
61         ^ new
62     )
63
64     - aCollection = ( "set difference"
65         | new |
66         new := Set new.
67         self do: [:it |
68             (aCollection contains: it) ifFalse: [ new add: it ].
69         ^ new
70     )
71
72     contains: anObject = (
73         items do: [ :it | it == anObject ifTrue: [ ^true ] ].
74         ^false
75     )
76
77     remove: anObject = (
78         | newItems |
79         newItems := Vector new.
80         items do: [:it |
81             it == anObject ifFalse: [ newItems append: it ].
82         items := newItems
83     )
84
85     "Sets do not have the notion of ordering, but
86     for convenience we provide those accessors"
87     first = (
88         ^items at: 1
89     )
90
91     isEmpty = (
92         ^items isEmpty
93     )
94
95     "Iteration"
96     do: block = ( items do: block )
97
98     "Collection"
99     collect: block = ( | coll |
100         coll := Vector new.
101         self do: [ :e | coll append: (block value: e) ].
102         ^coll
103     )
104
105     "Printing"
106     println = (
107         '(' print.
108         self do: [ :it | '(' print. it print. ')' print ].
109         ')' println
110     )
111
112     asString = (
113         | result |
114         result := 'a Set('.
```



```

115         items do: [:e | result := result + e asString + ', '].
116         result := result + ')'.
117         ^ result
118     )
119
120     size = (
121         ^ items size
122     )
123
124     "Private"
125     items: it = ( items := it )
126
127     ----
128
129     new = (
130         | newSet |
131         newSet := super new.
132         newSet items: Vector new.
133         ^newSet
134     )
135
136 )
137

```

Smalltalk/String.som

```
1 "  
2  
3 $Id: String.som 29 2009-07-31 11:28:44Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 String = (  
27     "Strings are immutable"  
28  
29     "Concatenate: returns a new string object"  
30     concatenate: argument = primitive  
31     + argument           = ( ^self concatenate: argument asString )  
32  
33     "Converting"  
34     asString = ( ^self )  
35     asSymbol = primitive  
36     hashCode = primitive  
37  
38     "Info"  
39     length = primitive  
40  
41     "Returns true if all characters in the string are whitespace.  
42     False otherwise, including for the empty string."  
43     isWhiteSpace = primitive  
44  
45     "Returns true if all characters in the string are letters.  
46     False otherwise, including for the empty string."  
47     isLetters = primitive  
48  
49     "Returns true if all characters in the string are digits.  
50     False otherwise, including for the empty string."  
51     isDigits = primitive  
52  
53     "Comparing"
```

```

54     = argument = primitive
55
56     "substring: from 'start' to (and including) 'end'."
57     primSubstringFrom: start to: end = primitive
58
59     substringFrom: start to: end = (
60         ((end <= self length) && (start > 0) && (start <= end))
61         ifTrue: [^self primSubstringFrom: start to: end]
62         ifFalse: [
63             self error: 'Attempting to index string out of its bounds
(start: ' + start asString + ' end: ' + end asString + ' length: ' + self
length asString + ')'] ]
64     )
65
66     beginsWith: prefix = (
67         self length < prefix length ifTrue: [ ^ false ].
68
69         1 to: prefix length do: [:i |
70             ((self charAt: i) = (prefix charAt: i)) ifFalse: [ ^ false ].
71         ].
72
73         ^ true
74     )
75
76     endsWith: suffix = (
77         | l sufL |
78         l := self length.
79         sufL := suffix length.
80         l < sufL ifTrue: [ ^ false ].
81
82         1 to: sufL do: [:i |
83             (self charAt: l - sufL + i) = (suffix charAt: i) ifFalse: [ ^
false ]
84         ].
85         ^ true
86     )
87
88     asInteger = (
89         ^ Integer fromString: self
90     )
91
92     charAt: argument = (
93         ^self substringFrom: argument to: argument
94     )
95
96     indexOf: aString = (
97         ^ self indexOf: aString startingAt: 1
98     )
99
100    indexOf: aString startingAt: start = (
101        | l |
102        l := aString length.
103        start + 1 > (self length + 1) ifTrue: [ ^ -1 ].
104
105        start to: self length - 1 + 1 do: [:i |
106            (self primSubstringFrom: i to: i + 1 - 1) = aString ifTrue: [ ^
i ].
107        ].
108
109        ^ -1
110    )

```

```

111
112     split: split = (
113         | start newStart result |
114         self length < split length ifTrue: [ ^ Array new: self ].
115         start := 1.
116         result := Vector new.
117
118         [start > 0] whileTrue: [
119             newStart := self indexOf: split startingAt: start.
120             newStart = -1
121                 ifTrue: [
122                     result append: (self primSubstringFrom: start to: self
length).
123                     ^ result asArray ]
124                 ifFalse: [
125                     result append: (self primSubstringFrom: start to: newStart -
1).
126                     start := newStart + split length ] ].
127         ^ result asArray
128     )
129
130     "Printing"
131     print = ( system printString: self )
132
133 )
134

```

```
1 "  
2  
3 $Id: Symbol.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 Symbol = String (  
27  
28     concatenate: argument = ( ^ (super concatenate: argument) asSymbol )  
29  
30     "Converting"  
31     asString = primitive  
32     asSymbol = ( ^self )  
33  
34     "Printing"  
35     print      = ( '#' print. super print )  
36  
37 )  
38
```

```

1 "
2
3 $Id: System.som 29 2009-07-31 11:28:44Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 System = (
27
28     "Accessing"
29     global: name                = primitive
30     global: name put: value     = primitive
31     hasGlobal: name            = primitive
32
33     "Initializing"
34     initialize: arguments = (
35         | application |
36
37         "Make sure we have got at least one argument"
38         (arguments length < 1) ifTrue: [ 'No class specified' println.
^nil ].
39
40         "Load the class with the specified name, create an instance of
it, and
41         run it. If there is more than only the class given on the
command line,
42         and the class has a method #run:, the arguments array is passed
to it,
43         otherwise, #run is sent."
44         application := (self resolve: (arguments at: 1) asSymbol) new.
45
46         ^ (application respondsTo: #run:)
47           ifTrue: [ application run: arguments ]
48           ifFalse: [ application run ]
49     )

```

```

50
51     "File support"
52
53     "Load a file identified by a path. Return content as a string"
54     loadFile: fileName = primitive
55
56     "Loading and resolving"
57     load:      symbol = primitive
58     resolve: symbol = (
59         | class |
60
61         "Check if we've already got the global"
62         (self global: symbol) == nil ifFalse: [ ^self global: symbol ].
63
64         "Try loading the class"
65         class := self load: symbol.
66         (class == nil) ifFalse: [
67             ^class ].
68         self error: 'Tried loading \'' + symbol + '\'' as a class, but
failed.'
69     )
70
71     "Exiting"
72     exit: error = primitive
73     exit      = ( self exit: 0 )
74
75     "Printing"
76     printString: string = primitive
77     printNewline      = primitive
78
79     errorPrintln = ( self errorPrintln: '' )
80     errorPrintln: string = primitive
81     errorPrint: string = primitive
82
83     printStackTrace = primitive
84
85     "Time"
86     time = primitive
87     ticks = primitive      "returns the microseconds since start"
88
89     "Force Garbage Collection"
90     fullGC = primitive
91
92     "To be implemented by SOM implementations that gather such
statistics."
93     gcStats = ( ^ #(
94         0 "Total number of GCs"
95         0 "Estimated total GC time in milliseconds"
96         0 "Approximate number of allocated bytes of current thread") )
97     totalCompilationTime = ( ^ 0 "Estimated total compilation time in
milliseconds" )
98
99     -----
100
101     "Allocation"
102     new = ( self error: 'The system object is singular' )
103
104 )
105

```

```
1 "  
2  
3 $Id: True.som 27 2009-07-31 11:17:53Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 True = Boolean (  
27  
28     "Converting"  
29     asString      = ( ^'true' )  
30  
31     "Conditional evaluation"  
32     ifTrue: block = ( ^block value )  
33     ifFalse: block = ( ^nil )  
34  
35     "Logical operations"  
36     not      = ( ^false )  
37     or: block = ( ^true )  
38     and: block = ( ^block value )  
39  
40 )  
41
```



```
1 "  
2  
3 $Id: Vector.som 29 2009-07-31 11:28:44Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "FIXME: Implement pushFront and popFront..."  
27  
28 Vector = (  
29  
30     | first last storage |  
31  
32     "Accessing"  
33     at: index = (  
34         | storeIdx |  
35         storeIdx := index + first - 1.  
36         ^ self checkIndex: storeIdx ifValid: [ storage at: storeIdx ]  
37     )  
38  
39     at: index put: value = (  
40         | storeIdx |  
41         storeIdx := index + first - 1.  
42         ^ self checkIndex: storeIdx ifValid: [ storage at: storeIdx put:  
value ]  
43     )  
44  
45     first = ( ^ (self size > 0) ifTrue: [storage at: first] ifFalse:  
[nil] )  
46     last = ( ^ (self size > 0) ifTrue: [storage at: last - 1] ifFalse:  
[nil] )  
47  
48     "Iterating"  
49     do: block = (  
50         first to: last - 1 do: [ :i | block value: (storage at: i) ]
```

```

51     )
52
53     doIndexes: block = (
54         1 to: last - first do: block
55     )
56
57     "Adding"
58     , element = ( ^self append: element )
59
60     append: element = (
61         (last > storage length) ifTrue: [
62             "Need to expand capacity first"
63             | newStorage |
64             newStorage := Array new: (2 * storage length).
65             storage doIndexes: [ :i | newStorage at: i put: (storage at:
i) ].
66             storage := newStorage. ].
67
68         storage at: last put: element.
69         last := last + 1.
70         ^self
71     )
72
73     appendAll: collection = (
74         collection do: [:e |
75             self append: e ]
76     )
77
78     "Removing"
79     remove = (
80         | value |
81         self isEmpty ifTrue: [ self error: 'Vector: Attempting to remove
the last element from an empty Vector' ].
82         last := last - 1.
83         value := storage at: last.
84         storage at: last put: nil.
85         ^ value
86     )
87
88     remove: object = (
89         | newArray newLast found |
90         newArray := Array new: self capacity.
91         newLast := 1.
92         found := false.
93
94         self do: [ :it |
95             (it == object)
96             ifTrue: [ found := true ]
97             ifFalse: [
98                 newArray at: newLast put: it.
99                 newLast := newLast + 1.
100             ]
101         ].
102
103         storage := newArray.
104         last := newLast.
105         first := 1.
106         ^found
107     )
108
109     contains: anObject = (

```

```

110         self do: [ :element |
111             element = anObject ifTrue: [ ^ true ] ].
112         ^ false
113     )
114
115     "If anObject is in vector, return index of first occurrence.
116     If it isn't, return -1."
117     indexOf: anObject = (
118         first to: last - 1 do: [ :i |
119             (storage at: i) = anObject ifTrue: [ ^ i - first + 1 ] ].
120         ^ -1
121     )
122
123     "Printing"
124     println = (
125         '(' print.
126         self do: [ :it | '(' print. it print. ')' print ].
127         ')' println
128     )
129
130     "Sizing"
131     isEmpty = ( ^last = first )
132     size = ( ^last - first )
133     capacity = ( ^storage length )
134
135     "Conversion"
136     asArray = ( | arr |
137         arr := Array new: self size.
138         self doIndexes: [ :i | arr at: i put: (self at: i) ].
139         ^arr
140     )
141
142     "Private"
143     initialize: size = (
144         storage := Array new: size.
145         first := 1.
146         last := 1.
147     )
148
149     checkIndex: index ifValid: block = (
150         ^ ((first <= index) && (index < last)
151             ifTrue: [ block value ]
152             ifFalse: [
153                 self error:
154                     'Vector[' + first asString + '...' + last asString +
155                     ']: Index ' + index asString + ' out of bounds' ])
156     )
157
158     "DeltaBlue"
159     removeFirst = (
160         | value |
161         self isEmpty ifTrue: [ self error: 'Vector: Attempting to remove
the first element from an empty Vector' ].
162         value := storage at: first.
163         storage at: first put: nil.
164         first := first + 1.
165         ^ value
166     )
167
168     "Conversion"
169     asSet = (

```

```

170         ^ Set new addAll: self
171     )
172
173     -----
174
175     "Allocation"
176     new          = ( ^ self new: 50 )
177     new: initialSize = ( ^ super new initialize: initialSize )
178
179     with: elem = (
180         | newVector |
181         newVector := self new: 1.
182         newVector append: elem.
183         ^ newVector
184     )
185
186 )
187

```

```

1 BytecodeGenerator = (
2   ----
3   emitPop: mgenc = (
4     self emit: mgenc bc: #pop
5   )
6
7   emit: mgenc pushArgument: idx in: ctx = (
8     self emit: mgenc bc: #pushArgument with: idx and: ctx
9   )
10
11   emitReturnLocal: mgenc = (
12     self emit: mgenc bc: #returnLocal
13   )
14
15   emitReturnNonLocal: mgenc = (
16     self emit: mgenc bc: #returnNonLocal
17   )
18
19   emitDup: mgenc = (
20     self emit: mgenc bc: #dup
21   )
22
23   emit: mgenc pushBlock: blockMethod = (
24     self emit: mgenc bc: #pushBlock with: (mgenc addLiteralIfAbsent:
25     blockMethod)
26   )
27
28   emit: mgenc pushLocal: idx in: ctx = (
29     idx negative ifTrue: [ self error: 'pushLocal: ' + idx asString].
30     self emit: mgenc bc: #pushLocal with: idx and: ctx
31   )
32
33   emit: mgenc pushField: aSymbol = (
34     (mgenc hasField: aSymbol) ifFalse: [ self error: 'pushField: field
35     unknown ' + aSymbol ].
36     self emit: mgenc bc: #pushField with: (mgenc fieldIndex: aSymbol)
37   )
38
39   emit: mgenc pushGlobal: aSymbol = (
40     self emit: mgenc bc: #pushGlobal with: (mgenc addLiteralIfAbsent:
41     aSymbol)
42   )
43
44   emit: mgenc popArgument: idx in: ctx = (
45     idx negative ifTrue: [ self error: 'popArgument: ' + idx asString].
46     self emit: mgenc bc: #popArgument with: idx and: ctx
47   )
48
49   emit: mgenc popLocal: idx in: ctx = (
50     idx negative ifTrue: [ self error: 'popLocal: ' + idx asString].
51     self emit: mgenc bc: #popLocal with: idx and: ctx
52   )
53
54   emit: mgenc popField: aSymbol = (
55     (mgenc hasField: aSymbol) ifFalse: [ self error: 'popField: field
56     unknown ' + aSymbol ].
57     self emit: mgenc bc: #popField with: (mgenc fieldIndex: aSymbol)
58   )
59 )

```

```

55
56   emit: mgenc superSend: aSymbol = (
57       self emit: mgenc bc: #superSend with: (mgenc addLiteralIfAbsent:
58           aSymbol)
59   )
60   emit: mgenc send: aSymbol = (
61       self emit: mgenc bc: #send with: (mgenc addLiteralIfAbsent: aSymbol)
62   )
63
64   emit: mgenc pushConstant: anAbstractObject = (
65       self emit: mgenc bc: #pushConstant with: (mgenc addLiteralIfAbsent:
66           anAbstractObject)
67   )
68   emit: mgenc pushConstantIdx: anInteger = (
69       self emit: mgenc bc: #pushConstant with: anInteger
70   )
71
72   emit: mgenc bc: aSymbol = (
73       mgenc addBytecode: aSymbol.
74   )
75
76   emit: mgenc bc: aSymbol with: anInteger = (
77       mgenc addBytecode: aSymbol.
78       mgenc addBytecode: anInteger.
79   )
80
81   emit: mgenc bc: aSymbol with: anInteger and: otherInteger = (
82       mgenc addBytecode: aSymbol.
83       mgenc addBytecode: anInteger.
84       mgenc addBytecode: otherInteger.
85   )
86 )
87

```

```

1 ClassGenerationContext = (
2   | universe
3   name superName
4   classSide
5   classFields instanceFields
6   classMethods instanceMethods |
7
8   initialize: aUniverse = (
9     universe := aUniverse.
10    classSide := false.
11    classFields := Vector new.
12    instanceFields := Vector new.
13    classMethods := Vector new.
14    instanceMethods := Vector new.
15  )
16
17  name = (
18    ^ name
19  )
20
21  name: aSSymbol = (
22    name := aSSymbol
23  )
24
25  superName = (
26    ^ superName
27  )
28
29  superName: aSymbol = (
30    superName := aSymbol
31  )
32
33  instanceFieldsOfSuper: aArrayOfFieldNames = (
34    | numFields |
35    numFields := aArrayOfFieldNames numberOfIndexableFields.
36    1 to: numFields do: [:i |
37      instanceFields append: (aArrayOfFieldNames indexableField: i) ]
38  )
39
40  classFieldsOfSuper: aArrayOfFieldNames = (
41    | numFields |
42    numFields := aArrayOfFieldNames numberOfIndexableFields.
43    1 to: numFields do: [:i |
44      classFields append: (aArrayOfFieldNames indexableField: i) ]
45  )
46
47  addField: aSymbol = (
48    classSide
49    ifTrue: [classFields append: aSymbol]
50    ifFalse: [instanceFields append: aSymbol]
51  )
52
53  hasField: aSymbol = (
54    ^ classSide
55    ifTrue: [classFields contains: aSymbol]
56    ifFalse: [instanceFields contains: aSymbol]
57  )
58

```

```

59   fieldIndex: aSymbol = (
60     ^ classSide
61     ifTrue: [classFields indexOf: aSymbol]
62     ifFalse: [instanceFields indexOf: aSymbol]
63   )
64
65   addMethod: anInvokable = (
66     classSide
67     ifTrue: [classMethods append: anInvokable]
68     ifFalse: [instanceMethods append: anInvokable]
69   )
70
71   startClassSide = (
72     classSide := true
73   )
74
75   assemble = (
76     | ccname superClass resultClass superMClass result |
77     "build class class name"
78     ccname := name string + ' class'.
79
80     "Load the super class"
81     superClass := universe loadClass: superName.
82
83     "Allocate the class of the resulting class"
84     resultClass := universe newClass: universe metaclassClass.
85
86     "Initialize the class of the resulting class"
87     resultClass instanceFields: (universe newArrayFromVector:
classFields).
88     resultClass instanceInvokables: (universe newArrayFromVector:
classMethods).
89     resultClass name: (universe symbolFor: ccname).
90
91     superMClass := superClass somClass.
92     resultClass superClass: superMClass.
93
94     "Allocate the resulting class"
95     result := universe newClass: resultClass.
96
97     "Initialize the resulting class"
98     result name: name.
99     result superClass: superClass.
100    result instanceFields: (universe newArrayFromVector: instanceFields).
101    result instanceInvokables: (universe newArrayFromVector:
instanceMethods).
102
103    ^ result
104  )
105
106  assembleSystemClass: systemClass = (
107    | superMClass |
108    systemClass instanceInvokables: (universe newArrayFromVector:
instanceMethods).
109    systemClass instanceFields: (universe newArrayFromVector:
instanceFields).
110
111    "class-bound == class-instance-bound"
112    superMClass := systemClass somClass.
113    superMClass instanceInvokables: (universe newArrayFromVector:
classMethods).

```



```
114      superMClass instanceFields: (universe newArrayFromVector:
classFields).
115
116      ^ systemClass
117  )
118
119  ----
120
121  new: aUniverse = (
122      ^ self new initialize: aUniverse
123  )
124 )
125
```

```

1 Disassembler = (
2
3 ----
4
5 dump: cl in: universe = (
6   1 to: cl numberOfInstanceInvokables do: [:i |
7     | inv |
8     inv := cl instanceInvokable: i.
9
10    "output header and skip if the Invokable is a Primitive"
11    Universe errorPrint: (cl name string + '>>' + inv signature string
+ ' = ').
12
13    inv isPrimitive
14      ifTrue: [ Universe errorPrintln: '<primitive>' ]
15      ifFalse: [ self dumpMethod: inv indent: '\t' in: universe ] ]
16  )
17
18 dumpInvokable: inv in: universe = (
19   | holderName |
20   holderName := inv holder == nil
21     ifTrue: ['nil']
22     ifFalse: [inv holder name string].
23   Universe errorPrint: (holderName + '>>#' + inv signature string + ' =
' ).
24   inv isPrimitive
25     ifTrue: [
26       Universe errorPrint: '<primitive>: '.
27       Universe errorPrintln: inv debugString ]
28     ifFalse: [ self dumpMethod: inv indent: '\t' in: universe ]
29  )
30
31 dumpMethod: m indent: indent in: universe = (
32   | b |
33   Universe errorPrintln: '('.
34
35   "output stack information"
36   Universe errorPrintln: indent + '<' + m numberOfLocals + ' locals, '
37     + m maximumNumberOfStackElements + ' stack, '
38     + m numberOfBytecodes + ' bc_count>'.
39
40   "output bytecodes"
41   b := 1.
42   [b <= m numberOfBytecodes] whileTrue: [
43     | bytecode |
44     Universe errorPrint: indent.
45
46     b < 10 ifTrue: [ Universe errorPrint: ' ' ].
47     b < 100 ifTrue: [ Universe errorPrint: ' ' ].
48
49     Universe errorPrint: ' ' + b + ':' .
50
51     "mnemonic"
52     bytecode := m bytecode: b.
53     Universe errorPrint: (Bytecodes paddedBytecodeName: bytecode) + '
' .
54
55     "parameters (if any)"

```

```

56         (Bytecodes length: bytecode) = 1
57         ifTrue: [ Universe errorPrintln ]
58         ifFalse: [ self dumpBytecode: bytecode idx: b method: m indent:
indent in: universe ].
59
60         b := b + (Bytecodes length: (m bytecode: b)) ].
61
62     Universe errorPrintln: indent + ')'
63 )
64
65     dumpBytecode: bc idx: b method: m indent: indent in: universe = (
66         bc == #pushLocal ifTrue: [
67             Universe errorPrintln: 'local: ' + (m bytecode: b + 1) + ',
context: ' + (m bytecode: b + 2).
68             ^ self ].
69         bc == #pushArgument ifTrue: [
70             Universe errorPrintln: 'argument: ' + (m bytecode: b + 1) + ',
context: ' + (m bytecode: b + 2).
71             ^ self ].
72         bc == #pushField ifTrue: [
73             | idx fieldName |
74             idx := m bytecode: b + 1.
75             fieldName := (m holder instanceFields indexableField: idx) string.
76             Universe errorPrintln: '(index: ' + idx + ') field: ' + fieldName.
77             ^ self ].
78         bc == #pushBlock ifTrue: [
79             Universe errorPrint: '(block: (index: ' + (m bytecode: b + 1) + ')
'.
80             self dumpMethod: (m constant: b) indent: indent + '\t' in: universe.
81             ^ self ].
82         bc == #pushConstant ifTrue: [
83             | constant |
84             constant := m constant: b.
85             Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ') value:
',
86             + '(' + (constant somClassIn: universe) name string + ') ' +
constant debugString.
87             ^ self ].
88         bc == #pushGlobal ifTrue: [
89             Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ') value:
#' + (m constant: b) string.
90             ^ self ].
91         bc == #popLocal ifTrue: [
92             Universe errorPrintln: 'local: ' + (m bytecode: b + 1) + ',
context: ' + (m bytecode: b + 2).
93             ^ self ].
94         bc == #popArgument ifTrue: [
95             Universe errorPrintln: 'argument: ' + (m bytecode: b + 1) + ',
context: ' + (m bytecode: b + 2).
96             ^ self ].
97         bc == #pushField ifTrue: [
98             | idx fieldName |
99             idx := m bytecode: b + 1.
100             fieldName := (m holder instanceFields indexableField: idx) string.
101             Universe errorPrintln: '(index: ' + idx + ') field: ' + fieldName.
102             ^ self ].
103         bc == #send ifTrue: [
104             Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ')
signature: #' + (m constant: b) string.
105             ^ self ].
106         bc == #superSend ifTrue: [

```

```
107         Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ')\nsignature: #' + (m constant: b) string.\n108         ^ self ].\n109\n110         Universe errorPrintln: '<unknown bytecode>'\n111     )\n112 )\n113
```

```

1 Lexer = (
2   | fileContent state stateAfterPeek peekDone
3   index
4   sym text
5   nextSym nextText
6   |
7
8   initialize: aString = (
9     fileContent := aString.
10    peekDone := false.
11    index := 1.
12  )
13
14  isPeekDone = ( ^ peekDone )
15  text = ( ^ text )
16
17  currentTextContext = (
18    | start end |
19    start := (index - 50) max: 1.
20    end := (index + 5) min: fileContent length.
21    ^ fileContent substringFrom: start to: end
22  )
23
24  peek = (
25    | savedSym savedText |
26    peekDone ifTrue: [
27      self error: 'SOM lexer cannot peek twice. Likely parser bug' ].
28
29    savedSym := sym.
30    savedText := text.
31    self sym.
32
33    nextSym := sym.
34    nextText := text.
35    peekDone := true.
36
37    sym := savedSym.
38    text := savedText.
39
40    ^ nextSym
41  )
42
43  sym = (
44    peekDone ifTrue: [
45      peekDone := false.
46      sym := nextSym.
47      text := nextText.
48      ^ sym ].
49
50    self hasMoreInput ifFalse: [
51      sym := #NONE.
52      text := nil.
53      ^ sym ].
54
55    [self currentChar isWhiteSpace or: [self currentChar = '"']]
whileTrue: [
56      self skipWhiteSpace.
57      self skipComment ].

```

```

58
59     self currentChar = '\\' ifTrue: [
60         ^ self lexString ].
61     self currentChar = '[' ifTrue: [
62         ^ self match: #newBlock ].
63     self currentChar = ']' ifTrue: [
64         ^ self match: #endBlock ].
65
66     self currentChar = ':' ifTrue: [
67         self nextChar = '='
68         ifTrue: [
69             index := index + 2.
70             sym := #assign.
71             text := ':'= ]
72         ifFalse: [
73             index := index + 1.
74             sym := #colon.
75             text := ':'
76         ].
77         ^ sym ].
78
79     self currentChar = '(' ifTrue: [
80         ^ self match: #newTerm ].
81     self currentChar = ')' ifTrue: [
82         ^ self match: #endTerm ].
83     self currentChar = '#' ifTrue: [
84         ^ self match: #pound ].
85     self currentChar = '^' ifTrue: [
86         ^ self match: #exit ].
87     self currentChar = '.' ifTrue: [
88         ^ self match: #period ].
89     self currentChar = '-' ifTrue: [
90         (self currentMatches: Lexer sepStr)
91         ifTrue: [
92             text := ''.
93             [self currentChar = '-'] whileTrue: [
94                 text := text + self currentChar.
95                 index := index + 1 ].
96
97             ^ sym := #separator ]
98         ifFalse: [
99             ^ self lexOperator ] ].
100     (Lexer isOperator: self currentChar) ifTrue: [
101         ^ self lexOperator ].
102     (self currentMatches: Lexer primStr) ifTrue: [
103         index := index + Lexer primStr length.
104         text := Lexer primStr.
105         ^ sym := #primitive ].
106     self currentChar isLetters ifTrue: [
107         text := ''.
108         [self currentChar isLetters or: [self currentChar isDigits or:
109 [self currentChar = '_']] whileTrue: [
110             text := text + self currentChar.
111             index := index + 1 ].
112         sym := #identifier.
113
114     self currentChar = ':' ifTrue: [
115         sym := #keyword.
116         index := index + 1.
117         text := text + ':'.
118         self currentChar isLetters ifTrue: [

```

```

118         sym := #keywordSequence.
119         [self currentChar isLetters or: [self currentChar = ':']]
whileTrue: [
120             text := text + self currentChar.
121             index := index + 1 ] ] ].
122     ^ sym ].
123     self currentChar isDigits ifTrue: [
124         ^ self lexNumber ].
125
126     text := self currentChar.
127     ^ sym := #NONE
128 )
129
130 lexNumber = (
131     | sawDecimalMark |
132     sym := #integer.
133     text := ''.
134
135     sawDecimalMark := false.
136
137     [self currentChar isDigits] whileTrue: [
138         text := text + self currentChar.
139         index := index + 1.
140         (sawDecimalMark not and: [
141             self currentChar = '.' and: [
142                 self nextChar isDigits]]) ifTrue: [
143             sym := #double.
144             text := text + self currentChar.
145             index := index + 1 ] ].
146     ^ sym
147 )
148
149 lexEscapeChar = (
150     self currentChar = 't' ifTrue: [ ^ text := text + '\\t' ].
151     self currentChar = 'b' ifTrue: [ ^ text := text + '\\b' ].
152     self currentChar = 'n' ifTrue: [ ^ text := text + '\\n' ].
153     self currentChar = 'r' ifTrue: [ ^ text := text + '\\r' ].
154     self currentChar = 'f' ifTrue: [ ^ text := text + '\\f' ].
155     self currentChar = '\\' ifTrue: [ ^ text := text + '\\\\' ].
156     self currentChar = '\\\\' ifTrue: [ ^ text := text + '\\\\' ].
157     self currentChar = '0' ifTrue: [ ^ text := text + '\\0' ].
158
159     self error: 'Unknown escape sequence \\' + self currentChar
160 )
161
162 lexStringChar = (
163     self currentChar = '\\\\'
164     ifTrue: [
165         index := index + 1.
166         self lexEscapeChar.
167         index := index + 1 ]
168     ifFalse: [
169         text := text + self currentChar.
170         index := index + 1 ]
171 )
172
173 lexString = (
174     sym := #string.
175     text := ''.
176     index := index + 1.
177

```

```

178     [self currentChar = '\\''] whileFalse: [
179         self lexStringChar ].
180
181     index := index + 1.
182     ^ sym
183 )
184
185 lexOperator = (
186     (Lexer isOperator: self nextChar) ifTrue: [
187         text := ''.
188         [Lexer isOperator: self currentChar] whileTrue: [
189             text := text + self currentChar.
190             index := index + 1 ].
191         ^ sym := #operatorSequence ].
192     self currentChar = '~' ifTrue: [
193         ^ self match: #not ].
194     self currentChar = '&' ifTrue: [
195         ^ self match: #and ].
196     self currentChar = '|' ifTrue: [
197         ^ self match: #or ].
198     self currentChar = '*' ifTrue: [
199         ^ self match: #star ].
200     self currentChar = '/' ifTrue: [
201         ^ self match: #div ].
202     self currentChar = '\\' ifTrue: [
203         ^ self match: #mod ].
204     self currentChar = '+' ifTrue: [
205         ^ self match: #plus ].
206     self currentChar = '=' ifTrue: [
207         ^ self match: #equal ].
208     self currentChar = '>' ifTrue: [
209         ^ self match: #more ].
210     self currentChar = '<' ifTrue: [
211         ^ self match: #less ].
212     self currentChar = ',' ifTrue: [
213         ^ self match: #comma ].
214     self currentChar = '@' ifTrue: [
215         ^ self match: #at ].
216     self currentChar = '%' ifTrue: [
217         ^ self match: #per ].
218     self currentChar = '-' ifTrue: [
219         ^ self match: #minus ].
220     self error: 'lexOperator ran out of options. This should not happen'
221 )
222
223 skipWhiteSpace = (
224     [self currentChar isWhiteSpace] whileTrue: [
225         index := index + 1 ]
226 )
227
228 skipComment = (
229     self currentChar = '"'
230     ifFalse: [ ^ self ].
231
232     index := index + 1.
233
234     [self currentChar = '"'] whileFalse: [
235         index := index + 1 ].
236
237     index := index + 1
238 )

```



```

239
240     currentChar = (
241         index > fileContent length ifTrue: [ ^ '\0' ].
242         ^ fileContent charAt: index
243     )
244
245     nextChar = (
246         (index + 1) > fileContent length ifTrue: [ ^ '\0' ].
247         ^ fileContent charAt: index + 1
248     )
249
250     hasMoreInput = (
251         ^ index <= fileContent length
252     )
253
254     currentMatches: str = (
255         (index + str length) <= fileContent length ifFalse: [ ^ false ].
256         ^ str = (fileContent substringFrom: index to: index - 1 + str length)
257     )
258
259     match: s = (
260         sym := s.
261         text := self currentChar.
262         index := index + 1.
263         ^ sym
264     )
265
266     ----
267
268     new: aString = (
269         ^ self new initialize: aString
270     )
271
272     isOperator: c = (
273         c = '~' ifTrue: [ ^ true ].
274         c = '&' ifTrue: [ ^ true ].
275         c = '|' ifTrue: [ ^ true ].
276         c = '*' ifTrue: [ ^ true ].
277         c = '/' ifTrue: [ ^ true ].
278         c = '\\' ifTrue: [ ^ true ].
279         c = '+' ifTrue: [ ^ true ].
280         c = '=' ifTrue: [ ^ true ].
281         c = '>' ifTrue: [ ^ true ].
282         c = '<' ifTrue: [ ^ true ].
283         c = ',' ifTrue: [ ^ true ].
284         c = '@' ifTrue: [ ^ true ].
285         c = '%' ifTrue: [ ^ true ].
286         c = '-' ifTrue: [ ^ true ].
287         ^ false
288     )
289
290     sepStr = ( ^ '----' )
291     primStr = ( ^ 'primitive' )
292 )
293

```

```

1 MethodGenerationContext = (
2   | holderGenc outerGenc
3   arguments locals literals
4   signature
5   finished prim blockMethod
6   bytecode |
7
8   initializeWith: aHolderGenc and: aOuterGenc = (
9     holderGenc := aHolderGenc.
10    outerGenc := aOuterGenc.
11    arguments := Vector new.
12    locals := Vector new.
13    literals := Vector new.
14    finished := false.
15    prim := false.
16    blockMethod := false.
17    bytecode := Vector new.
18  )
19
20  holder = (
21    ^ holderGenc
22  )
23
24  signature: aSymbol = (
25    signature := aSymbol
26  )
27
28  addArgument: aString = (
29    arguments append: aString
30  )
31
32  addArgumentIfAbsent: aString = (
33    (arguments contains: aString)
34    ifTrue: [^ false].
35
36    arguments append: aString.
37    ^ true
38  )
39
40  numberOfArguments = (
41    ^ arguments size
42  )
43
44  addLocalIfAbsent: aString = (
45    (locals contains: aString)
46    ifTrue: [^ false].
47
48    locals append: aString.
49    ^ true
50  )
51
52  addLiteralIfAbsent: anAbstractObject = (
53    | idx |
54    idx := literals indexOf: anAbstractObject.
55    idx <> -1 ifTrue: [
56      ^ idx ].
57
58    ^ self addLiteral: anAbstractObject

```

```

59 )
60
61 addLiteral: anAbstractObject = (
62     literals append: anAbstractObject.
63     ^ literals size
64 )
65
66 updateLiteral: oldVal at: idx put: newVal = (
67     (literals at: idx) == oldVal ifFalse: [
68         self error: 'updateLiteral saw wrong oldVal, indicates bug in
parser' ].
69     literals at: idx put: newVal
70 )
71
72 findVar: var with: searchResult = (
73     "searchResult: index, context, isArgument"
74     searchResult at: 1 put: (locals indexOf: var).
75     (searchResult at: 1) = -1 ifTrue: [
76         searchResult at: 1 put: (arguments indexOf: var).
77         (searchResult at: 1) = -1
78         ifTrue: [
79             outerGenc == nil
80             ifTrue: [^ false]
81             ifFalse: [
82                 searchResult at: 2 put: (searchResult at: 2) + 1.
83                 ^ outerGenc findVar: var with: searchResult ] ]
84         ifFalse: [
85             searchResult at: 3 put: true ] ].
86
87     ^ true
88 )
89
90 markAsFinished = (
91     finished := true
92 )
93
94 isFinished = (
95     ^ finished
96 )
97
98 markAsPrimitive = (
99     prim := true
100 )
101
102 isBlockMethod = (
103     ^ blockMethod
104 )
105
106 markAsBlockMethod = (
107     blockMethod := true
108 )
109
110 addBytecode: code = (
111     bytecode append: code
112 )
113
114 removeLastBytecode = (
115     bytecode remove
116 )
117
118 hasBytecodes = (

```

```

119     ^ bytecode isEmpty not
120 )
121
122 hasField: aSymbol = (
123     ^ holderGenc hasField: aSymbol
124 )
125
126 fieldIndex: aSymbol = (
127     ^ holderGenc fieldIndex: aSymbol
128 )
129
130 assemble: universe = (
131     prim
132     ifTrue: [
133         ^ SPrimitive emptyPrimitive: signature string in: universe ]
134     ifFalse: [
135         ^ self assembleMethod: universe ]
136 )
137
138 assembleMethod: universe = (
139     | numLocals meth i |
140     "create a method instance with the given number of bytecodes"
141     numLocals := locals size.
142
143     meth := universe newMethod: signature
144         bc: bytecode asArray literals: literals asArray
145         numLocals: numLocals maxStack: self computeStackDepth.
146
147     "return the method - the holder field is to be set later on!"
148     ^ meth
149 )
150
151 computeStackDepth = (
152     | depth maxDepth i |
153     depth := 0.
154     maxDepth := 0.
155     i := 1.
156
157     [i <= bytecode size] whileTrue: [
158         | bc |
159         bc := bytecode at: i.
160
161         (bc == #dup          or: [
162         bc == #pushLocal    or: [
163         bc == #pushArgument or: [
164         bc == #pushField    or: [
165         bc == #pushBlock    or: [
166         bc == #pushConstant or: [
167         bc == #pushGlobal ] ] ] ] ] ]) ifTrue: [
168         depth := depth + 1 ] ifFalse: [
169
170         (bc == #pop          or: [
171         bc == #popLocal      or: [
172         bc == #popArgument or: [
173         bc == #popField ] ] ]) ifTrue: [
174         depth := depth - 1 ] ifFalse: [
175
176         (bc == #send or: [bc == #superSend]) ifTrue: [
177             | sig |
178             "these are special: they need to look at the number of
179             arguments (extractable from the signature)"

```

```

180         sig := literals at: (bytecode at: i + 1).
181         depth := depth - sig numberOfSignatureArguments.
182         depth := depth + 1 "return value" ] ] ].
183
184     i := i + (Bytecodes length: bc).
185
186     depth > maxDepth ifTrue: [
187         maxDepth := depth ] ].
188
189     ^ maxDepth
190 )
191
192 ----
193
194 new: holderGenc = (
195     ^ self new initializeWith: holderGenc and: nil
196 )
197
198 new: holderGenc with: outerGenc = (
199     ^ self new initializeWith: holderGenc and: outerGenc
200 )
201 )
202

```

```

1 Parser = (
2   | lexer sym text nextSym filename cgener universe bcGen |
3
4   initializeWith: aString for: aFilename in: aUniverse = (
5     filename := aFilename.
6     lexer := Lexer new: aString.
7     universe := aUniverse.
8     cgener := ClassGenerationContext new: universe.
9     self takeSymbolFromLexer.
10
11     "This is just a convenient abbreviation."
12     bcGen := BytecodeGenerator.
13   )
14
15   takeSymbolFromLexer = (
16     sym := lexer sym.
17     text := lexer text.
18     nextSym := #none.
19   )
20
21   classdef = (
22     cgener name: (universe symbolFor: text).
23     self expect: #identifier.
24     self expect: #equal.
25
26     self superclass.
27
28     self expect: #newTerm.
29     self classBody.
30
31     (self accept: #separator) ifTrue: [
32       cgener startClassSide.
33       self classBody ].
34
35     self expect: #endTerm.
36     ^ cgener
37   )
38
39   classBody = (
40     self fields.
41     [self symIsMethod] whileTrue: [
42       | mgener |
43       mgener := MethodGenerationContext new: cgener.
44       mgener addArgument: 'self'.
45       self method: mgener.
46       cgener addMethod: (mgener assemble: universe) ].
47   )
48
49   superclass = (
50     | superName |
51     sym == #identifier
52     ifTrue: [
53       superName := universe symbolFor: text.
54       self accept: #identifier ]
55     ifFalse: [
56       superName := universe symbolFor: 'Object' ].
57
58     cgener superName: superName.

```

```

59
60     superName string = 'nil' ifFalse: [
61         self initializeFromSuperClass: superName ].
62     )
63
64     initializeFromSuperClass: superName = (
65         | superClass |
66         superClass := universe loadClass: superName.
67         superClass == nil ifTrue: [
68             self error: 'Was not able to load super class: ' + superName string
+ ' in ' + filename ].
69         cgenc instanceFieldsOfSuper: superClass instanceFields.
70         cgenc classFieldsOfSuper: superClass somClass instanceFields.
71     )
72
73     fields = (
74         (self accept: #or) ifTrue: [
75             [sym == #identifier] whileTrue: [
76                 | var |
77                 var := self variable.
78                 cgenc addField: (universe symbolFor: var) ].
79             self expect: #or ]
80     )
81
82     method: mgenc = (
83         self pattern: mgenc.
84         self expect: #equal.
85
86         sym == #primitive
87         ifTrue: [
88             mgenc markAsPrimitive.
89             self primBlock ]
90         ifFalse: [
91             self methodBlock: mgenc ]
92     )
93
94     primBlock = (
95         self expect: #primitive
96     )
97
98     pattern: mgenc = (
99         sym == #identifier ifTrue: [
100             ^ self unaryPattern: mgenc ].
101         sym == #keyword ifTrue: [
102             ^ self keywordPattern: mgenc ].
103         self binaryPattern: mgenc
104     )
105
106     unaryPattern: mgenc = (
107         mgenc signature: self unarySelector
108     )
109
110     binaryPattern: mgenc = (
111         mgenc signature: self binarySelector.
112         mgenc addArgumentIfAbsent: self argument
113     )
114
115     keywordPattern: mgenc = (
116         | kw |
117         kw := ''.
118

```

```

119     [sym == #keyword] whileTrue: [
120         kw := kw + self keyword.
121         mgenc addArgumentIfAbsent: self argument ].
122
123     mgenc signature: (universe symbolFor: kw)
124 )
125
126     methodBlock: mgenc = (
127         self expect: #newTerm.
128
129         self blockContents: mgenc.
130
131         " if no return has been generated so far, we can be sure there was
no . (dot)
132         terminating the last expression, so the last expression's value
must be
133         popped off the stack and a ^self be generated "
134         mgenc isFinished ifFalse: [
135             bcGen emitPop: mgenc.
136             bcGen emit: mgenc pushArgument: 1 in: 0.
137             bcGen emitReturnLocal: mgenc.
138             mgenc markAsFinished ].
139
140         self expect: #endTerm.
141     )
142
143     blockContents: mgenc = (
144         (self accept: #or) ifTrue: [
145             self locals: mgenc.
146             self expect: #or ].
147         self blockBody: mgenc sawPeriod: false
148     )
149
150     locals: mgenc = (
151         [sym == #identifier] whileTrue: [
152             mgenc addLocalIfAbsent: self variable ]
153     )
154
155     blockBody: mgenc sawPeriod: seenPeriod = (
156         (self accept: #exit) ifTrue: [
157             ^ self result: mgenc ].
158
159         sym == #endBlock ifTrue: [
160             seenPeriod ifTrue: [
161                 "a POP has been generated which must be elided (blocks always
162                 return the value of the last expression, regardless of
163                 whether it was terminated with a . or not)"
164                 mgenc removeLastBytecode ].
165
166             (mgenc isBlockMethod and: [ mgenc hasBytecodes not ]) ifTrue: [
167                 | nilSym |
168                 "if the block is empty, we need to return nil"
169                 nilSym := universe symbolFor: 'nil'.
170                 bcGen emit: mgenc pushGlobal: nilSym. ].
171
172             bcGen emitReturnLocal: mgenc.
173             mgenc markAsFinished.
174             ^ self ].
175
176         sym == #endTerm ifTrue: [
177             "it does not matter whether a period has been seen, as the end of

```



```

178         the method has been found (EndTerm) - so it is safe to emit a
179         'return self'"
180         bcGen emit: mgenc pushArgument: 1 in: 0.
181         bcGen emitReturnLocal: mgenc.
182         mgenc markAsFinished.
183         ^ self ].
184
185     self expression: mgenc.
186     (self accept: #period) ifTrue: [
187         bcGen emitPop: mgenc.
188         self blockBody: mgenc sawPeriod: true ]
189     )
190
191     unarySelector = (
192         ^ universe symbolFor: self identifier
193     )
194
195     binarySelector = (
196         | s |
197         s := text.
198
199         (self accept: #operatorSequence) or: [
200             (self acceptOneOf: Parser singleOpSyms) or: [
201                 self expect: #none ] ].
202
203         ^ universe symbolFor: s
204     )
205
206     variable = (
207         ^ self identifier
208     )
209
210     argument = (
211         ^ self variable
212     )
213
214     identifier = (
215         | s |
216         s := text.
217         (self accept: #primitive)
218             ifFalse: [self expect: #identifier].
219         ^ s
220     )
221
222     keyword = (
223         | s |
224         s := text.
225         self expect: #keyword.
226         ^ s
227     )
228
229     string = (
230         | s |
231         s := text.
232         self expect: #string.
233         ^ s
234     )
235
236     selector = (
237         (sym == #operatorSequence or: [self symIn: Parser singleOpSyms])
238             ifTrue: [^ self binarySelector].

```

```

239     (sym == #keyword or: [sym == #keywordSequence])
240     ifTrue: [^ self keywordSelector].
241
242     ^ self unarySelector
243 )
244
245 keywordSelector = (
246     | s |
247     s := text.
248     self expectOneOf: Parser keywordSelectorSyms.
249     ^ universe symbolFor: s
250 )
251
252 result: mgenc = (
253     self expression: mgenc.
254
255     mgenc isBlockMethod
256     ifTrue: [bcGen emitReturnNonLocal: mgenc ]
257     ifFalse: [bcGen emitReturnLocal: mgenc ].
258     mgenc markAsFinished.
259
260     self accept: #period
261 )
262
263 expression: mgenc = (
264     self peekForNextSymbolFromLexer.
265
266     nextSym == #assign
267     ifTrue: [self assignation: mgenc]
268     ifFalse: [self evaluation: mgenc]
269 )
270
271 assignation: mgenc = (
272     | variables |
273     variables := Vector new.
274
275     self assignments: mgenc to: variables.
276     self evaluation: mgenc.
277
278     variables do: [:v | bcGen emitDup: mgenc ].
279     variables do: [:v | self gen: mgenc popVariable: v ]
280 )
281
282 assignments: mgenc to: variables = (
283     sym == #identifier ifTrue: [
284         variables append: (self assignment: mgenc).
285         self peekForNextSymbolFromLexer.
286         nextSym == #assign ifTrue: [
287             self assignments: mgenc to: variables ] ]
288 )
289
290 assignment: mgenc = (
291     | v |
292     v := self variable.
293     self expect: #assign.
294     ^ v
295 )
296
297 evaluation: mgenc = (
298     | superSend |
299     superSend := self primary: mgenc.

```

```

300     self symIsMethod ifTrue: [
301         self messages: mgenc with: superSend ]
302     )
303
304     primary: mgenc = (
305         | superSend |
306         superSend := false.
307
308         sym == #identifier ifTrue: [
309             | v |
310             v := self variable.
311             v = 'super' ifTrue: [
312                 superSend := true.
313                 " sends to super, but pushes self as receiver"
314                 v := 'self' ].
315
316             self gen: mgenc pushVariable: v.
317             ^ superSend ].
318
319         sym == #newTerm ifTrue: [
320             self nestedTerm: mgenc.
321             ^ superSend ].
322
323         sym == #newBlock ifTrue: [
324             | bgenc blockMethod |
325             bgenc := MethodGenerationContext new: mgenc holder with: mgenc.
326             bgenc markAsBlockMethod.
327
328             self nestedBlock: bgenc.
329
330             blockMethod := bgenc assembleMethod: universe.
331             bcGen emit: mgenc pushBlock: blockMethod.
332             ^ superSend ].
333
334         self literal: mgenc.
335         ^ superSend
336     )
337
338     messages: mgenc with: superSend = (
339         sym == #identifier ifTrue: [
340             "only the first message in a sequence can be a super send"
341             self unaryMessage: mgenc with: superSend.
342
343             [sym == #identifier] whileTrue: [
344                 self unaryMessage: mgenc with: false ].
345
346             [sym == #operatorSequence or: [self symIn: Parser binaryOpSyms]]
whileTrue: [
347                 self binaryMessage: mgenc with: false ].
348
349             sym == #keyword ifTrue: [
350                 self keywordMessage: mgenc with: false ].
351
352             ^ self ].
353
354         (sym == #operatorSequence or: [self symIn: Parser binaryOpSyms])
ifTrue: [
355             self binaryMessage: mgenc with: superSend.
356
357             [sym == #operatorSequence or: [self symIn: Parser binaryOpSyms]]
whileTrue: [

```

```

358         self binaryMessage: mgenc with: false ].
359
360     sym == #keyword ifTrue: [
361         self keywordMessage: mgenc with: false ].
362
363     ^ self ].
364
365     self keywordMessage: mgenc with: superSend
366 )
367
368 unaryMessage: mgenc with: superSend = (
369     | msg |
370     msg := self unarySelector.
371
372     superSend ifTrue: [ bcGen emit: mgenc superSend: msg ]
373     ifFalse: [ bcGen emit: mgenc send: msg ]
374 )
375
376 binaryMessage: mgenc with: superSend = (
377     | msg |
378     msg := self binarySelector.
379     self binaryOperand: mgenc.
380
381     superSend ifTrue: [ bcGen emit: mgenc superSend: msg ]
382     ifFalse: [ bcGen emit: mgenc send: msg ]
383 )
384
385 binaryOperand: mgenc = (
386     | superSend |
387     superSend := self primary: mgenc.
388
389     [sym == #identifier] whileTrue: [
390         self unaryMessage: mgenc with: superSend.
391         superSend := false ].
392
393     ^ superSend
394 )
395
396 keywordMessage: mgenc with: superSend = (
397     | kw msg |
398     kw := self keyword.
399     self formula: mgenc.
400
401     [sym == #keyword] whileTrue: [
402         kw := kw + self keyword.
403         self formula: mgenc ].
404
405     msg := universe symbolFor: kw.
406     superSend ifTrue: [ bcGen emit: mgenc superSend: msg ]
407     ifFalse: [ bcGen emit: mgenc send: msg ]
408 )
409
410 formula: mgenc = (
411     | superSend |
412     superSend := self binaryOperand: mgenc.
413
414     "only the first message in a sequence can be a super send"
415     [sym == #operatorSequence or: [self symIn: Parser binaryOpSyms]]
whileTrue: [
416         self binaryMessage: mgenc with: superSend.
417         superSend := false ].

```

```

418 )
419
420 nestedTerm: mgenc = (
421     self expect: #newTerm.
422     self expression: mgenc.
423     self expect: #endTerm.
424 )
425
426 nestedBlock: mgenc = (
427     | blockSig argSize |
428     mgenc addArgumentIfAbsent: '$block self'.
429
430     self expect: #newBlock.
431
432     sym == #colon ifTrue: [
433         self blockPattern: mgenc ].
434
435     "generate block signature"
436     blockSig := '$block method'.
437     argSize := mgenc numberOfArguments.
438     (argSize - 1) timesRepeat: [
439         blockSig := blockSig + ':' ].
440
441     mgenc signature: (universe symbolFor: blockSig).
442
443     self blockContents: mgenc.
444
445     "if no return has been generated, we can be sure that the last
expression
446     in the block was not terminated by ., and can generate a return"
447     mgenc isFinished ifFalse: [
448         bcGen emitReturnLocal: mgenc.
449         mgenc markAsFinished ].
450
451     self expect: #endBlock
452 )
453
454 blockPattern: mgenc = (
455     self blockArguments: mgenc.
456     self expect: #or.
457 )
458
459 blockArguments: mgenc = (
460     self expect: #colon.
461     mgenc addArgumentIfAbsent: self argument.
462
463     [sym == #colon] whileTrue: [
464         self expect: #colon.
465         mgenc addArgumentIfAbsent: self argument ]
466 )
467
468 literal: mgenc = (
469     sym == #pound ifTrue: [
470         self peekForNextSymbolFromLexerIfNecessary.
471         nextSym == #newTerm
472             ifTrue: [ self literalArray: mgenc ]
473             ifFalse: [ self literalSymbol: mgenc ].
474         ^ self ].
475
476     sym == #string ifTrue: [
477         self literalString: mgenc.

```

```

478     ^ self ].
479
480     self literalNumber: mgenc
481 )
482
483 literalArray: mgenc = (
484     | arrayClassName arraySizePlaceholder
485     newMessage atPutMessage arraySizeLiteralIndex i |
486     self expect: #pound.
487     self expect: #newTerm.
488
489     arrayClassName := universe symbolFor: 'Array'.
490     arraySizePlaceholder := universe symbolFor:
'ArraySizeLiteralPlaceholder'.
491     newMessage := universe symbolFor: 'new:'.
492     atPutMessage := universe symbolFor: 'at:put:'.
493
494     "need the array size at a know idx so that we don't need a second pass
495     over the array elements"
496     arraySizeLiteralIndex := mgenc addLiteral: arraySizePlaceholder.
497
498     "create empty array"
499     bcGen emit: mgenc pushGlobal: arrayClassName.
500     bcGen emit: mgenc pushConstantIdx: arraySizeLiteralIndex.
501     bcGen emit: mgenc send: newMessage.
502
503     i := 1.
504
505     [sym == #endTerm] whileFalse: [
506         | pushIndex |
507         pushIndex := universe newInteger: i.
508         bcGen emit: mgenc pushConstant: pushIndex.
509         self literal: mgenc.
510         bcGen emit: mgenc send: atPutMessage.
511         i := i + 1 ].
512
513     "replace the placeholder with the actual array size"
514     mgenc updateLiteral: arraySizePlaceholder at: arraySizeLiteralIndex
put: (universe newInteger: i - 1).
515     self expect: #endTerm.
516 )
517
518 literalSymbol: mgenc = (
519     | symb |
520     self expect: #pound.
521     sym == #string
522     ifTrue: [
523         | s |
524         s := self string.
525         symb := universe symbolFor: s ]
526     ifFalse: [
527         symb := self selector ].
528     bcGen emit: mgenc pushConstant: symb
529 )
530
531 literalString: mgenc = (
532     | s str |
533     s := self string.
534     str := universe newString: s.
535     bcGen emit: mgenc pushConstant: str
536 )

```

```

537
538 literalNumber: mgenc = (
539     | lit |
540     sym == #minus
541     ifTrue: [lit := self negativeDecimal]
542     ifFalse: [lit := self literalDecimal: false].
543
544     bcGen emit: mgenc pushConstant: lit
545 )
546
547 negativeDecimal = (
548     self expect: #minus.
549     ^ self literalDecimal: true
550 )
551
552 literalDecimal: isNegative = (
553     sym == #integer
554     ifTrue: [^ self literalInteger: isNegative]
555     ifFalse: [^ self literalDouble: isNegative]
556 )
557
558 literalInteger: isNegative = (
559     | i |
560     i := Integer fromString: text.
561     isNegative ifTrue: [
562         i := i negated].
563
564     self expect: #integer.
565     ^ universe newInteger: i
566 )
567
568 literalDouble: isNegative = (
569     | d |
570     d := Double fromString: text.
571     isNegative ifTrue: [
572         d := d negated ].
573
574     self expect: #double.
575     ^ universe newDouble: d
576 )
577
578 accept: s = (
579     sym == s ifTrue: [
580         self takeSymbolFromLexer.
581         ^ true ].
582     ^ false
583 )
584
585 acceptOneOf: ss = (
586     (self symIn: ss) ifTrue: [
587         self takeSymbolFromLexer.
588         ^ true ].
589     ^ false
590 )
591
592 expect: s = (
593     (self accept: s) ifTrue: [ ^ true ].
594
595     self error: 'Parsing of ' + filename + ' failed, expected ' + s + '
but found ' + sym +
596     ' (' + text + ').\nCurrent parser context: ' + lexer

```

```

currentTextContext
597   )
598
599   expectOneOf: ss = (
600     | err |
601     (self acceptOneOf: ss) ifTrue: [ ^ true ].
602
603     err := 'Parsing of ' + filename + ' failed, expected one of '.
604
605     ss do: [
606       err := err + s + ', ' ].
607     err := err + 'but found: ' + sym + ' (' + text + ').\nCurrent parser
context: ' + lexer currentTextContext.
608
609     self error: err
610   )
611
612   symIn: ss = (
613     ^ ss contains: sym
614   )
615
616   symIsMethod = (
617     sym == #identifier      ifTrue: [^ true].
618     sym == #keyword        ifTrue: [^ true].
619     sym == #operatorSequence ifTrue: [^ true].
620     (self symIn: Parser binaryOpSyms) ifTrue: [^ true].
621     ^ false
622   )
623
624   peekForNextSymbolFromLexer = (
625     nextSym := lexer peek
626   )
627
628   peekForNextSymbolFromLexerIfNecessary = (
629     lexer isPeekDone ifFalse: [
630       self peekForNextSymbolFromLexer ]
631   )
632
633   gen: mgenc popVariable: var = (
634     | searchResult |
635     "Needs to determine whether the variable that is to be popped off the
stack
636     is a local variable, argument, or object field.
637     This is done by examining all available lexical contexts, starting
with
638     the innermost (i.e., the one represented by mgenc).".
639
640     "index, context, isArgument"
641     searchResult := Array with: 0 with: 0 with: false. "TODO support: #(0
0 false)"
642
643     (mgenc findVar: var with: searchResult)
644     ifTrue: [
645       (searchResult at: 3) "isArgument"
646       ifTrue: [bcGen emit: mgenc popArgument: (searchResult at: 1)
in: (searchResult at: 2)]
647       ifFalse: [bcGen emit: mgenc popLocal: (searchResult at: 1) in:
(searchResult at: 2)]
648     ]
649     ifFalse: [
650       | varSym |

```



```

651         varSym := universe symbolFor: var.
652         (mgenc hasField: varSym) ifFalse: [
653             ^ self error: 'Write to variable with the name ' + var + ', but
there is no variable or field defined with this name' ].
654         bcGen emit: mgenc popField: varSym ].
655     )
656
657     gen: mgenc pushVariable: var = (
658         "Needs to determine whether the variable to be pushed on the stack
659         is a local variable, argument, or object field.
660         This is done by examining all available lexical contexts, starting
with
661         the innermost (i.e., the one represented by mgenc).".
662         "index, context, isArgument"
663         | searchResult |
664         searchResult := Array with: 0 with: 0 with: false. "TODO support: #(0
0 false)"
665
666         (mgenc findVar: var with: searchResult)
667         ifTrue: [
668             (searchResult at: 3) "isArgument"
669             ifTrue: [
670                 bcGen emit: mgenc pushArgument: (searchResult at: 1) in:
(searchResult at: 2) ]
671             ifFalse: [
672                 bcGen emit: mgenc pushLocal: (searchResult at: 1) in:
(searchResult at: 2) ] ]
673         ifFalse: [
674             | varSym |
675             varSym := universe symbolFor: var.
676             (mgenc hasField: varSym)
677             ifTrue: [
678                 bcGen emit: mgenc pushField: varSym ]
679             ifFalse: [
680                 bcGen emit: mgenc pushGlobal: varSym ] ]
681     )
682
683
684
685     ----
686     | singleOpSyms binaryOpSyms keywordSelectorSyms |
687
688     singleOpSyms = (
689         singleOpSyms == nil ifTrue: [
690             singleOpSyms := #( #not #and #or #star #div #mod #plus #equal
691                 #more #less #comma #at #per #minus #none) ].
692         ^ singleOpSyms
693     )
694
695     binaryOpSyms = (
696         binaryOpSyms == nil ifTrue: [
697             binaryOpSyms := #( #or #comma #minus #equal #not #and #or #star
698                 #div #mod #plus #equal #more #less #comma #at
699                 #per #none) ].
700         ^ binaryOpSyms
701     )
702
703     keywordSelectorSyms = (
704         keywordSelectorSyms == nil ifTrue: [
705             keywordSelectorSyms := #( #keyword #keywordSequence) ].
706         ^ keywordSelectorSyms

```

```
707 )
708
709 newWith: aString for: aFilename in: universe = (
710   ^ self new initializeWith: aString for: aFilename in: universe
711 )
712
713 load: aFileName in: universe = (
714   | fileContent |
715   fileContent := system loadFile: aFileName.
716   fileContent == nil ifTrue: [ ^ nil ].
717
718   ^ self new initializeWith: fileContent for: aFileName in: universe
719 )
720 )
721
```

```

1 SourcecodeCompiler = (
2   ----
3   compileClass: path name: fileName into: systemClass in: universe = (
4     | fname parser result cname |
5     fname := path + '/' + fileName + '.som'.
6
7     parser := Parser load: fname in: universe.
8     parser ifNil: [ ^ nil ].
9
10    result := self compile: parser into: systemClass.
11
12    cname := result name string.
13
14    fileName ~= cname ifTrue: [
15      self error: 'File name ' + fname
16        + ' does not match class name (' + cname + ') in it.' ].
17    ^ result
18  )
19
20  compileClass: stmt into: systemClass in: universe = (
21    | parser |
22    parser := Parser newWith: stmt for: '$string$' in: universe.
23    ^ self compile: parser into: systemClass.
24  )
25
26  compile: parser into: systemClass = (
27    | cgc |
28    cgc := parser classdef.
29
30    systemClass == nil
31      ifTrue: [ ^ cgc assemble ]
32      ifFalse: [ ^ cgc assembleSystemClass: systemClass ]
33  )
34 )
35

```

```

1 Bytecodes = (
2   ----
3
4   length: bytecode = (
5     bytecode == #halt           ifTrue: [ ^ 1 ].
6     bytecode == #dup            ifTrue: [ ^ 1 ].
7     bytecode == #pushLocal      ifTrue: [ ^ 3 ].
8     bytecode == #pushArgument  ifTrue: [ ^ 3 ].
9     bytecode == #pushField      ifTrue: [ ^ 2 ].
10    bytecode == #pushBlock      ifTrue: [ ^ 2 ].
11    bytecode == #pushConstant   ifTrue: [ ^ 2 ].
12    bytecode == #pushGlobal     ifTrue: [ ^ 2 ].
13    bytecode == #pop            ifTrue: [ ^ 1 ].
14    bytecode == #popLocal       ifTrue: [ ^ 3 ].
15    bytecode == #popArgument    ifTrue: [ ^ 3 ].
16    bytecode == #popField       ifTrue: [ ^ 2 ].
17    bytecode == #send           ifTrue: [ ^ 2 ].
18    bytecode == #superSend      ifTrue: [ ^ 2 ].
19    bytecode == #returnLocal    ifTrue: [ ^ 1 ].
20    bytecode == #returnNonLocal ifTrue: [ ^ 1 ].
21
22    self error: 'Unknown bytecode' + bytecode asString
23  )
24
25  paddedBytecodeName: bytecodeSymbol = (
26    | max padded |
27    max := #returnNonLocal length.
28    padded := bytecodeSymbol asString.
29    [padded length < max] whileTrue: [
30      padded := padded + ' '.
31    ^ padded
32  )
33 )
34

```

```

1 Frame = (
2 "
3 Frame layout:
4 +-----+
5 | Arguments      | 1
6 +-----+
7 | Local Variables | <-- localOffset
8 +-----+
9 | Stack          | <-- stackPointer
10 | ...            |
11 +-----+
12 "
13 |
14   "Points at the top element"
15   stackPointer
16   bytecodeIndex
17
18   "the offset at which local variables start"
19   localOffset
20
21   method
22   context
23   previousFrame
24   stack
25 |
26   initialize: nilObject previous: prevFrame context: contextFrame method:
asMethod maxStack: stackElements = (
27     previousFrame := prevFrame.
28     context := contextFrame.
29     method := asMethod.
30     stack := Array new: stackElements withAll: nilObject.
31
32     "Reset the stack pointer and the bytecode index"
33     self resetStackPointer.
34     bytecodeIndex := 1.
35   )
36
37   previousFrame = (
38     ^ previousFrame
39   )
40
41   clearPreviousFrame = (
42     previousFrame := nil
43   )
44
45   hasPreviousFrame = (
46     ^ previousFrame ~= nil
47   )
48
49   isBootstrapFrame = (
50     ^ self hasPreviousFrame not
51   )
52
53   context = (
54     ^ context
55   )
56
57   hasContext = (

```

```

58     ^ context ~= nil
59 )
60
61 context: level = (
62     | frame |
63     "Get the context frame at the given level"
64     frame := self.
65
66     "Iterate through the context chain until the given level is reached"
67     [level > 0] whileTrue: [
68         "Get the context of the current frame"
69         frame := frame context.
70
71         "Go to the next level"
72         level := level - 1 ].
73
74     ^ frame
75 )
76
77 outerContext = (
78     | frame |
79     "Compute the outer context of this frame"
80     frame := self.
81
82     "Iterate through the context chain until null is reached"
83     [frame hasContext] whileTrue: [
84         frame := frame context ].
85
86     ^ frame
87 )
88
89 method = (
90     ^ method
91 )
92
93 pop = (
94     | sp |
95     "Pop an object from the expression stack and return it"
96     sp := stackPointer.
97     stackPointer := stackPointer - 1.
98     ^ stack at: sp.
99 )
100
101 push: aSAbstractObject = (
102     "Push an object onto the expression stack"
103     | sp |
104     sp := stackPointer + 1.
105     stack at: sp put: aSAbstractObject.
106     stackPointer := sp
107 )
108
109 resetStackPointer = (
110     "arguments are stored in front of local variables"
111     localOffset := method numberOfArguments + 1.
112
113     "Set the stack pointer to its initial value thereby clearing the
stack"
114     stackPointer := localOffset + method numberOfLocals - 1
115 )
116
117 bytecodeIndex = (

```

```

118     "Get the current bytecode index for this frame"
119     ^ bytecodeIndex
120 )
121
122 bytecodeIndex: value = (
123     "Set the current bytecode index for this frame"
124     bytecodeIndex := value
125 )
126
127 stackElement: index = (
128     "Get the stack element with the given index
129     (an index of zero yields the top element)"
130     ^ stack at: stackPointer - index
131 )
132
133 stackElement: index put: value = (
134     "Set the stack element with the given index to the given value
135     (an index of zero yields the top element)"
136     stack at: stackPointer - index put: value
137 )
138
139 local: index = (
140     ^ stack at: localOffset + index - 1
141 )
142
143 local: index put: value = (
144     stack at: localOffset + index - 1 put: value
145 )
146
147 local: index at: contextLevel = (
148     "Get the local with the given index in the given context"
149     ^ (self context: contextLevel) local: index
150 )
151
152 local: index at: contextLevel put: value = (
153     "Set the local with the given index in the given context to the given
154     value"
155     (self context: contextLevel) local: index put: value
156 )
157
158 argument: index = (
159     ^ stack at: index
160 )
161
162 argument: index put: value = (
163     ^ stack at: index put: value
164 )
165
166 argument: index at: contextLevel = (
167     | context |
168     "Get the context"
169     context := self context: contextLevel.
170
171     "Get the argument with the given index"
172     ^ context argument: index
173 )
174
175 argument: index at: contextLevel put: value = (
176     | context |
177     "Get the context"
178     context := self context: contextLevel.

```

```

178
179     "Set the argument with the given index to the given value"
180     context argument: index put: value
181 )
182
183 copyArgumentsFrom: frame = (
184     | numArgs |
185     "copy arguments from frame:
186     - arguments are at the top of the stack of frame.
187     - copy them into the argument area of the current frame"
188     numArgs := method numberOfArguments.
189     0 to: numArgs - 1 do: [:i |
190         stack at: i + 1 put: (frame stackElement: numArgs - 1 - i) ]
191 )
192
193 printStackTrace = (
194     | className methodName |
195     "Print a stack trace starting in this frame"
196     self hasPreviousFrame ifTrue: [
197         previousFrame printStackTrace ].
198
199     className := method holder name string.
200     methodName := method signature string.
201     Universe println: className + '>>#' + methodName + ' @bi: ' +
bytecodeIndex
202 )
203
204 ----
205
206 new: nilObject previous: prevFrame context: contextFrame method:
aSMethod maxStack: stackElements = (
207     ^ self new initialize: nilObject previous: prevFrame context:
contextFrame method: aSMethod maxStack: stackElements
208 )
209 )
210

```



```

1 Interpreter = (
2   | universe frame |
3
4   initializeWith: aUniverse = (
5     universe := aUniverse
6   )
7
8   doDup = (
9     frame push: (frame stackElement: 0)
10  )
11
12  doPushLocal: bytecodeIndex = (
13    frame push: (
14      frame local: (frame method bytecode: bytecodeIndex + 1)
15      at: (frame method bytecode: bytecodeIndex + 2))
16  )
17
18  doPushArgument: bytecodeIndex = (
19    frame push: (
20      frame argument: (frame method bytecode: bytecodeIndex + 1)
21      at: (frame method bytecode: bytecodeIndex + 2))
22  )
23
24  doPushField: bytecodeIndex = (
25    | fieldIndex |
26    fieldIndex := frame method bytecode: bytecodeIndex + 1.
27
28    "Push the field with the computed index onto the stack"
29    frame push: (self getSelf field: fieldIndex)
30  )
31
32  doPushBlock: bytecodeIndex = (
33    | blockMethod |
34    blockMethod := frame method constant: bytecodeIndex.
35
36    "Push a new block with the current frame as context onto the stack"
37    frame push: (
38      universe newBlock: blockMethod
39      with: frame
40      numArgs: blockMethod numberOfArguments)
41  )
42
43  doPushConstant: bytecodeIndex = (
44    frame push: (frame method constant: bytecodeIndex)
45  )
46
47  doPushGlobal: bytecodeIndex = (
48    | globalName global |
49    globalName := frame method constant: bytecodeIndex.
50
51    "Get the global from the universe"
52    global := universe global: globalName.
53
54    global ~= nil
55    ifTrue: [ frame push: global ]
56    ifFalse: [
57      "Send 'unknownGlobal:' to self"
58      self getSelf sendUnknownGlobal: globalName in: universe using:

```

```

self ]
59  )
60
61  doPop = (
62      frame pop
63  )
64
65  doPopLocal: bytecodeIndex = (
66      frame local: (frame method bytecode: bytecodeIndex + 1)
67      at: (frame method bytecode: bytecodeIndex + 2)
68      put: frame pop
69  )
70
71  doPopArgument: bytecodeIndex = (
72      frame argument: (frame method bytecode: bytecodeIndex + 1)
73      at: (frame method bytecode: bytecodeIndex + 2)
74      put: frame pop
75  )
76
77  doPopField: bytecodeIndex = (
78      | fieldIndex |
79      fieldIndex := frame method bytecode: bytecodeIndex + 1.
80
81      "Set the field with the computed index to the value popped from the
stack"
82      self getSelf field: fieldIndex put: frame pop
83  )
84
85  doSend: bytecodeIndex = (
86      | signature numberOfArguments receiver |
87      signature := frame method constant: bytecodeIndex.
88      numberOfArguments := signature numberOfSignatureArguments.
89      receiver := frame stackElement: numberOfArguments - 1.
90      self send: signature rcvrClass: (receiver somClassIn: universe)
91  )
92
93  doSuperSend: bytecodeIndex = (
94      | signature holderSuper invokable |
95      signature := frame method constant: bytecodeIndex.
96
97      "Send the message
98      Lookup the invokable with the given signature"
99      holderSuper := frame method holder superClass.
100     invokable := holderSuper lookupInvokable: signature.
101
102     self activate: invokable orDnu: signature
103  )
104
105  doReturnLocal = (
106      | result |
107      result := frame pop.
108
109      "Pop the top frame and push the result"
110      self popFrameAndPushResult: result
111  )
112
113  doReturnNonLocal = (
114      | result context |
115      result := frame pop.
116
117      "Compute the context for the non-local return"

```

```

118     context := frame outerContext.
119
120     "Make sure the block context is still on the stack"
121     context hasPreviousFrame ifFalse: [
122         | block sender method numArgs |
123         "Try to recover by sending 'escapedBlock:' to the sending object
124         this can get a bit nasty when using nested blocks. In this case
125         the 'sender' will be the surrounding block and not the object
126         that actually sent the 'value' message."
127         block := frame argument: 1 at: 0.
128         sender := frame previousFrame outerContext argument: 1 at: 0.
129
130         "pop the frame of the currently executing block..."
131         self popFrame.
132
133         "pop old arguments from stack"
134         method := frame method.
135         numArgs := method numberOfArguments.
136         numArgs timesRepeat: [ frame pop ].
137
138         "... and execute the escapedBlock message instead"
139         sender sendEscapedBlock: block in: universe using: self.
140         ^ self ].
141
142     "Unwind the frames"
143     [frame ~= context] whileTrue: [
144         self popFrame ].
145
146     self popFrameAndPushResult: result
147 )
148
149 start = (
150     [true] whileTrue: [
151         | bytecodeIndex bytecode bytecodeLength nextBytecodeIndex result |
152         bytecodeIndex := frame bytecodeIndex.
153         bytecode := frame method bytecode: bytecodeIndex.
154         bytecodeLength := Bytecodes length: bytecode.
155         nextBytecodeIndex := bytecodeIndex + bytecodeLength.
156         frame bytecodeIndex: nextBytecodeIndex.
157
158         result := self dispatch: bytecode idx: bytecodeIndex.
159         result ~= nil
160         ifTrue: [ ^ result ] ]
161 )
162
163 dispatch: bytecode idx: bytecodeIndex = (
164     bytecode == #halt ifTrue: [
165         ^ frame stackElement: 0 ].
166
167     bytecode == #dup ifTrue: [
168         self doDup.
169         ^ nil ].
170
171     bytecode == #pushLocal ifTrue: [
172         self doPushLocal: bytecodeIndex.
173         ^ nil ].
174
175     bytecode == #pushArgument ifTrue: [
176         self doPushArgument: bytecodeIndex.
177         ^ nil ].
178

```

```

179     bytecode == #pushField ifTrue: [
180         self doPushField: bytecodeIndex.
181         ^ nil ].
182
183     bytecode == #pushBlock ifTrue: [
184         self doPushBlock: bytecodeIndex.
185         ^ nil ].
186
187     bytecode == #pushConstant ifTrue: [
188         self doPushConstant: bytecodeIndex.
189         ^ nil ].
190
191     bytecode == #pushGlobal ifTrue: [
192         self doPushGlobal: bytecodeIndex.
193         ^ nil ].
194
195     bytecode == #pop ifTrue: [
196         self doPop.
197         ^ nil ].
198
199     bytecode == #popLocal ifTrue: [
200         self doPopLocal: bytecodeIndex.
201         ^ nil ].
202
203     bytecode == #popArgument ifTrue: [
204         self doPopArgument: bytecodeIndex.
205         ^ nil ].
206
207     bytecode == #popField ifTrue: [
208         self doPopField: bytecodeIndex.
209         ^ nil ].
210
211     bytecode == #send ifTrue: [
212         self doSend: bytecodeIndex.
213         ^ nil ].
214
215     bytecode == #superSend ifTrue: [
216         self doSuperSend: bytecodeIndex.
217         ^ nil ].
218
219     bytecode == #returnLocal ifTrue: [
220         self doReturnLocal.
221         ^ nil ].
222
223     bytecode == #returnNonLocal ifTrue: [
224         self doReturnNonLocal.
225         ^ nil ].
226
227     self error: 'Unknown bytecode' + bytecode asString
228 )
229
230 pushNewFrame: method with: contextFrame = (
231     frame := universe newFrame: frame with: method with: contextFrame.
232     ^ frame
233 )
234
235 pushNewFrame: method = (
236     ^ self pushNewFrame: method with: nil
237 )
238
239 frame = (

```

```

240     ^ frame
241   )
242
243   method = (
244     ^ frame method
245   )
246
247   getSelf = (
248     "Get the self object from the interpreter"
249     ^ frame outerContext argument: 1 at: 0
250   )
251
252   send: selector rcvrClass: receiverClass = (
253     | invokable |
254     invokable := receiverClass lookupInvokable: selector.
255     self activate: invokable orDnu: selector
256   )
257
258   activate: invokable orDnu: signature = (
259     invokable ~= nil
260     ifTrue: [
261       "Invoke the invokable in the current frame"
262       invokable invoke: frame using: self ]
263     ifFalse: [
264       | numberOfArguments receiver |
265       numberOfArguments := signature numberOfSignatureArguments.
266       receiver := frame stackElement: numberOfArguments - 1.
267       receiver sendDoesNotUnderstand: signature in: universe using:
self ]
268   )
269
270   popFrame = (
271     | result |
272     "Save a reference to the top frame"
273     result := frame.
274
275     "Pop the top frame from the frame stack"
276     frame := frame previousFrame.
277
278     "Destroy the previous pointer on the old top frame"
279     result clearPreviousFrame.
280
281     "Return the popped frame"
282     ^ result
283   )
284
285   popFrameAndPushResult: result = (
286     | numberOfArguments |
287     "Pop the top frame from the interpreter frame stack and
288     get the number of arguments"
289     numberOfArguments := self popFrame method numberOfArguments.
290
291     "Pop the arguments"
292     numberOfArguments
293     timesRepeat: [ frame pop ].
294
295     frame push: result
296   )
297
298   ----
299

```

```
300     new: universe = (  
301         ^ self new initializeWith: universe  
302     )  
303 )  
304
```

```

1 ArrayPrimitives = Primitives (
2
3   installPrimitives = (
4     self installInstancePrimitive: (
5       SPrimitive new: 'at:' in: universe with: [:frame :interp |
6         | idx rcvr |
7         idx := frame pop.
8         rcvr := frame pop.
9         frame push: (rcvr indexableField: idx integer)]).
10
11     self installInstancePrimitive: (
12       SPrimitive new: 'at:put:' in: universe with: [:frame :interp |
13         | rcvr idx value |
14         value := frame pop.
15         idx := frame pop.
16         rcvr := frame stackElement: 0.
17         rcvr indexableField: idx integer put: value ])).
18
19     self installInstancePrimitive: (
20       SPrimitive new: 'length' in: universe with: [:frame :interp |
21         | rcvr |
22         rcvr := frame pop.
23
24         frame push: (universe newInteger: rcvr numberOfIndexableFields) ])).
25
26     self installClassPrimitive: (
27       SPrimitive new: 'new:' in: universe with: [:frame :interp |
28         | arg |
29         arg := frame pop.
30         frame pop.
31
32         frame push: (universe newArray: arg integer) ])).
33   )
34
35   ----
36
37   new: universe = (
38     ^ self new initialize: universe
39   )
40 )
41

```

SomSom/src/primitives/BlockPrimitives.som

```
1 BlockPrimitives = Primitives (  
2  
3   installPrimitives = (  
4     self installInstancePrimitive: (  
5       SPrimitive new: 'restart' in: universe with: [:frame :interp |  
6         frame bytecodeIndex: 1.  
7         frame resetStackPointer. ] ).  
8   )  
9  
10  ----  
11  
12  new: universe = (  
13    ^ self new initialize: universe  
14  )  
15 )  
16
```



```

1 ClassPrimitives = Primitives (
2
3   installPrimitives = (
4     self installInstancePrimitive: (
5       SPrimitive new: 'new' in: universe with: [:frame :interp |
6         | rcvr |
7         rcvr := frame pop.
8         frame push: (universe newInstance: rcvr) ])).
9
10    self installInstancePrimitive: (
11      SPrimitive new: 'name' in: universe with: [:frame :interp |
12        | rcvr |
13        rcvr := frame pop.
14        frame push: rcvr name ])).
15
16    self installInstancePrimitive: (
17      SPrimitive new: 'superclass' in: universe with: [:frame :interp |
18        | rcvr |
19        rcvr := frame pop.
20        frame push: rcvr superClass ])).
21
22    self installInstancePrimitive: (
23      SPrimitive new: 'fields' in: universe with: [:frame :interp |
24        | rcvr |
25        rcvr := frame pop.
26        frame push: rcvr instanceFields ])).
27
28    self installInstancePrimitive: (
29      SPrimitive new: 'methods' in: universe with: [:frame :interp |
30        | rcvr |
31        rcvr := frame pop.
32        frame push: rcvr instanceInvokables ])).
33  )
34
35  ----
36
37  new: universe = (
38    ^ self new initialize: universe
39  )
40 )
41

```

```

1 DoublePrimitives = Primitives (
2
3   coerceToDouble: anSAbstractObject = (
4     anSAbstractObject class == SDouble ifTrue: [
5       ^ anSAbstractObject double ].
6     anSAbstractObject class == SInteger ifTrue: [
7       ^ anSAbstractObject integer asDouble ].
8     self error: 'Cannot coerce ' + anSAbstractObject debugString + ' to
double'.
9   )
10
11  installPrimitives = (
12    self installInstancePrimitive: (
13      SPrimitive new: 'asString' in: universe with: [:frame :interp |
14        | rcvr |
15        rcvr := frame pop.
16        frame push: (universe newString: rcvr double asString) ]).
17
18    self installInstancePrimitive: (
19      SPrimitive new: 'asInteger' in: universe with: [:frame :interp |
20        | rcvr |
21        rcvr := frame pop.
22        frame push: (universe newInteger: rcvr double asInteger) ]).
23
24    self installInstancePrimitive: (
25      SPrimitive new: 'sqrt' in: universe with: [:frame :interp |
26        | rcvr |
27        rcvr := frame pop.
28        frame push: (universe newDouble: rcvr double sqrt) ]).
29
30    self installInstancePrimitive: (
31      SPrimitive new: '+' in: universe with: [:frame :interp |
32        | rcvr arg |
33        arg := self coerceToDouble: frame pop.
34        rcvr := frame pop.
35        frame push: (universe newDouble: rcvr double + arg) ]).
36
37    self installInstancePrimitive: (
38      SPrimitive new: '-' in: universe with: [:frame :interp |
39        | rcvr arg |
40        arg := self coerceToDouble: frame pop.
41        rcvr := frame pop.
42        frame push: (universe newDouble: rcvr double - arg) ]).
43
44    self installInstancePrimitive: (
45      SPrimitive new: '*' in: universe with: [:frame :interp |
46        | rcvr arg |
47        arg := self coerceToDouble: frame pop.
48        rcvr := frame pop.
49        frame push: (universe newDouble: rcvr double * arg) ]).
50
51    self installInstancePrimitive: (
52      SPrimitive new: '/' in: universe with: [:frame :interp |
53        | rcvr arg |
54        arg := self coerceToDouble: frame pop.
55        rcvr := frame pop.
56        frame push: (universe newDouble: rcvr double // arg) ]).
57

```

```

58     self installInstancePrimitive: (
59         SPrimitive new: '%' in: universe with: [:frame :interp |
60             | rcvr arg |
61             arg := self coerceToDouble: frame pop.
62             rcvr := frame pop.
63             frame push: (universe newDouble: rcvr double % arg) ])).
64
65     self installInstancePrimitive: (
66         SPrimitive new: '=' in: universe with: [:frame :interp |
67             | argument rcvr left |
68             argument := frame pop.
69             rcvr := frame pop.
70             left := rcvr double.
71
72             frame push: (self somBool: (
73                 (argument class == SDouble)
74                 ifTrue: [left = argument double]
75                 ifFalse: [
76                     argument class == SInteger
77                     ifTrue: [left = argument integer]
78                     ifFalse: [ false ] ])) ])).
79
80     self installInstancePrimitive: (
81         SPrimitive new: '<' in: universe with: [:frame :interp |
82             | rcvr arg |
83             arg := self coerceToDouble: frame pop.
84             rcvr := frame pop.
85             frame push: (self somBool: rcvr double < arg) ])).
86
87     self installInstancePrimitive: (
88         SPrimitive new: 'round' in: universe with: [:frame :interp |
89             | rcvr |
90             rcvr := frame pop.
91             frame push: (universe newInteger: rcvr double round) ])).
92
93     self installInstancePrimitive: (
94         SPrimitive new: 'sin' in: universe with: [:frame :interp |
95             | rcvr |
96             rcvr := frame pop.
97             frame push: (universe newDouble: rcvr double sin) ])).
98
99     self installInstancePrimitive: (
100         SPrimitive new: 'cos' in: universe with: [:frame :interp |
101             | rcvr |
102             rcvr := frame pop.
103             frame push: (universe newDouble: rcvr double cos) ])).
104
105     self installClassPrimitive: (
106         SPrimitive new: 'PositiveInfinity' in: universe with:
107         [:frame :interp |
108             | rcvr |
109             rcvr := frame pop.
110             frame push: (universe newDouble: Double PositiveInfinity) ])).
111
112     self installClassPrimitive: (
113         SPrimitive new: 'fromString:' in: universe with: [:frame :interp |
114             | rcvr arg |
115             arg := frame pop.
116             rcvr := frame pop.
117             frame push: (universe newDouble: (Double fromString: arg
118 string)) ])).

```

```
117     )
118
119     ----
120
121     new: universe = (
122         ^ self new initialize: universe
123     )
124 )
125
```

```

1 IntegerPrimitives = Primitives (
2
3   installPrimitives = (
4     self installInstancePrimitive: (
5       SPrimitive new: 'asString' in: universe with: [:frame :interp |
6         | rcvr |
7         rcvr := frame pop.
8         frame push: (universe newString: rcvr integer asString) ])).
9
10
11    self installInstancePrimitive: (
12      SPrimitive new: 'sqrt' in: universe with: [:frame :interp |
13        | rcvr result |
14        rcvr := frame pop.
15        result := rcvr integer sqrt.
16        result class == Integer
17          ifTrue: [frame push: (universe newInteger: result)]
18          ifFalse: [frame push: (universe newDouble: result)] ])).
19
20    self installInstancePrimitive: (
21      SPrimitive new: 'atRandom' in: universe with: [:frame :interp |
22        | rcvr |
23        rcvr := frame pop.
24        frame push: (universe newInteger: rcvr integer atRandom) ])).
25
26    self installInstancePrimitive: (
27      SPrimitive new: 'asDouble' in: universe with: [:frame :interp |
28        | rcvr |
29        rcvr := frame pop.
30        frame push: (universe newDouble: rcvr integer asDouble) ])).
31
32    self installInstancePrimitive: (
33      SPrimitive new: '+' in: universe with: [:frame :interp |
34        | argument rcvr |
35        argument := frame pop.
36        rcvr := frame pop.
37
38        frame push: (argument class == SDouble
39          ifTrue: [universe newDouble: rcvr integer + argument double]
40          ifFalse: [universe newInteger: rcvr integer + argument
integer]) ])).
41
42    self installInstancePrimitive: (
43      SPrimitive new: '-' in: universe with: [:frame :interp |
44        | argument rcvr |
45        argument := frame pop.
46        rcvr := frame pop.
47
48        frame push: (argument class == SDouble
49          ifTrue: [universe newDouble: rcvr integer - argument double]
50          ifFalse: [universe newInteger: rcvr integer - argument
integer]) ])).
51
52    self installInstancePrimitive: (
53      SPrimitive new: '*' in: universe with: [:frame :interp |
54        | argument rcvr |
55        argument := frame pop.
56        rcvr := frame pop.

```

```

57
58     frame push: (argument class == SDouble
59         ifTrue: [universe newDouble: rcvr integer * argument double]
60         ifFalse: [universe newInteger: rcvr integer * argument
integer])) ]).
61
62     self installInstancePrimitive: (
63         SPrimitive new: '/' in: universe with: [:frame :interp |
64             | argument rcvr |
65             argument := frame pop.
66             rcvr := frame pop.
67
68             frame push: (universe newDouble:
69                 (argument class == SDouble
70                     ifTrue: [rcvr integer // argument double]
71                     ifFalse: [rcvr integer // argument integer])) ]).
72
73     self installInstancePrimitive: (
74         SPrimitive new: '/' in: universe with: [:frame :interp |
75             | argument rcvr |
76             argument := frame pop.
77             rcvr := frame pop.
78
79             frame push: (universe newInteger:
80                 (argument class == SDouble
81                     ifTrue: [rcvr integer / argument double]
82                     ifFalse: [rcvr integer / argument integer])) ]).
83
84     self installInstancePrimitive: (
85         SPrimitive new: '%' in: universe with: [:frame :interp |
86             | argument rcvr |
87             argument := frame pop.
88             rcvr := frame pop.
89
90             frame push: (argument class == SDouble
91                 ifTrue: [universe newDouble: rcvr integer % argument double]
92                 ifFalse: [universe newInteger: rcvr integer % argument
integer])) ]).
93
94
95     self installInstancePrimitive: (
96         SPrimitive new: 'rem:' in: universe with: [:frame :interp |
97             | argument rcvr |
98             argument := frame pop.
99             rcvr := frame pop.
100             frame push: (universe newInteger: (rcvr integer rem: argument
integer))]).
101
102     self installInstancePrimitive: (
103         SPrimitive new: '&' in: universe with: [:frame :interp |
104             | argument rcvr |
105             argument := frame pop.
106             rcvr := frame pop.
107             frame push: (universe newInteger: (rcvr integer & argument
integer)) ]).
108
109     self installInstancePrimitive: (
110         SPrimitive new: '=' in: universe with: [:frame :interp |
111             | argument rcvr left |
112             argument := frame pop.
113             rcvr := frame pop.

```

```

114         left := rcvr integer.
115
116         frame push: (self somBool: (
117             (argument class == SDouble)
118             ifTrue: [left = argument double]
119             ifFalse: [
120                 argument class == SInteger
121                 ifTrue: [left = argument integer]
122                 ifFalse: [ false ] ])) ]).
123
124     self installInstancePrimitive: (
125         SPrimitive new: '==' in: universe with: [:frame :interp |
126             | argument rcvr left |
127             argument := frame pop.
128             rcvr := frame pop.
129             left := rcvr integer.
130
131             frame push: (self somBool: (
132                 argument class == SInteger
133                 ifTrue: [left = argument integer]
134                 ifFalse: [ false ] )) ]) dontWarn: true.
135
136     self installInstancePrimitive: (
137         SPrimitive new: '<<' in: universe with: [:frame :interp |
138             | argument rcvr |
139             argument := frame pop.
140             rcvr := frame pop.
141             frame push: (universe newInteger: (rcvr integer << argument
142 integer)) ]).
143
144     self installInstancePrimitive: (
145         SPrimitive new: '<' in: universe with: [:frame :interp |
146             | argument rcvr left |
147             argument := frame pop.
148             rcvr := frame pop.
149             left := rcvr integer.
150
151             frame push: (self somBool: (
152                 (argument class == SDouble)
153                 ifTrue: [left < argument double]
154                 ifFalse: [
155                     argument class == SInteger
156                     ifTrue: [left < argument integer]
157                     ifFalse: [ false ] ])) ]).
158
159     self installInstancePrimitive: (
160         SPrimitive new: 'bitXor:' in: universe with: [:frame :interp |
161             | argument rcvr |
162             argument := frame pop.
163             rcvr := frame pop.
164             frame push: (universe newInteger: (rcvr integer bitXor: argument
165 integer)) ]).
166
167     self installInstancePrimitive: (
168         SPrimitive new: 'as32BitSignedValue' in: universe with:
169         [:frame :interp |
170             | rcvr |
171             rcvr := frame pop.
172             frame push: (universe newInteger: (rcvr integer
173 as32BitSignedValue)) ]).
174

```

```

171     self installInstancePrimitive: (
172         SPrimitive new: 'as32BitUnsignedValue' in: universe with:
173         [:frame :interp |
174             | rcvr |
175             rcvr := frame pop.
176             frame push: (universe newInteger: (rcvr integer
177 as32BitUnsignedValue)) ]).
178
179     self installInstancePrimitive: (
180         SPrimitive new: '>>>' in: universe with: [:frame :interp |
181             | argument rcvr |
182             argument := frame pop.
183             rcvr := frame pop.
184             frame push: (universe newInteger: (rcvr integer >>> argument
185 integer)) ]).
186
187     self installClassPrimitive: (
188         SPrimitive new: 'fromString:' in: universe with: [:frame :interp |
189             | argument |
190             argument := frame pop.
191             frame pop.
192             frame push: (universe newInteger: (Integer fromString: argument
193 string)) ]).
194
195     )
196
197     ----
198
199     new: universe = (
200         ^ self new initialize: universe
201     )
202
203     )
204
205     )

```



```
1 MethodPrimitives = Primitives (
2
3   installPrimitives = (
4     self installInstancePrimitive: (
5       SPrimitive new: 'holder' in: universe with: [:frame :interp |
6         | rcvr |
7         rcvr := frame pop.
8         frame push: rcvr holder ])).
9
10    self installInstancePrimitive: (
11      SPrimitive new: 'signature' in: universe with: [:frame :interp |
12        | rcvr |
13        rcvr := frame pop.
14        frame push: rcvr signature ])).
15  )
16
17  ----
18
19  new: universe = (
20    ^ self new initialize: universe
21  )
22 )
23
```

```

1 ObjectPrimitives = Primitives (
2
3   installPrimitives = (
4     self installInstancePrimitive: (
5       SPrimitive new: '=' in: universe with: [:frame :interp |
6         | op1 op2 |
7         op1 := frame pop.
8         op2 := frame pop.
9
10        frame push: (self somBool: op1 == op2) ])).
11
12    self installInstancePrimitive: (
13      SPrimitive new: 'hashCode' in: universe with: [:frame :interp |
14        | rcvr |
15        rcvr := frame pop.
16        frame push: (universe newInteger: rcvr hashCode) ])).
17
18    self installInstancePrimitive: (
19      SPrimitive new: 'objectSize' in: universe with: [:frame :interp |
20        | rcvr size clazz |
21        rcvr := frame pop.
22
23        size := 1.
24        clazz := (rcvr somClassIn: universe).
25        clazz == SArray ifTrue: [
26          size := size + rcvr numberOfIndexableFields ].
27        clazz == SObject ifTrue: [
28          size := size + rcvr numberOfFields ].
29
30        frame push: (universe newInteger: size) ])).
31
32    self installInstancePrimitive: (
33      SPrimitive new: 'perform:' in: universe with: [:frame :interp |
34        | selector rcvr invocable |
35        selector := frame pop.
36        rcvr := frame stackElement: 0.
37
38        invocable := (rcvr somClassIn: universe) lookupInvokable: selector.
39        invocable invoke: frame using: interp ])).
40
41    self installInstancePrimitive: (
42      SPrimitive new: 'perform:inSuperclass:' in: universe with:
43      [:frame :interp |
44        | selector clazz invocable |
45        clazz := frame pop.
46        selector := frame pop.
47
48        invocable := clazz lookupInvokable: selector.
49        invocable invoke: frame using: interp ])).
50
51    self installInstancePrimitive: (
52      SPrimitive new: 'perform:withArguments:' in: universe with:
53      [:frame :interp |
54        | args selector rcvr invocable |
55        args := frame pop.
56        selector := frame pop.
57        rcvr := frame stackElement: 0.

```

```

57         1 to: args numberOfIndexableFields do: [:i |
58             frame push: (args indexableField: i) ].
59
60         invokable := (rcvr somClassIn: universe) lookupInvokable: selector.
61         invokable invoke: frame using: interp ]).
62
63     self installInstancePrimitive: (
64         SPrimitive new: 'instVarAt:' in: universe with: [:frame :interp |
65             | idx rcvr invokable |
66             idx := frame pop.
67             rcvr := frame pop.
68
69             frame push: (rcvr field: idx integer) ]).
70
71     self installInstancePrimitive: (
72         SPrimitive new: 'instVarAt:put:' in: universe with: [:frame :interp |
73             | idx rcvr invokable val |
74             val := frame pop.
75             idx := frame pop.
76             rcvr := frame stackElement: 0.
77
78             rcvr field: idx integer put: val ]).
79
80     self installInstancePrimitive: (
81         SPrimitive new: 'class' in: universe with: [:frame :interp |
82             | rcvr |
83             rcvr := frame pop.
84             frame push: (rcvr somClassIn: universe) ]).
85
86     self installInstancePrimitive: (
87         SPrimitive new: 'halt' in: universe with: [:frame :interp |
88             | rcvr |
89             rcvr := frame stackElement: 0.
90             rcvr halt ]).
91 )
92
93 ----
94
95 new: universe = (
96     ^ self new initialize: universe
97 )
98 )
99

```

```
1 PrimitivePrimitives = Primitives (
2
3   installPrimitives = (
4     self installInstancePrimitive: (
5       SPrimitive new: 'holder' in: universe with: [:frame :interp |
6         | rcvr |
7         rcvr := frame pop.
8         frame push: rcvr holder ])).
9
10    self installInstancePrimitive: (
11      SPrimitive new: 'signature' in: universe with: [:frame :interp |
12        | rcvr |
13        rcvr := frame pop.
14        frame push: rcvr signature ])).
15  )
16
17  ----
18
19  new: universe = (
20    ^ self new initialize: universe
21  )
22 )
23
```

```
1 Primitives = (  
2   | universe holder |  
3   initialize: aUniverse = (  
4     universe := aUniverse  
5   )  
6  
7   installPrimitivesIn: aSClass = (  
8     holder := aSClass.  
9  
10    self installPrimitives  
11  )  
12  
13  installInstancePrimitive: prim = (  
14    self installInstancePrimitive: prim dontWarn: false  
15  )  
16  
17  installInstancePrimitive: prim dontWarn: suppressWarning = (  
18    holder addInstancePrimitive: prim dontWarn: suppressWarning  
19  )  
20  
21  installClassPrimitive: prim = (  
22    holder somClass addInstancePrimitive: prim  
23  )  
24  
25  somBool: aBool = (  
26    ^ aBool  
27      ifTrue: [ universe trueObject ]  
28      ifFalse: [ universe falseObject ]  
29  )  
30 )  
31
```

```

1 StringPrimitives = Primitives (
2
3   installPrimitives = (
4     self installInstancePrimitive: (
5       SPrimitive new: 'concatenate:' in: universe with: [:frame :interp |
6         | rcvr argument |
7         argument := frame pop.
8         rcvr := frame pop.
9
10        frame push: (universe newString: rcvr string + argument string) ]).
11
12    self installInstancePrimitive: (
13      SPrimitive new: 'asSymbol' in: universe with: [:frame :interp |
14        | rcvr |
15        rcvr := frame pop.
16        frame push: (universe symbolFor: rcvr string) ]).
17
18    self installInstancePrimitive: (
19      SPrimitive new: 'length' in: universe with: [:frame :interp |
20        | rcvr |
21        rcvr := frame pop.
22        frame push: (universe newInteger: rcvr string length) ]).
23
24    self installInstancePrimitive: (
25      SPrimitive new: '=' in: universe with: [:frame :interp |
26        | rcvr argument argCls |
27        argument := frame pop.
28        rcvr := frame pop.
29
30        argCls := argument somClassIn: universe.
31
32        frame push: (self somBool:
33          ((argCls == universe stringClass or: [argCls == universe
34            symbolClass])
35            and: [rcvr string = argument string])) ).
36
37    self installInstancePrimitive: (
38      SPrimitive new: 'primSubstringFrom:to:' in: universe with:
39      [:frame :interp |
40        | rcvr from to |
41        to := frame pop.
42        from := frame pop.
43        rcvr := frame pop.
44
45        frame push: (universe newString: (rcvr string primSubstringFrom:
46          from integer to: to integer)) ).
47
48    self installInstancePrimitive: (
49      SPrimitive new: 'hashCode' in: universe with: [:frame :interp |
50        | rcvr |
51        rcvr := frame pop.
52        frame push: (universe newInteger: rcvr string hashCode) ).
53
54    self installInstancePrimitive: (
55      SPrimitive new: 'isWhiteSpace' in: universe with: [:frame :interp |
56        | rcvr |
57        rcvr := frame pop.
58        frame push: (self somBool: rcvr string isWhiteSpace) ).

```

```

56
57     self installInstancePrimitive: (
58         SPrimitive new: 'isLetters' in: universe with: [:frame :interp |
59             | rcvr |
60             rcvr := frame pop.
61             frame push: (self somBool: rcvr string isLetters) ])).
62
63     self installInstancePrimitive: (
64         SPrimitive new: 'isDigits' in: universe with: [:frame :interp |
65             | rcvr |
66             rcvr := frame pop.
67             frame push: (self somBool: rcvr string isDigits) ])).
68 )
69
70 ----
71
72 new: universe = (
73     ^ self new initialize: universe
74 )
75 )
76

```

SomSom/src/primitives/SymbolPrimitives.som

```
1 SymbolPrimitives = Primitives (  
2  
3   installPrimitives = (  
4     self installInstancePrimitive: (  
5       SPrimitive new: 'asString' in: universe with: [:frame :interp |  
6         | rcvr |  
7         rcvr := frame pop.  
8  
9         frame push: (universe newString: rcvr string) ]).  
10  )  
11  
12  ----  
13  
14  new: universe = (  
15    ^ self new initialize: universe  
16  )  
17 )  
18
```



```

1 SystemPrimitives = Primitives (
2   installPrimitives = (
3
4     self installInstancePrimitive: (
5       SPrimitive new: 'load:' in: universe with: [:frame :interp |
6         | arg result |
7         arg := frame pop.
8         frame pop.
9
10        result := universe loadClass: arg.
11
12        frame push: (result == nil
13          ifTrue: [ universe nilObject ]
14          ifFalse: [ result ]) ).
15
16    self installInstancePrimitive: (
17      SPrimitive new: 'exit:' in: universe with: [:frame :interp |
18        | error |
19        frame printStackTrace.
20        error := frame pop.
21        universe exit: error integer ]).
22
23    self installInstancePrimitive: (
24      SPrimitive new: 'global:' in: universe with: [:frame :interp |
25        | argument result |
26        argument := frame pop.
27        frame pop.
28
29        result := universe global: argument.
30        frame push: (result == nil
31          ifTrue: [ universe nilObject ]
32          ifFalse: [ result ]) ).
33
34    self installInstancePrimitive: (
35      SPrimitive new: 'global:put:' in: universe with: [:frame :interp |
36        | value argument |
37        value := frame pop.
38        argument := frame pop.
39
40        universe global: argument put: value ]).
41
42    self installInstancePrimitive: (
43      SPrimitive new: 'printString:' in: universe with: [:frame :interp |
44        | arg |
45        arg := frame pop.
46        "Universe print: arg somClass asString."
47        Universe print: arg string ]).
48
49    self installInstancePrimitive: (
50      SPrimitive new: 'printNewline' in: universe with: [:frame :interp |
51        Universe println ]).
52
53    self installInstancePrimitive: (
54      SPrimitive new: 'errorPrint:' in: universe with: [:frame :interp |
55        | arg |
56        arg := frame pop.
57        Universe errorPrint: arg string ]).
58

```

```

59     self installInstancePrimitive: (
60         SPrimitive new: 'errorPrintln:' in: universe with: [:frame :interp |
61             | arg |
62             arg := frame pop.
63             Universe errorPrintln: arg string ])).
64
65     self installInstancePrimitive: (
66         SPrimitive new: 'time' in: universe with: [:frame :interp |
67             | time |
68             frame pop. "ignore"
69             time := system time.
70             frame push: (universe newInteger: time) ])).
71
72     self installInstancePrimitive: (
73         SPrimitive new: 'ticks' in: universe with: [:frame :interp |
74             | ticks |
75             frame pop. "ignore"
76             ticks := system ticks.
77             frame push: (universe newInteger: ticks) ])).
78
79     self installInstancePrimitive: (
80         SPrimitive new: 'gcStats' in: universe with: [:frame :interp |
81             | gcStats arr |
82             frame pop. "ignore"
83             gcStats := system gcStats.
84             arr := universe newArray: 3.
85             arr indexableField: 1 put: (universe newInteger: (gcStats at: 1)).
86             arr indexableField: 2 put: (universe newInteger: (gcStats at: 2)).
87             arr indexableField: 3 put: (universe newInteger: (gcStats at: 3)).
88
89             frame push: arr ])).
90
91     self installInstancePrimitive: (
92         SPrimitive new: 'totalCompilationTime' in: universe with:
93         [:frame :interp |
94             | cTime |
95             frame pop. "ignore"
96             cTime := system totalCompilationTime.
97             frame push: (universe newInteger: cTime) ])).
98
99     self installInstancePrimitive: (
100         SPrimitive new: 'fullGC' in: universe with: [:frame :interp |
101             frame pop. "ignore"
102             system fullGC.
103             frame push: (universe trueObject) ])).
104
105     self installInstancePrimitive: (
106         SPrimitive new: 'loadFile:' in: universe with: [:frame :interp |
107             | fileName content |
108             fileName := frame pop.
109             frame pop.
110
111             content := system loadFile: fileName string.
112             content == nil
113                 ifTrue: [frame push: universe nilObject]
114                 ifFalse: [frame push: (universe newString: content)] ])).
115
116     self installInstancePrimitive: (
117         SPrimitive new: 'printStackTrace' in: universe with:
118         [:frame :interp |
119             frame pop. "ignore"

```

```
118         frame printStackTrace.
119         frame push: (universe trueObject) ]).
120     )
121
122     ----
123
124     new: universe = (
125         ^ self new initialize: universe
126     )
127 )
128
```

```
1 Main = (  
2   run: args = (  
3     | u args2 |  
4     u := Universe new.  
5     args2 := args copyFrom: 2.  
6     u interpret: args2.  
7     u exit: 0.  
8   )  
9 )  
10
```

```

1 MainLoadAll = (
2   loadAllSomSomSources = (
3     #(
4       #Bytecodes
5       #Interpreter
6       #Frame
7       #SString
8       #SObject
9       #SAbstractObject
10      #SSymbol
11      #SBlock
12      #SDouble
13      #SArray
14      #SPrimitive
15      #SMethod
16      #SClass
17      #SInteger
18      #SystemPrimitives
19      #ClassPrimitives
20      #DoublePrimitives
21      #Primitives
22      #IntegerPrimitives
23      #PrimitivePrimitives
24      #SymbolPrimitives
25      #MethodPrimitives
26      #StringPrimitives
27      #BlockPrimitives
28      #ObjectPrimitives
29      #ArrayPrimitives
30      #Main
31      #Universe
32      #MainLoadAll
33      #Parser
34      #BytecodeGenerator
35      #ClassGenerationContext
36      #Lexer
37      #SourcecodeCompiler
38      #Disassembler
39      #MethodGenerationContext
40    ) do: [:className |
41      (system load: className) println. ]
42  )
43  run: args = (
44    | u args2 |
45    u := Universe new.
46    args2 := args copyFrom: 2.
47    u interpret: args2.
48    u exit: 0.
49  )
50 )
51

```

```
1 Universe = (  
2   | symbolTable globals classPath dumpBytecodes interpreter  
3  
4   avoidExit  
5   lastExitCode  
6   exitBlock  
7  
8   nilObject  
9   trueObject  
10  falseObject  
11  
12  objectClass  
13  classClass  
14  metaclassClass  
15  
16  nilClass  
17  integerClass  
18  arrayClass  
19  methodClass  
20  symbolClass  
21  primClass  
22  stringClass  
23  systemClass  
24  blockClass  
25  doubleClass  
26  
27  trueClass  
28  falseClass  
29  |  
30  
31  initialize = (  
32    symbolTable := Dictionary new.  
33    globals := Dictionary new.  
34    interpreter := Interpreter new: self.  
35    dumpBytecodes := false.  
36    avoidExit := false  
37  )  
38  
39  initialize: aBool = (  
40    self initialize.  
41    avoidExit := aBool  
42  )  
43  
44  exit: errorCode = (  
45    "Exit from the Java system"  
46    avoidExit  
47    ifTrue: [  
48      lastExitCode := errorCode.  
49      exitBlock value: errorCode ]  
50    ifFalse: [system exit: errorCode]  
51  )  
52  
53  lastExitCode = (  
54    ^ lastExitCode  
55  )  
56  
57  errorExit: message = (  
58    Universe errorPrintln: 'Runtime Error: ' + message.
```

```

59     self exit: 1
60 )
61
62 nilObject    = ( ^ nilObject )
63 trueObject   = ( ^ trueObject )
64 falseObject  = ( ^ falseObject )
65 metaclassClass = ( ^ metaclassClass )
66
67 arrayClass   = ( ^ arrayClass )
68 blockClass   = ( ^ blockClass )
69 doubleClass  = ( ^ doubleClass )
70 integerClass = ( ^ integerClass )
71 methodClass  = ( ^ methodClass )
72 primClass    = ( ^ primClass )
73 stringClass  = ( ^ stringClass )
74 symbolClass  = ( ^ symbolClass )
75
76 defaultClassPath = (
77     ^ #('.')
78 )
79
80 setupClassPath: cp = (
81     | paths cps |
82     "Create a new tokenizer to split up the string of directories"
83     paths := cp split: ':'.
84
85     cps := Vector new.
86     cps appendAll: self defaultClassPath.
87     cps appendAll: paths.
88
89     classPath := cps asArray
90 )
91
92 handleArguments: args = (
93     | gotClasspath remainingArgs cnt i sawOthers |
94     gotClasspath := false.
95     remainingArgs := Vector new.
96
97     "read dash arguments only while we haven't seen other kind of
arguments"
98     sawOthers := false.
99
100    i := 1.
101
102    [i <= args length] whileTrue: [
103        ((args at: i) = '-cp' and: sawOthers not)
104        ifTrue: [
105            i + 1 > args length ifTrue: [
106                self printUsageAndExit ].
107            self setupClassPath: (args at: i + 1).
108            i := i + 1.
109            gotClasspath := true ]
110        ifFalse: [
111            ((args at: i) = '-d' and: sawOthers not)
112            ifTrue: [ dumpBytecodes := true ]
113            ifFalse: [
114                sawOthers := true.
115                remainingArgs append: (args at: i) ] ].
116        i := i + 1 ].
117
118    gotClasspath ifFalse: [

```

```

119         classPath := self defaultClassPath ].
120
121     remainingArgs isEmpty ifFalse: [
122         | split |
123         split := self pathClassExtension: (remainingArgs at: 1).
124         (split at: 1) = '' ifFalse: [
125             classPath := classPath prependedWith: (split at: 1) ].
126         remainingArgs at: 1 put: (split at: 2) ].
127
128     ^ remainingArgs asArray
129 )
130
131 pathClassExtension: str = (
132     | pathElements fileName parentPath nameParts |
133     pathElements := str split: '/'.
134     fileName := pathElements last.
135
136     parentPath := ''.
137     1 to: pathElements length - 1 do: [:i |
138         parentPath = '' ifFalse: [
139             parentPath := parentPath + '/' ].
140         parentPath := parentPath + (pathElements at: i) ].
141
142     nameParts := fileName split: '.'.
143     ^ Array with: parentPath with: (nameParts at: 1)
144 )
145
146 interpret: args = (
147     | remainingArgs result |
148     remainingArgs := self handleArguments: args.
149     result := self initializeInterpreter: remainingArgs.
150     result class == SInteger
151         ifTrue: [ ^ result integer ]
152         ifFalse: [ ^ 1 ]
153 )
154
155 interpret: className with: selector = (
156     | clazz initialize |
157     self initializeObjectSystem.
158
159     clazz := self loadClass: (self symbolFor: className).
160
161     "Lookup the initialize invokable on the system class"
162     initialize := (clazz somClassIn: self) lookupInvokable: (self
symbolFor: selector).
163
164     initialize == nil ifTrue: [
165         self error: 'Lookup of ' + className + '>>#' + selector + '
failed' ].
166
167     ^ self interpret: initialize in: clazz with: nil
168 )
169
170 initializeInterpreter: arguments = (
171     | systemObject initialize argumentsArray |
172     systemObject := self initializeObjectSystem.
173
174     "Start the shell if no filename is given"
175     arguments length == 0 ifTrue: [
176         | shell |
177         shell := Shell for: self using: interpreter.

```



```

178     shell bootstrapMethod: self createBootstrapMethod.
179     ^ shell start ].
180
181     "Lookup the initialize invokable on the system class"
182     initialize := systemClass lookupInvokable: (self symbolFor:
'initialize:').
183
184     "Convert the arguments into an array"
185     argumentsArray := self newArrayFromStrings: arguments.
186
187     ^ self interpret: initialize in: systemObject with: argumentsArray
188 )
189
190 createBootstrapMethod = (
191     | bootstrapMethod |
192     "Create a fake bootstrap method to simplify later frame traversal"
193     bootstrapMethod := self newMethod: (self symbolFor: 'bootstrap')
194     bc: #(#halt) literals: #() numLocals: 0 maxStack: 2.
195
196     bootstrapMethod holder: systemClass.
197     ^ bootstrapMethod
198 )
199
200 interpret: invokable in: receiver with: arguments = (
201     | bootstrapMethod bootstrapFrame |
202     exitBlock := [:errorCode | ^ errorCode ].
203
204     bootstrapMethod := self createBootstrapMethod.
205
206     "Create a fake bootstrap frame with the system object on the stack"
207     bootstrapFrame := interpreter pushNewFrame: bootstrapMethod.
208     bootstrapFrame push: receiver.
209
210     arguments ~= nil ifTrue: [
211         bootstrapFrame push: arguments ].
212
213     "Invoke the initialize invokable"
214     invokable invoke: bootstrapFrame using: interpreter.
215
216     "Start the interpreter"
217     ^ interpreter start
218 )
219
220 initializeObjectSystem = (
221     | trueSymbol falseSymbol systemObject |
222
223     "Allocate the nil object"
224     nilObject := SObject new.
225
226     "Allocate the Metaclass classes"
227     metaclassClass := self newMetaclassClass.
228
229     "Allocate the rest of the system classes"
230     objectClass := self newSystemClass.
231     nilClass := self newSystemClass.
232     classClass := self newSystemClass.
233     arrayClass := self newSystemClass.
234     symbolClass := self newSystemClass.
235     methodClass := self newSystemClass.
236     integerClass := self newSystemClass.
237     primClass := self newSystemClass.

```

```

238     stringClass := self newSystemClass.
239     doubleClass := self newSystemClass.
240
241     "Setup the class reference for the nil object"
242     nilObject somClass: nilClass.
243
244     "Initialize the system classes."
245     self initializeSystemClass: objectClass superClass: nil name:
'Object'.
246     self initializeSystemClass: classClass superClass: objectClass name:
'Class'.
247     self initializeSystemClass: metaclassClass superClass: classClass
name: 'Metaclass'.
248     self initializeSystemClass: nilClass superClass: objectClass name:
'Nil'.
249     self initializeSystemClass: arrayClass superClass: objectClass name:
'Array'.
250     self initializeSystemClass: methodClass superClass: arrayClass name:
'Method'.
251     self initializeSystemClass: stringClass superClass: objectClass name:
'String'.
252     self initializeSystemClass: symbolClass superClass: stringClass name:
'Symbol'.
253     self initializeSystemClass: integerClass superClass: objectClass
name: 'Integer'.
254     self initializeSystemClass: primClass superClass: objectClass name:
'Primitive'.
255     self initializeSystemClass: doubleClass superClass: objectClass name:
'Double'.
256
257     "Load methods and fields into the system classes"
258     self loadSystemClass: objectClass.
259     self loadSystemClass: classClass.
260     self loadSystemClass: metaclassClass.
261     self loadSystemClass: nilClass.
262     self loadSystemClass: arrayClass.
263     self loadSystemClass: methodClass.
264     self loadSystemClass: symbolClass.
265     self loadSystemClass: integerClass.
266     self loadSystemClass: primClass.
267     self loadSystemClass: stringClass.
268     self loadSystemClass: doubleClass.
269
270     "Fix up objectClass"
271     objectClass superClass: nilObject.
272
273     "Load the generic block class"
274     blockClass := self loadClass: (self symbolFor: 'Block').
275
276     "Setup the true and false objects"
277     trueSymbol := self symbolFor: 'True'.
278     trueClass := self loadClass: trueSymbol.
279     trueObject := self newInstance: trueClass.
280
281     falseSymbol := self symbolFor: 'False'.
282     falseClass := self loadClass: falseSymbol.
283     falseObject := self newInstance: falseClass.
284
285     "Load the system class and create an instance of it"
286     systemClass := self loadClass: (self symbolFor: 'System').
287     systemObject := self newInstance: systemClass.

```

```

288
289     "Put special objects and classes into the dictionary of globals"
290     self global: (self symbolFor: 'nil') put: nilObject.
291     self global: (self symbolFor: 'true') put: trueObject.
292     self global: (self symbolFor: 'false') put: falseObject.
293     self global: (self symbolFor: 'system') put: systemObject.
294     self global: (self symbolFor: 'System') put: systemClass.
295     self global: (self symbolFor: 'Block') put: blockClass.
296     self global: trueSymbol put: trueClass.
297     self global: falseSymbol put: falseClass.
298     ^ systemObject
299 )
300
301 symbolFor: aString = (
302     | result |
303     result := symbolTable at: aString.
304     result == nil ifFalse: [
305         ^ result ].
306     ^ self newSymbol: aString
307 )
308
309
310 newArray: size = (
311     ^ SArray new: size with: nilObject
312 )
313
314 newArrayFromStrings: strArray = (
315     | sArr |
316     sArr := self newArray: strArray length.
317     1 to: strArray length do: [:i |
318         sArr indexableField: i put: (self newString: (strArray at: i))].
319     ^ sArr
320 )
321
322 newArrayFromVector: vector = (
323     | result |
324     "Allocate a new array with the same length as the list"
325     result := self newArray: vector size.
326
327     "Copy all elements from the list into the array"
328     vector doIndexes: [:i |
329         result indexableField: i put: (vector at: i) ].
330
331     "Return the allocated and initialized array"
332     ^ result
333 )
334
335 newBlock: method with: context numArgs: arguments = (
336     ^ SBlock new: method in: context with: (self blockClass: arguments)
337 )
338
339 newClass: classClass = (
340     | result |
341     "Allocate a new class and set its class to be the given class class"
342     result := SClass new: classClass numberOfInstanceFields in: self.
343     result somClass: classClass.
344
345     "Return the freshly allocated class"
346     ^ result
347 )
348

```

```

349   newFrame: previousFrame with: method with: contextFrame = (
350       | length result |
351       "Compute the maximum number of stack locations (including arguments,
352       locals and extra buffer to support doesNotUnderstand) and set the
number
353       of indexable fields accordingly"
354       length := method numberOfArguments
355               + method numberOfLocals
356               + method maximumNumberOfStackElements + 2.
357
358       result := Frame new: nilObject previous: previousFrame context:
contextFrame method: method maxStack: length.
359
360       "Return the freshly allocated frame"
361       ^ result
362   )
363
364   newSymbol: aString = (
365       | result |
366       result := SSymbol new: aString.
367       symbolTable at: aString put: result.
368       ^ result
369   )
370
371   newInstance: instanceClass = (
372       | result |
373       result := SObject new: instanceClass numberOfInstanceFields with:
nilObject.
374       result somClass: instanceClass.
375
376       ^ result
377   )
378
379   newInteger: anInteger = (
380       ^ SInteger for: anInteger
381   )
382
383   newDouble: aDouble = (
384       ^ SDouble for: aDouble
385   )
386
387   newMetaclassClass = (
388       | result |
389       "Allocate the metaclass classes"
390       result := SClass new: self.
391       result somClass: (SClass new: self).
392
393       "Setup the metaclass hierarchy"
394       result somClass somClass: result.
395
396       "Return the freshly allocated metaclass class"
397       ^ result
398   )
399
400   newMethod: aSSymbol bc: bcArray literals: literalsArray numLocals:
numLocals maxStack: maxStack = (
401       ^ SMethod new: aSSymbol bc: bcArray literals: literalsArray
numLocals: numLocals maxStack: maxStack
402   )
403
404   newString: aString = (

```

```

405     ^ SString new: aString
406 )
407
408 newSystemClass = (
409     | symbolClass |
410     "Allocate the new system class"
411     systemClass := SClass new: self.
412
413     "Setup the metaclass hierarchy"
414     systemClass somClass: (SClass new: self).
415     systemClass somClass somClass: metaclassClass.
416
417     "Return the freshly allocated system class"
418     ^ systemClass
419 )
420
421 initializeSystemClass: systemClass superClass: superClass name: name = (
422     "Initialize the superclass hierarchy"
423     superClass ~= nil
424     ifTrue: [
425         systemClass superClass: superClass.
426         systemClass somClass superClass: (superClass somClass) ]
427     ifFalse: [
428         systemClass somClass superClass: classClass ].
429
430     "Initialize the array of instance fields"
431     systemClass instanceFields: (self newArray: 0).
432     systemClass somClass instanceFields: (self newArray: 0).
433
434     "Initialize the array of instance invokables"
435     systemClass instanceInvokables: (self newArray: 0).
436     systemClass somClass instanceInvokables: (self newArray: 0).
437
438     "Initialize the name of the system class"
439     systemClass name: (self symbolFor: name).
440     systemClass somClass name: (self symbolFor: name + ' class').
441
442     "Insert the system class into the dictionary of globals"
443     self global: systemClass name put: systemClass.
444 )
445
446 global: aSSymbol = (
447     "Return the global with the given name if it's in the dictionary of
globals"
448     (self hasGlobal: aSSymbol) ifTrue: [
449         ^ globals at: aSSymbol ].
450
451     "Global not found"
452     ^ nil
453 )
454
455 global: aSSymbol put: aSAbstractObject = (
456     "Insert the given value into the dictionary of globals"
457     globals at: aSSymbol put: aSAbstractObject
458 )
459
460 hasGlobal: aSSymbol = (
461     "Returns if the universe has a value for the global of the given name"
462     ^ globals containsKey: aSSymbol
463 )
464

```

```

465   blockClass: numberOfArguments = (
466       | name result |
467       "Determine the name of the block class with the given number of
arguments"
468       name := self symbolFor: 'Block' + numberOfArguments.
469
470       "Lookup the block class in the dictionary of globals and return it"
471       (self hasGlobal: name) ifTrue: [
472           ^ self global: name ].
473
474       result := self loadClass: name into: nil.
475
476       "Add the appropriate value primitive to the block class"
477       result addInstancePrimitive:
478           (SBlock evaluationPrimitive: numberOfArguments in: self).
479
480       self global: name put: result.
481       ^ result
482   )
483
484   loadClass: name = (
485       | result |
486       "Check if the requested class is already in the dictionary of globals"
487       (self hasGlobal: name) ifTrue: [
488           ^ self global: name ].
489
490       "Load the class"
491       result := self loadClass: name into: nil.
492
493       "Load primitives (if necessary) and return the resulting class"
494       (result ~= nil and: [result hasPrimitives]) ifTrue: [
495           result loadPrimitives ].
496
497       self global: name put: result.
498       ^ result
499   )
500
501   loadSystemClass: systemClass = (
502       | result |
503       "Load the system class"
504       result := self loadClass: systemClass name into: systemClass.
505
506       "Load primitives if necessary"
507       result hasPrimitives ifTrue: [
508           result loadPrimitives ].
509   )
510
511   loadClass: name into: systemClass = (
512       "Try loading the class from all different paths"
513       classPath do: [:cpEntry |
514           | result |
515           "Load the class from a file and return the loaded class"
516           result := SourcecodeCompiler compileClass: cpEntry name: name
string into: systemClass in: self.
517
518           (result notNil and: dumpBytecodes) ifTrue: [
519               Disassembler dump: result somClass in: self.
520               Disassembler dump: result in: self ].
521
522           result ifNotNil: [ ^ result ].
523

```

```

524     "The class could not be found."
525     ^ nil
526 )
527
528 loadShellClass: stmt = (
529     | result |
530     "Load the class from a stream and return the loaded class"
531     result := SourcecodeCompiler compileClass: stmt into: nil in: self.
532     dumpBytecodes ifTrue: [
533         Disassembler dump: result in: self ].
534     ^ result
535 )
536
537 ----
538
539 new = (
540     ^ super new initialize
541 )
542
543 new: avoidExit = (
544     ^ super new initialize: avoidExit
545 )
546
547 errorPrint: msg = (
548     system errorPrint: msg
549 )
550
551 errorPrintln: msg = (
552     system errorPrintln: msg
553 )
554
555 errorPrintln = (
556     system errorPrintln: ''
557 )
558
559 print: msg = (
560     system errorPrint: msg
561 )
562
563 println: msg = (
564     system errorPrintln: msg
565 )
566
567 println = (
568     system errorPrintln
569 )
570 )
571

```

```

1 SAbstractObject = (
2   send: selectorString with: arguments in: universe using: interpreter = (
3     | selector invocable |
4     selector := universe symbolFor: selectorString.
5
6     interpreter frame push: self.
7
8     arguments do: [:arg |
9       interpreter frame push: arg ].
10
11     invocable := (self somClassIn: universe) lookupInvokable: selector.
12
13     invocable invoke: interpreter frame using: interpreter
14   )
15
16   sendDoesNotUnderstand: selector in: universe using: interpreter = (
17     | numberOfArguments frame argumentsArray args |
18     numberOfArguments := selector numberOfSignatureArguments.
19
20     frame := interpreter frame.
21     frame printStackTrace.
22
23     "Allocate an array with enough room to hold all arguments
24     except for the receiver, which is passed implicitly, as receiver of
25     #dnu."
26     argumentsArray := universe newArray: numberOfArguments - 1.
27
28     "Remove all arguments and put them in the freshly allocated array"
29     numberOfArguments - 1 downTo: 1 do: [:i |
30       argumentsArray indexableField: i put: frame pop ].
31
32     frame pop. "pop receiver"
33
34     args := Array with: selector with: argumentsArray.
35     self send: 'doesNotUnderstand:arguments:' with: args in: universe
36   using: interpreter
37   )
38
39   sendUnknownGlobal: globalName in: universe using: interpreter = (
40     | arguments |
41     arguments := Array with: globalName.
42     self send: 'unknownGlobal:' with: arguments in: universe using:
43     interpreter
44   )
45
46   sendEscapedBlock: block in: universe using: interpreter = (
47     | arguments |
48     arguments := Array with: block.
49     self send: 'escapedBlock:' with: arguments in: universe using:
50     interpreter
51   )
52 )

```



```

1 SArray = SAbstractObject (
2   | indexableFields |
3
4   initializeWith: length and: nilObject = (
5     indexableFields := Array new: length withAll: nilObject.
6   )
7
8   somClassIn: universe = (
9     ^ universe arrayClass
10  )
11
12  indexableField: idx = (
13    ^ indexableFields at: idx
14  )
15
16  indexableField: idx put: val = (
17    ^ indexableFields at: idx put: val
18  )
19
20  numberOfIndexableFields = (
21    ^ indexableFields length
22  )
23
24  copyAndExtendWith: value in: universe = (
25    | result newLength |
26    newLength := indexableFields length + 1.
27    result := universe newArray: newLength.
28
29    self copyIndexableFieldsTo: result.
30
31    result indexableField: newLength put: value.
32    ^ result
33  )
34
35  copyIndexableFieldsTo: destination = (
36    indexableFields doIndexes: [:i |
37      destination indexableField: i put: (indexableFields at: i) ]
38  )
39
40  "For using in debugging tools such as the Diassembler"
41  debugString = (
42    | elems |
43    elems := ''.
44    indexableFields do: [:e |
45      elems = '' ifTrue: [elems := e debugString]
46      ifFalse: [ elems := elems + ', ' + e debugString] ].
47    ^ 'SArray(' + indexableFields length + '; ' + elems + ')' )
48
49  ----
50
51  new: length with: nilObject = (
52    ^ self new initializeWith: length and: nilObject
53  )
54 )
55

```

SomSom/src/vmobjects/SBlock.som

```
1 SBlock = SAbstractObject (
2   | method context blockClass |
3
4   initialize: aSMethod in: aContext with: aBlockClass = (
5     method := aSMethod.
6     context := aContext.
7     blockClass := aBlockClass.
8   )
9
10  method = (
11    ^ method
12  )
13
14  context = (
15    ^ context
16  )
17
18  somClassIn: universe = (
19    ^ blockClass
20  )
21
22  "For using in debugging tools such as the Diassembler"
23  debugString = ( ^ 'SBlock(' + method asString + '))' )
24
25  ----
26
27  new: aSMethod in: aContext with: aBlockClass = (
28    ^ self new initialize: aSMethod in: aContext with: aBlockClass
29  )
30
31  evaluationPrimitive: numberOfArguments in: universe = (
32    ^ SPrimitive new: (self computeSignatureString: numberOfArguments)
33      in: universe
34      with: [:frame :interp |
35        | rcvr context newFrame |
36        "Get the block (the receiver) from the stack"
37        rcvr := frame stackElement: numberOfArguments - 1.
38
39        "Get the context of the block"
40        context := rcvr context.
41
42        "Push a new frame and set its context to be the one specified in
43        the block"
44        newFrame := interp pushNewFrame: rcvr method with: context.
45        newFrame copyArgumentsFrom: frame ]
46  )
47
48  computeSignatureString: numberOfArguments = (
49    | signatureString |
50    signatureString := 'value'.
51    numberOfArguments > 1 ifTrue: [
52      signatureString := signatureString + ':' ].
53
54    "Add extra with: selector elements if necessary"
55    2 to: numberOfArguments - 1 do: [:i |
56      signatureString := signatureString + 'with:' ].
57
58    ^ signatureString
```

59)
60)
61

```
1 SClass = SObject (
2   | universe
3   | superClass
4   | name
5   | instanceInvokables instanceFields|
6
7   initialize: aUniverse = (
8     universe := aUniverse
9   )
10
11   initialize: numberOfFields in: aUniverse = (
12     super initialize: numberOfFields with: aUniverse nilObject.
13     universe := aUniverse
14   )
15
16   superClass = (
17     ^ superClass
18   )
19
20   superClass: aSClass = (
21     superClass := aSClass
22   )
23
24   hasSuperClass = (
25     ^ superClass ~= universe nilObject
26   )
27
28   name = (
29     ^ name
30   )
31
32   name: aSSymbol = (
33     name := aSSymbol
34   )
35
36   instanceFields = (
37     ^ instanceFields
38   )
39
40   instanceFields: aSArray = (
41     instanceFields := aSArray
42   )
43
44   instanceInvokables = (
45     ^ instanceInvokables
46   )
47
48   instanceInvokables: aSArray = (
49     instanceInvokables := aSArray.
50
51     "Make sure this class is the holder of all invokables in the array"
52     1 to: self numberOfInstanceInvokables do: [:i |
53       (instanceInvokables indexableField: i) holder: self ]
54   )
55
56   numberOfInstanceInvokables = (
57     ^ instanceInvokables numberOfIndexableFields
58   )
```

```

59
60 instanceInvokable: idx = (
61   ^ instanceInvokables indexableField: idx
62 )
63
64 instanceInvokable: idx put: aSInvokable = (
65   aSInvokable holder: self.
66   instanceInvokables indexableField: idx put: aSInvokable
67 )
68
69 lookupInvokable: signature = (
70   | invokable |
71
72   "Lookup invokable with given signature in array of instance
invokables"
73   1 to: instanceInvokables numberOfIndexableFields do: [:i |
74     "Get the next invokable in the instance invokable array"
75     invokable := instanceInvokables indexableField: i.
76
77     "Return the invokable if the signature matches"
78     invokable signature == signature ifTrue: [
79       ^ invokable ] ].
80
81   "Traverse the super class chain by calling lookup on the super class"
82   self hasSuperClass ifTrue: [
83     invokable := superClass lookupInvokable: signature.
84     invokable ~= nil ifTrue: [
85       ^ invokable ] ].
86
87   "Invokable not found"
88   ^ nil
89 )
90
91 lookupFieldIndex: fieldName = (
92   "Lookup field with given name in array of instance fields"
93
94   self numberOfInstanceFields downTo: 1 do: [:i |
95     "Return the current index if the name matches"
96     fieldName == (self instanceFieldName: i)
97     ifTrue: [ ^ i ] ].
98
99   "Field not found"
100   ^ -1
101 )
102
103 addInstanceInvokable: value = (
104   "Add the given invokable to the array of instance invokables"
105   1 to: self numberOfInstanceInvokables do: [:i |
106     "Get the next invokable in the instance invokable array"
107     | invokable |
108     invokable := self instanceInvokable: i.
109
110     "Replace the invokable with the given one if the signature matches"
111     invokable signature == value signature ifTrue: [
112       self instanceInvokable: i put: value.
113       ^ false ] ].
114
115   "Append the given method to the array of instance methods"
116   instanceInvokables := instanceInvokables copyAndExtendWith: value in:
universe.
117   ^ true

```

```

118 )
119
120 addInstancePrimitive: value = (
121     self addInstancePrimitive: value dontWarn: false
122 )
123
124 addInstancePrimitive: value dontWarn: suppressWarning = (
125     value holder: self.
126     ((self addInstanceInvokable: value) and: [suppressWarning not])
127     ifTrue: [
128         Universe print: 'Warning: Primitive ' + value signature string.
129         Universe println: ' is not in class definition for class ' + name
130         string ]
131     )
132
133 instanceFieldName: index = (
134     "Get the name of the instance field with the given index"
135     index > self numberOfSuperInstanceFields
136     ifTrue: [
137         | idx |
138         "Adjust the index to account for fields defined in the super
139         class"
140         idx := index - self numberOfSuperInstanceFields.
141         "Return the symbol representing the instance fields name"
142         ^ instanceFields indexableField: idx ]
143     ifFalse: [
144         "Ask the super class to return the name of the instance field"
145         ^ superClass instanceFieldName: index ]
146 )
147
148 numberOfInstanceFields = (
149     "Get the total number of instance fields in this class"
150     ^ instanceFields numberOfIndexableFields + self
151     numberOfSuperInstanceFields
152 )
153
154 numberOfSuperInstanceFields = (
155     self hasSuperClass
156     ifTrue: [ ^ self superClass numberOfInstanceFields ]
157     ifFalse: [ ^ 0 ]
158 )
159
160 hasPrimitives = (
161     "Lookup invokable with given signature in array of instance
162     invokables"
163     1 to: self numberOfInstanceInvokables do: [:i |
164         "Get the next invokable in the instance invokable array"
165         (self instanceInvokable: i) isPrimitive
166         ifTrue: [ ^ true ] ].
167     ^ false
168 )
169
170 loadPrimitives = (
171     | className primsClass |
172     className := (name string + 'Primitives') asSymbol.
173     "Try loading the primitives"
174
175     primsClass := system load: className.
176     primsClass ~= nil

```

```

174         ifTrue: [
175             (primsClass new: universe) installPrimitivesIn: self ]
176         ifFalse: [
177             Universe println: 'Primitives class ' + className + ' not found' ]
178     )
179
180
181     "For using in debugging tools such as the Diassembler"
182     debugString = ( ^ 'SClass(' + name string + ' )' )
183
184     ----
185
186     new: universe = (
187         ^ self new initialize: universe
188     )
189
190     new: numberOfFields in: universe = (
191         ^ self new initialize: numberOfFields in: universe
192     )
193 )
194

```

```
1 SDouble = SAbstractObject (
2   | value |
3
4   initialize: aDouble = (
5     value := aDouble
6   )
7
8   double = ( ^ value )
9
10  somClassIn: universe = (
11    ^ universe doubleClass
12  )
13
14  "For using in debugging tools such as the Diassembler"
15  debugString = ( ^ 'SDouble(' + value asString + ')' )
16
17  ----
18
19  for: aDouble = (
20    ^ self new initialize: aDouble
21  )
22 )
23
```


SomSom/src/vmobjects/SInteger.som

```
1 SInteger = SAbstractObject (
2   | value |
3
4   initialize: anInteger = (
5     value := anInteger
6   )
7
8   integer = ( ^ value )
9
10  somClassIn: universe = (
11    ^ universe integerClass
12  )
13
14  "For using in debugging tools such as the Diassembler"
15  debugString = ( ^ 'SInteger(' + value asString + ')' )
16
17  ----
18
19  "TODO: see whether it makes sense to have a cache"
20  for: anInteger = (
21    ^ self new initialize: anInteger
22  )
23 )
24
```

```

1 SMethod = SAbstractObject (
2   | signature
3   | holder
4   | bytecodes literals
5   | numberOfLocals maximumNumberOfStackElements |
6
7   initializeWith: aSSymbol bc: bcArray literals: literalsArray numLocals:
numLocals maxStack: maxStack = (
8     signature := aSSymbol.
9     bytecodes := bcArray.
10    literals := literalsArray.
11    numberOfLocals := numLocals.
12    maximumNumberOfStackElements := maxStack.
13  )
14
15  isPrimitive = (
16    ^ false
17  )
18
19  numberOfLocals = (
20    ^ numberOfLocals
21  )
22
23  maximumNumberOfStackElements = (
24    ^ maximumNumberOfStackElements
25  )
26
27  signature = (
28    ^ signature
29  )
30
31  holder = (
32    ^ holder
33  )
34
35  holder: value = (
36    holder := value.
37
38    literals == nil ifTrue: [ ^ self ].
39
40    "Make sure all nested invokables have the same holder"
41    literals do: [:l |
42      (l class == SMethod or: [l class == SPrimitive]) ifTrue: [
43        l holder: value ] ]
44  )
45
46  constant: bytecodeIndex = (
47    "Get the constant associated to a given bytecode index"
48    ^ literals at: (bytecodes at: bytecodeIndex + 1)
49  )
50
51  numberOfArguments = (
52    "Get the number of arguments of this method"
53    ^ signature numberOfSignatureArguments
54  )
55
56  numberOfBytecodes = (
57    "Get the number of bytecodes in this method"

```

```

58     ^ bytecodes length
59 )
60
61 bytecode: index = (
62     "Get the bytecode at the given index"
63     ^ bytecodes at: index
64 )
65
66 invoke: frame using: interpreter = (
67     | newFrame |
68     "Allocate and push a new frame on the interpreter stack"
69     newFrame := interpreter pushNewFrame: self.
70     newFrame copyArgumentsFrom: frame
71 )
72
73 somClassIn: universe = (
74     ^ universe methodClass
75 )
76
77 "For using in debugging tools such as the Diassembler"
78 debugString = ( ^ 'SMethod(' + holder name + '>>#' + signature string +
79 ') ' )
80 ----
81
82 new: aSSymbol bc: bcArray literals: literalsArray numLocals: numLocals
maxStack: maxStack = (
83     ^ self new
84     initializeWith: aSSymbol bc: bcArray literals: literalsArray
numLocals: numLocals maxStack: maxStack
85 )
86
87 )
88

```

SomSom/src/vmobjects/SObject.som

```
1 SObject = SAbstractObject (
2   | fields clazz |
3
4   initialize: numberOfFields with: nilObject = (
5     fields := Array new: numberOfFields withAll: nilObject
6   )
7
8   somClass = (
9     ^ clazz
10  )
11
12  somClass: aSClass = (
13    clazz := aSClass
14  )
15
16  somClassIn: universe = (
17    ^ clazz
18  )
19
20  fieldName: index = (
21    "Get the name of the field with the given index"
22    ^ clazz instanceFieldName: index
23  )
24
25  fieldIndex: name = (
26    "Get the index for the field with the given name"
27    ^ clazz lookupFieldIndex: name
28  )
29
30  numberOfFields = (
31    "Get the number of fields in this object"
32    ^ fields length
33  )
34
35  field: index = (
36    "Get the field with the given index"
37    ^ fields at: index
38  )
39
40  field: index put: value = (
41    "Set the field with the given index to the given value"
42    fields at: index put: value
43  )
44
45  "For using in debugging tools such as the Diassembler"
46  debugString = ( ^ 'SObject(' + clazz name string + ')' )
47
48  ----
49
50  new: numberOfFields with: nilObject = (
51    ^ self new initialize: numberOfFields with: nilObject
52  )
53 )
54
```

```

1 SPrimitive = SAbstractObject (
2   | signature holder isEmpty operation |
3
4   initialize: aSSymbol with: aBlock = (
5     signature := aSSymbol.
6     isEmpty := false.
7     operation := aBlock.
8   )
9
10  initializeEmpty: aSSymbol in: universe = (
11    signature := aSSymbol.
12    isEmpty := true.
13    operation := [:frame :interp |
14      | receiver msg |
15      signature numberOfSignatureArguments timesRepeat: [
16        receiver := frame pop ].
17      msg := 'Undefined primitive ' + (receiver somClassIn: universe) name
string +
18        '>>#' + signature string + ' called'.
19      self send: 'error:' with: (Array with: receiver with: (universe
newString: msg))
20        in: universe using: interp ].
21  )
22
23  isPrimitive = ( ^ true )
24
25  signature = (
26    ^ signature
27  )
28
29  holder = (
30    ^ holder
31  )
32
33  holder: aSClass = (
34    holder := aSClass
35  )
36
37  isEmpty = (
38    ^ isEmpty
39  )
40
41  invoke: frame using: interp = (
42    ^ operation value: frame with: interp
43  )
44
45  somClassIn: universe = (
46    ^ universe primClass
47  )
48
49  "For using in debugging tools such as the Diassembler"
50  debugString = ( ^ 'SPrimitive(' + holder name string + '>>#' + signature
string + ')' )
51
52  ----
53
54  new: signatureString in: universe with: aBlock = (
55    ^ self new initialize: (universe symbolFor: signatureString)

```

```
56             with: aBlock
57         )
58
59     emptyPrimitive: signatureString in: universe = (
60         ^ self new initializeEmpty: (universe symbolFor: signatureString)
61           in: universe
62     )
63 )
64
```

SomSom/src/vmobjects/SString.som

```
1 SString = SAbstractObject (
2   | string |
3
4   initializeWith: aString = (
5     string := aString.
6   )
7
8   string = ( ^ string )
9
10  somClassIn: universe = (
11    ^ universe stringClass
12  )
13
14  "For using in debugging tools such as the Diassembler"
15  debugString = ( ^ 'SString(' + string + ')' )
16
17  ----
18
19  new: aString = (
20    ^ self new initializeWith: aString
21  )
22 )
23
```

```
1 SSymbol = SString (
2   | numSignatureArguments |
3
4   initializeWith: aString = (
5     super initializeWith: aString.
6     numSignatureArguments := self determineNumberOfArguments
7   )
8
9   determineNumberOfArguments = (
10    | numColons |
11    self isBinarySignature ifTrue: [ ^ 2 ].
12
13    numColons := 0.
14
15    1 to: string length do: [:i |
16      ':' = (string charAt: i) ifTrue: [
17        numColons := numColons + 1 ] ].
18    ^ numColons + 1
19  )
20
21  isBinarySignature = (
22    1 to: string length do: [:i |
23      (Lexer isOperator: (string charAt: i))
24        ifFalse: [ ^ false ] ].
25    ^ true
26  )
27
28  numberOfSignatureArguments = (
29    ^ numSignatureArguments
30  )
31
32  somClassIn: universe = (
33    ^ universe symbolClass
34  )
35
36  "For using in debugging tools such as the Diassembler"
37  debugString = ( ^ 'SSymbol(' + string + ')' )
38
39  ----
40
41  new: aString = (
42    ^ self new initializeWith: aString
43  )
44 )
45
```


SomSom/tests/BasicInterpreterTests.som

```
1 BasicInterpreterTests = TestCase (
2   | testClass testSel
3   expectedResult resultType |
4
5   testBasicInterpreter = (
6     | arr |
7     arr := BasicInterpreterTests nextTest.
8
9     testClass := arr at: 1.
10    testSel := arr at: 2.
11    expectedResult := arr at: 3.
12    resultType := arr at: 4.
13
14    testSelector := ' ' + testClass + '>>#' + testSel.
15
16    self doBasicInterpreterBehavior.
17  )
18
19  doBasicInterpreterBehavior = (
20    | u actualResult |
21    u := Universe new: true.
22    u setupClassPath: 'Smalltalk:TestSuite/BasicInterpreterTests'.
23
24    actualResult := u interpret: testClass with: testSel.
25
26    self assertExpectedEqualsSOMValue: actualResult.
27  )
28
29  assertExpectedEqualsSOMValue: actualResult = (
30    resultType ~= actualResult class ifTrue: [
31      self signalFailure: 'Unexpected result type: ' + actualResult
32      debugString.
33      ^ self ].
34
35    resultType == SInteger ifTrue: [
36      self assert: expectedResult equals: actualResult integer.
37      ^ self ].
38
39    resultType == SDouble ifTrue: [
40      "TODO: allow for small errors/inaccuracies"
41      self assert: expectedResult equals: actualResult double.
42      ^ self ].
43
44    resultType == SClass ifTrue: [
45      self assert: expectedResult equals: actualResult name string.
46      ^ self ].
47
48    resultType == SSymbol ifTrue: [
49      self assert: expectedResult equals: actualResult string.
50      ^ self ].
51
52    self signalFailure: 'resultType not currently supported: ' +
53    resultType name string
54  )
55
56  ----
57  | basicTests next |
```

```

57
58 tests = (
59   | tests |
60   next := 1.
61   basicTests := Vector new.
62   tests := Vector new.
63
64   self setupBasicTest.
65   self setupBasicTest2.
66
67   basicTests size timesRepeat: [
68     tests append: (self for: #testBasicInterpreter) ].
69   ^ tests
70 )
71
72 nextTest = (
73   | test |
74   test := basicTests at: next.
75   next := next + 1.
76   ^ test
77 )
78
79 setupBasicTest = (
80   self c: 'MethodCall' t: 'test' e: 42 t: SInteger.
81   self c: 'MethodCall' t: 'test2' e: 42 t: SInteger.
82
83   self c: 'NonLocalReturn' t: 'test1' e: 42 t: SInteger.
84   self c: 'NonLocalReturn' t: 'test2' e: 43 t: SInteger.
85   self c: 'NonLocalReturn' t: 'test3' e: 3 t: SInteger.
86   self c: 'NonLocalReturn' t: 'test4' e: 42 t: SInteger.
87   self c: 'NonLocalReturn' t: 'test5' e: 22 t: SInteger.
88
89   self c: 'Blocks' t: 'testArg1' e: 42 t: SInteger.
90   self c: 'Blocks' t: 'testArg2' e: 77 t: SInteger.
91   self c: 'Blocks' t: 'testArgAndLocal' e: 8 t: SInteger.
92   self c: 'Blocks' t: 'testArgAndContext' e: 8 t: SInteger.
93   self c: 'Blocks' t: 'testEmptyZeroArg' e: 1 t: SInteger.
94   self c: 'Blocks' t: 'testEmptyOneArg' e: 1 t: SInteger.
95   self c: 'Blocks' t: 'testEmptyTwoArg' e: 1 t: SInteger.
96
97   self c: 'Return' t: 'testReturnSelf' e: 'Return' t: SClass.
98   self c: 'Return' t: 'testReturnSelfImplicitly' e: 'Return' t: SClass.
99   self c: 'Return' t: 'testNoReturnReturnsSelf' e: 'Return' t: SClass.
100  self c: 'Return' t: 'testBlockReturnsImplicitlyLastValue' e: 4 t:
SInteger.
101
102  self c: 'IfTrueIfFalse' t: 'test' e: 42 t: SInteger.
103  self c: 'IfTrueIfFalse' t: 'test2' e: 33 t: SInteger.
104  self c: 'IfTrueIfFalse' t: 'test3' e: 4 t: SInteger.
105
106  self c: 'IfTrueIfFalse' t: 'testIfTrueTrueResult' e: 'Integer' t:
SClass.
107  self c: 'IfTrueIfFalse' t: 'testIfTrueFalseResult' e: 'Nil' t:
SClass.
108  self c: 'IfTrueIfFalse' t: 'testIfFalseTrueResult' e: 'Nil' t:
SClass.
109  self c: 'IfTrueIfFalse' t: 'testIfFalseFalseResult' e: 'Integer' t:
SClass.
110 )
111
112 setupBasicTest2 = (

```

```

113     self c: 'CompilerSimplification' t: 'testReturnConstantSymbol' e:
'constant' t: SSymbol.
114     self c: 'CompilerSimplification' t: 'testReturnConstantInt' e: 42 t:
SInteger.
115     self c: 'CompilerSimplification' t: 'testReturnSelf' e:
'CompilerSimplification' t: SClass.
116     self c: 'CompilerSimplification' t: 'testReturnSelfImplicitly' e:
'CompilerSimplification' t: SClass.
117
118     self c: 'CompilerSimplification' t: 'testReturnArgumentN' e: 55 t:
SInteger.
119     self c: 'CompilerSimplification' t: 'testReturnArgumentA' e: 44 t:
SInteger.
120     self c: 'CompilerSimplification' t: 'testSetField' e: 'foo'
t: SSymbol.
121     self c: 'CompilerSimplification' t: 'testGetField' e: 40 t:
SInteger.
122
123     self c: 'Hash' t: 'testHash' e: 444 t: SInteger.
124
125     self c: 'Arrays' t: 'testEmptyToInts' e: 3 t: SInteger.
126     self c: 'Arrays' t: 'testPutAllInt' e: 5 t: SInteger.
127     self c: 'Arrays' t: 'testPutAllNil' e: 'Nil' t: SClass.
128     self c: 'Arrays' t: 'testPutAllBlock' e: 3 t: SInteger.
129     self c: 'Arrays' t: 'testNewWithAll' e: 1 t: SInteger.
130
131     self c: 'BlockInlining' t: 'testNoInlining' e: 1 t: SInteger.
132     self c: 'BlockInlining' t: 'testOneLevelInlining' e: 1 t: SInteger.
133     self c: 'BlockInlining' t:
'testOneLevelInliningWithLocalShadowTrue' e: 2 t: SInteger.
134     self c: 'BlockInlining' t:
'testOneLevelInliningWithLocalShadowFalse' e: 1 t: SInteger.
135
136     self c: 'BlockInlining' t: 'testShadowDoesntStoreWrongLocal' e: 33
t: SInteger.
137     self c: 'BlockInlining' t: 'testShadowDoesntReadUnrelated' e:
'Nil' t: SClass.
138
139     self c: 'BlockInlining' t: 'testBlockNestedInIfTrue' e: 2 t:
SInteger.
140     self c: 'BlockInlining' t: 'testBlockNestedInIfFalse' e: 42 t:
SInteger.
141
142     self c: 'BlockInlining' t: 'testStackDisciplineTrue' e: 1 t:
SInteger.
143     self c: 'BlockInlining' t: 'testStackDisciplineFalse' e: 2 t:
SInteger.
144
145     self c: 'BlockInlining' t: 'testDeepNestedInlinedIfTrue' e: 3 t:
SInteger.
146     self c: 'BlockInlining' t: 'testDeepNestedInlinedIfFalse' e: 42 t:
SInteger.
147
148     self c: 'BlockInlining' t: 'testDeepNestedBlocksInInlinedIfTrue'
e: 5 t: SInteger.
149     self c: 'BlockInlining' t: 'testDeepNestedBlocksInInlinedIfFalse'
e: 43 t: SInteger.
150
151     self c: 'BlockInlining' t: 'testDeepDeepNestedTrue' e: 9 t:
SInteger.
152     self c: 'BlockInlining' t: 'testDeepDeepNestedFalse' e: 43 t:

```

```

SInteger.
153
154     self c: 'BlockInlining' t: 'testToDoNestDoNestIfTrue' e: 2 t:
SInteger.
155
156     self c: 'NonLocalVars' t: 'testWriteDifferentTypes' e: 3.75 t:
SDouble.
157
158     "self c: 'ObjectCreation' t: 'test' e: 1000000 t: SInteger."
159
160     self c: 'Regressions' t: 'testSymbolEquality' e: 1 t:
SInteger.
161     self c: 'Regressions' t: 'testSymbolReferenceEquality' e: 1 t:
SInteger.
162     self c: 'Regressions' t: 'testUninitializedLocal' e: 1 t:
SInteger.
163     self c: 'Regressions' t: 'testUninitializedLocalInBlock' e: 1 t:
SInteger.
164
165     self c: 'BinaryOperation' t: 'test' e: 3 + 8 t: SInteger.
166
167     self c: 'NumberOfTests' t: 'numberOfTests' e: 65 t: SInteger.
168 )
169
170 c: className t: testName e: value t: resultClass = (
171 | arr |
172     arr := Array new: 4.
173     arr at: 1 put: className.
174     arr at: 2 put: testName.
175     arr at: 3 put: value.
176     arr at: 4 put: resultClass.
177
178     basicTests append: arr.
179 )
180 )
181

```

```

1 FrameTests = TestCase (
2   | u a b c d e f g h |
3
4   initialize = (
5     u := Universe new: true.
6     u setupClassPath: 'Smalltalk:TestSuite/BasicInterpreterTests'.
7     u initializeObjectSystem.
8
9     a := SSymbol new: 'a'.
10    b := SSymbol new: 'b'.
11    c := SSymbol new: 'c'.
12    d := SSymbol new: 'd'.
13    e := SSymbol new: 'e'.
14    g := SSymbol new: 'g'.
15    h := SSymbol new: 'h'.
16  )
17
18  method: name numLocals: numLocals = (
19    | sym classz method |
20    sym := SSymbol new: name.
21    classz := SClass new: u.
22    classz name: (SSymbol new: 'Holder').
23    method := SMethod new: sym bc: #() literals: #() numLocals: numLocals
maxStack: 4.
24    method holder: classz.
25    ^ method
26  )
27
28  testPushPop = (
29    | f length m |
30    m := self method: 'testPushPop' numLocals: 0.
31    length := 4 + m numberOfArguments + m numberOfLocals.
32    f := Frame new: u nilObject previous: nil context: nil method: m
maxStack: length.
33    f resetStackPointer.
34
35    f push: a.
36    self assert: 'a' equals: (f stackElement: 0) string.
37    f push: b.
38    self assert: 'b' equals: (f stackElement: 0) string.
39    f push: c.
40    self assert: 'c' equals: (f stackElement: 0) string.
41    f pop.
42    self assert: 'b' equals: (f stackElement: 0) string.
43    f pop.
44    self assert: 'a' equals: (f stackElement: 0) string.
45    f pop.
46  )
47
48  testArgsAndLocal = (
49    | f length m |
50    m := self method: 'testArgsAndLocal' numLocals: 2.
51    length := 4 + m numberOfArguments + m numberOfLocals.
52    f := Frame new: u nilObject previous: nil context: nil method: m
maxStack: length.
53    f resetStackPointer.
54
55    f argument: 1 put: a. "rcvr"

```

```

56     f argument: 2 put: b. "local 1"
57     f argument: 3 put: c. "local 2"
58     f push: d.
59     f push: e.
60     f push: g.
61     f push: h.
62
63     self assert: 'a' equals: (f argument: 1) string.
64     self assert: 'b' equals: (f local: 1) string.
65     self assert: 'c' equals: (f local: 2) string.
66     self assert: 'h' equals: (f stackElement: 0) string.
67     self assert: 'g' equals: (f stackElement: 1) string.
68     self assert: 'e' equals: (f stackElement: 2) string.
69     self assert: 'd' equals: (f stackElement: 3) string.
70     self assert: 'c' equals: (f stackElement: 4) string.
71 )
72
73 testCopyArgs = (
74     | f length m1 m2 copyF |
75     m1 := self method: 'sourceTest:copy:args:' numLocals: 2.
76     m2 := self method: 'targetTest:copy:args:' numLocals: 2.
77     length := 5 + m1 numberOfArguments + m1 numberOfLocals.
78     f := Frame new: u nilObject previous: nil context: nil method: m1
maxStack: length.
79     f resetStackPointer.
80
81     f local: 1 put: e. "local 1"
82     f local: 2 put: g. "local 2"
83     f push: h. "stack 1"
84     "stuff to be copied"
85     f push: a. "rcvr"
86     f push: b. "arg test:"
87     f push: c. "arg copy:"
88     f push: d. "arg args:"
89
90     copyF := Frame new: u nilObject previous: nil context: nil method: m2
maxStack: length.
91     copyF resetStackPointer.
92     copyF copyArgumentsFrom: f.
93
94     self assert: 'a' equals: (copyF argument: 1) string.
95     self assert: 'b' equals: (copyF argument: 2) string.
96     self assert: 'c' equals: (copyF argument: 3) string.
97     self assert: 'd' equals: (copyF argument: 4) string.
98     self assert: u nilObject is: (copyF local: 1).
99     self assert: u nilObject is: (copyF local: 2).
100    self assert: u nilObject is: (copyF stackElement: 0).
101
102    copyF push: e. "arg args:"
103
104    self assert: 'a' equals: (copyF argument: 1) string.
105    self assert: 'b' equals: (copyF argument: 2) string.
106    self assert: 'c' equals: (copyF argument: 3) string.
107    self assert: 'd' equals: (copyF argument: 4) string.
108    self assert: u nilObject is: (copyF local: 1).
109    self assert: u nilObject is: (copyF local: 2).
110    self assert: 'e' equals: (copyF stackElement: 0) string.
111
112 )
113 ----
114

```

```
115     new = (  
116         ^ super new initialize  
117     )  
118 )  
119
```

```

1 LexerTests = TestCase (
2
3   testEmptyClass = (
4     | 1 |
5     l := Lexer new: 'Foo = ()'.
6
7     self assert: #identifier is: 1 sym.
8     self assert: 'Foo' equals: 1 text.
9
10    self assert: #equal is: 1 sym.
11    self assert: '=' equals: 1 text.
12
13    self assert: #newTerm is: 1 sym.
14    self assert: '(' equals: 1 text.
15
16    self assert: #endTerm is: 1 sym.
17    self assert: ')' equals: 1 text.
18
19    self assert: #NONE is: 1 sym.
20  )
21
22  testKeywordSymbol = (
23    | 1 |
24    l := Lexer new: '#key:word:'.
25
26    self assert: #pound is: 1 sym.
27    self assert: '#' equals: 1 text.
28
29    self assert: #keywordSequence is: 1 sym.
30    self assert: 'key:word:' equals: 1 text.
31
32    self assert: #NONE is: 1 sym.
33  )
34
35  testIntMessage = (
36    | 1 |
37    l := Lexer new: '314 println'.
38
39    self assert: #integer is: 1 sym.
40    self assert: '314' equals: 1 text.
41
42    self assert: #identifier is: 1 sym.
43    self assert: 'println' equals: 1 text.
44
45    self assert: #NONE is: 1 sym.
46  )
47
48  testAssignDouble = (
49    | 1 |
50    l := Lexer new: 'var := 3.14.'.
51
52    self assert: #identifier is: 1 sym.
53    self assert: 'var' equals: 1 text.
54
55    self assert: #assign is: 1 sym.
56    self assert: ':= ' equals: 1 text.
57
58    self assert: #double is: 1 sym.

```



```

59     self assert: '3.14' equals: 1 text.
60
61     self assert: #period is: 1 sym.
62     self assert: '.' equals: 1 text.
63
64     self assert: #NONE is: 1 sym.
65 )
66
67 testString = (
68     | 1 str |
69     str := '\some string with new\nline\''.
70     l := Lexer new: str.
71
72     self assert: #string is: 1 sym.
73     self assert: 'some string with new\nline' equals: 1 text.
74
75     self assert: #NONE is: 1 sym.
76 )
77
78 testEscapeChars = (
79     | 1 str |
80     str := '\\n\\0\\t\''.
81     l := Lexer new: str.
82
83     self assert: #string is: 1 sym.
84     self assert: '\\n\\0\\t' equals: 1 text.
85     self assert: 3 equals: 1 text length.
86
87     self assert: #NONE is: 1 sym.
88 )
89
90 testPrimitiveMethod = (
91     | 1 |
92     l := Lexer new: ' foo: bar = primitive '.
93
94     self assert: #keyword is: 1 sym.
95     self assert: 'foo:' equals: 1 text.
96
97     self assert: #identifier is: 1 sym.
98     self assert: 'bar' equals: 1 text.
99
100     self assert: #equal is: 1 sym.
101     self assert: '=' equals: 1 text.
102
103     self assert: #primitive is: 1 sym.
104     self assert: 'primitive' equals: 1 text.
105 )
106
107 testMathBlock = (
108     | 1 syms |
109     l := Lexer new: '[ 0 ~ 1 & 2 | 3 * 4 / 5 \\ 6 + 7 - 8 > 9 < 10 , 11 @
110 12 % 13]'.
111     syms := (#not #and #or #star #div #mod #plus #minus #more #less
112 #comma #at #per).
113
114     self assert: #newBlock is: 1 sym.
115     self assert: '[' equals: 1 text.
116
117     0 to: 12 do: [:i |
118         self assert: #integer is: 1 sym.

```

```

118         self assert: i asString equals: 1 text.
119         self assert: (syms at: i + 1) is: 1 sym ].
120
121     self assert: #integer is: 1 sym.
122     self assert: '13' equals: 1 text.
123
124     self assert: #endBlock is: 1 sym.
125     self assert: ']' equals: 1 text.
126 )
127
128 testOperatorSequence = (
129     | 1 |
130     1 := Lexer new: '1 ----- ---- --> 2'.
131
132     self assert: #integer is: 1 sym.
133     self assert: '1' equals: 1 text.
134
135     self assert: #separator is: 1 sym.
136     self assert: '-----' equals: 1 text.
137
138     self assert: #separator is: 1 sym.
139     self assert: '----' equals: 1 text.
140
141     self assert: #operatorSequence is: 1 sym.
142     self assert: '-->' equals: 1 text.
143
144     self assert: #integer is: 1 sym.
145     self assert: '2' equals: 1 text.
146 )
147
148 testBlock = (
149     | 1 |
150     1 := Lexer new: '[:x | ^ 1]'.
151
152     self assert: #newBlock is: 1 sym.
153     self assert: '[' equals: 1 text.
154
155     self assert: #colon is: 1 sym.
156     self assert: ':' equals: 1 text.
157
158     self assert: #identifier is: 1 sym.
159     self assert: 'x' equals: 1 text.
160
161     self assert: #or is: 1 sym.
162     self assert: '|' equals: 1 text.
163
164     self assert: #exit is: 1 sym.
165     self assert: '^' equals: 1 text.
166
167     self assert: #integer is: 1 sym.
168     self assert: '1' equals: 1 text.
169
170     self assert: #endBlock is: 1 sym.
171     self assert: ']' equals: 1 text.
172 )
173
174
175 " Colon, Exit "
176 )

```

```

1 ParserTests = TestCase (
2   | universe |
3   testEmptyClass = (
4     | cgenic parser u |
5     u := self initUniverse.
6     parser := Parser newWith: 'Foo = ()' for: 'Foo.som' in: u.
7     cgenic := parser classdef.
8   )
9
10  testSpaceBeforeEmptyClass = (
11    | cgenic parser u |
12    u := self initUniverse.
13    parser := Parser newWith: '
14      Foo = ()' for: 'Foo.som' in: u.
15    cgenic := parser classdef.
16  )
17
18  testCommentBeforeEmptyClass = (
19    | cgenic parser u |
20    u := self initUniverse.
21    parser := Parser newWith: '
22      "This is a Foo Class"
23      Foo = ()' for: 'Foo.som' in: u.
24    cgenic := parser classdef.
25  )
26
27  testEmptyWithNilSuperClass = (
28    | cgenic parser u |
29    u := self initUniverse.
30    parser := Parser newWith: 'Foo = nil ()' for: 'Foo.som' in: u.
31    cgenic := parser classdef.
32  )
33
34  testEmptyWithObjectSuperClass = (
35    | cgenic parser u |
36    u := self initUniverse.
37    parser := Parser newWith: 'Foo = Object ()' for: 'Foo.som' in: u.
38    cgenic := parser classdef.
39  )
40
41  parseAndCaptureError: parser = (
42    parser errorHandler: [:msg | ^ msg ].
43    ^ parser classdef.
44  )
45
46  testEmptyClassMissingEqual = (
47    | cgenic parser u |
48    u := self initUniverse.
49    parser := ParserWithError newWith: 'Foo ()' for: 'Foo.som' in: u.
50    cgenic := self parseAndCaptureError: parser.
51    self assert: (cgenic beginsWith: 'Parsing of Foo.som failed, expected
equal but found newTerm')
52  )
53
54  testEmptyClassWithComment = (
55    | cgenic parser u |
56    u := self initUniverse.
57    parser := Parser newWith: 'Foo = ( "comment" )' for: 'Foo.som' in: u.

```

```

58     cgener := parser classdef.
59 )
60
61 testClassWithFields = (
62   | cgener parser u |
63   u := self initUniverse.
64   parser := Parser newWith: 'Foo = (|a b c|)' for: 'Foo.som' in: u.
65   cgener := parser classdef.
66 )
67
68 testClassWithUnaryMethod = (
69   | cgener parser u |
70   u := self initUniverse.
71   parser := Parser newWith: 'Foo = ( m = () )' for: 'Foo.som' in: u.
72   cgener := parser classdef.
73 )
74
75 testClassWithBinaryMethod = (
76   | cgener parser u |
77   u := self initUniverse.
78   parser := Parser newWith: 'Foo = ( * o = () )' for: 'Foo.som' in: u.
79   cgener := parser classdef.
80 )
81
82 testClassWithKeywordMethod = (
83   | cgener parser u |
84   u := self initUniverse.
85   parser := Parser newWith: 'Foo = ( m: o = () )' for: 'Foo.som' in: u.
86   cgener := parser classdef.
87 )
88
89 testClassWithKeywordPrimitive = (
90   | cgener parser u |
91   u := self initUniverse.
92   parser := Parser newWith: 'Foo = ( m: o = primitive )' for: 'Foo.som'
in: u.
93   cgener := parser classdef.
94 )
95
96 testClassWithVariousMethods = (
97   | cgener parser u |
98   u := self initUniverse.
99   parser := Parser newWith: '
100   ClassWithVariousMethods = (
101     a: o = ( | s n v | )
102     b: o = ( ^ 1 )
103     bn: o = ( ^ -1 )
104     c: o = ( ^ 2.2 )
105     cn: o = ( ^ -2.2 )
106     d: o = ( ^ \'ss\' )
107     e: o = ( ^ #sym )
108     f: o = ( | a | a := a := 2 )
109     g: o = ( o )
110     h: o = ( self foo )
111     i: o = ( super foo )
112     j: o = ( (1) )
113     k: o = ( [1] )
114     l: o = ( 1 foo: 4 )
115     m: o = ( 1 + 4 )
116     n: o = ( 1 ++ 2 )
117     o: o = ( #(2 3 4 5) )

```

```

118     )' for: 'ClassWithVariousMethods.som' in: u.
119     cgener := parser classdef.
120 )
121
122 testSmalltalkFolder = (
123     | files |
124     files := #(
125         'Array.som'
126         'Block.som'
127         'Block1.som'
128         'Block2.som'
129         'Block3.som'
130         'Boolean.som'
131         'Class.som'
132         'Dictionary.som'
133         'Double.som'
134         'False.som'
135         'HashEntry.som'
136         'Hashtable.som'
137         'Integer.som'
138         'Metaclass.som'
139         'Method.som'
140         'Nil.som'
141         'Object.som'
142         'Pair.som'
143         'Primitive.som'
144         'Set.som'
145         'String.som'
146         'Symbol.som'
147         'System.som'
148         'True.som'
149         'Vector.som' ).
150
151     files do: [:f |
152         | cgener parser u |
153         u := self initUniverse.
154         parser := Parser load: 'Smalltalk/' + f in: u.
155         self deny: parser isNil.
156         cgener := parser classdef ].
157 )
158
159 testTestSuiteFolder = (
160     | files |
161     files := #(
162         'ArrayTest.som'
163         'BlockTest.som'
164         'ClassA.som'
165         'ClassB.som'
166         'ClassC.som'
167         'ClassLoadingTest.som'
168         'ClassStructureTest.som'
169         'ClosureTest.som'
170         'CoercionTest.som'
171         'CompilerReturnTest.som'
172         'DoesNotUnderstandMessage.som'
173         'DoesNotUnderstandTest.som'
174         'DoubleTest.som'
175         'EmptyTest.som'
176         'GlobalTest.som'
177         'HashTest.som'
178         'IntegerTest.som'

```

```

179     'PreliminaryTest.som'
180     'ReflectionTest.som'
181     'SelfBlockTest.som'
182     'SetTest.som'
183     'SpecialSelectorsTest.som'
184     'StringTest.som'
185     'SuperTest.som'
186     'SuperTestSuperClass.som'
187     'SymbolTest.som'
188     'SystemTest.som'
189     'TestCase.som'
190     'TestHarness.som'
191     'TestRunner.som'
192     'VectorTest.som'
193 ).
194
195 files do: [:f |
196     | cgen parser u |
197     u := self initUniverse.
198     parser := Parser load: 'TestSuite/' + f in: u.
199     self deny: parser isNil.
200     cgen := parser classdef ].
201 )
202
203 initUniverse = (
204     | u |
205     universe ifNil: [
206         u := Universe new.
207         u setupClassPath: 'Smalltalk:TestSuite'.
208         u initializeObjectSystem.
209         universe := u ].
210     ^ universe
211 )
212 )
213

```

SomSom/tests/ParserWithError.som

```
1 ParserWithError = Parser (  
2   | errorHandler |  
3  
4   errorHandler: aBlock = (  
5     errorHandler := aBlock  
6   )  
7  
8   error: message = (  
9     errorHandler value: message  
10  )  
11 )  
12
```

SomSom/tests/SomSomTests.som

```
1 SomSomTests = TestHarness (  
2   tests = (  
3     ^ EmptyTest,  
4       FrameTests,  
5       LexerTests,  
6       ParserTests,  
7       BasicInterpreterTests,  
8       SomTests  
9   )  
10 )  
11
```



```

1 SomTests = TestCase (
2
3   testArray          = ( self doTest: 'Array' )
4   testBlock          = ( self doTest: 'Block' )
5   testBoolean        = ( self doTest: 'Boolean' )
6   testClassLoading   = ( self doTest: 'ClassLoading' )
7   testClassStructure = ( self doTest: 'ClassStructure' )
8
9   testClosure         = ( self doTest: 'Closure' )
10  testCoercion        = ( self doTest: 'Coercion' )
11  testCompilerReturn  = ( self doTest: 'CompilerReturn' )
12  testDictionary      = ( self doTest: 'Dictionary' )
13  testDoesNotUnderstand = ( self doTest: 'DoesNotUnderstand' )
14  testDouble          = ( self doTest: 'Double' )
15
16  testEmpty           = ( self doTest: 'Empty' )
17  testGlobal          = ( self doTest: 'Global' )
18  testHash            = ( self doTest: 'Hash' )
19  testInteger         = ( self doTest: 'Integer' )
20
21  testPreliminary     = ( self doTest: 'Preliminary' )
22  testReflection      = ( self doTest: 'Reflection' )
23  testSelfBlock       = ( self doTest: 'SelfBlock' )
24  testSpecialSelectorsTest = ( self doTest: 'SpecialSelectorsTest' )
25  testSuper           = ( self doTest: 'Super' )
26
27  testSet             = ( self doTest: 'Set' )
28  testString          = ( self doTest: 'String' )
29  testSymbol          = ( self doTest: 'Symbol' )
30  testSystem          = ( self doTest: 'System' )
31  testVector          = ( self doTest: 'Vector' )
32
33  doTest: testName = (
34    | args u exitCode |
35    args := Array new: 4.
36    args at: 1 put: '-cp'.
37    args at: 2 put: 'Smalltalk'.
38    args at: 3 put: 'TestSuite/TestHarness.som'.
39    args at: 4 put: testName.
40
41    u := Universe new: true.
42
43    exitCode := u interpret: args.
44
45    self assert: 0 equals: exitCode.
46  )
47 )
48

```

TestSuite/ArrayTest.som

```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 ArrayTest = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Array new: 3.  
33     a at: 1 put: 'hello'.  
34     a at: 2 put: #world.  
35     a at: 3 put: 23.  
36   )  
37  
38   testAt = (  
39     self assert: 'hello' equals: (a at: 1).  
40     self assert: #world equals: (a at: 2).  
41     self assert: 23 equals:      (a at: 3).  
42   )  
43  
44   testLength = (  
45     self assert: 3 equals: a length  
46   )  
47  
48   testPutAllBlock = (  
49     | arr i |  
50     arr := Array new: 10.  
51  
52     i := 0.  
53     arr putAll: [ i := i + 1. i ].
```

```

54
55     1 to: 10 do: [:j |
56         self assert: j equals: (arr at: j) ].
57 )
58
59 testPutAllInt = (
60     | arr i |
61     arr := Array new: 10.
62     arr putAll: 0.
63
64     1 to: 10 do: [:j |
65         self assert: 0 equals: (arr at: j) ].
66 )
67
68 testPutAllSym = (
69     | arr i |
70     arr := Array new: 10.
71     arr putAll: #sym.
72
73     1 to: 10 do: [:j |
74         self assert: #sym is: (arr at: j) ].
75 )
76
77 testFirst = (
78     self assert: 'hello' equals: a first.
79 )
80
81 testLast = (
82     self assert: 23 equals: a last.
83 )
84
85 testContains = (
86     self assert: (a contains: 23).
87     self deny: (a contains: #notInThere).
88 )
89
90 testDo = (
91     | j |
92     j := 1.
93
94     a do: [:i |
95         self assert: (a at: j) is: i.
96         j := j + 1.
97     ]
98 )
99
100 testDoIndexes = (
101     | i arr |
102     i := 1.
103     a doIndexes: [:j |
104         self assert: i equals: j.
105         i := i + 1.
106     ].
107     self assert: 4 equals: i.
108
109     arr := Array new: 10.
110     i := 1.
111     arr doIndexes: [:j |
112         self assert: i equals: j.
113         i := i + 1.
114     ].

```

```

115     self assert: 11 equals: i.
116 )
117
118 testFromToDo = (
119     | arr i |
120     a from: 2 to: 2 do: [:e | self assert: #world is: e ].
121
122     i := 0.
123     arr := Array new: 10 withAll: [ i := i + 1. i ].
124
125     i := 3.
126     arr from: 3 to: 7 do: [:e |
127         self assert: i equals: e.
128         i := i + 1 ].
129
130     self assert: 8 equals: i.
131 )
132
133 testSumAndAverage = (
134     | arr |
135     arr := Array new: 3.
136     1 to: 3 do: [ :i | arr at: i put: i ].
137
138     self assert: 6 equals: arr sum.
139     self assert: 2 equals: arr average.
140 )
141
142 testCopyFrom = (
143     | arr b |
144     arr := Array new: 5.
145     1 to: 5 do: [ :i | arr at: i put: i ].
146
147     b := arr copyFrom: 2 to: 4.
148     self assert: 2 equals: (b at: 1).
149     self assert: 3 equals: (b at: 2).
150     self assert: 4 equals: (b at: 3).
151
152     b := arr copyFrom: 3.
153     self assert: 3 equals: (b at: 1).
154     self assert: 4 equals: (b at: 2).
155     self assert: 5 equals: (b at: 3).
156 )
157
158 testCopy = (
159     | arr |
160     arr := a copy.
161     self assert: 3 equals: arr length.
162     self assert: 'hello' equals: (arr at: 1).
163     self assert: #world equals: (arr at: 2).
164     self assert: 23 equals: (arr at: 3).
165 )
166
167 testReplaceFrom = (
168     | arr1 arr2 i |
169     arr1 := Array new: 10 withAll: 0.
170
171     i := 0.
172     arr2 := Array new: 10 withAll: [ i := i + 1. i ].
173
174     arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.
175

```

```

176     i := 1.
177     3 to: 7 do: [:j |
178         self assert: i equals: (arr1 at: j).
179         i := i + 1.
180     ]
181 )
182
183 testExtendedWith = (
184     | arr newArr |
185     arr := Array new: 0.
186     newArr := arr extendedWith: 33.
187
188     self assert: 1 equals: newArr length.
189     self assert: 0 equals: arr length.
190     self assert: 33 equals: (newArr at: 1).
191
192     self testAt. "confirm a is correct"
193     self testLength.
194
195     newArr := a extendedWith: 44.
196
197     self testAt. "confirm a is correct"
198     self testLength.
199
200     self assert: 4 equals: newArr length.
201     self assert: 0 equals: arr length.
202     self assert: 44 equals: (newArr at: 4).
203 )
204
205 testPrependedWith = (
206     | arr newArr |
207     arr := Array new: 0.
208     newArr := arr prependedWith: 33.
209
210     self assert: 1 equals: newArr length.
211     self assert: 0 equals: arr length.
212     self assert: 33 equals: (newArr at: 1).
213
214     self testAt. "confirm a is correct"
215     self testLength.
216
217     newArr := a prependedWith: 44.
218
219     self testAt. "confirm a is correct"
220     self testLength.
221
222     self assert: 4 equals: newArr length.
223     self assert: 0 equals: arr length.
224     self assert: 44 equals: (newArr at: 1).
225 )
226
227 testIndexOf = (
228     | arr |
229     arr := Array new: 6.
230     arr at: 1 put: #one.
231     arr at: 2 put: #two.
232     arr at: 3 put: #three.
233     arr at: 4 put: #four.
234     arr at: 5 put: #five.
235     arr at: 6 put: #one.
236

```

```

237     self assert: 2 equals: (arr indexOf: #two).
238     self assert: 4 equals: (arr indexOf: #four).
239     self assert: 5 equals: (arr indexOf: #five).
240
241     self assert: nil equals: (arr indexOf: #notIncluded).
242
243     self assert: 1 equals: (arr indexOf: #one).
244 )
245
246 testLastIndexOf = (
247     | arr |
248     arr := Array new: 6.
249     arr at: 1 put: #one.
250     arr at: 2 put: #two.
251     arr at: 3 put: #three.
252     arr at: 4 put: #four.
253     arr at: 5 put: #five.
254     arr at: 6 put: #one.
255
256     self assert: 2 equals: (arr lastIndexOf: #two).
257     self assert: 4 equals: (arr lastIndexOf: #four).
258     self assert: 5 equals: (arr lastIndexOf: #five).
259
260     self assert: nil equals: (arr indexOf: #notIncluded).
261
262     self assert: 6 equals: (arr lastIndexOf: #one).
263 )
264
265 testCollect = (
266     | arr i col |
267     i := 0.
268     arr := Array new: 10 withAll: [ i := i + 1. i ].
269     col := arr collect: [:e | e + 1 ].
270
271     self assert: 10 equals: col length.
272
273     1 to: 10 do: [:i |
274         self assert: i + 1 equals: (col at: i) ].
275 )
276
277 testInject = (
278     | arr result |
279     arr := Array new: 10 withAll: 1.
280
281     result := arr inject: 100 into: [:sum :e | sum + e ].
282
283     self assert: 110 equals: result.
284 )
285
286 testReject = (
287     | arr i result |
288     i := 0.
289     arr := Array new: 10 withAll: [ i := i + 1. i ].
290
291     result := arr reject: [:e | e % 2 = 0 ].
292
293     self assert: 5 equals: result size.
294     self assert: 1 equals: (result at: 1).
295     self assert: 3 equals: (result at: 2).
296     self assert: 5 equals: (result at: 3).
297     self assert: 7 equals: (result at: 4).

```

```

298     self assert: 9 equals: (result at: 5).
299 )
300
301 testRejectEmpty = (
302     | result |
303     result := (Array new: 0) reject: [:e | false ].
304     self assert: 0 equals: result size.
305
306     result := (Array new: 0) reject: [:e | true ].
307     self assert: 0 equals: result size.
308
309     result := (Array new: 10 withAll: 4) reject: [:e | true ].
310     self assert: 0 equals: result size.
311
312     result := (Array new: 10 withAll: 4) reject: [:e | true ].
313     self assert: 0 equals: result size.
314 )
315
316 testDontRejectAny = (
317     | result |
318     result := (Array new: 10 withAll: 0) reject: [:e | false ].
319     self assert: 10 equals: result size.
320
321     self assert: 0 equals: (result at: 5).
322 )
323
324 testSelect = (
325     | arr i result |
326     i := 0.
327     arr := Array new: 10 withAll: [ i := i + 1. i ].
328
329     result := arr select: [:e | e % 2 = 0 ].
330
331     self assert: 5 equals: result size.
332     self assert: 2 equals: (result at: 1).
333     self assert: 4 equals: (result at: 2).
334     self assert: 6 equals: (result at: 3).
335     self assert: 8 equals: (result at: 4).
336     self assert: 10 equals: (result at: 5).
337 )
338
339 testSelectEmpty = (
340     | result |
341     result := (Array new: 0) select: [:e | false ].
342     self assert: 0 equals: result size.
343
344     result := (Array new: 0) select: [:e | true ].
345     self assert: 0 equals: result size.
346
347     result := (Array new: 10 withAll: 4) select: [:e | false ].
348     self assert: 0 equals: result size.
349
350     result := (Array new: 10 withAll: 4) select: [:e | false ].
351     self assert: 0 equals: result size.
352 )
353
354 testSelectAll = (
355     | result |
356     result := (Array new: 10 withAll: 0) select: [:e | true ].
357     self assert: 10 equals: result size.
358

```

```

359     self assert: 0 equals: (result at: 5).
360 )
361
362 testUnion = (
363     | result |
364     result := a union: a.
365     self assert: 3 equals: result size.
366
367     self assert: (result contains: #world).
368     self assert: (result contains: 23).
369
370     result := a union: #(21 22 23 #world).
371
372     self assert: 5 equals: result size.
373     self assert: (result contains: 21).
374     self assert: (result contains: 22).
375     self assert: (result contains: 23).
376     self assert: (result contains: #world).
377 )
378
379 testNewWithAll = (
380     | arr |
381     arr := Array new: 5 withAll: [1].
382     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
383
384     arr := Array new: 5 withAll: 1.
385     1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
386 )
387
388 testNewWithAllIntAndObjects = (
389     | arr o |
390     arr := Array new: 5 withAll: 5.
391     self assert: 5 equals: (arr at: 3).
392     arr at: 3 put: nil.
393     self assert: nil equals: (arr at: 3).
394
395     o := Object new.
396     arr at: 2 put: o.
397     self assert: o equals: (arr at: 2).
398 )
399
400 testLiteralArrays = (
401     -6VÆb 76W'Cç ,2f " ' Cç ' W V Ç3ç à
402     -6VÆb 76W'Cç ,2f " ' Cç " ' W V Ç3ç "à
403
404     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç ' W V Ç3ç Ó à
405     -6VÆb 76W'Cç ,2,Ó Ó# ã ' Cç " ' W V Ç3ç Ó# ã à
406 )
407
408 testJoin = (
409     | arr |
410     arr := Array new: 0.
411     self assert: nil is: (arr join: ', ').
412
413     arr := Array with: 1 with: 10 with: 100.
414     self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).
415
416     arr := Array with: 'a' with: 'b' with: 'c'.
417     self assert: 'a, b, c' equals: (arr join: ', ').
418 )
419 )

```



```

1 Arrays = (
2   ----
3
4   testEmptyToInts = (
5     | arr |
6     arr := Array new: 5.
7     (arr at: 1) ifNotNil: [self error: 'should be initialized to nil'].
8
9     1 to: 5 do: [:i |
10      arr at: i put: i.
11      (arr at: i) = i ifFalse: [self error: 'should be i'].
12    ].
13
14    (arr at: 1) = 1 ifFalse: [self error: 'should be 1'].
15    (arr at: 5) = 5 ifFalse: [self error: 'should be 1'].
16    ^ arr at: 3
17  )
18
19  testPutAllInt = (
20    | arr |
21    arr := Array new: 5.
22    arr putAll: 5.
23    ^ arr at: 3
24  )
25
26  testPutAllNil = (
27    | arr |
28    arr := Array new: 5.
29    (arr at: 4) ifNotNil: [self error: 'should be initialized to nil'].
30
31    arr putAll: 5.
32    (arr at: 4) = 5 ifFalse: [self error: 'should be set to 5'].
33
34    arr putAll: nil.
35
36    ^ (arr at: 3) class
37  )
38
39  testPutAllBlock = (
40    | arr b cnt |
41    cnt := 0.
42    b := [cnt := cnt + 1. cnt].
43    arr := Array new: 5.
44    arr putAll: b.
45
46    1 to: 5 do: [:i |
47      (arr at: i) = i ifFalse: [self error: 'block not properly
evaluated?']
48    ].
49
50    ^ arr at: 3
51  )
52
53  testNewWithAll = (
54    | arr |
55    arr := Array new: 5 withAll: [1].
56    1 to: 5 do: [:i | (arr at: i) = 1 ifFalse: [self error: 'wrong
result']]

```

```
57      ^ arr at: 3
58    )
59
60 )
```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 BinaryOperation = (  
24  
25     ----  
26  
27     test = (  
28         ^ (self foo: 1) + (self foo2: 2)  
29     )  
30  
31     foo: aNumber = (  
32         ^ 3  
33     )  
34  
35     foo2: aNumber = (  
36         ^ 8  
37     )  
38  
39 )  
40 )  
41
```

```

1 "
2 Copyright (c) 2015 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 BlockInlining = (
24     ----
25
26     testNoInlining = (
27         | a block |
28         a := 1.
29         block := [ a ].
30         ^ block value
31     )
32
33     testOneLevelInlining = (
34         | a |
35         a := 1.
36         ^ true ifTrue: [ a ] ifFalse: [ 42 ]
37     )
38
39     oneLevelInliningWithLocalShadow: bool = (
40         | a |
41         a := 1.
42         ^ bool
43             ifTrue: [ | a |
44                 a := 2.
45                 a
46             ] ifFalse: [ a "that's outer a" ]
47     )
48
49     testOneLevelInliningWithLocalShadowTrue = (
50         ^ self oneLevelInliningWithLocalShadow: true
51     )
52
53     testOneLevelInliningWithLocalShadowFalse = (

```

```

54     ^ self oneLevelInliningWithLocalShadow: false
55 )
56
57 testShadowDoesntStoreWrongLocal = (
58     | a |
59     a := 33.
60     true ifTrue: [
61         | a |
62         a := 2 ].
63     ^ a
64 )
65
66 testShadowDoesntReadUnrelated = (
67     | a |
68     a := 33.
69     true ifTrue: [
70         | a |
71         a := 2 ].
72
73     true ifTrue: [
74         | a |
75         ^ a class ].
76 )
77
78 deepNestedInlinedIf: bool = (
79     | block a block2 |
80     a := 1.
81     block := [ "not inlined"
82         a := a + 1.
83         block2 := [ "not inlined"
84             bool ifTrue: [ ^ a := a + 1.]
85             ifFalse:[ |a| a := 42. a ]
86         ].
87     block2 value
88     ].
89     ^ block value
90 )
91
92 testDeepNestedInlinedIfTrue = ( ^ self deepNestedInlinedIf: true )
93 testDeepNestedInlinedIfFalse = ( ^ self deepNestedInlinedIf: false )
94
95 blockNestedInIf: bool = (
96     | a |
97     a := 1.
98     bool ifTrue: [
99         | block |
100         block := [ a := a + 1 ].
101         block value
102     ] ifFalse: [
103         a := 42.
104     ].
105     ^ a
106 )
107
108 testBlockNestedInIfTrue = ( ^ self blockNestedInIf: true )
109 testBlockNestedInIfFalse = ( ^ self blockNestedInIf: false )
110
111 testStackDisciplineTrue = (
112     | result |
113     result := 0 max: (1 > 0 ifTrue: [1] ifFalse: [2]).
114     ^ result

```

```

115     )
116
117     testStackDisciplineFalse = (
118         | result |
119         result := 0 max: (1 < 0 ifTrue: [1] ifFalse: [2]).
120         ^ result
121     )
122
123     deepNestedBlocksInInlinedIf: bool = (
124         | block a block2 block3 |
125         a := 1.
126         block := [ "not inlined"
127             a := a + 1.
128             block2 := [ "not inlined"
129                 bool ifTrue: [ a := a + 1. "inlined"
130                     block3 := [ |block4|
131                         a := a + 1.
132                         block4 := [ "not inlined"
133                             a := a + 1.
134                             a
135                         ].
136                         block4 value
137                     ].
138                     block3 value
139                 ] ifFalse: [ |a block4| "inlined"
140                     a := 42.
141                     block4 := [ ^ a := a + 1 ]. "not inlined"
142                     block4 value
143                 ]
144             ].
145             block2 value
146         ].
147         ^ block value
148     )
149
150     testDeepNestedBlocksInInlinedIfTrue = ( ^ self
151     deepNestedBlocksInInlinedIf: true )
152
153     testDeepNestedBlocksInInlinedIfFalse = ( ^ self
154     deepNestedBlocksInInlinedIf: false )
155
156     deepDeepNested: bool = (
157         | block a block2 block3 |
158         a := 1.
159         block := [ "not inlined"
160             a := a + 1.
161             block2 := [ "not inlined"
162                 bool ifTrue: [ a := a + 1. "inlined"
163                     block3 := [ |block4|
164                         a := a + 1.
165                         block4 := [ "not inlined"
166                             a := a + 1.
167                             block := [ "not inlined"
168                                 a := a + 1.
169                                 block2 := [ "not inlined"
170                                     bool ifTrue: [ a := a + 1. "inlined"
171                                         block3 := [ |block4|
172                                             a := a + 1.
173                                             block4 := [ "not inlined"

```

```

174             a
175         ].
176         block4 value
177     ].
178     block3 value
179     ] ifFalse:[ |a block4| a := 42. "inlined"
180         block4 := [^ a := a + 1]. "not inlined"
181         block4 value
182     ]
183 ].
184 block2 value
185 ].
186 block value
187
188
189     ].
190     block4 value
191 ].
192 block3 value
193 ] ifFalse:[ |a block4| a := 42. "inlined"
194     block4 := [^ a := a + 1]. "not inlined"
195     block4 value
196 ]
197 ].
198 block2 value
199 ].
200 ^ block value
201 )
202
203 testDeepDeepNestedTrue = ( ^ self deepDeepNested: true )
204 testDeepDeepNestedFalse = ( ^ self deepDeepNested: false )
205
206 testToDoNestDoNestIfTrue = (
207     "from the bounce benchmark"
208     | balls bounces |
209     balls := Array new: 1 withAll: true.
210     bounces := 0.
211
212     1 to: 2 do: [ :i |
213         balls do: [ :ball |
214             ball ifTrue: [ bounces := bounces + 1 ] ] ].
215
216     ^ bounces
217 )
218 )
219

```



```

1 "
2 Copyright (c) 2001-2013 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 Blocks = (
24
25     ----
26
27     testArg1 = ( ^ [:a | a - 1] value: 43 )
28
29     testArg2 = ( ^ [:a :b | a * b ] value: 11 with: 7 )
30
31     testArgAndLocal = (
32         ^ ([:a |
33             | blockLocal |
34             blockLocal := 3.
35             a + blockLocal] value: 5)
36     )
37
38     testArgAndContext = (
39         | methodLocal |
40         ^ [:a |
41             methodLocal := 3.
42             a + methodLocal] value: 5
43     )
44
45     testEmptyZeroArg = (
46         [] value == nil ifTrue: [ ^ 1 ].
47         ^ 2
48     )
49
50     testEmptyOneArg = (
51         ([:x | ] value: 4) == nil ifTrue: [ ^ 1 ].
52         ^ 2
53     )
54

```

```
55     testEmptyTwoArg = (  
56         ([:x :y | ] value: 4 with: 5) == nil ifTrue: [ ^ 1 ].  
57         ^ 2  
58     )  
59 )  
60
```

```

1 "
2 Copyright (c) 2014 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 CompilerSimplification = (
24     ---
25     | aField |
26
27     testReturnConstantSymbol = ( ^ #constant )
28     testReturnConstantInt    = ( ^ 42 )
29
30     testReturnSelf           = ( ^ self )
31     testReturnSelfImplicitly = ( )
32
33     testReturnArgument: n      = ( ^ n )
34     testReturnArgument: n a: a = ( ^ a )
35
36     testReturnArgumentN = ( ^ self testReturnArgument: 55 )
37     testReturnArgumentA = ( ^ self testReturnArgument: 55 a: 44 )
38
39
40     setField: val = ( aField := val )
41     testSetField = (
42         aField := #bar.
43         self setField: #foo.
44         ^ aField
45     )
46
47     getField = ( ^ aField )
48     testGetField = (
49         aField := 40.
50         ^ self getField
51     )
52 )
53

```

```

1 "
2 Copyright (c) 2001-2013 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 Hash = (
24
25     ----
26
27     testHash = (
28         | ht string array t |
29
30         ht := Hashtable new.
31         ht isEmpty
32             ifFalse: [ 'New Hashtable not empty!'. ^ #notEmpty ].
33
34         ht at: 'a' put: 'b'.
35         (ht containsValue: 'b')
36             ifFalse: [ '1 not in Hashtable'. ^ 1 ].
37         ht isEmpty
38             ifTrue: [ 'Nonempty Hashtable empty!'. ^ #notEmpty ].
39         ((ht size) = 1)
40             ifFalse: [ 'Hashtable has wrong size!'. ^ #wrongSize ].
41
42         ht at: 'c' put: 'd'.
43         ((ht size) = 2)
44             ifFalse: [ 'Hashtable has wrong size!'. ^ #wrongSize ].
45
46         ht at: 1 put: 2.
47         t := Hashtable new.
48         ht at: Hashtable put: t.
49         (ht containsValue: 'b')
50             ifFalse: [ '1 not in Hashtable'. ^ 1 ].
51         (ht containsValue: 'd')
52             ifFalse: [ '2 not in Hashtable'. ^ 2 ].
53
54         (ht containsValue: 2)

```

```

55         ifFalse: [ '3 not in Hashtable'. ^ 3 ].
56     (ht containsValue: t)
57         ifFalse: [ '4 not in Hashtable'. ^ 4 ].
58     (ht containsKey: Hashtable)
59         ifFalse: [ 'key not found'. ^ #keyNotFound ].
60
61     ht clear.
62     ht isEmpty ifFalse: [ 'cleared hashtable is not empty!'. ^
#notEmpty ].
63     ht size = 0 ifFalse: [ 'cleared hashtable has elements!'. ^
#hasElementsAfterCleaning ].
64
65     string := (ht get: 'a').
66     (string = 'b') ifTrue: [ 'get from Hashtable'. ^ 5 ].
67
68     ^ 444
69 )
70 )
71

```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 IfTrueIfFalse = (  
24  
25     ----  
26  
27     test = (  
28         ^ self resolve: 42.  
29     )  
30  
31     test2 = (  
32         self resolve: 42.  
33         self resolve: 42.  
34         ^ self resolve: 33  
35     )  
36  
37     test3 = (  
38         | i a |  
39         i := 4.  
40         [ i > 0 ] whileTrue: [  
41             a := self resolve: 4.  
42             i := i - 1.  
43         ].  
44         ^ 4  
45     )  
46  
47     testIfTrueTrueResult = (  
48         | result |  
49         result := true ifTrue: [ 1 ].  
50         ^ result class  
51     )  
52  
53     testIfTrueFalseResult = (  
54         | result |
```

```

55     result := false ifTrue: [ 1 ].
56     ^ result class
57 )
58
59 testIfFalseTrueResult = (
60     | result |
61     result := true ifFalse: [ 1 ].
62     ^ result class
63 )
64
65 testIfFalseFalseResult = (
66     | result |
67     result := false ifFalse: [ 1 ].
68     ^ result class
69 )
70
71 resolve: a = (
72     (a == nil) ifFalse: [ ^ a ].
73 )
74
75 value: aBlock = (
76     ^ aBlock value
77 )
78 )
79

```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 MethodCall = (  
24     ----  
25  
26     test = (  
27         ^ self test2  
28     )  
29  
30     test2 = ( ^ 42 )  
31 )  
32
```



```

1 "
2 Copyright (c) 2001-2013 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 NonLocalReturn = (
24
25     ----
26
27     test1 = ( | t1Frame |
28         [ | nlrFrame |
29             ^ 42 ] value
30     )
31     test2 = ( ^ self test1 + 1 )
32     test3 = ( [ self test1. ^ 3 ] value )
33
34     test4 = ( ^ self at: 11 )
35     test5 = ( ^ self at: 10000 )
36
37     "Test case borrowed from Vector"
38     at: index = ( self checkIndex: index ifValid: [ ^ 42 ].
39         "else" ^ 22 )
40     checkIndex: index ifValid: block = (
41         (10 <= index) && (index <= 100)
42         ifTrue: [ ^ block value ]
43         ifFalse: [ #dontcare ]
44     )
45 )
46

```

```
1 "  
2 Copyright (c) 2016 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 NonLocalVars = (  
24     ----  
25  
26     testWriteDifferentTypes = (  
27         | value |  
28         1 to: 10 do: [:index |  
29             value := 0.  
30             self collection do: [:index | value := value + index].  
31             value := value // 4.  
32         ].  
33         ^value.  
34     )  
35  
36     collection = (^#(7 8))  
37 )  
38
```

```
1 "  
2 Copyright (c) 2019 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 NumberOfTests = (  
24  
25     ----  
26  
27     "Return the known number of tests,  
28     should be used in basic interpreter test harness to confirm  
completeness"  
29     numberOfTests = ( ^ 65 )  
30 )  
31
```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 ObjectCreation = (  
24  
25     ----  
26  
27     test = (  
28         | i |  
29         i := 0.  
30  
31         [i < 1000000] whileTrue: [  
32             self new.  
33             i := i + 1.  
34         ].  
35         ^ i  
36     )  
37 )  
38
```

```

1 "
2 Copyright (c) 2019 see AUTHORS file
3
4 Permission is hereby granted, free of charge, to any person obtaining a
copy
5 of this software and associated documentation files (the 'Software'), to
deal
6 in the Software without restriction, including without limitation the
rights
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8 copies of the Software, and to permit persons to whom the Software is
9 furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 Regressions = (
24
25     ----
26
27     testSymbolEquality = (
28         'foo:' asSymbol = #foo: ifTrue: [ ^ 1 ].
29         ^ 2
30     )
31
32     testSymbolReferenceEquality = (
33         'foo:' asSymbol == #foo: ifTrue: [ ^ 1 ].
34         ^ 2
35     )
36
37     testUninitializedLocal = (
38         | local |
39         local == nil ifTrue: [ ^ 1 ].
40         ^ 2
41     )
42
43     testUninitializedLocalInBlock = (
44         [ | local |
45           local == nil ifTrue: [ ^ 1 ] ] value.
46         ^ 2
47     )
48 )
49

```

```
1 "  
2 Copyright (c) 2001-2013 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 Return = (  
24  
25     ----  
26  
27     testReturnSelf = ( ^ self )  
28  
29     testReturnSelfImplicitly = ( )  
30  
31     testNoReturnReturnsSelf = ( 1 )  
32  
33     testBlockReturnsImplicitlyLastValue = ( ^ ([4] value) )  
34 )  
35
```

```
1 "  
2 Copyright (c) 2001-2018 see AUTHORS file  
3  
4 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
5 of this software and associated documentation files (the 'Software'), to  
deal  
6 in the Software without restriction, including without limitation the  
rights  
7 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
8 copies of the Software, and to permit persons to whom the Software is  
9 furnished to do so, subject to the following conditions:  
10  
11 The above copyright notice and this permission notice shall be included in  
12 all copies or substantial portions of the Software.  
13  
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
20 THE SOFTWARE.  
21 "  
22  
23 Self = (  
24  
25     ----  
26  
27     testAssignSuper = (  
28         super := 42.  
29         ^ super  
30     )  
31  
32     testAssignSelf = (  
33         self := 42.  
34         ^ self  
35     )  
36 )  
37
```

TestSuite/BasicInterpreterTests/number-of-tests.sh

```
1 #!/bin/sh
2 SCRIPT_PATH=`dirname $0`
3 TEST_FILE="${SCRIPT_PATH}/NumberOfTests.som"
4
5 # find all tests, count them, trim whitespace from result
6 NUM_TESTS=`grep -R "test[^[:space:]]*[[[:space:]]\+= ( " "${SCRIPT_PATH}" |
wc -l | tr -d '[:space:]'`
7
8 TEST_CODE="    numberOfTests = ( ^ ${NUM_TESTS} )"
9
10 sed -i'.old' -e 's/. *numberOfTests.*/' "${TEST_CODE}"/" "${TEST_FILE}"
11
12 git --no-pager diff --exit-code "${TEST_FILE}"
13
```



```

1 BlockTest = TestCase (
2   |escape_count escaped_block|
3
4   simpleBlock = (
5     ^ [ 42 ]
6   )
7
8   incBlock = (
9     ^ [:val | val + 1 ]
10  )
11
12  "This requires a closure"
13  adderBlock: amount = (
14    ^ [:val | amount + val ]
15  )
16
17  "Closure with mutable state in block"
18  counterBlock = (
19    |count|
20    count := 0.
21    ^ [ count := count + 1. count ]
22  )
23
24  selfKeeper = (
25    ^ [ self ]
26  )
27
28  escapingBlock = (
29    ^ [ ^ 42 ]
30  )
31
32  escapingNestedBlock = (
33    [ [ ^ [ ^ 43 ] ] value ] value
34  )
35
36  testSimpleBlocks = (
37    self assert: 42 equals: self simpleBlock value.
38    self assert: 4  equals: (self incBlock value: 3).
39    self assert: 43 equals: ((self adderBlock: 13) value: 30).
40  )
41
42  testClosure = (
43    | counter |
44    counter := self counterBlock.
45    self assert: 1 equals: counter value.
46    self assert: 2 equals: counter value.
47    self assert: 1 equals: self counterBlock value. "make sure each
copy is independent"
48    self assert: 3 equals: counter value.
49  )
50
51  testSelfInBlock = (
52    | test_inst |
53    test_inst := BlockTest new.
54    self assert: test_inst equals: test_inst selfKeeper value.
55    self assert: self      equals: self selfKeeper value.
56  )
57

```

```

58     testEscapedBlock = (
59         | escaping_block |
60
61         escape_count := 0.
62
63         escaping_block := self escapingBlock.
64
65         self assert: 0 equals: escape_count.
66         self assert: 666 equals: escaping_block value.
67         self assert: 1 equals: escape_count.
68
69         self assert: escaping_block is: escaped_block.
70
71
72         escaping_block := self escapingNestedBlock.
73         self assert: 1 equals: escape_count.
74         self assert: 666 equals: escaping_block value.
75         self assert: 2 equals: escape_count.
76         self assert: escaping_block is: escaped_block.
77     )
78
79     escapedBlock: block = (
80         escape_count := escape_count + 1.
81         escaped_block := block.
82
83         "return some dummy value to the object that sent 'value' to block"
84         ^666
85     )
86
87     testWhileTrue = (
88         | i |
89         i := 1.
90         [ i < 10 ] whileTrue: [ i := i + 1 ].
91         self assert: 10 equals: i.
92     )
93
94     testWhileFalse = (
95         | i |
96         i := 1.
97         [ i >= 10 ] whileFalse: [ i := i + 1 ].
98         self assert: 10 equals: i.
99     )
100
101     returnVal: val predicate: p = (
102         p ifNil: [ ^ val ].
103         ^ 0
104     )
105
106     testToDoAsResultOfIfTrue = (
107         | result |
108         result := true ifTrue: [
109             1 to: 2 do: [:i | 4 ]
110         ].
111
112         self assert: 1 equals: result
113     )
114
115     testReturnWithSomething = (
116         | result |
117         result := self returnVal: 3 predicate: nil.
118         self assert: 3 equals: result.

```

```

119
120     result := self returnVal: 4 predicate: self.
121     self assert: 0 equals: result.
122 )
123
124 testSuperExpressionInBlock = (
125     | result |
126     result := BlockTest helperForTestSuperExpressionInBlock.
127     self assert: BlockTest is: result.
128 )
129
130 ----
131
132 doInTest: block = (
133     ^ block value
134 )
135
136 helperForTestSuperExpressionInBlock = (
137     ^ self doInTest: [ (super new) class ]
138 )
139 )
140

```

```

1 BooleanTest = TestCase (
2
3   testIfTrueIfFalse = (
4     | b1 b2 |
5     b1 := false.
6     b2 := false.
7
8     true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
9     self assert: b1.
10    self deny: b2.
11
12    b1 := false.
13    b2 := false.
14    false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
15    self assert: b2.
16    self deny: b1.
17  )
18
19  testIfTrue = (
20    | b |
21    b := false.
22
23    true ifTrue: [ b := true ].
24    self assert: b.
25
26    b := false.
27    false ifTrue: [ b := true ].
28    self deny: b.
29  )
30
31  testIfTrueWithValueBlock = (
32    | b block |
33    b := false.
34    block := [ b := true ].
35
36    true ifTrue: block.
37    self assert: b.
38
39    b := false.
40    false ifTrue: block.
41    self deny: b.
42  )
43
44  testIfFalse = (
45    | b |
46    b := false.
47
48    true ifFalse: [ b := true ].
49    self deny: b.
50
51    b := false.
52    false ifFalse: [ b := true ].
53    self assert: b.
54  )
55
56  testIfFalseWithValueBlock = (
57    | b block |
58    b := false.

```

```

59     block := [ b := true ].
60
61     true ifFalse: block.
62     self deny: b.
63
64     b := false.
65     false ifFalse: block.
66     self assert: b.
67 )
68
69 testNot = (
70     self deny: true not.
71     self assert: false not.
72 )
73
74 andBoolTrueTrue   = ( ^ true  and: [ true  ] )
75 andBoolTrueFalse  = ( ^ true  and: [ false ] )
76 andBoolFalseTrue  = ( ^ false and: [ true  ] )
77 andBoolFalseFalse = ( ^ false and: [ false ] )
78
79 testAnd = (
80     self assert: self andBoolTrueTrue.
81     self deny:   self andBoolTrueFalse.
82     self deny:   self andBoolFalseTrue.
83     self deny:   self andBoolFalseFalse.
84 )
85
86 ampBoolTrueTrue   = ( ^ true  && [ true  ] )
87 ampBoolTrueFalse  = ( ^ true  && [ false ] )
88 ampBoolFalseTrue  = ( ^ false && [ true  ] )
89 ampBoolFalseFalse = ( ^ false && [ false ] )
90
91 testAmp = (
92     self assert: self ampBoolTrueTrue.
93     self deny:   self ampBoolTrueFalse.
94     self deny:   self ampBoolFalseTrue.
95     self deny:   self ampBoolFalseFalse.
96 )
97
98 orBoolTrueTrue   = ( ^ true  or: [ true  ] )
99 orBoolTrueFalse  = ( ^ true  or: [ false ] )
100 orBoolFalseTrue  = ( ^ false or: [ true  ] )
101 orBoolFalseFalse = ( ^ false or: [ false ] )
102
103 testOr = (
104     self assert: self orBoolTrueTrue.
105     self assert: self orBoolTrueFalse.
106     self assert: self orBoolFalseTrue.
107     self deny:   self orBoolFalseFalse.
108 )
109
110 pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
111 pipeBoolTrueFalse  = ( ^ true  || [ false ] )
112 pipeBoolFalseTrue  = ( ^ false || [ true  ] )
113 pipeBoolFalseFalse = ( ^ false || [ false ] )
114
115 testPipe = (
116     self assert: self pipeBoolTrueTrue.
117     self assert: self pipeBoolTrueFalse.
118     self assert: self pipeBoolFalseTrue.
119     self deny:   self pipeBoolFalseFalse.

```

```

120     )
121
122     testBlockSideEffectsOnLogicOps = (
123         | changed unused |
124         changed := false.
125
126         "#and:"
127         unused := true and: [ changed := #case1. true ].
128         self assert: changed is: #case1.
129
130         unused := false and: [ changed := #no. true ].
131         self assert: changed is: #case1.
132
133         "&&"
134         unused := true && [ changed := #case2. true ].
135         self assert: changed is: #case2.
136
137         unused := false && [ changed := #no. true ].
138         self assert: changed is: #case2.
139
140         "#or:"
141         unused := true or: [ changed := #no. true ].
142         self assert: changed is: #case2.
143
144         unused := false or: [ changed := #case3. true ].
145         self assert: changed is: #case3.
146
147         "||"
148         unused := true || [ changed := #no. true ].
149         self assert: changed is: #case3.
150
151         unused := false || [ changed := #case4. true ].
152         self assert: changed is: #case4.
153     )
154
155     testAsString = (
156         self assert: 'true' equals: true asString.
157         self assert: 'false' equals: false asString.
158     )
159
160     testIfNil = (
161         self assert: (nil ifNil: [ true ]).
162         self deny: (nil ifNil: [ false ]).
163
164         self assert: (self ifNil: [ #notExec ]) is: self.
165         self assert: (self ifNil: [ #notExec ]) is: self.
166     )
167
168     testIfNotNil = (
169         self assert: (self ifNotNil: [ true ]).
170         self deny: (self ifNotNil: [ false ]).
171
172         self assert: (nil ifNotNil: [ #notExec ]) is: nil.
173         self assert: (nil ifNotNil: [ #notExec ]) is: nil.
174     )
175 )
176

```

TestSuite/ClassA.som

```
1 ClassA = (  
2   | a b |  
3   result = (  
4     ^42  
5   )  
6   ----  
7   | c1 c2 c3 |  
8 )  
9
```

TestSuite/ClassB.som

```
1 ClassB = ClassA (  
2   | c d |  
3   ----  
4   | c4 c5 c6 |  
5 )  
6
```



```

1 ClassC = ClassB (
2   | e f |
3   a      = ( ^ a )
4   a: val = ( a := val )
5
6   f      = ( ^ f )
7   f: val = ( f := val )
8
9   ----
10
11  | c7 c8 c9 |
12
13  setAllAndInc: anInt = (
14    c1 := anInt.
15    c2 := c1 + 1.
16    c3 := c2 + 1.
17    c4 := c3 + 1.
18    c5 := c4 + 1.
19    c6 := c5 + 1.
20    c7 := c6 + 1.
21    c8 := c7 + 1.
22    c9 := c8 + 1.
23  )
24
25  getAll = (
26    | arr |
27    arr := Array new: 9.
28    arr at: 1 put: c1.
29    arr at: 2 put: c2.
30    arr at: 3 put: c3.
31    arr at: 4 put: c4.
32    arr at: 5 put: c5.
33    arr at: 6 put: c6.
34    arr at: 7 put: c7.
35    arr at: 8 put: c8.
36    arr at: 9 put: c9.
37    ^ arr
38  )
39 )
40

```

TestSuite/ClassLoadingTest.som

```
1 ClassLoadingTest = TestCase (
2   testEqualityOfClasses = (
3     | a b c |
4     b := ClassB new.
5     a := ClassA new.
6     c := ClassC new.
7
8     self assert: 42 equals: b result.
9     self assert: 42 equals: c result.
10
11     self assert: a class equals: b class superclass.
12     self assert: b class equals: c class superclass.
13   )
14 )
15
```

```

1 ClassStructureTest = TestCase (
2
3   testClassIdentity = (
4     self assert: Array equals: Array new class.
5     self assert: Integer equals: 1 class.
6     self assert: Integer equals: 10000000000 class.
7     self assert: Double equals: (1 // 2) class.
8     self assert: Double equals: 0.5 class.
9     self assert: Block1 equals: [42] class.
10    self assert: Object equals: Object new class.
11    self assert: Set equals: Set new class.
12    self assert: String equals: 'foo' class.
13    self assert: Symbol equals: #foo class.
14    self assert: True equals: true class.
15    self assert: False equals: false class.
16    self assert: Nil equals: nil class.
17
18    self assert: True superclass equals: False superclass.
19    self assert: True superclass equals: Boolean.
20    self assert: True superclass equals: Boolean.
21  )
22
23  testThatCertainMethodsArePrimitives = (
24    | m |
25    "This is a little fragile.
26     Index needs to be adapted with changing Class definition."
27    m := Object methods at: 1.
28    "self expect: #class equals: m signature."
29
30    self optional: #invokableTypes assert: Primitive equals: m class.
31    "Class>>#name should be a primitive."
32
33    m := Object methods at: 7.
34    "self expect: #asString equals: m signature."
35
36    self optional: #invokableTypes assert: Method equals: m class.
37    "Class>>#asString should be a normal method."
38  )
39
40  testAccessToInstanceFields = (
41    | o |
42    o := ClassC new.
43    o a: 333.
44    self assert: 333 equals: o a.
45
46    o f: 333.
47    self assert: 333 equals: o f.
48  )
49
50  testAccessToClassFields = (
51    | arr |
52    ClassC setAllAndInc: 4.
53    arr := ClassC getAll.
54    1 to: 9 do: [:i |
55      self assert: i + (4 - 1) equals: (arr at: i).
56    ].
57
58    "We do that here to make sure that class fields do not interfere with

```

```

57     other class properties."
58     self assert: ClassB      is: ClassC superclass.
59     self assert: Metaclass is: ClassC class class.
60     self assert: #ClassC     equals: ClassC name.
61 )
62
63 testMetaClasses = (
64     self assert: nil          is: Object superclass.
65     self assert: Integer      is: 1 class.
66     self assert: #'Integer class' is: 1 class class name.
67     self assert: Metaclass     is: 1 class class class.
68
69     self assert: #'Metaclass class' is: Metaclass class name.
70     self assert: Metaclass        is: Metaclass class class.
71
72     self assert: Object          is: 1 class superclass.
73     self assert: #'Object class' is: 1 class class superclass name.
74     self assert: Class           is: Object class superclass.
75     self assert: Metaclass       is: Class class class.
76 )
77
78 testInstanceFields = (
79     self assert: 2 equals: ClassA fields length.
80     self assert: 4 equals: ClassB fields length.
81     self assert: 6 equals: ClassC fields length.
82 )
83 )
84

```

TestSuite/ClosureTest.som

```
1 "  
2  
3 $Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "This test verifies that SOM blocks are indeed closures. The test was  
found on  
27 Eliot Miranda's Cog Blog."  
28  
29 ClosureTest = TestCase (  
30   testClosureProperty = (  
31     | factorial result facts |  
32  
33     facts := Array new: 10.  
34     facts at: 1 put: 1.  
35     facts at: 2 put: 2.  
36     facts at: 3 put: 6.  
37     facts at: 4 put: 24.  
38     facts at: 5 put: 120.  
39     facts at: 6 put: 720.  
40     facts at: 7 put: 5040.  
41     facts at: 8 put: 40320.  
42     facts at: 9 put: 362880.  
43     facts at: 10 put: 3628800.  
44  
45     factorial := [ :n |  
46       n = 1  
47         ifTrue: [ 1 ]  
48         ifFalse: [ (factorial value: n - 1) * n ] ].  
49  
50     result := (1 to: 10) collect: factorial.  
51     result doIndexes: [ :i |  
52       self assert: (facts at: i) equals: (result at: i) ]  
53   )
```

54)
55

TestSuite/CoercionTest.som

```
1 "  
2  
3 $Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 CoercionTest = TestCase (  
29  
30     testBasicNumberCoercion = (  
31         self assert: 5 equals: 25 sqrt.  
32         self assert: 1 equals: (2 // 4) * 2.  
33         self assert: 1 equals: 2 * (2 // 4).  
34     )  
35 )  
36
```

TestSuite/CompilerReturnTest.som

```
1 "  
2  
3 $Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2009-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 CompilerReturnTest = TestCase (  
29  
30     return1 = ( ^self )  
31     return2 = (      )  
32  
33     return3: arg = ( ^self )  
34     return4: arg = (      )  
35  
36     testExplicitAndImplicitReturns = (  
37         self assert: self is: self return1.  
38         self assert: self is: self return2.  
39         self assert: self is: (self return3: 23).  
40         self assert: self is: (self return4: 23).  
41     )  
42  
43  
44     "In SOM++, code after the #ifTrue: does not seem to be executed, if  
the  
45     block expression ends with a dot."  
46     testIfTrueWithDot = (  
47         | arr |  
48         arr := Array new: 3.  
49         self usesIfTrueWithDot: arr.  
50         self assertArrayCorrectness: arr.  
51     )  
52
```



```

53     assertArrayCorrectness: arr = (
54         self assert: 1 equals: (arr at: 1). "method was not executed"
55         self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
56         self assert: 3 equals: (arr at: 3). "remainder was not
executed"
57     )
58
59     testIfTrueWithoutDot = (
60         | arr |
61         arr := Array new: 3.
62         self usesIfTrueWithoutDot: arr.
63         self assertArrayCorrectness: arr.
64     )
65
66     testIfFalseWithDot = (
67         | arr |
68         arr := Array new: 3.
69         self usesIfFalseWithDot: arr.
70         self assertArrayCorrectness: arr.
71     )
72
73     testIfFalseWithoutDot = (
74         | arr |
75         arr := Array new: 3.
76         self usesIfFalseWithoutDot: arr.
77         self assertArrayCorrectness: arr.
78     )
79
80     usesIfTrueWithDot: arr = (
81         arr at: 1 put: 1.
82         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
83         arr at: 3 put: 3.
84     )
85
86     usesIfTrueWithoutDot: arr = (
87         arr at: 1 put: 1.
88         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
89         arr at: 3 put: 3.
90     )
91
92     usesIfFalseWithDot: arr = (
93         arr at: 1 put: 1.
94         (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
95         arr at: 3 put: 3.
96     )
97
98     usesIfFalseWithoutDot: arr = (
99         arr at: 1 put: 1.
100        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITHOUT DOT"].
101        arr at: 3 put: 3.
102    )
103
104    testWriteArgument = (
105        self assert: 42 equals: (self dec: 43).
106    )
107
108    dec: anInt = (
109        anInt := anInt - 1.
110        ^ anInt
111    )
112 )

```



```

1 DictionaryTest = TestCase (
2   testAtAndAtPut = (
3     | dict val |
4     dict := Dictionary new.
5     val := dict at: 1.
6
7     self assert: nil is: val.
8
9     val := dict at: 1 put: #foo.
10    self assert: dict is: val.
11    val := dict at: 1.
12    self assert: #foo is: val.
13
14    val := dict at: 1 put: #foo.
15    self assert: dict is: val.
16    val := dict at: 1.
17    self assert: #foo is: val.
18
19    val := dict at: 1 put: 42.
20    self assert: dict is: val.
21    val := dict at: 1.
22    self assert: 42 equals: val.
23  )
24
25  testContainsKey = (
26    | dict |
27    dict := Dictionary new.
28    self deny: (dict containsKey: 4).
29
30    dict at: 4 put: 34234.
31    self assert: (dict containsKey: 4).
32  )
33
34  testKeys = (
35    | dict keys |
36    dict := Dictionary new.
37
38    self assert: 0 equals: dict keys size.
39
40    dict at: 4 put: 423.
41    self assert: 1 equals: dict keys size.
42
43    dict at: 4 put: #gdfgd.
44    self assert: 1 equals: dict keys size.
45
46    dict at: 'as' put: Object new.
47    self assert: 2 equals: dict keys size.
48
49    keys := dict keys.
50    self assert: 4 equals: (keys at: 1).
51    self assert: 'as' equals: (keys at: 2).
52  )
53
54  testValues = (
55    | dict values v2 |
56    dict := Dictionary new.
57
58    self assert: 0 equals: dict values size.

```

```

59
60 dict at: 4 put: 423.
61 self assert: 1 equals: dict values size.
62
63 dict at: 4 put: #gdfgd.
64 self assert: 1 equals: dict values size.
65
66 dict at: 'as' put: #(1 2 3).
67 self assert: 2 equals: dict values size.
68
69 values := dict values.
70 self assert: #gdfgd is: (values at: 1).
71
72 v2 := values at: 2.
73 self assert: 1 equals: (v2 at: 1).
74 self assert: 2 equals: (v2 at: 2).
75 self assert: 3 equals: (v2 at: 3).
76 )
77
78 testDo = (
79 | dict expectedKs expectedVs i |
80 i := 1.
81 dict := Dictionary new.
82 dict at: #e put: 344.
83 dict at: 1 put: 545.
84 dict at: 0 put: 123.
85
86 expectedKs := #(#e 1 0).
87 expectedVs := #(344 545 123).
88
89 dict do: [:p |
90     self assert: (expectedKs at: i) equals: p key.
91     self assert: (expectedVs at: i) equals: p value.
92     i := i + 1.
93 ].
94
95 self assert: 4 equals: i.
96 )
97 )

```

```
1 DoesNotUnderstandMessage = (  
2   | target selector arguments |  
3  
4   initializeWith: targetObj selector: aSelector arguments: argArray = (  
5     target      := targetObj.  
6     selector    := aSelector.  
7     arguments   := argArray.  
8   )  
9  
10  target      = ( ^ target )  
11  selector    = ( ^ selector )  
12  arguments   = ( ^ arguments )  
13  
14  ----  
15  
16  to: target selector: selector arguments: args = (  
17    | m |  
18    m := self new.  
19    m initializeWith: target selector: selector arguments: args.  
20    ^ m  
21  )  
22 )
```

```

1 DoesNotUnderstandTest = TestCase (
2
3   testSimpleUnknownFoo = (
4     | result |
5     result := self foo.
6     self assert: DoesNotUnderstandMessage is: result class.
7     self assert: self is: result target.
8     self assert: #foo is: result selector.
9   )
10
11  testArguments = (
12    | result |
13    result := self foo.
14    self assert: Array is: result arguments class.
15    self assert: 0 equals: result arguments length.
16
17    result := self foo: 1.
18    self assert: 1 equals: result arguments length.
19    self assert: 1 equals: (result arguments at: 1).
20
21    result := self foo: 1 bar: 2 baz: 3.
22    self assert: 3 equals: result arguments length.
23    self assert: 1 equals: (result arguments at: 1).
24    self assert: 2 equals: (result arguments at: 2).
25    self assert: 3 equals: (result arguments at: 3).
26  )
27
28  testRepeat = (
29    | result |
30    result := Array new: 5.
31    1 to: result length do: [:i |
32      result at: i put: self foo.
33
34      i > 1 ifTrue: [
35        self assert: (result at: i - 1) ~= (result at: i).
36      ]
37    ].
38  )
39
40  doesNotUnderstand: selector arguments: arguments = (
41    ^ DoesNotUnderstandMessage to: self selector: selector arguments:
arguments.
42  )
43 )
44

```

TestSuite/DoubleTest.som

```
1 "  
2  
3 $Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 DoubleTest = TestCase (  
27  
28     testAdd = (  
29         self assert: 1.0 equals: 0.5 + 0.5.  
30         self assert: 0.0 equals: 0.5 + -0.5.  
31  
32         self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.  
33         self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.  
34     )  
35  
36     testSubtract = (  
37         self assert: 0.0 equals: 0.5 - 0.5.  
38         self assert: 1.0 equals: 0.5 - -0.5.  
39  
40         self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.  
41         self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.  
42     )  
43  
44     testMultiply = (  
45         self assert: 4.0 equals: 2.0 * 2.0.  
46         self assert: -4.0 equals: 2.0 * -2.0.  
47  
48         self assert: 1.0 equals: 4.0 * 0.25.  
49         self assert: -1.0 equals: -4.0 * 0.25.  
50     )  
51  
52     testIntegerDivision = (  
53         self assert: 1 equals: (4/3) + (4/5)
```

```

54 )
55
56 testDoubleDivision = (
57     self assert: 32 // 15 equals: (4//3) + (4//5)
58 )
59
60 testModulo = (
61     self assert: 1.0 equals: 3.0 % 2.0.
62     self assert: 0.0 equals: 3.0 % 3.0.
63
64     self assert: -1.0 equals: -3.0 % 2.0.
65     self assert: -1.0 equals: -3.0 % -2.0.
66     self assert: 1.0 equals: 3.0 % -2.0.
67
68     self assert: 0.0 equals: 3.0 % 3.0.
69     self assert: 0.0 equals: -3.0 % -3.0.
70     self assert: 0.0 equals: 3.0 % -3.0.
71 )
72
73 testAbs = (
74     self assert: 1.0 equals: 1.0 abs.
75     self assert: 1.0 equals: -1.0 abs.
76
77     self assert: 9007199254740992.0 equals: 9007199254740992.0 abs.
78     self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.
79
80     self assert: 19007199254740992.0 equals: 19007199254740992.0 abs.
81     self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
82 )
83
84 testSqrt = (
85     self assert: 3.0 equals: 9.0 sqrt.
86     self assert: 16.0 equals: 256.0 sqrt.
87
88     self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
89 )
90
91 testRaisedTo = (
92     self assert: 1.0 equals: (2.0 raisedTo: 0).
93     self assert: 2.0 equals: (2.0 raisedTo: 1).
94     self assert: 8.0 equals: (2.0 raisedTo: 3).
95     self assert: 256.0 equals: (2.0 raisedTo: 8).
96
97     self assert: 256.0 equals: (-2.0 raisedTo: 8).
98     self assert: -128.0 equals: (-2.0 raisedTo: 7).
99
100     self assert: 6.25 equals: (2.5 raisedTo: 2).
101     self assert: 5.0625 equals: (1.5 raisedTo: 4).
102
103     self assert: 0.0 equals: (0.0 raisedTo: 5).
104     self assert: 1.0 equals: (0.0 raisedTo: 0).
105
106     "Negative exponents are not yet supported"
107     self assert: 1.0 equals: (0.0 raisedTo: -1).
108     self assert: 1.0 equals: (0.0 raisedTo: -2).
109     self assert: 1.0 equals: (10.0 raisedTo: -1).
110     self assert: 1.0 equals: (10.0 raisedTo: -2).
111
112     "Double exponents are not yet supported"
113     self assert: 2.0 equals: (2.0 raisedTo: 1.5).
114     self assert: 4.0 equals: (2.0 raisedTo: 2.4).

```



```

115     self assert: 4.0 equals: ( 2.0 raisedTo: 2.9).
116     self assert: 1.0 equals: ( 2.0 raisedTo: -2.2).
117 )
118
119 testNegated = (
120     self assert: 0.0 equals: 0.0 negated.
121     self assert: -1.0 equals: 1.0 negated.
122     self assert: 1.0 equals: -1.0 negated.
123
124     self assert: -9007199254740992.0 equals: 9007199254740992.0 negated.
125     self assert: 9007199254740992.0 equals: -9007199254740992.0 negated.
126
127     self assert: -19007199254740992.0 equals: 19007199254740992.0
negated.
128     self assert: 19007199254740992.0 equals: -19007199254740992.0
negated.
129 )
130
131 testAsString = (
132     self assert: '0.5' equals: (1//2) asString.
133     self assert: '0.5' equals: 0.5 asString.
134 )
135
136 testEquals = (
137     self assert: (1.0 = 1).
138 )
139
140 testRound = (
141     self assert: 1 equals: 1.0 round.
142     self assert: 1 equals: 1.4 round.
143     self assert: 1 equals: 1.4999 round.
144     self assert: 2 equals: 1.5 round.
145     self assert: 2 equals: 1.5000001 round.
146     self assert: 1 equals: (5//10) round.
147     self assert: 1 equals: (14//10) round.
148     self assert: 445 equals: (44534//100) round.
149 )
150
151 testAsInteger = (
152     self assert: 1 equals: 1.0 asInteger.
153     self assert: 1 equals: 1.1 asInteger.
154     self assert: 1 equals: 1.999 asInteger.
155
156     self assert: -1 equals: -1.0 asInteger.
157     self assert: -1 equals: -1.999 asInteger.
158 )
159
160 testSin = (
161     | pi |
162     pi := 3.141592653589.
163     self assert: 0.0 equals: 0.0 sin.
164     self assert: pi sin abs < 0.00000000001.
165     self assert: (pi // 2.0) sin > 0.9999999999.
166 )
167
168 testCos = (
169     | pi |
170     pi := 3.141592653589.
171     self assert: 1.0 equals: 0.0 cos.
172     self assert: (pi // 2.0) cos abs < 0.00000000001.
173     self assert: pi cos < -0.9999999999.

```

```

174 )
175
176 testInfinity = (
177     self assert: Double PositiveInfinity > 1.
178     self assert: Double PositiveInfinity equals: Double PositiveInfinity
+ 1.
179     self assert: Double PositiveInfinity equals: Double PositiveInfinity
- 1.
180
181     self assert: Double PositiveInfinity > (999999 * 999999 * 999999 *
999999).
182 )
183
184 testFromString = (
185     self assert: 0.0 equals: (Double fromString: '0.0').
186     self assert: -1.1 equals: (Double fromString: '-1.1').
187
188     self assert: 3423.54656 equals: (Double fromString: '3423.54656').
189     self assert: -672.433244 equals: (Double fromString: '-672.433244').
190 )
191
192 testEqual = (
193     self assert: 0.0 = 0.0.
194     self assert: 1.0 = 1.0.
195     self assert: 0.0 = -0.0.
196     self assert: -0.0 = 0.0.
197 )
198
199 testLessThan = (
200     self assert: 0.0 < 1.0.
201     self assert: 0.499999999 < 0.5.
202     self deny: 1.0 < 0.0.
203     self deny: 0.5 < 0.499999999.
204 )
205
206 testGreaterThan = (
207     self deny: 0.0 > 1.0.
208     self deny: 0.499999999 > 0.5.
209     self assert: 1.0 > 0.0.
210     self assert: 0.5 > 0.499999999.
211 )
212
213 testLessThanOrEqual = (
214     self assert: 0.0 <= 1.0.
215     self assert: 0.499999999 <= 0.5.
216     self assert: 0.5 <= 0.5.
217     self deny: 1.0 < 0.0.
218     self deny: 0.5 < 0.499999999.
219 )
220
221 testGreaterThanOrEqual = (
222     self deny: 0.0 >= 1.0.
223     self deny: 0.499999999 >= 0.5.
224     self assert: 0.5 >= 0.5.
225     self assert: 1.0 >= 0.0.
226     self assert: 0.5 >= 0.499999999.
227 )
228
229 testNegative = (
230     self assert: -0.000000001 negative.
231     self assert: -1.0 negative.

```

```

232     self assert: -123123.000000001 negative.
233     self deny: 0.000000001 negative.
234     self deny: 1.0 negative.
235     self deny: 123123.000000001 negative.
236 )
237
238 testBetween = (
239     self assert: (1.0 between: 0.0 and: 2.0).
240     self assert: (0.000001 between: 0.0 and: 2.0).
241     self assert: (1.999999 between: 0.0 and: 2.0).
242     self deny: (0.0 between: 0.0 and: 2.0).
243     self deny: (2.0 between: 0.0 and: 2.0).
244 )
245
246 testToDo = (
247     | d |
248     d := 0.0.
249     0.0 to: 10.0 do: [:ii |
250         d := d + ii.
251     ].
252
253     self assert: 55.0 equals: d.
254
255     d := 0.0.
256     0.0 to: 10.1 do: [:ii |
257         d := d + ii.
258     ].
259
260     self assert: 55.0 equals: d.
261
262     d := 0.0.
263     0.1 to: 10.1 do: [:ii |
264         d := d + ii.
265     ].
266
267     self assert: 55.0 + 1.1 equals: d.
268 )
269
270 testDownToDo = (
271     | d |
272     d := 0.0.
273     10.0 downTo: 0.0 do: [:ii |
274         d := d + ii.
275     ].
276
277     self assert: 55.0 equals: d.
278
279     d := 0.0.
280     10.1 downTo: 0.0 do: [:ii |
281         d := d + ii.
282     ].
283
284     self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
285 )
286 )
287

```

TestSuite/EmptyTest.som

```
1 "  
2  
3 $Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 EmptyTest = TestCase (  
27  
28     "This is just an empty TestCase.  
29     It only tests the basic infrastructure"  
30  
31 )  
32
```

```

1 GlobalTest = TestCase (
2   | doesntKnow |
3   unknownGlobal: name = ( doesntKnow := name. ^ name )
4
5   testUnknownGlobalHandler = (
6     self assert: #foobar equals: foobar.      "should return the unknown
globals name"
7     self assert: #foobar equals: doesntKnow. "and should have set it in
the field"
8   )
9
10  testKnownGlobals = (
11    self assert: True   equals: true   class.
12    self assert: False  equals: false  class.
13    self assert: Nil    equals: nil    class.
14    self assert: System equals: system class.
15  )
16
17  escapingBlock = (
18    ^ [ EscapingBlockGlobal ]
19  )
20
21  testUnknownGlobalSemanticsInBlocks = (
22    self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
23    self assert: #NormalBlockGlobal is: doesntKnow.
24
25    self assert: #EscapingBlockGlobal is: self escapingBlock value.
26    self assert: #EscapingBlockGlobal is: doesntKnow.
27
28    self assert: #NestedBlockGlobal is: [
29      [ [ NestedBlockGlobal ] value ] value ] value.
30    self assert: #NestedBlockGlobal is: doesntKnow.
31  )
32 )
33

```

```

1 "
2
3 $Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5 Copyright (c) 2001-2013 see AUTHORS file
6
7 Permission is hereby granted, free of charge, to any person obtaining a
copy
8 of this software and associated documentation files (the 'Software'), to
deal
9 in the Software without restriction, including without limitation the
rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in
15 all copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23 THE SOFTWARE.
24 "
25
26 HashTest = TestCase (
27
28     testHashtable = (
29         | ht string array t |
30
31         ht := Hashtable new.
32         self assert: ht isEmpty description: 'new ht needs to be empty'.
33
34         ht at: 'a' put: 'b'.
35         self assert: (ht containsValue: 'b') description: 'needs to contain
"b"'.
36         self deny: ht isEmpty.
37
38         self assert: 1 equals: ht size.
39
40         ht at: 'c' put: 'd'.
41         self assert: 2 equals: ht size.
42
43         ht at: 1 put: 2.
44         t := Hashtable new.
45         ht at: Hashtable put: t.
46         system fullGC.
47
48         self assert: (ht containsValue: 'b') description: 'needs to contain
"b"'.
49         self assert: (ht containsValue: 'd') description: 'needs to contain
"d"'.
50         self assert: (ht containsValue: 2) description: 'needs to contain
"2"'.

```

```
51     self assert: (ht containsValue: t)    description: 'needs to contain t'.
52     self assert: (ht containsKey: Hashtable) description: 'needs to
contain Hashtable'.
53
54     ht clear.
55     self assert: ht isEmpty.
56     self assert: 0 equals: ht size.
57
58     self assert: nil equals: (ht get: 'a').
59 )
60 )
61
62
```

TestSuite/IntegerTest.som

```
1 "  
2  
3 $Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 IntegerTest = TestCase (  
29  
30   testEqualityAndIdentity = (  
31     | a b |  
32     a := 42.  
33     b := 42.  
34  
35     self assert: a = b    description: 'Integers are equal based on their  
value'.  
36     self assert: a == b  description: 'Integers do not have pointer/  
reference equality. It is also supposed to be value equality'.  
37  
38     "Sometimes it can be hard to implement efficiently, but it SHOULD  
really  
39     be true for all values of integers."  
40     a := 1 << 30.  b := 1 << 30.  
41     self optional: #integerIdentity assert: a is: b.  
42  
43     a := 1 << 32.  b := 1 << 32.  
44     self optional: #integerIdentity assert: a is: b.  
45  
46     a := 1 << 60.  b := 1 << 60.  
47     self optional: #integerIdentity assert: a is: b.  
48  
49     a := 1 << 70.  b := 1 << 70.  
50     self optional: #integerIdentity assert: a is: b.
```



```

51
52     a := 1 << 100.  b := 1 << 100.
53     self optional: #integerIdentity assert: a is: b.
54 )
55
56 testClassAndValueRanges = (
57     | i |
58     self assert: Integer equals: -42 class.
59     self assert: Integer equals: 0 class.
60     self assert: Integer equals: 23 class.
61     self assert: Integer equals: 1073741823 class.
62     self assert: Integer equals: 1073741824 class.
63
64     "Let's test for size behavior and corresponding class"
65     i := 1 << 30.
66     self assert: Integer equals: i class.
67     self assert: i > 0 description: 'should not overflow'.
68     self assert: '1073741824' equals: i asString.
69
70     i := 1 << 32.
71     self assert: Integer equals: i class.
72     self assert: i > 0 description: 'should not overflow'.
73     self assert: '4294967296' equals: i asString.
74
75     i := 1 << 60.
76     self assert: Integer equals: i class.
77     self assert: i > 0 description: 'should not overflow'.
78     self assert: '1152921504606846976' equals: i asString.
79
80     i := 1 << 70.
81     self assert: Integer equals: i class.
82     self assert: i > 0 description: 'should not overflow'.
83     self optional: #bigIntShifts assert: '1180591620717411303424' equals:
i asString.
84
85     i := -1 << 30.
86     self assert: Integer equals: i class.
87     self assert: i < 0 description: 'should not underflow'.
88     self assert: '-1073741824' equals: i asString.
89
90     i := -1 << 32.
91     self assert: Integer equals: i class.
92     self assert: i < 0 description: 'should not underflow'.
93     self assert: '-4294967296' equals: i asString.
94
95     i := -1 << 60.
96     self assert: Integer equals: i class.
97     self assert: i < 0 description: 'should not underflow'.
98     self assert: '-1152921504606846976' equals: i asString.
99
100    i := -1 << 70.
101    self assert: Integer equals: i class.
102    self assert: i < 0 description: 'should not underflow'.
103    self optional: #bigIntShifts assert: '-1180591620717411303424'
equals: i asString.
104 )
105
106 testStringConversion = (
107     self assert: '0' equals: ( 0 asString).
108     self assert: '1' equals: ( 1 asString).
109     self assert: '2' equals: ( 2 asString).

```

```

110     self assert: '-1' equals: (-1 asString).
111     self assert: '-2' equals: (-2 asString).
112
113     self assert: 42 equals: '42' asInteger.
114     self assert: -2 equals: '-2' asInteger.
115 )
116
117 testIntegerLiterals = (
118     "Make sure the parser reads literals correctly. So, check some basic
properties"
119     self assert: 1 equals: 2 / 2.
120     self assert: 100 equals: 50 + 50.
121     self assert: 9223372036854775807 equals: 92233720368 * 100000000
+ 54775807.
122     self assert: -50 equals: 50 - 100.
123     self assert: -2147483648 equals: 21474 * -100000 - 83648.
124
125     self optional: #bigInteger
126     assert: 922337203685477580700 equals: 92233720368 * 100000000
+ 54775807 * 100.
127     self optional: #bigInteger
128     assert: -922337203685477580700 equals: 92233720368 * 100000000
+ 54775807 * -100.
129 )
130
131 testFromString = (
132     self assert: 1 equals: (Integer
fromString: '1').
133     self assert: 100 equals: (Integer
fromString: '100').
134     self assert: 9223372036854775807 equals: (Integer fromString:
'9223372036854775807').
135     self assert: -50 equals: (Integer
fromString: '-50').
136     self assert: -2147483648 equals: (Integer
fromString: '-2147483648').
137
138     self optional: #bigInteger
139     assert: 922337203685477580700 equals: (Integer fromString:
'922337203685477580700').
140     self optional: #bigInteger
141     assert: -922337203685477580700 equals: (Integer fromString:
'-922337203685477580700').
142 )
143
144 testRangeBorders = (
145     self assert: '536870911' equals: 536870911 asString.
146     self assert: '536870912' equals: 536870912 asString.
147     self assert: '536870913' equals: 536870913 asString.
148     self assert: '1073741823' equals: 1073741823 asString.
149     self assert: '1073741824' equals: 1073741824 asString.
150     self assert: '1073741825' equals: 1073741825 asString.
151     self assert: '2147483647' equals: 2147483647 asString.
152     self assert: '-536870911' equals: -536870911 asString.
153     self assert: '-536870912' equals: -536870912 asString.
154     self assert: '-536870913' equals: -536870913 asString.
155     self assert: '-1073741823' equals: -1073741823 asString.
156     self assert: '-1073741824' equals: -1073741824 asString.
157     self assert: '-1073741825' equals: -1073741825 asString.
158     self assert: '-2147483647' equals: -2147483647 asString.
159     self assert: '-2147483648' equals: -2147483648 asString.

```

```

160 )
161
162 testComparisons = (
163     self assert: ( 9 = 9).
164     self deny:   ( 1 = 2).
165     self deny:   ( 0 < 0).
166     self assert: ( 1 < 2).
167     self deny:   ( 2 < 1).
168     self assert: (-3 < 2).
169     self deny:   ( 3 < -2).
170     self deny:   ( 0 > 0).
171     self deny:   ( 1 > 2).
172     self assert: ( 2 > 1).
173     self deny:   (-3 > 2).
174     self assert: ( 3 > -2).
175     self assert: ( 4 >= 3).
176     self assert: ( 3 >= 3).
177     self deny:   ( 2 >= 3).
178     self assert: ( 2 <= 4).
179     self assert: ( 3 <= 3).
180     self deny:   ( 4 <= 3).
181 )
182
183 testAddition = (
184     self assert: 0 equals: ( 0+0).
185     self assert: 1 equals: ( 1+0).
186     self assert: 1 equals: ( 0+1).
187     self assert: 2 equals: ( 1+1).
188     self assert: 0 equals: (-1+1).
189     self assert: 1 equals: (-1+2).
190 )
191
192 testSubtraction = (
193     self assert: 1 equals: (1-0).
194     self assert: -1 equals: (0-1).
195     self assert: 1 equals: (2-1).
196 )
197
198 testMultiplication = (
199     self assert: 0 equals: ( 1* 0).
200     self assert: -1 equals: (-1* 1).
201     self assert: -25 equals: ( 5* -5).
202     self assert: 12 equals: (-3* -4).
203 )
204
205 testDivision = (
206     self assert: 1 equals: ( 1/ 1).
207     self assert: 1 equals: ( 3/ 2).
208     self assert: -2 equals: ( 4/ -2).
209     self assert: -2 equals: (-6/ 3).
210     self assert: 3 equals: (-12/ -4).
211 )
212
213 testDouble = (
214     self assert: 6 equals: ( 36// 6).
215     self assert: -5 equals: (-10// 2).
216     self assert: -4 equals: ( 20// -5).
217     self assert: 1 equals: ( -5// -5).
218 )
219
220 testModulo = (

```

```

221     self assert: 1 equals: ( 10 % 3).
222     self assert: -2 equals: ( 10 % -3).
223     self assert: 2 equals: (-10 % 3).
224     self assert: -1 equals: (-10 % -3).
225     self assert: 0 equals: ( 10 % 5).
226
227     self assert: 1 equals: ( 10 rem: 3).
228     self assert: 1 equals: ( 10 rem: -3).
229     self assert: -1 equals: (-10 rem: 3).
230     self assert: -1 equals: (-10 rem: -3).
231     self assert: 0 equals: ( 10 rem: 5).
232 )
233
234 testAbsoluteValue = (
235     self assert: 4 equals: -4 abs.
236     self assert: 4 equals: 4 abs.
237
238     self assert: 9223372036854775296 equals: -9223372036854775296 abs.
239     self assert: 9223372036854775296 equals: 9223372036854775296 abs.
240 )
241
242 testNegated = (
243     self assert: -23 equals: ( 23 negated).
244     self assert: 23 equals: (-23 negated).
245 )
246
247 testSquareRoot = (
248     self assert: 5 equals: (25 sqrt).
249     self assert: Integer equals: (25 sqrt class).
250 )
251
252 testRaisedTo = (
253     self assert: 1 equals: ( 2 raisedTo:
254 0).
255     self assert: 2 equals: ( 2 raisedTo:
256 1).
257     self assert: 8 equals: ( 2 raisedTo:
258 3).
259     self assert: 256 equals: ( 2 raisedTo:
260 8).
261     self assert: 1267650600228229401496703205376 equals: ( 2 raisedTo:
262 100).
263     self assert: 256 equals: (-2 raisedTo:
264 8).
265     self assert: -128 equals: (-2 raisedTo:
266 7).
267     self assert: 0 equals: ( 0 raisedTo:
268 5).
269     self assert: 1 equals: ( 0 raisedTo:
270 0).
271
272     "Negative exponents are not yet supported"
273     self assert: 1 equals: ( 0 raisedTo:
274 -1).
275     self assert: 1 equals: ( 0 raisedTo:
276 -2).
277     self assert: 1 equals: (10 raisedTo:
278 -1).
279     self assert: 1 equals: (10 raisedTo:

```

```

-2).
270
271     "Double exponents are not yet supported"
272     self assert:                                2 equals: ( 2 raisedTo:
1.5).
273     self assert:                                4 equals: ( 2 raisedTo:
2.4).
274     self assert:                                4 equals: ( 2 raisedTo:
2.9).
275     self assert:                                1 equals: ( 2 raisedTo:
-2.2).
276 )
277
278 testAnd = (
279     self assert: 0 equals: (2 & 1).
280     self assert: 2 equals: (2 & 2).
281 )
282
283 testBitXor = (
284     self assert: 0 equals: (1 bitXor: 1).
285     self assert: 3 equals: (2 bitXor: 1).
286 )
287
288 testAs32BitUnsignedValue = (
289     self assert: 1 << 1 equals: (1 << 1) as32BitUnsignedValue.
290     self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
291     self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
292     self assert: 0 equals: (1 << 32) as32BitUnsignedValue.
293     self assert: 4294967295 equals: -1 as32BitUnsignedValue.
294     self assert: 512 equals: -9223372036854775296
as32BitUnsignedValue.
295     self assert: 4294966784 equals: 9223372036854775296
as32BitUnsignedValue.
296 )
297
298 testAs32BitSignedValue = (
299     self assert: 1 << 1 equals: (1 << 1) as32BitSignedValue.
300     self assert: 1 << 10 equals: (1 << 10) as32BitSignedValue.
301     self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
302     self assert: 0 equals: (1 << 32) as32BitSignedValue.
303
304     self assert: 512 equals: -9223372036854775296 as32BitSignedValue.
305     self assert: -512 equals: 9223372036854775296 as32BitSignedValue.
306 )
307
308 testAsDouble = (
309     self assert: 0.0 equals: 0 asDouble.
310     self assert: Double is: 0 asDouble class.
311
312     self assert: 2147483648.0 equals: 2147483648 asDouble.
313     self assert: Double is: 2147483648 asDouble class.
314
315     self assert: -2147483648.0 equals: -2147483648 asDouble.
316     self assert: Double is: -2147483648 asDouble class.
317 )
318
319 testUnsignedRightShift = (
320     self assert: 0 equals: 1 >>> 1.
321     self assert: 512 equals: 1024 >>> 1.
322     self assert: 127 equals: 1023 >>> 3.
323

```

```

324     "not sure whether we should really insist on this"
325     self optional: #toBeSpecified assert: 9223372036854775807 equals:
-1 >>> 1.
326     self optional: #toBeSpecified assert: 9223372036854775296 equals:
-1024 >>> 1.
327 )
328
329 testMin = (
330     "We need to test numbers that are 64bit or less, larger than 64bit,
331     positive, and negative"
332     | big small |
333     big := #(1 100 9223372036854775807 -50 -2147483648).
334     small := #(0 52 9223372036854775296 -51 -2147483650).
335
336     big doIndexes: [:i |
337         self assert: (small at: i) equals: ((big at: i) min: (small at:
338         i)).
339         self assert: (small at: i) equals: ((small at: i) min: (big at:
340         i)) ].
341     "not sure whether we should really insist on this"
342     big := #( 922337203685477580700 922337203685477580700
343             -922337203685477580700 922337203685477580700).
344     small := #( 922337203685477529600 1
345                -922337203685477580701 -922337203685477580701).
346     big doIndexes: [:i |
347         self optional: #toBeSpecified
348         assert: (small at: i) equals: ((big at: i) min: (small at:
349         i)).
350         self optional: #toBeSpecified
351         assert: (small at: i) equals: ((small at: i) min: (big at:
352         i)) ].
353 )
354
355 testMax = (
356     "We need to test numbers that are 64bit or less, larger than 64bit,
357     positive, and negative"
358     | big small |
359     big := #(1 100 9223372036854775807 -50 -2147483648).
360     small := #(0 52 9223372036854775296 -51 -2147483650).
361     big doIndexes: [:i |
362         self assert: (big at: i) equals: ((big at: i) max: (small at:
363         i)).
364         self assert: (big at: i) equals: ((small at: i) max: (big at:
365         i)) ].
366     big := #( 922337203685477580700
367             922337203685477580700
368             -922337203685477580700
369             922337203685477580700).
370     small := #( 922337203685477529600
371                1
372                -922337203685477580701
373                -922337203685477580701).
374     big doIndexes: [:i |
375         self optional: #toBeSpecified
376         assert: (big at: i) equals: ((big at: i) max: (small at: i)).
377         self optional: #toBeSpecified
378         assert: (big at: i) equals: ((small at: i) max: (big at:
379         i)) ].

```

376)
377)
378

```
1 "  
2  
3 $Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 "... something just a bit complicated that tests iteration with  
27 blocks, so that we might fail here rather than when the other tests  
28 start, in case things are broken."  
29  
30 PreliminaryTest = TestCase (  
31  
32     testBasicSanity = (  
33         | sum |  
34         sum := 0.  
35         1, 2, 3 do: [ :i |  
36             sum := sum + i.  
37             i<2 ifTrue: [ sum := sum*2 ].  
38             i>2 ifFalse: [ sum := sum*2 ] ].  
39         self assert: 15 equals: sum  
40     )  
41  
42 )  
43
```


TestSuite/ReflectionTest.som

```
1 "  
2  
3 $Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 ReflectionTest = TestCase (  
29   testRespondsTo = (  
30     self assert: (Object new respondsTo: #isNil).  
31     self assert: (23 respondsTo: #isNil).  
32     self assert: (23 respondsTo: #+).  
33   )  
34  
35   testMethods = (  
36     "First method in Object should be #class."  
37     self assert: #class equals: (Object methods at: 1) signature.  
38     self assert: (Object hasMethod: #==).  
39   )  
40  
41   testPerform = (  
42     | o |  
43     self assert: Integer equals: (23 perform: #class).  
44     self assert: (23 perform: #between:and: withArguments: (Array with: 22  
with: 24)).  
45  
46     o := SuperTest new.  
47     self assert: #super equals: (o perform: #something inSuperclass:  
SuperTestSuperClass).  
48  
49     "Trying to see whether the stack in bytecode-based SOMs works properly"  
50     self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a]  
ifFalse: [#b]).  
51
```

```

52     self assert: 28 equals: 5 + (23 perform: #value).
53 )
54
55 testInstVarAtAndPut = (
56     | tmp |
57     "Testing #at: and #at:put:"
58     tmp := Pair withKey: 3 andValue: 42.
59
60     self assert: tmp key equals: (tmp instVarAt: 1).
61
62     tmp instVarAt: 1 put: #foo.
63     self assert: #foo equals: tmp key.
64 )
65
66 testName = (
67     self assert: #Object equals: Object name.
68     self assert: #'Object class' equals: Object class name.
69     self assert: #Integer equals: 1 class name.
70 )
71
72 testAsString = (
73     self assert: 'Object' equals: Object asString.
74     self assert: 'Object class' equals: Object class asString.
75     self assert: 'Integer' equals: 1 class asString.
76 )
77
78 testSelectors = (
79     | sels |
80     sels := Object selectors.
81     self assert: 32 equals: sels length.
82
83     sels contains: #=.
84     self assert: (Object hasMethod: #=).
85
86     sels contains: #value.
87     self assert: (Object hasMethod: #value).
88
89     sels contains: #notNil.
90     self assert: (Object hasMethod: #notNil).
91 )
92 )
93

```

TestSuite/SelfBlockTest.som

```
1 "  
2  
3 $Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SelfBlockTest = TestCase (  
29  
30     testEscapedBlock = (  
31         self assert: 42 equals: self give42 value  
32     )  
33  
34     give42 = (  
35         ^[ self giveBlock value ]  
36     )  
37  
38     giveBlock = (  
39         ^self returnBlock value  
40     )  
41  
42     returnBlock = (  
43         ^[ self returnBlock2 value ]  
44     )  
45  
46     returnBlock2 = (  
47         ^[ 42 ]  
48     )  
49 )  
50
```

```

1 SetTest = TestCase (
2   testBasics = (
3     | a b t |
4     a := Set new.
5     b := Set new.
6
7     a add: #a.
8     b add: #b.
9
10    self deny: a = b.
11
12    t := Set new.
13    t add: #a.
14
15    self deny: a == t. "different objects"
16    self assert: a equals: t. "but with equal value"
17  )
18
19  testEquality = (
20    | a b |
21    a := Set new.
22    a addAll: #(1 2 3 4).
23
24    b := Set new.
25    b addAll: #(1 2 3 4).
26
27    self assert: a = b.
28    self deny: a == b.
29
30    a add: 5.
31
32    self deny: a = b.
33    b add: 5.
34
35    self assert: a = b.
36
37    b add: #foo.
38    self deny: a = b.
39    a add: #foo.
40    self assert: a = b.
41  )
42
43  testUnion = (
44    | a b u |
45    a := Set new.
46    b := Set new.
47    a addAll: #(1 2 3 4).
48    b addAll: #(1 2 3 4).
49
50    u := a union: b.
51    self assert: a = b.
52    self assert: u = a.
53    self assert: u = b.
54
55    self deny: a == b.
56    self deny: u == a.
57    self deny: u == b.
58

```

```

59     self assert: 4 equals: u size.
60
61     a add: #mm.
62     u := a union: b.
63     self assert: 5 equals: u size.
64
65     b add: #oo.
66     u := a union: b.
67     self assert: 6 equals: u size.
68
69     b add: #mm.
70     u := a union: b.
71     self assert: 6 equals: u size.
72 )
73
74 testIntersection = (
75     | a b i |
76     a := Set new.
77     b := Set new.
78     a addAll: #(43 64 730 667).
79     b addAll: #(43 64 730 667).
80
81     i := a intersection: b.
82     self assert: 4 equals: i size.
83
84     a do: [:e |
85         self assert: (i contains: e) ].
86
87     b := Set new.
88     b add: 64.
89     b add: 667.
90
91     i := a intersection: b.
92     self assert: 2 equals: i size.
93
94     b do: [:e |
95         self assert: (i contains: e) ].
96 )
97
98 testSetDifference = (
99     | a b d |
100    a := Set new.
101    b := Set new.
102    a addAll: #(43 64 730 667).
103    b addAll: #(43 64 730 667).
104
105    d := a - b.
106    self assert: d isEmpty.
107    self assert: 0 equals: d size.
108
109    b := Set new.
110    b add: 43.
111    b add: 667.
112    b add: 345345.
113
114    d := a - b.
115    self assert: 2 equals: d size.
116    self assert: (d contains: 64).
117    self assert: (d contains: 730).
118 )
119

```

```

120 testContains = (
121     | s |
122     s := Set new.
123
124     self deny: (s contains: #'333').
125     s add: #'333'.
126
127     self assert: (s contains: #'333').
128     s add: 333.
129     self assert: (s contains: #'333').
130     self assert: (s contains: 333).
131 )
132
133 testRemove = (
134     | s o |
135     o := Object new.
136     s := Set new.
137     s add: #sfsdf.
138     s add: 323.
139     s add: 545.
140     s add: self.
141     s add: o.
142
143     self assert: (s contains: o).
144     self assert: 5 equals: s size.
145
146     s remove: 323.
147     self assert: 4 equals: s size.
148     self deny: (s contains: 323).
149     s add: 65767.
150     self assert: 5 equals: s size.
151
152     s remove: o.
153     self assert: 4 equals: s size.
154     self deny: (s contains: o).
155     s remove: self.
156     s remove: #sfsdf.
157     s remove: 323.
158     s remove: 545.
159     s remove: 65767.
160
161     self assert: s isEmpty.
162 )
163
164 testFirst = (
165     | s |
166     s := Set new.
167     s addAll: #(233 545 665).
168
169     self assert: 233 equals: s first.
170
171     s remove: 233.
172     self assert: 545 equals: s first.
173
174     s remove: 545.
175     self assert: 665 equals: s first.
176 )
177
178 testCollect = (
179     | s r |
180     s := Set new.

```

```
181     s addAll: #(21 54642 6753 344 655).
182
183     r := s collect: [:e | e % 10 ].
184
185     self assert: 5 equals: r size.
186     self assert: 1 equals: r first.
187     self assert: 2 equals: (r at: 2).
188 )
189 )
190
```

```
1 SpecialSelectorsTest = TestCase (
2   testMinusMinsPrefix = (
3     self assert: self --> 1 equals: 1.
4     self assert: self -- 1 equals: 1.
5   )
6
7   --> aValue = (
8     ^1
9   )
10
11   -- aValue = (
12     •ã
13   )
14 )
15
```



```

1  "
2  Copyright (c) 2001-2013 see AUTHORS file
3
4  Permission is hereby granted, free of charge, to any person obtaining a
copy
5  of this software and associated documentation files (the 'Software'), to
deal
6  in the Software without restriction, including without limitation the
rights
7  to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
8  copies of the Software, and to permit persons to whom the Software is
9  furnished to do so, subject to the following conditions:
10
11 The above copyright notice and this permission notice shall be included in
12 all copies or substantial portions of the Software.
13
14 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
17 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
18 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
19 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
20 THE SOFTWARE.
21 "
22
23 StringTest = TestCase (
24
25   testEquality = (
26     | str1 str2 |
27     str1 := 'foo'.
28     str2 := 'bar'.
29
30     self assert: str1 = str1.
31     self assert: str1 = 'foo'.
32     self assert: str1 = ('f' + 'oo').
33     self deny: str1 = str2.
34     self assert: str2 = str2.
35
36     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
37     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
38     self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
39     self assert: ('f' + 'o' + 'o') = #foo.
40   )
41
42   testEqualEqual = (
43     | str1 |
44     str1 := 'foo'.
45     self assert: str1 == str1.
46     self deny: str1 == str1 asSymbol.
47     self deny: str1 == #foo.
48   )
49
50   testLength = (
51     self assert: 1 equals: 't' length.
52     self assert: 6 equals: ('foo' + 'bar') length.
53   )

```

```

54
55 testCharAt = (
56     | str |
57     str := 'foobar'.
58     self assert: 'f' equals: (str charAt: 1).
59     self assert: 'o' equals: (str charAt: 2).
60     self assert: 'o' equals: (str charAt: 3).
61     self assert: 'b' equals: (str charAt: 4).
62     self assert: 'a' equals: (str charAt: 5).
63     self assert: 'r' equals: (str charAt: 6).
64 )
65
66 testStringLiteralLineBreak = (
67     | str |
68     "Some parsers get the literals and line boundaries wrong"
69     str := '
70 '.
71     self assert: '\n' equals: (str charAt: 1).
72     self assert: 1 equals: str length.
73 )
74
75 testPrimSubstringFrom = (
76     | str |
77     str := 'foobar'.
78     self assert: 'foo' equals: (str primSubstringFrom: 1 to: 3).
79     self assert: 'bar' equals: (str primSubstringFrom: 4 to: 6).
80     self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
81     self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
82 )
83
84 testSplit = (
85     | r |
86     r := 'aaaa' split: ','.
87     self assert: 1 equals: r length.
88     self assert: 'aaaa' equals: (r at: 1).
89
90     r := 'foo.bar' split: '.'.
91     self assert: 2 equals: r length.
92     self assert: 'foo' equals: (r at: 1).
93     self assert: 'bar' equals: (r at: 2).
94
95     r := 'foo..bar' split: '..'.
96     self assert: 3 equals: r length.
97     self assert: 'foo' equals: (r at: 1).
98     self assert: '' equals: (r at: 2).
99     self assert: 'bar' equals: (r at: 3).
100
101     r := 'foo..bar' split: '...'.
102     self assert: 2 equals: r length.
103     self assert: 'foo' equals: (r at: 1).
104     self assert: 'bar' equals: (r at: 2).
105
106     r := 'foo' split: 'bar'.
107     self assert: 1 equals: r length.
108     self assert: 'foo' equals: (r at: 1).
109
110     self assert: Array is: r class
111 )
112
113 testIndexOf = (
114     self assert: -1 equals: ('foo' indexOf: 'b').

```

```

115     self assert: 1 equals: ('foo' indexOf: 'f').
116     self assert: 2 equals: ('foo' indexOf: 'o').
117     self assert: 3 equals: ('foo' indexOf: 'o' startingAt: 3).
118
119     self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).
120
121     self assert: 2 equals: ('foo' indexOf: 'oo').
122 )
123
124 testBeginsWith = (
125     self deny: ('foo' beginsWith: 'oo').
126     self assert: ('foo' beginsWith: 'foo').
127 )
128
129 testEndsWith = (
130     self assert: ('foo' endsWith: 'foo').
131     self assert: ('foo' endsWith: 'oo').
132     self deny: ('f' endsWith: 'bar').
133     self deny: ('f' endsWith: 'foo').
134 )
135
136 testMultiLineString = (
137     "Test whether the parser will parse multi-line strings correctly."
138     self assert: '
139 1234567890
140 1234567890
141 1234567890
142 1234567890
143 1234567890' equals: '
144 1234567890
145 1234567890
146 1234567890
147 1234567890
148 1234567890'
149 )
150
151 testEscapeSequences = (
152     "Tests for escape sequences, not all of them are reliable represented
153 as
154 proper strings. So, we do a simple equality test, and check
155 substring or
156 length.
157
158 \t'    F " 6† & 7FW
159 \b'    & 6·7 6R 6† & 7FW
160 \n'    æWvÆ-æR 6† & 7FW
161 \r'    6 '&- vR &WGW&â 6† & 7FW
162 \f'    f÷&ÖfVVB 6† & 7FW
163 \'     6-ævÆR V÷FR 6† & 7FW
164 \\'    & 6·6Æ 6, 6† & 7FW
165 \0     zero byte character
166
167 self assert: '\t' equals: '\t'.
168 self assert: 1 equals: '\t' length.
169
170 self assert: '\b' equals: '\b'.
171 self assert: 1 equals: '\b' length.
172
173 self assert: '\n' equals: '\n'.
174 self assert: 1 equals: '\n' length.

```

```

174     self deny: ('\n' endsWith: 'n').
175
176     self assert: '\r' equals: '\r'.
177     self assert: 1 equals: '\n' length.
178     self deny: ('\r' endsWith: 'r').
179
180     self assert: '\f' equals: '\f'.
181     self assert: 1 equals: '\f' length.
182     self deny: ('\f' endsWith: 'f').
183
184     self assert: '\\' equals: '\\'.
185     self assert: 1 equals: '\\' length.
186
187     self assert: '\\\\' equals: '\\\\'.
188     self assert: 1 equals: '\\\\' length.
189
190     self assert: '\\0' equals: '\\0'.
191     self assert: 1 equals: '\\0' length.
192     self assert: 5 equals: '\\0rest' length.
193 )
194
195 testHash = (
196     | str |
197     "Hash should be identical for strings that are identical,
198     whether given literal or composed at runtime"
199     self assert: 'foobar' hashCode equals: 'foobar' hashCode.
200     self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode
201         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashCode.
202
203     str := 'foo' + 'bar'.
204     str := str + str.
205     self assert: 'foobarfoobar' hashCode equals: str hashCode.
206
207     str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw
erltnwrtln'.
208     self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw
erltnwrtln' hashCode
209         equals: str hashCode.
210 )
211
212 testWhiteSpace = (
213     self assert: ' ' isWhiteSpace.
214     self assert: '\t' isWhiteSpace.
215     self assert: '\t\n \n \n' isWhiteSpace.
216
217     self deny: '' isWhiteSpace.
218     self deny: '\t\n N \n \n' isWhiteSpace.
219     self deny: 'N' isWhiteSpace.
220     self deny: '3' isWhiteSpace.
221 )
222
223 testLetters = (
224     self assert: 'a' isLetters.
225     self assert: 'all' isLetters.
226     self optional: #unicode assert: 'aOoöéÉíä' isLetters description:
'Does not support Unicode'.
227
228     self deny: '' isLetters.
229     self deny: ' ' isLetters.
230     self deny: '3' isLetters.
231     self deny: '3333' isLetters.

```

```

232     self deny: 'aOo öéÉíä' isLetters.
233     self deny: 'aOolöéÉíä' isLetters.
234 )
235
236 testDigits = (
237     self assert: '0' isDigits.
238     self assert: '0123' isDigits.
239     self assert: '0123456789' isDigits.
240
241     self deny: '' isDigits.
242     self deny: ' ' isDigits.
243     self deny: 'S' isDigits.
244     self deny: '333 3' isDigits.
245     self deny: '66i77' isDigits.
246     self deny: '66e7' isDigits.
247     self deny: 'aOolöéÉíä' isDigits.
248 )
249
250 testAsInteger = (
251     self assert: 0 equals: '0' asInteger.
252     self assert: 100 equals: '100' asInteger.
253     self assert: 923 equals: '923' asInteger.
254
255     self assert: -0 equals: '-0' asInteger.
256     self assert: -100 equals: '-100' asInteger.
257     self assert: -923 equals: '-923' asInteger.
258
259     self assert: 123342353453453456456456 equals:
'123342353453453456456456' asInteger.
260 )
261 )
262

```

```
1 "  
2  
3 Copyright (c) 2007-2018 see AUTHORS file  
4 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
5 http://www.hpi.uni-potsdam.de/swa/  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 SuperTest = SuperTestSuperClass (  
27  
28   testSuper = (  
29     self assert: 42 equals: self give42.  
30     self assert: 42 equals: self blockGive42.  
31   )  
32  
33   yourself = (  
34     record := record + 1000.  
35     ^ self  
36   )  
37  
38   give42 = (  
39     ^super give42  
40   )  
41  
42   blockGive42 = (  
43     ^[ super give42 ] value  
44   )  
45  
46   something = (  
47     ^ #sub  
48   )  
49  
50   number = (  
51     ^ 10  
52   )  
53
```

```

54   + other = (
55       ^ 11
56   )
57
58   ++++ other = (
59       ^ 111
60   )
61
62   keyword: other = (
63       ^ 1111
64   )
65
66   testBasicUnary = (
67       self assert: 10 equals: self number.
68       self assert: 1 equals: super number.
69   )
70
71   testBasicBinary = (
72       self assert: 11 equals: self + 3.
73       self assert: 22 equals: super + 5.
74   )
75
76
77   testBasicBinaryNonStandardOperator = (
78       self assert: 111 equals: self ++++ 3.
79       self assert: 222 equals: super ++++ 5.
80   )
81
82   testBasicKeyword = (
83       self assert: 1111 equals: (self keyword: 3).
84       self assert: 2222 equals: (super keyword: 5).
85   )
86
87   testWithBinaryUnaryMessage = (
88       | val |
89       record := 0.
90       val := super number * super number.
91       self assert: 1 equals: val.
92   )
93
94   testWithBinaryUnaryUnaryMessage = (
95       | val |
96       record := 0.
97       super yourself yourself @ super yourself yourself.
98       self assert: 2002 equals: record.
99   )
100
101   testWithKeywordUnaryUnaryMessage = (
102       | val |
103       record := 0.
104       super key: super yourself yourself key: super yourself yourself.
105       self assert: 2002 equals: record.
106
107       record := 0.
108       self key: super yourself yourself key: super yourself yourself.
109       self assert: 2002 equals: record.
110   )
111
112   "Note: testing assigning self was moved to basic interpreter tests"
113
114   testGlobalSelfDoesNotShadowKeyword = (

```

```
115     | that |
116     that := self.
117     system global: #self put: 42.
118     that optional: #selfSuperBug assert: that is: self.
119
120     self assert: 42 equals: (system global: #self)
121 )
122
123 testGlobalSuperDoesNotShadowKeyword = (
124     | that |
125     that := super.
126     system global: #super put: 42.
127     that optional: #selfSuperBug assert: that is: super.
128
129     self assert: 42 equals: (system global: #super)
130 )
131 )
132
```


TestSuite/SuperTestSuperClass.som

```
1 "  
2  
3 $Id: SuperTestSuperClass.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 SuperTestSuperClass = TestCase (  
29   | record |  
30  
31   yourself = (  
32     record := record + 1.  
33     ^ self  
34   )  
35  
36   give42 = (  
37     ^ 42  
38   )  
39  
40   something = (  
41     ^ #super  
42   )  
43  
44   number = (  
45     ^ 1  
46   )  
47  
48   + other = (  
49     ^ 22  
50   )  
51  
52   ++++ other = (  
53     ^ 222  
54   )
```

```
55
56     keyword: other = (
57         ^ 2222
58     )
59
60     key: a key: b = (
61         ^ self
62     )
63
64     @ o = ( ^ self )
65 )
66
67
```

```

1  "
2
3  $Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $
4
5  Copyright (c) 2007-2013 see AUTHORS file
6  Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
7  http://www.hpi.uni-potsdam.de/swa/
8
9  Permission is hereby granted, free of charge, to any person obtaining a
copy
10 of this software and associated documentation files (the 'Software'), to
deal
11 in the Software without restriction, including without limitation the
rights
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
13 copies of the Software, and to permit persons to whom the Software is
14 furnished to do so, subject to the following conditions:
15
16 The above copyright notice and this permission notice shall be included in
17 all copies or substantial portions of the Software.
18
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
25 THE SOFTWARE.
26 "
27
28 SymbolTest = TestCase (
29
30   testConcatenation = (
31     self assert: #a + #b is: #ab.
32     self assert: #a + 'b' is: #ab.
33   )
34
35   testConversion = (
36     self assert: 'gunk' equals: 'gunk' asSymbol asString.
37     self assert: 'oink' equals: #oink asString.
38   )
39
40   testEquality = (
41     self assert: #oink is: #oink.
42     self assert: #oink is: 'oink' asSymbol.
43     self assert: #oink equals: #oink.
44     self assert: #oink equals: 'oink' asSymbol.
45
46     self deny: #foo equals: #fooo.
47     self deny: #foo is:      #fooo.
48
49     self assert: #foo equals: 'foo'.
50     self deny: #foo is: 'fooo'.
51     self deny: #foo is: #foo asString.
52   )
53
54   testSymbolIsString = (

```

```

55     self assert: (#oink beginsWith: 'oink').
56     self assert: 100 equals: #'100' asInteger.
57     self assert: String equals: #foo class superclass
58 )
59
60 testOperatorSymbols = (
61     self assert: #~ equals: '~' asSymbol.
62     self assert: #& equals: '&' asSymbol.
63     self assert: #| equals: '|' asSymbol.
64     self assert: #* equals: '*' asSymbol.
65     self assert: #/ equals: '/' asSymbol.
66     self assert: #\ equals: '\\' asSymbol.
67     self assert: #+ equals: '+' asSymbol.
68     self assert: #= equals: '=' asSymbol.
69     self assert: #> equals: '>' asSymbol.
70     self assert: #< equals: '<' asSymbol.
71     self assert: #, equals: ',' asSymbol.
72     self assert: #@ equals: '@' asSymbol.
73     self assert: #% equals: '%' asSymbol.
74     self assert: #- equals: '-' asSymbol.
75 )
76 )
77

```

TestSuite/SystemTest.som

```
1 SystemTest = TestCase (
2
3   testFullGCsupport = (
4     "Test whether #fullGC is support. We expect the VM now to return true,
5     to indicate the a GC was done."
6     self optional: #fullGCWithEffect assert: system fullGC description:
'#fullGC is not supported or has not immediate effect.'
7   )
8
9   testTicks = (
10    | ticks |
11    ticks := system ticks.
12    self assert: ticks class equals: Integer.
13    self assert: ticks > 0 description: 'Should return the microseconds
since the start'
14  )
15 )
16
```

```

1 TestCase = (
2   | testSelector runner failed |
3
4   selector      = ( ^ testSelector )
5   selector: aSym = ( testSelector := aSym )
6
7   "asserting"
8   assert: aBoolean = (
9     runner countAssert.
10    aBoolean ifFalse: [
11      self signalFailure: 'Assertion failed' ] )
12
13   assert: aBoolean description: aStringOrBlock = (
14     runner countAssert.
15     aBoolean ifFalse: [
16       self signalFailure: aStringOrBlock value ] )
17
18   assert: expected equals: actual = (
19     "test value equality"
20     self assert: (expected = actual)
21     description: [self comparingStringBetween: expected and:
actual]
22   )
23
24   assert: expected equals: actual description: aStringOrBlock = (
25     "test value equality"
26     self assert: (expected = actual)
27     description: aStringOrBlock
28   )
29
30   assert: expected is: actual = (
31     "test reference equality"
32     self assert: (expected == actual)
33     description: [self comparingStringBetween: expected and:
actual]
34   )
35
36   optional: aSymbol assert: aBoolean = (
37     runner countAssert.
38     aBoolean ifFalse: [
39       self signalUnsupported: aSymbol description: nil ] )
40
41   optional: aSymbol assert: expected equals: actual = (
42     self optional: aSymbol
43     assert: (expected = actual)
44     description: [self comparingStringBetween: expected and:
actual]
45   )
46
47   optional: aSymbol assert: expected is: actual = (
48     self optional: aSymbol
49     assert: (expected == actual)
50     description: [self comparingStringBetween: expected and:
actual]
51   )
52
53   optional: aSymbol assert: aBoolean description: aStringOrBlock = (
54     runner countAssert.

```

```

55         aBoolean ifFalse: [
56             self signalUnsupported: aSymbol description: aStringOrBlock
value ] )
57
58     deny: aBoolean = (
59         self assert: aBoolean not
60     )
61
62     deny: aBooleanOrBlock description: aString = (
63         self assert: aBooleanOrBlock value not description: aString
64     )
65
66     deny: expected equals: actual = (
67         "test value equality"
68         self deny: (expected = actual)
69         description: [
70             'Expected ' + expected asString +
71             ' to differ from ' + actual asString + '.' ]
72     )
73
74     deny: expected is: actual = (
75         "test value equality"
76         self deny: (expected == actual)
77         description: [
78             'Expected ' + expected asString +
79             ' to have different identity from ' + actual asString +
'.' ]
80     )
81
82     optional: aSymbol deny: aBoolean = (
83         self optional: aSymbol assert: aBoolean not
84     )
85
86     optional: aSymbol deny: aBooleanOrBlock description: aString = (
87         self optional: aSymbol assert: aBooleanOrBlock value not
description: aString
88     )
89
90     signalFailure: aString = (
91         failed := true.
92         runner fail: self class name + '>>#' + testSelector
93             because: aString.
94     )
95
96     signalUnsupported: aSymbol description: aDescription = (
97         runner unsupported: aSymbol
98             test: self class name + '>>#' + testSelector
99             because: aDescription.
100     )
101
102     comparingStringBetween: expected and: actual = (
103         ^ 'Expected ' + expected asString +
104         ' but was ' + actual asString + '.'
105     )
106
107     "running"
108     run: aRunner = (
109         runner := aRunner.
110         failed := false.
111
112         self setUp.

```

```

113         self performTest.
114         self tearDown.
115
116         failed ifFalse: [
117             runner passed: self class name + '>>#' + testSelector
118         ].
119     )
120
121     setUp      = ()
122     tearDown = ()
123
124     performTest = ( self perform: testSelector )
125
126     ----
127
128     for: aSelector = (
129         | case |
130         case := self new.
131         case selector: aSelector.
132         ^ case
133     )
134
135     tests = (
136         | tests |
137         tests := Vector new: self methods length.
138         self methods do: [:m |
139             (m signature beginsWith: #test) ifTrue: [
140                 tests append: (self for: m signature).
141             ].
142         ].
143         ^ tests
144     )
145 )
146 )
147

```


TestSuite/TestHarness.som

```
1 "  
2  
3 $Id: TestHarness.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2001-2013 see AUTHORS file  
6  
7 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
8 of this software and associated documentation files (the 'Software'), to  
deal  
9 in the Software without restriction, including without limitation the  
rights  
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
11 copies of the Software, and to permit persons to whom the Software is  
12 furnished to do so, subject to the following conditions:  
13  
14 The above copyright notice and this permission notice shall be included in  
15 all copies or substantial portions of the Software.  
16  
17 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
23 THE SOFTWARE.  
24 "  
25  
26 TestHarness = (  
27     | failOnUnsupportedOptionals |  
28  
29  
30     tests = ( "Now ordered by alphabetical order to improve  
maintainability"  
31         ^ EmptyTest,  
32           SpecialSelectorsTest,  
33           ArrayTest,  
34           BlockTest,  
35           BooleanTest,  
36           ClassLoadingTest,  
37           ClassStructureTest,  
38           ClosureTest,  
39           CoercionTest,  
40           CompilerReturnTest,  
41           DictionaryTest,  
42           DoesNotUnderstandTest,  
43           DoubleTest,  
44           GlobalTest,  
45           HashTest,  
46           IntegerTest,  
47           PreliminaryTest,  
48           ReflectionTest,  
49           SelfBlockTest,  
50           SetTest,  
51           StringTest,  
52           SuperTest,
```

```

53         SymbolTest,
54         SystemTest,
55         VectorTest
56     )
57
58
59     runAllSuites = (
60         | totalTestNum successfulTestNum unsupportedTestNum
totalAssertionNum |
61         totalTestNum := 0.
62         unsupportedTestNum := 0.
63         successfulTestNum := 0.
64         totalAssertionNum := 0.
65
66         self tests do: [ :test |
67             | runner |
68             runner := TestRunner new.
69             runner initializeOn: test.
70             runner runAllTests.
71             (runner hasUnsupported or: [runner hasFailures])
72             ifTrue: [
73                 'Test Suite: ' print.
74                 test name println.
75                 runner overviewReport.
76                 '' println ].
77
78                 totalTestNum := totalTestNum + runner expectedPasses.
79                 unsupportedTestNum := unsupportedTestNum + runner
actualUnsupported.
80                 successfulTestNum := successfulTestNum + runner actualPasses.
81                 totalAssertionNum := totalAssertionNum + runner numAsserts.
82             ].
83
84             'Total number of tests:          ' print.
85             totalTestNum println.
86             'Number of unsupported optionals: ' print.
87             unsupportedTestNum println.
88             'Number of successful tests:      ' print.
89             successfulTestNum println.
90             'Number of assertions tested:    ' print.
91             totalAssertionNum println.
92
93             (failOnUnsupportedOptionals and: [unsupportedTestNum > 0])
94             ifTrue: [system exit: 1].
95             totalTestNum = successfulTestNum
96             ifFalse: [system exit: 1].
97         )
98
99     runOneSuite: name = (
100         | testName runner |
101         testName := name.
102         (testName endsWith: 'Test') ifFalse: [
103             testName := testName + 'Test'].
104
105         runner := TestRunner new.
106         runner initializeOn: (system resolve: testName asSymbol).
107         runner run.
108         runner hasFailures ifTrue: [system exit: 1]
109     )
110
111     run: args = (

```

```

112         failOnUnsupportedOptionals := false.
113
114         args length = 1 ifTrue: [ self runAllSuites. ].
115         args length = 2 ifTrue: [
116             ((args at: 2) beginsWith: '--') ifTrue: [
117                 (args at: 2) = '--help' ifTrue: [
118                     'TestHarness.som [--help] [--fail-on-optionals]
[TestSuiteName]' println.
119                     system exit: 0 ].
120
121                 (args at: 2) = '--fail-on-optionals' ifTrue: [
122                     failOnUnsupportedOptionals := true ].
123
124                 self runAllSuites
125             ] ifFalse: [
126                 self runOneSuite: (args at: 2) ].
127         ^ 0
128     )
129 )
130

```

```

1 TestRunner = (
2   | suite passes unsupported failures numAsserts |
3
4   initializeOn: aSuite = (
5     suite := aSuite.
6
7     passes      := Vector new.
8     unsupported  := Vector new.
9     failures     := Vector new.
10
11     numAsserts := 0.
12   )
13
14   hasUnsupported = ( ^ unsupported size > 0 )
15   hasFailures    = ( ^ failures size > 0 )
16
17   actualUnsupported = ( ^ unsupported size )
18   expectedPasses    = ( ^ suite tests size )
19   actualPasses      = ( ^ passes size )
20
21   run = (
22     self reportPreRun.
23     self runAllTests.
24     self reportPostRun.
25     self overviewReport.
26   )
27
28   countAssert = (
29     numAsserts := numAsserts + 1.
30   )
31
32   numAsserts = (
33     ^ numAsserts
34   )
35
36   reportPreRun = (
37     ('TestSuite ' + suite name + ':') println.
38     ('Tests: ' + suite tests size asString) println.
39   )
40
41   reportPostRun = (
42     self hasUnsupported ifTrue: [
43       ('Unsupported optional: ' + unsupported size asString) println
44     ].
45     self hasFailures ifTrue: [
46       ('Failures: ' + failures size asString) println
47     ].
48   )
49
50   runAllTests = (
51     suite tests do: [ :each |
52       each run: self ].
53   )
54
55   overviewReport = (
56     ('Tests passed: ' + passes size asString) println.
57
58     (self hasFailures or: [self hasUnsupported]) ifTrue: [

```

```

59         '-----' println ].
60
61     self hasUnsupported ifTrue: [
62         | lastCategory |
63         ('Unsupported optional features: ' + unsupported size asString)
println.
64         unsupported do: [:each |
65             | cat |
66             cat := each at: 1.
67             cat == lastCategory ifFalse: [
68                 lastCategory := cat.
69                 ('\t' + cat) println ].
70             ('\t\t' + (each at: 2) asString) println.
71             ('\t\t\t' + (each at: 3) value asString) println ].
72     ].
73
74     self hasFailures ifTrue: [
75         ('Failures: ' + failures size asString) println.
76         failures do: [:each |
77             (' ' + each key asString) println.
78             (' ' + each value asString) println ].
79     ].
80 )
81
82 fail: aSignature because: aReason = (
83     | pair |
84     pair := Pair withKey: aSignature andValue: aReason.
85     failures append: pair.
86 )
87
88 unsupported: aSymbol test: aSignature because: aReason = (
89     | array |
90     array := Array with: aSymbol with: aSignature with: aReason.
91     unsupported append: array.
92 )
93
94 passed: aSignature = (
95     passes append: aSignature
96 )
97 )
98

```

TestSuite/VectorTest.som

```
1 "  
2  
3 $Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $  
4  
5 Copyright (c) 2007-2013 see AUTHORS file  
6 Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany  
7 http://www.hpi.uni-potsdam.de/swa/  
8  
9 Permission is hereby granted, free of charge, to any person obtaining a  
copy  
10 of this software and associated documentation files (the 'Software'), to  
deal  
11 in the Software without restriction, including without limitation the  
rights  
12 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
13 copies of the Software, and to permit persons to whom the Software is  
14 furnished to do so, subject to the following conditions:  
15  
16 The above copyright notice and this permission notice shall be included in  
17 all copies or substantial portions of the Software.  
18  
19 THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
20 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
21 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL  
THE  
22 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
23 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING  
FROM,  
24 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
25 THE SOFTWARE.  
26 "  
27  
28 VectorTest = TestCase (  
29   | a |  
30  
31   setUp = (  
32     a := Vector new.  
33     a append: 'hello'.  
34     a append: #world.  
35     a append: 23.  
36   )  
37  
38   testSize = (  
39     self assert: 3 equals: a size.  
40   )  
41  
42   testAt = (  
43     self assert: #world equals: (a at: 2).  
44     self assert:      23 equals: (a at: 3).  
45   )  
46  
47   testAtPut = (  
48     self assert: 'hello' equals: (a at: 1).  
49     a at: 1 put: 11.  
50  
51     self assert: 11 equals: (a at: 1).  
52  
53     self assert: #world equals: (a at: 2).
```

```

54     self assert:      23 equals: (a at: 3).
55
56     a at: 3 put: 33.
57     self assert: 33 equals: (a at: 3).
58 )
59
60 testAtAfterRemoveFirst = (
61     self assert: 'hello' equals: (a at: 1).
62     a removeFirst.
63     self assert: #world equals: (a at: 1).
64     self assert:      23 equals: (a at: 2).
65 )
66
67 testAtPutAfterRemoveFirst = (
68     a removeFirst.
69     a at: 1 put: 11.
70     a at: 2 put: 22.
71
72     self assert: 11 equals: (a at: 1).
73     self assert: 22 equals: (a at: 2).
74
75     self assert: 2 equals: a size
76 )
77
78 testFirst = (
79     | v |
80     self assert: 'hello' equals: a first.
81
82     v := Vector new.
83     1 to: 10 do: [:i |
84         v append: i.
85         self assert: 1 equals: v first ].
86
87     1 to: 10 do: [:i |
88         self assert: i equals: v first.
89         v removeFirst ]
90 )
91
92 testLast = (
93     | v |
94     self assert: 23 equals: a last.
95
96     v := Vector new.
97     1 to: 10 do: [:i |
98         v append: i.
99         self assert: i equals: v last ].
100
101     10 downTo: 1 do: [:i |
102         self assert: i equals: v last.
103         v remove ]
104 )
105
106 testContains = (
107     self assert: (a contains: 'hello').
108     self assert: (a contains: #world).
109     self assert: (a contains: 23).
110     self deny: (a contains: #nono).
111 )
112
113 testContainsAfterRemoveFirst = (
114     a removeFirst.

```

```

115
116     self deny: (a contains: 'hello').
117     self assert: (a contains: #world).
118     self assert: (a contains: 23).
119     self deny: (a contains: #nono).
120
121     a removeFirst.
122
123     self deny: (a contains: 'hello').
124     self deny: (a contains: #world).
125     self assert: (a contains: 23).
126     self deny: (a contains: #nono).
127
128     a removeFirst.
129
130     self deny: (a contains: 'hello').
131     self deny: (a contains: #world).
132     self deny: (a contains: 23).
133     self deny: (a contains: #nono).
134 )
135
136 testContainsAfterRemovals = (
137     self deny: (a contains: nil).
138
139     a removeFirst.
140     self deny: (a contains: nil).
141
142     a remove.
143     self deny: (a contains: nil).
144
145     self assert: (a contains: #world).
146 )
147
148 testAppendAndRemoveFirst = (
149     | v |
150     v := Vector new: 10.
151     1 to: 100 do: [:i |
152         v append: i ].
153
154     "160 is implementation dependent, just here for orientation"
155     self assert: 160 equals: v capacity.
156     self assert: 100 equals: v size.
157
158     1 to: 100 do: [:i |
159         v removeFirst ].
160     1 to: 100 do: [:i |
161         v append: i ].
162
163     self assert: 320 equals: v capacity.
164     self assert: 100 equals: v size.
165 )
166
167 testIndexOf = (
168     | v |
169     v := Vector new: 3.
170     1 to: 17 do: [:i |
171         v append: i ].
172
173     self assert: -1 equals: (v indexOf: nil).
174     self assert: 1 equals: (v indexOf: 1).
175

```



```

176     self assert: 13 equals: (v indexOf: 13).
177     v at: 13 put: #test.
178
179     self assert: 13 equals: (v indexOf: #test).
180
181     v removeFirst.
182     self assert: -1 equals: (v indexOf: 1).
183
184     1 to: 12 do: [:i |
185         v removeFirst ].
186     self assert: -1 equals: (v indexOf: #test).
187 )
188
189 testAppendAll = (
190     | v c i |
191     v := Vector new: 2.
192     v append: 1.
193     v append: 2.
194     v append: 3.
195
196     c := Array with: 4 with: 5 with: 6.
197
198     v appendAll: c.
199     i := 1.
200
201     v do: [:e |
202         self assert: i equals: e.
203         i := i + 1 ]
204 )
205
206 testAsArray = (
207     | v arr |
208     v := Vector new.
209     self assert: 0 equals: v asArray length.
210
211     v append: 1.
212     v append: 2.
213
214     arr := v asArray.
215     self assert: 2 equals: arr length.
216     self assert: 1 equals: (arr at: 1).
217     self assert: 2 equals: (arr at: 2).
218 )
219
220 testAsArrayAfterRemoveFirst = (
221     | v arr |
222     v := Vector new.
223     v append: 1.
224     v append: 2.
225     v append: 5.
226
227     v removeFirst.
228
229     arr := v asArray.
230     self assert: 2 equals: arr length.
231     self assert: 2 equals: (arr at: 1).
232     self assert: 5 equals: (arr at: 2).
233 )
234
235 testAsSet = (
236     | v set |

```

```

237     v := Vector new.
238     v append: 1.
239     v append: 2.
240     v append: 3.
241     v append: 4.
242     self assert: 4 equals: v size.
243
244     set := v asSet.
245     self assert: 4 equals: set size.
246
247     v append: 1.
248     v append: 1.
249     v append: 1.
250
251     self assert: 4 + 3 equals: v size.
252
253     set := v asSet.
254     self assert: 4 equals: set size.
255 )
256
257 testIsEmpty = (
258     | v |
259     v := Vector new.
260     self assert: v isEmpty.
261
262     v append: 1.
263     self deny: v isEmpty.
264
265     v removeFirst.
266     self assert: v isEmpty.
267
268     v append: #ee.
269     self deny: v isEmpty.
270
271     v removeFirst.
272     self assert: v isEmpty.
273 )
274
275 testRemoveObj = (
276     | v |
277     v := Vector new.
278     v append: #a.
279     v append: #b.
280     v append: #c.
281     v append: #d.
282     v append: #e.
283     v append: #f.
284     v append: #g.
285     v append: #h.
286
287     self assert: 8 equals: v size.
288
289     self deny: (v remove: #aa).
290     self assert: (v remove: #e).
291     self assert: 7 equals: v size.
292 )
293
294 testAppendComma = (
295     | v |
296     v := Vector new.
297     v, #a.

```

```

298     v, #b.
299
300     self assert: 2 equals: v size.
301     self assert: (v contains: #a).
302     self assert: (v contains: #b).
303 )
304
305 testDoIndexes = (
306     | i v |
307     v := Vector new.
308     v doIndexes: [:j |
309         self assert: false ].
310
311     v appendAll: #(1 2 3 4 5).
312     i := 1.
313     v doIndexes: [:j |
314         self assert: i equals: j.
315         i := i + 1.
316     ].
317     self assert: 6 equals: i.
318 )
319
320 testDoIndexesAfterRemoveFirst = (
321     | i v |
322     v := Vector new.
323     v appendAll: #(1 2 3 4 5).
324
325     v removeFirst.
326
327     i := 1.
328     v doIndexes: [:j |
329         self assert: i equals: j.
330         i := i + 1 ].
331
332     self assert: 5 equals: i.
333 )
334
335 testDo = (
336     | i v |
337     v := Vector new.
338     v appendAll: #(1 2 3 4 5).
339     i := 1.
340     v do: [:v |
341         self assert: i equals: v.
342         i := i + 1.
343     ].
344     self assert: 6 equals: i.
345 )
346 )
347

```

codespeed.conf

```
1 # -*- mode: yaml -*-
2 # Config file for ReBench
3 default_experiment: all
4 default_data_file: 'codespeed.data'
5
6 reporting:
7   codespeed:
8     url: https://som-speed.stefan-marr.de/result/add/json/
9
10 runs:
11   max_invocation_time: 6000
12
13 benchmark_suites:
14   macro-startup:
15     gauge_adapter: RebenchLog
16     command: &MACRO_CMD "-cp Smalltalk:/home/smarr/.local/SOM/
Examples/Benchmarks/Richards:/home/smarr/.local/SOM/Examples/Benchmarks/
DeltaBlue:/home/smarr/.local/SOM/Examples/Benchmarks/NBody:/home/smarr/.local/
SOM/Examples/Benchmarks/Json:/home/smarr/.local/SOM/Examples/Benchmarks/
GraphSearch /home/smarr/.local/SOM/Examples/Benchmarks/BenchmarkHarness.som
%(benchmark)s "
17     benchmarks:
18       - Richards:
19         extra_args: "1 0 1"
20         codespeed_name: "Richards [>"
21       - DeltaBlue:
22         extra_args: "1 0 1000"
23         codespeed_name: "DeltaBlue [>"
24       - Mandelbrot:
25         extra_args: "1 0 300"
26         codespeed_name: "Mandelbrot [>"
27       - NBody:
28         extra_args: "1 0 30000"
29         codespeed_name: "NBody [>"
30       - Json:
31         extra_args: "1 0 80"
32         codespeed_name: "Json [>"
33       - GraphSearch:
34         extra_args: "1 0 30"
35         codespeed_name: "GraphSearch [>"
36       - PageRank:
37         extra_args: "1 0 1400"
38         codespeed_name: "PageRank [>"
39
40   macro-steady:
41     gauge_adapter: RebenchLog
42     command: *MACRO_CMD
43     benchmarks:
44       - Richards:
45         extra_args: "130 0 60"
46         codespeed_name: "Richards >]"
47         warmup: 30
48       - DeltaBlue:
49         extra_args: "120 0 20000"
50         codespeed_name: "DeltaBlue >]"
51         warmup: 20
52       - Mandelbrot:
53         extra_args: "110 0 1000"
```

```

54         codespeed_name: "Mandelbrot >]"
55         warmup: 10
56     - NBody:
57         extra_args: "120 0 500000"
58         codespeed_name: "NBody >]"
59         warmup: 20
60     - Json:
61         extra_args: "120 0 80"
62         codespeed_name: "Json >]"
63         warmup: 20
64     - GraphSearch:
65         extra_args: "250 0 30"
66         codespeed_name: "GraphSearch >]"
67         warmup: 100
68     - PageRank:
69         extra_args: "120 0 1400"
70         codespeed_name: "PageRank >]"
71         warmup: 20
72
73     micro-startup-100:
74         gauge_adapter: RebenchLog
75         command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/
Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/
BenchmarkHarness.som %(benchmark)s "
76         benchmarks:
77             - Fibonacci:
78                 extra_args: "1 0 100"
79                 codespeed_name: "Fibonacci 100x [>]"
80             - Dispatch:
81                 extra_args: "1 0 1000"
82                 codespeed_name: "Dispatch 100x [>]"
83             - Bounce:
84                 extra_args: "1 0 100"
85                 codespeed_name: "Bounce 100x [>]"
86             - Loop:
87                 extra_args: "1 0 500"
88                 codespeed_name: "Loop 100x [>]"
89             - Permute:
90                 extra_args: "1 0 50"
91                 codespeed_name: "Permute 100x [>]"
92             - Queens:
93                 extra_args: "1 0 50"
94                 codespeed_name: "Queens 100x [>]"
95             - List:
96                 extra_args: "1 0 50"
97                 codespeed_name: "List 100x [>]"
98             - Recurse:
99                 extra_args: "1 0 100"
100                 codespeed_name: "Recurse 100x [>]"
101             - Storage:
102                 extra_args: "1 0 20"
103                 codespeed_name: "Storage 100x [>]"
104             - Sieve:
105                 extra_args: "1 0 100"
106                 codespeed_name: "Sieve 100x [>]"
107             - BubbleSort:
108                 extra_args: "1 0 100"
109                 codespeed_name: "BubbleSort 100x [>]"
110             - QuickSort:
111                 extra_args: "1 0 20"
112                 codespeed_name: "QuickSort 100x [>]"

```

```

113         - Sum:
114             extra_args: "1 0 500"
115             codespeed_name: "Sum 100x [>]"
116         - Towers:
117             extra_args: "1 0 20"
118             codespeed_name: "Towers 100x [>]"
119         - TreeSort:
120             extra_args: "1 0 10"
121             codespeed_name: "TreeSort 100x [>]"
122         - IntegerLoop:
123             extra_args: "1 0 400"
124             codespeed_name: "IntegerLoop 100x [>]"
125         - FieldLoop:
126             extra_args: "1 0 50"
127             codespeed_name: "FieldLoop 100x [>]"
128         - WhileLoop:
129             extra_args: "1 0 1000"
130             codespeed_name: "WhileLoop 100x [>]"
131
132     micro-startup:
133         gauge_adapter: RebenchLog
134         command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/
Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/
BenchmarkHarness.som %(benchmark)s "
135         benchmarks:
136             - Fibonacci:
137                 extra_args: "1 0 3"
138                 codespeed_name: "Fibonacci [>]"
139             - Dispatch:
140                 extra_args: "1 0 20"
141                 codespeed_name: "Dispatch [>]"
142             - Bounce:
143                 extra_args: "1 0 2"
144                 codespeed_name: "Bounce [>]"
145             - Loop:
146                 extra_args: "1 0 10"
147                 codespeed_name: "Loop [>]"
148             - Permute:
149                 extra_args: "1 0 3"
150                 codespeed_name: "Permute [>]"
151             - Queens:
152                 extra_args: "1 0 2"
153                 codespeed_name: "Queens [>]"
154             - List:
155                 extra_args: "1 0 2"
156                 codespeed_name: "List [>]"
157             - Recurse:
158                 extra_args: "1 0 3"
159                 codespeed_name: "Recurse [>]"
160             - Storage:
161                 extra_args: "1 0 2"
162                 codespeed_name: "Storage [>]"
163             - Sieve:
164                 extra_args: "1 0 5"
165                 codespeed_name: "Sieve [>]"
166             - BubbleSort:
167                 extra_args: "1 0 3"
168                 codespeed_name: "BubbleSort [>]"
169             - QuickSort:
170                 extra_args: "1 0 3"
171                 codespeed_name: "QuickSort [>]"

```

```

172         - Sum:
173             extra_args: "1 0 10"
174             codespeed_name: "Sum [>"
175         - Towers:
176             extra_args: "1 0 2"
177             codespeed_name: "Towers [>"
178         - TreeSort:
179             extra_args: "1 0 1"
180             codespeed_name: "TreeSort [>"
181         - IntegerLoop:
182             extra_args: "1 0 8"
183             codespeed_name: "IntegerLoop [>"
184         - FieldLoop:
185             extra_args: "1 0 3"
186             codespeed_name: "FieldLoop [>"
187         - WhileLoop:
188             extra_args: "1 0 30"
189             codespeed_name: "WhileLoop [>"
190
191     micro-steady-100:
192         gauge_adapter: RebenchLog
193         command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/
Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/
BenchmarkHarness.som %(benchmark)s "
194         benchmarks:
195             - Fannkuch:
196                 extra_args: "55 0 9"
197                 codespeed_name: "Fannkuch 100x >]"
198                 warmup: 5
199             - Fibonacci:
200                 extra_args: "60 0 1000"
201                 codespeed_name: "Fibonacci 100x >]"
202                 warmup: 10
203             - Dispatch:
204                 extra_args: "55 0 10000"
205                 codespeed_name: "Dispatch 100x >]"
206                 warmup: 5
207             - Bounce:
208                 extra_args: "60 0 4000"
209                 codespeed_name: "Bounce 100x >]"
210                 warmup: 10
211             - Loop:
212                 extra_args: "55 0 10000"
213                 codespeed_name: "Loop 100x >]"
214                 warmup: 5
215             - Permute:
216                 extra_args: "55 0 1500"
217                 codespeed_name: "Permute 100x >]"
218                 warmup: 5
219             - Queens:
220                 extra_args: "55 0 1000"
221                 codespeed_name: "Queens 100x >]"
222                 warmup: 5
223             - List:
224                 extra_args: "65 0 1000"
225                 codespeed_name: "List 100x >]"
226                 warmup: 15
227             - Recurse:
228                 extra_args: "65 0 2000"
229                 codespeed_name: "Recurse 100x >]"
230                 warmup: 15

```

```

231         - Storage:
232             extra_args: "60 0 1000"
233             codespeed_name: "Storage 100x >]"
234             warmup: 10
235         - Sieve:
236             extra_args: "60 0 2500"
237             codespeed_name: "Sieve 100x >]"
238             warmup: 10
239         - BubbleSort:
240             extra_args: "60 0 3000"
241             codespeed_name: "BubbleSort 100x >]"
242             warmup: 10
243         - QuickSort:
244             extra_args: "60 0 2000"
245             codespeed_name: "QuickSort 100x >]"
246             warmup: 10
247         - Sum:
248             extra_args: "55 0 10000"
249             codespeed_name: "Sum 100x >]"
250             warmup: 5
251         - Towers:
252             extra_args: "55 0 1000"
253             codespeed_name: "Towers 100x >]"
254             warmup: 5
255         - TreeSort:
256             extra_args: "60 0 1000"
257             codespeed_name: "TreeSort 100x >]"
258             warmup: 10
259         - IntegerLoop:
260             extra_args: "55 0 8000"
261             codespeed_name: "IntegerLoop 100x >]"
262             warmup: 5
263         - FieldLoop:
264             extra_args: "55 0 900"
265             codespeed_name: "FieldLoop 100x >]"
266             warmup: 5
267         - WhileLoop:
268             extra_args: "55 0 9000"
269             codespeed_name: "WhileLoop 100x >]"
270             warmup: 5
271     micro-steady:
272         gauge_adapter: RebenchLog
273         command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/
Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/
BenchmarkHarness.som %(benchmark)s "
274         benchmarks:
275             - Fannkuch:
276                 extra_args: "14 0 6"
277                 codespeed_name: "Fannkuch >]"
278                 warmup: 4
279             - Fibonacci:
280                 extra_args: "15 0 3"
281                 codespeed_name: "Fibonacci >]"
282                 warmup: 5
283             - Dispatch:
284                 extra_args: "12 0 20"
285                 codespeed_name: "Dispatch >]"
286                 warmup: 2
287             - Bounce:
288                 extra_args: "22 0 2"
289                 codespeed_name: "Bounce >]"

```



```

290         warmup: 12
291     - Loop:
292         extra_args: "14 0 10"
293         codespeed_name: "Loop >]"
294         warmup: 4
295     - Permute:
296         extra_args: "16 0 3"
297         codespeed_name: "Permute >]"
298         warmup: 6
299     - Queens:
300         extra_args: "13 0 2"
301         codespeed_name: "Queens >]"
302         warmup: 3
303     - List:
304         extra_args: "16 0 2"
305         codespeed_name: "List >]"
306         warmup: 6
307     - Recurse:
308         extra_args: "14 0 3"
309         codespeed_name: "Recurse >]"
310         warmup: 4
311     - Storage:
312         extra_args: "17 0 2"
313         codespeed_name: "Storage >]"
314         warmup: 7
315     - Sieve:
316         extra_args: "18 0 5"
317         codespeed_name: "Sieve >]"
318         warmup: 8
319     - BubbleSort:
320         extra_args: "16 0 3"
321         codespeed_name: "BubbleSort >]"
322         warmup: 6
323     - QuickSort:
324         extra_args: "15 0 3"
325         codespeed_name: "QuickSort >]"
326         warmup: 5
327     - Sum:
328         extra_args: "20 0 10"
329         codespeed_name: "Sum >]"
330         warmup: 10
331     - Towers:
332         extra_args: "20 0 2"
333         codespeed_name: "Towers >]"
334         warmup: 10
335     - TreeSort:
336         extra_args: "15 0 1"
337         codespeed_name: "TreeSort >]"
338         warmup: 5
339     - IntegerLoop:
340         extra_args: "14 0 8"
341         codespeed_name: "IntegerLoop >]"
342         warmup: 4
343     - FieldLoop:
344         extra_args: "12 0 3"
345         codespeed_name: "FieldLoop >]"
346         warmup: 2
347     - WhileLoop:
348         extra_args: "13 0 30"
349         codespeed_name: "WhileLoop >]"
350         warmup: 3

```

```

351
352 executors:
353     SOM:
354         path: .
355         executable: som.sh
356
357     TruffleSOM-interpreter:
358         path: .
359         executable: som.sh
360     TruffleSOM-graal:
361         path: .
362         executable: som
363         args: "-E"
364
365     TruffleSOM-interpreter-exp:
366         path: .
367         executable: som.sh
368     TruffleSOM-graal-exp:
369         path: .
370         executable: som
371         args: "-E"
372
373     CSOM:
374         path: .
375         executable: CSOM
376     SOMpp:
377         path: .
378         executable: som.sh
379     PySOM:
380         path: .
381         executable: som.sh
382     RPySOM-interpreter:
383         path: .
384         executable: RPySOM-no-jit
385     RPySOM-jit:
386         path: .
387         executable: RPySOM-jit
388     RTruffleSOM-interpreter:
389         path: .
390         executable: RTruffleSOM-no-jit
391     RTruffleSOM-jit:
392         path: .
393         executable: RTruffleSOM-jit
394
395 # define the benchmarks to be executed for a re-executable benchmark run
396 experiments:
397     SOM:
398         description: All benchmarks on SOM (Java, bytecode-based)
399         suites:
400             - micro-startup-100
401             - micro-steady-100
402             - micro-startup
403             - micro-steady
404             - macro-startup
405             - macro-steady
406         executions:
407             - SOM
408     TruffleSOM:
409         description: All benchmarks on TruffleSOM (Java, AST Interpreter)
410         suites:
411             - micro-startup-100

```

```

412         - micro-steady-100
413         - macro-startup
414         - macro-steady
415     executions:
416         #- TruffleSOM-interpreter
417         - TruffleSOM-graal
418 TruffleSOM-exp:
419     description: All benchmarks on TruffleSOM (Java, AST Interpreter)
420     suites:
421         - micro-startup-100
422         - micro-steady-100
423         - macro-startup
424         - macro-steady
425     executions:
426         #- TruffleSOM-interpreter
427         - TruffleSOM-graal-exp
428
429 CSOM:
430     description: All benchmarks on CSOM
431     suites:
432         - micro-startup
433         - macro-startup
434     executions:
435         - CSOM
436 SOMpp:
437     description: All benchmarks on SOM++
438     suites:
439         - micro-startup
440         - micro-startup-100
441         - macro-startup
442     executions:
443         - SOMpp
444 PySOM:
445     description: All benchmarks on PySOM
446     suites:
447         - micro-startup
448         - micro-steady
449         - macro-startup
450     executions:
451         - PySOM
452 RPySOM:
453     description: All benchmarks on RPySOM
454     suites:
455         #- micro-startup
456         #- micro-steady
457         - micro-startup-100
458         - micro-steady-100
459         - macro-startup
460         - macro-steady
461     executions:
462         #- RPySOM-interpreter
463         - RPySOM-jit
464 RTruffleSOM:
465     description: All benchmarks on RTruffleSOM
466     suites:
467         #- micro-startup
468         #- micro-steady
469         - micro-startup-100
470         - micro-steady-100
471         - macro-startup
472         - macro-steady

```

```

473         executions:
474             #- RTruffleSOM-interpreter
475             - RTruffleSOM-jit
476 RTruffleSOM-OMOP:
477     description: All benchmarks on RTruffleSOM
478     suites:
479         #- micro-startup
480         #- micro-steady
481         - micro-startup-100
482         - micro-steady-100
483         - macro-startup
484         - macro-steady
485     executions:
486         #- RTruffleSOM-interpreter
487         - RTruffleSOM-jit
488 TruffleSOM-OMOP:
489     suites:
490         - micro-startup-100
491         - micro-steady-100
492         - macro-startup
493         - macro-steady
494     executions:
495         - TruffleSOM-graal
496

```

```
1 # SMOG (2)
2
3 SOM and _A Little Smalltalk_ (which has always fascinated me, since about
1983) are small, OOP languages, in a pretty _pure_ sense.
4
5 **Smog** is small Go implementation of SOM.
6
7 I'm going to try to advance thru the creation using a each step as a
"build it from scratch" project. (we will see if I can actually do that).
8
9 This _smog_ is a 1.5v attempt. 1.0 was to tranliterate Java/C to goLang.
That was the first _throw-away_. This might be the first prototype or maybe
just the second throw-away.
10
11
12 ### Step 0
13
14 _What's the go version of hierachical classes?_ Well, because of the
_composition_ way of doing things, one can nest _structs_. (but not
interfaces.)
15 This is the first goLang porject I'm thinking thru that is being done in
the <smirk> _generics era_ (like baseball's _deadball/liveball_ era divide).
16
17
18 ### LSP support for SOM
19
20 There is an LSP support for _vscode_ [effortless-language-servers](https://
marketplace.visualstudio.com/items?itemName=MetaConcProject.effortless-
language-servers)
21
22 ### Why did you organize the SOM code differently?
23
24 See #4 in[Structuring Applications in Go](https://www.gobeyond.dev/
structuring-applications/).
25 We all need to be goLang coders like Ben Johnson.
26
```

smog/cmd/main.go

```
1 package main
2
3 import (
4     '&÷2
5
6     '&v-F†V"æ6öÖ÷†C fW"÷6öÖ÷6öÖr
7 )
8
9 func main() {
10 -R fÖ g6öÖräVæ-fW'6W•Ð
11 - &w3" fÖ ÷2ä &w5³ ¥Ð
12 -Rä-çFW' &WB† &w3"•
13 -RäW†-Bf •
14 }
15
16 // Main = (
17 // -'Väç &w2 Ö €
18 // ' Â R &w3" Ä
19 // ' R fÖ Væ-fW'6R æWrà
20 // ' &w3" fÖ &w2 6÷ "g&öÖç "à
21 // ' R -çFW' &WCç &w3"à
22 // ' R W†-Cç à
23 // '•
24 // )
25
```

smog/compiler.go

```
1 package smog
```

smog/go.mod

```
1 module github.com/xt0fer/som/smog
2
3 go 1.20
4
```


smog/interpreter.go

```
1 package smog
2
3 type Interpreter struct {
4     -Væ-fW'6R ¥Væ-fW'6P
5     -g& ÖR      æg& ÖP
6 }
7
8 func NewInterpreter(u *Universe) *Interpreter {
9     --' fð d-çFW' &WFW'•Ð
10    --'çVæ-fW'6R ò P
11    -&WGW&â -•
12 }
13
14 // "
15 // Frame layout:
16 // +-----+
17 // | Arguments          | 1
18 // +-----+
19 // | Local Variables   | <-- localOffset
20 // +-----+
21 // | Stack              | <-- stackPointer
22 // | ...                |
23 // +-----+
24 // "
25 // |
26 //   "Points at the top element"
27 //   stackPointer
28 //   bytecodeIndex
29
30 //   "the offset at which local variables start"
31 //   localOffset
32
33 //   method
34 //   context
35 //   previousFrame
36 //   stack
37 // |
38
39 type Frame struct {
40     -7F 6µ ö-çFW" æððö&|V7@
41     -'FV6öFT-æFW, -çB
42     -Æö6 Äöfg6WB -çB
43     -ÖWF†öB æððö&|V7@
44     -6öçFW‡B æððö&|V7@
45     - &Wf-÷W4g& ÖR æððö&|V7@
46     -7F 6² æððö&|V7@
47 }
48
49 func NewFrame() *Frame {
50     -b fð dg& ÖW•Ð
51
52     -&WGW&â `
53 }
```

smog/primitives.go

```
1 package smog
2
```

smog/vm.go

```
1 package smog
2
3 import "os"
4
5 // | symbolTable globals classPath dumpBytecodes interpreter
6
7 // avoidExit
8 // lastExitCode
9 // exitBlock
10
11 // nilObject
12 // trueObject
13 // falseObject
14
15 // objectClass
16 // classClass
17 // metaclassClass
18
19 // nilClass
20 // integerClass
21 // arrayClass
22 // methodClass
23 // symbolClass
24 // primClass
25 // stringClass
26 // systemClass
27 // blockClass
28 // doubleClass
29
30 // trueClass
31 // falseClass
32 // |
33
34 // initialize = (
35 // symbolTable := Dictionary new.
36 // globals := Dictionary new.
37 // interpreter := Interpreter new: self.
38 // dumpBytecodes := false.
39 // avoidExit := false
40 // )
41
42 type ObjToObjMap map[*Object]*Object
43
44 type Universe struct {
45     -7-Ö&öÄF &ÆR      Ö&¥FôÖ&¤Ö
46     -vÆÖ& Ç2          Ö&¥FôÖ&¤Ö
47     --ÇFW' &WFW"      ¤-ÇFW' &WFW
48     -GV× '-FV6öFW2 &ööÄ
49     - fö-DW†-B        &ööÄ
50 }
51
52 // UNIVERSE
53
54 func (u *Universe) Exit(code int) {
55     -÷2äW†-B†6öFR•
56 }
57 func (u *Universe) Interpret(args []string) {
58
```

59 }
60

smog/vmobjects.go

```
1 package smog
2
3 // OObject - an attempt to make All objects in this project this struct
4 tied to its interface
5 type OObject struct {
6 }
7 type Object interface {
8     -6öÖ6Æ 72,' ¥46Æ 70
9     -6WE6öÖ6Æ 72† 46Æ 72 ¥46Æ 72•
10 }
11
12 type Sender interface {
13     -6VæB†6VÆV7F÷%7G&-ær 7G&-ærÂ &wVÖVçG2 µÒðôö&|V7BÂ Væ-fW'6R ¥Væ-fW'6RÂ
14     interpreter *Interpreter)
15     -6VæDFÖW4æ÷EVæFW'7F æB†6VÆV7F÷" 7G&-ærÂ Væ-fW'6R ¥Væ-fW'6RÂ -çFW' &WFW"
16     *Interpreter)
17     -6VæEVæ¶æ÷vävÆö& Â†vÆö& Äæ ÖR òôö&|V7BÂ Væ-fW'6R ¥Væ-fW'6RÂ -çFW' &WFW"
18     *Interpreter)
19     -6VæDW66 VD&Æö6²†&Æö6² òôö&|V7BÂ Væ-fW'6R ¥Væ-fW'6RÂ -çFW' &WFW"
20     *Interpreter)
21 }
22
23 // SSymbol = SString (
24 // -Â çVÖ6-væ GW&T &wVÖVçG2 Â
25
26 // SObject = SAbstractObject (
27 // -Â f-VÆG2 6Æ §ç Â
28
29 type SObject struct {
30     "ôö&|V7@
31     "f-VÆG2 µÖö&|V7@
32     "6Æ §ç ¥46Æ 70
33 }
34
35 func NewSObject(n int32, with *OObject) *SObject {
36     -6ð fÖ e4ö&|V7G•Ð
37     -6ðäf-VÆG2 Ö Ö ¶R...µÖö&|V7BÂ â•
38     •
39     -&WGW&â 6ö
40 }
41
42 // initialize: numberOfFields with: nilObject = (
43 //     fields := Array new: numberOfFields withAll: nilObject
44 // )
45
46 // somClass = (
47 //     •â 6Æ §
48 // )
49
50 func (so *SObject) somClass() *SClass {
51     -&WGW&â 6ðä6Æ §
52 }
53
54 // somClass: aSClass = (
55 //
56 // -6Æ §ç fÖ 46Æ 70
```

```

54 //
55 // )
56 func (so *SObject) setSomClass(aSClass *SClass) {
57 -6ðä6Æ §ç Ò 46Æ 70
58 }
59
60 // somClassIn: universe = (
61 //
62 //•â 6Æ §
63 //
64 // )
65 func (so *SObject) somClassIn(u *Universe) *SClass {
66 -&WGW&â 6ðä6Æ §
67 }
68
69 // fieldName: index = (
70 //
71 //'$vWB F†R æ ÖR öb F†R f-VÆB v-F, F†R v-fVâ -æFW,
72 //•â 6Æ §ç -ç7F æ6Tf-VÆDæ ÖSç -æFW€
73 //
74 // )
75 func (so *SObject) fieldName(index int32) string {
76 -&WGW&â 6ðä6Æ §çæf-VÆDæ ÖR†-æFW,•
77 }
78
79 // fieldIndex: name = (
80 //
81 //'$vWB F†R -æFW, f÷" F†R f-VÆB v-F, F†R v-fVâ æ ÖR
82 //•â 6Æ §ç ÆÖÖ•W f-VÆD-æFWfç æ ÖP
83 //
84 // )
85 func (so *SObject) fieldIndex(name string) int32 {
86 -&WGW&â 6ðä6Æ §çæf-VÆD-æFW,†æ ÖR•
87 }
88
89 // numberOfFields = (
90 // "Get the number of fields in this object"
91 // ^ fields length
92 // )
93
94 // field: index = (
95 //
96 //'$vWB F†R f-VÆB v-F, F†R v-fVâ -æFW,
97 //•â f-VÆG2 Cç -æFW€
98 //
99 // )
100 func (so *SObject) field(index int32) Object {
101 -&WGW&â 6ðäf-VÆG5¶-æFW...Ð
102 }
103
104 // field: index put: value = (
105 //
106 //'%6WB F†R f-VÆB v-F, F†R v-fVâ -æFW, Fò F†R v-fVâ f ÇVR
107 //-f-VÆG2 Cç -æFW, WCç f ÇVP
108 //
109 // )
110 func (so *SObject) fieldPut(index int32, value Object) Object {
111 -6ðäf-VÆG5¶-æFW...Ò Ò f ÇVP
112 }
113
114 // "For using in debugging tools such as the Diassembler"

```

```

115 //  debugString = ( ^ 'SObject(' + clazz name string + '))' )
116
117 //  ----
118
119
120 // ??
121 type Invokable *OOObject
122
123 // SClass = SObject (
124 //
125 // -Â Væ-fW'6P
126 // ' 7W W$6Æ 70
127 // ' æ ÖP
128 // ' -ç7F æ6T-çfö¶ &ÆW2 -ç7F æ6Tf-VÆG7À
129 type SClass struct {
130 •4ö&|V7@
131 •Væ-fW'6R          ¥Væ-fW'6P
132 •7W W$6Æ 72       ¥46Æ 70
133 "æ ÖR             ¥57-Ö&öÀ
134 "-ç7F æ6Tf-VÆG2 µÔ-çfö¶ &ÆP
135 }
136
137 // SSymbol = SString (
138 //
139 // -Â çVÖ6-væ GW&T &wVÖVçG2 À
140 type SSymbol struct {
141 •57G&-æp
142 "çVÖ6-væ GW&T &wVÖVçG2 -ç@
143 }
144
145 // SString = SAbstractObject (
146 //
147 // -Â 7G&-ær À
148 type SString struct {
149 •4ö&|V7@
150 •2 7G&-æp
151 }
152
153 func (S *SString) string() string { return S.S }
154
155 // "For using in debugging tools such as the Diassembler"
156 func (S *SString) debugString() string {
157 -B fð %57G&-ær," ² 2â2 ² "'
158 -&WGW&â @
159 }
160
161 // somClassIn: universe = (
162 //
163 // ' â Væ-fW'6R 7G&-æt6Æ 70
164 // '•
165 func (S *SString) somClassIn(u *Universe) *SClass {
166 -&WGW&â 2â4ö&|V7Bâ6Æ §
167 }
168
169 // initializeWith: aString = (
170 func NewString(aString string) *SString {
171 -2 fð e57G&-æw•Ð
172 -2â2 ò 7G&-æp
173 -&WGW&â 0
174 }
175

```

specification/Makefile

```

1 SHELL = '/bin/bash'
2
3 all: SOMParser.class
4
5 antlr.jar:
6 -vvWB Ôð çFÇ"æ| " ‡GG 3çð÷wwræ çFÇ"æ÷&röF÷væÆö Bö çFÇ"ÓBãrã"Ö6ö× ÆWFRæ|
7
8 SOMParser.java: antlr.jar SOM.g4
9 -| f Ö7 çFÇ"æ| " ÷&ræ çFÇ"çcBâFööÂ 4ôðæs@
10
11 SOMParser.class: SOMParser.java
12 -| f 2 Ö7 çFÇ"æ| " çæ| f
13
14 test: are-we-fast-yet SOMParser.class
15 -6WB ÖS² Æ
16 -f-æB ââ Öæ ÖR rçç6öðr × &-çC Â v†-ÆR &V B ÖB BBuÃ r "² Fò Æ
17 ™-V6†ð "BG¶-Ö#² Æ
18 ™"öUCÖ | f Ö7 çFÇ"æ| #çâ ÷&ræ çFÇ"çcBæwV'âFW7E&-r 4ôð 6Æ 76FVb Ð
diagnostics "$${i}" 2>&1`; \
19 ™--b ² ×ç "BDöUB" Ó² F†Vâ Æ
20 ™™echo "$$OUT"; \
21 ™™exit 1; \
22 ™-f"² Æ
23 -FöæP
24
25 are-we-fast-yet:
26 -v-B 6ÆöæR ÖÖFW Ffó ‡GG 3çðöV-F†V"æ6öð÷6ö "'ö &R×vRÖf 7B×-WB &R×vRÖf 7Bð
yet
27
28 clean:
29 -&ð çæ| f çæ6Æ 72 ççFö¶Vç2 çæ-çFW'
30
31 clobber: clean
32 -&ð çFÇ"æ|
33 -&ð Ö&b &R×vRÖf 7B×-W@
34

```



```

1 grammar SOM;
2
3 /* This parser accepts valid programs adhering to the following grammar.
Comments
4 and white space are not dealt with in the grammar. Names of non-terminals
begin
5 with a lower-case letter; terminals, with an upper-case one. */
6
7 classdef:
8     Identifier Equal superclass
9     instanceFields method*
10    ( Separator classFields method* )?
11    EndTerm;
12
13 superclass:
14     Identifier? NewTerm;
15
16 instanceFields:
17     ( Or variable* Or )?;
18
19 classFields:
20     ( Or variable* Or )?;
21
22 method:
23     pattern Equal ( Primitive | methodBlock );
24
25 pattern:
26     unaryPattern | keywordPattern | binaryPattern;
27
28 unaryPattern:
29     unarySelector;
30
31 binaryPattern:
32     binarySelector argument;
33
34 keywordPattern:
35     ( keyword argument )+;
36
37 methodBlock:
38     NewTerm blockContents? EndTerm;
39
40 unarySelector:
41     identifier;
42
43 binarySelector:
44     Or | Comma | Minus | Equal | Not | And | Star | Div | Mod | Plus |
More |
45     Less | At | Per | OperatorSequence;
46
47 identifier:
48     Primitive | Identifier;
49
50 keyword:
51     Keyword;
52
53 argument:
54     variable;
55

```

```

56 blockContents:
57     ( Or localDefs Or )?
58     blockBody;
59
60 localDefs:
61     variable*;
62
63 blockBody:
64     Exit result
65     | expression ( Period blockBody? )?;
66
67 result:
68     expression Period?;
69
70 expression:
71     assignation | evaluation;
72
73 assignation:
74     assignments evaluation;
75
76 assignments:
77     assignment+;
78
79 assignment:
80     variable Assign;
81
82 evaluation:
83     primary messages?;
84
85 primary:
86     variable | nestedTerm | nestedBlock | literal;
87
88 variable:
89     identifier;
90
91 messages:
92     unaryMessage+ binaryMessage* keywordMessage?
93     | binaryMessage+ keywordMessage?
94     | keywordMessage;
95
96 unaryMessage:
97     unarySelector;
98
99 binaryMessage:
100     binarySelector binaryOperand;
101
102 binaryOperand:
103     primary unaryMessage*;
104
105 keywordMessage:
106     ( keyword formula )+;
107
108 formula:
109     binaryOperand binaryMessage*;
110
111 nestedTerm:
112     NewTerm expression EndTerm;
113
114 literal:
115     literalArray | literalSymbol | literalString | literalNumber;
116

```

```

117 literalArray:
118     Pound NewTerm
119     literal*
120     EndTerm;
121
122 literalNumber:
123     negativeDecimal | literalDecimal;
124
125 literalDecimal:
126     literalInteger | literalDouble;
127
128 negativeDecimal:
129     Minus literalDecimal;
130
131 literalInteger:
132     Integer;
133
134 literalDouble:
135     Double;
136
137 literalSymbol:
138     Pound ( string | selector );
139
140 literalString:
141     string;
142
143 selector:
144     binarySelector | keywordSelector | unarySelector;
145
146 keywordSelector:
147     Keyword | KeywordSequence;
148
149 string:
150     STString;
151
152 nestedBlock:
153     NewBlock blockPattern? blockContents? EndBlock;
154
155 blockPattern:
156     blockArguments Or;
157
158 blockArguments:
159     ( Colon argument )+;
160
161 /* Lexer */
162
163 Comment:    '"' ~["]* '"' -> skip;
164 Whitespace : [ \t\r\n]+ -> skip ;
165
166 Primitive: 'primitive';
167 Identifier: [\p{Alpha}] [\p{Alpha}0-9_]*;
168
169 Equal: '=';
170
171 Separator: '----' '-'*;
172
173 NewTerm: '(';
174 EndTerm: ')';
175 Or: '|';
176
177

```

```

178 Comma: ',';
179 Minus: '-';
180 Not: '~';
181 And: '&';
182 Star: '*';
183 Div: '/';
184 Mod: '\\';
185 Plus: '+';
186 More: '>';
187 Less: '<';
188 At: '@';
189 Per: '%';
190
191 OperatorSequence: (
192     Not | And | Or | Star | Div |
193     Mod | Plus | Equal | More | Less |
194     Comma | At | Per | Minus )+;
195
196 Colon: ':';
197
198 NewBlock: '[';
199 EndBlock: ']';
200
201 Pound: '#';
202 Exit: '^';
203 Period: '.';
204 Assign: '=';
205
206 Integer: [0-9]+;
207 Double: [0-9]+ '.' [0-9]+;
208
209 Keyword: Identifier Colon;
210
211 KeywordSequence: Keyword+;
212
213 STString:
214     '\\'
215     (
216         '\\t'
217         | '\\b'
218         | '\\n'
219         | '\\r'
220         | '\\f'
221         | '\\0'
222         | '\\\\'
223         | ~('\\'| '\\\\')
224     )*
225     '\\';
226

```

