AUTHORS

This file lists all people who have contributed to the SOM VM.

SOM was originally implemented at the University of Aarhus (Denmark) in 2001/2002. The implementation of SOM was done by Jakob Roland Andersen, Kasper Verdich Lund, Lars Bak, Mads Torgersen, and Ulrik Pagh Schultz. They also wrote the original versions of the SOM Smalltalk libraries, test suites, and benchmarks, that are (in extended versions) bundled with SOM.

SOM was used by Michael Haupt in courses on virtual machines at Lancaster University and Technische Universitaet Darmstadt (Germany) in VM courses in 2006. During that time, some changes were applied to SOM by Michael Haupt and Sebastian Kanthak.

SOM is currently maintained by Michael Haupt at the Hasso-Plattner-Institut, University of Potsdam, Germany.

2009-08-14, Michael Haupt
michael.haupt@hpi.uni-potsdam.de

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Ball = (

  | x y xVel yVel |

  bounce = (
    | xLimit yLimit bounced |
    xLimit  := yLimit := 500.
    bounced := false.

    x := x + xVel.
    y := y + yVel.
    x > xLimit ifTrue: [
      x := xLimit. xVel := 0 - xVel abs. bounced := true ].
    x < 0 ifTrue: [
      x := 0.      xVel := xVel abs.     bounced := true ].
    y > yLimit ifTrue: [
      y := yLimit. yVel := 0 - yVel abs. bounced := true ].
    y < 0 ifTrue: [
      y := 0.      yVel := yVel abs.     bounced := true ].
    ^ bounced
  )

  initialize: random = (
    x := random next % 500.
    y := random next % 500.
    xVel := (random next % 300) - 150.
    yVel := (random next % 300) - 150.
  )

  ----------------

  new: random = ( ^ self new initialize: random )

```

```
)
```

Examples/AreWeFastYet/Benchmark.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Benchmark = (

  innerBenchmarkLoop: innerIterations = (
      1 to: innerIterations do: [:i |
        (self verifyResult: self benchmark) ifFalse: [ ^ false ].
      ].
      ^ true
  )

  benchmark  = ( self subclassResponsibility )
  verifyResult: result = ( self subclassResponsibility )
)

```

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Bounce = Benchmark (

  benchmark = (
    | ballCount balls bounces random |

    random := SomRandom new.

    ballCount := 100.
    bounces   := 0.
    balls     := Array new: ballCount withAll: [ Ball new: random ].

    1 to: 50 do: [ :i |
      balls do: [ :ball |
        (ball bounce) ifTrue: [ bounces := bounces + 1 ] ] ].

    ^ bounces
  )

  verifyResult: result = (
    ^ 1331 = result
  )
)

```

Examples/AreWeFastYet/CD/Aircraft.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Aircraft = (
  | callsign position |

  init: aCallsign pos: aPosition = (
    callsign := aCallsign.
    position := aPosition.
  )

  callsign = ( ^ callsign )
  position = ( ^ position )

  ----

  new: callsign pos: position = (
    ^ self new init: callsign pos: position
  )
)
```

Examples/AreWeFastYet/CD/CD.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
CD = Benchmark (

  benchmark: numAircrafts = (
    | numFrames simulator detector actualCollisions |
    numFrames := 200.

    simulator := Simulator new: numAircrafts.
    detector  := CollisionDetector new.

    actualCollisions := 0.

    0 to: numFrames - 1 do: [:i |
      | time collisions |
      time := i // 10.0.
      collisions := detector handleNewFrame: (simulator simulate: time).
      actualCollisions := actualCollisions + collisions size ].

    ^ actualCollisions
  )

  innerBenchmarkLoop: innerIterations = (
    ^ self verify: (self benchmark: innerIterations) resultFor: innerIterations
  )

  verify: actualCollisions resultFor: numAircrafts = (
```

```
    numAircrafts = 1000 ifTrue: [ ^ actualCollisions = 14484 ].
    numAircrafts =  500 ifTrue: [ ^ actualCollisions = 14484 ].
    numAircrafts =  250 ifTrue: [ ^ actualCollisions = 10830 ].
    numAircrafts =  200 ifTrue: [ ^ actualCollisions =  8655 ].
    numAircrafts =  100 ifTrue: [ ^ actualCollisions =  4305 ].
    numAircrafts =   10 ifTrue: [ ^ actualCollisions =   390 ].
    numAircrafts =    2 ifTrue: [ ^ actualCollisions =    42 ].

    ('No verification result for ' + numAircrafts + ' found.') println.
    ('Result is: ' + actualCollisions) println.
    ^ false
  )

  ----

  new = (
    Constants initialize.
    ^ super new
  )
)
```

Examples/AreWeFastYet/CD/CallSign.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
  * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.
  * Neither the name of the Purdue University nor the
    names of its contributors may be used to endorse or promote products
    derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
CallSign = (
  | value |

  init: val = ( value := val )

  value = ( ^ value )

  compareTo: other = (
    ^ value = other value
        ifTrue:  [ 0 ]
        ifFalse: [
          value < other value ifTrue: [ -1 ] ifFalse: [ 1 ]]
  )

  ----

  new: val = (
    ^ self new init: val
  )
)

```

Examples/AreWeFastYet/CD/Collision.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Collision = (
  | aircraftA aircraftB position |

  init: anA b: aB pos: aPos = (
    aircraftA := anA.
    aircraftB := aB.
    position  := aPos
  )

  aircraftA = ( ^ aircraftA )
  aircraftB = ( ^ aircraftB )
  position  = ( ^ position  )

  ----

  a: aircraftA b: aircraftB pos: position = (
    ^ self new init: aircraftA b: aircraftB pos: position
  )
)
```

Examples/AreWeFastYet/CD/CollisionDetector.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
CollisionDetector = (
  | state |

  initialize = (
    state := RedBlackTree new
  )

  handleNewFrame: frame = (
    | motions seen toRemove allReduced collisions |
    motions := Vector new.
    seen := RedBlackTree new.

    frame forEach: [:aircraft |
      | oldPosition newPosition |
      oldPosition := state at: aircraft callsign put: aircraft position.
      newPosition := aircraft position.
      seen at: aircraft callsign put: true.

      oldPosition isNil ifTrue: [
        "Treat newly introduced aircraft as if they were stationary"
        oldPosition := newPosition ].

      motions append: (Motion new: aircraft callsign old: oldPosition new: newPosition) ].
```

```

    " Remove aircraft that are no longer present "
    toRemove := Vector new.
    state forEach: [:e |
      (seen at: e key) ifFalse: [ toRemove append: e key ] ].

    toRemove forEach: [:e | state remove: e ].

    allReduced := self reduceCollisionSet: motions.
    collisions := Vector new.
    allReduced forEach: [:reduced |
      1 to: reduced size do: [:i |
        | motion1 |
        motion1 := reduced at: i.
        i + 1 to: reduced size do: [:j |
          | motion2 collision |
          motion2 := reduced at: j.
          collision := motion1 findIntersection: motion2.
          collision notNil ifTrue: [
            collisions append: (Collision a: motion1 callsign b: motion2 callsign pos: collision) ] ] ] ].

    ^ collisions
  )

  isInVoxel: voxel motion: motion = (
    | init fin v_s r v_x x0 xv v_y y0 yv low_x high_x low_y high_y |
    (voxel x > Constants MaxX or: [
     voxel x < Constants MinX or: [
     voxel y > Constants MaxY or: [
     voxel y < Constants MinY ]]]) ifTrue: [ ^ false ].

    init := motion posOne.
    fin  := motion posTwo.

    v_s := Constants GoodVoxelSize.
    r   := Constants ProximityRadius // 2.0.

    v_x := voxel x.
    x0  := init x.
    xv  := fin x - init x.

    v_y := voxel y.
    y0  := init y.
    yv  := fin y - init y.

    low_x  := (v_x - r - x0) // xv.
    high_x := (v_x + v_s + r - x0) // xv.

    xv < 0.0 ifTrue: [
      | tmp |
      tmp    := low_x.
      low_x  := high_x.
      high_x := tmp ].

    low_y  := (v_y - r - y0) // yv.
    high_y := (v_y + v_s + r - y0) // yv.

    yv < 0.0 ifTrue: [
      | tmp |
      tmp    := low_y.
```

```
      low_y  := high_y.
      high_y := tmp ].

    ^ (((xv = 0.0 and: [v_x <= (x0 + r) and: [(x0 - r) <= (v_x + v_s)]]) or: [ "no motion in x"
        (low_x <= 1.0 and: [1.0 <= high_x]) or: [
        (low_x <= 0.0 and: [0.0 <= high_x]) or: [
        (0.0 <= low_x and: [high_x <= 1.0])]]]) and: [

        (yv = 0.0 and: [v_y <= (y0 + r) and: [(y0 - r) <= (v_y + v_s)]]) or: [ "no motion in y"
          (low_y <= 1.0 and: [1.0 <= high_y]) or: [
          (low_y <= 0.0 and: [0.0 <= high_y]) or: [
          (0.0   <= low_y and: [high_y <= 1.0])]]]]) and: [

         xv = 0.0 or: [
         yv = 0.0 or: [ "no motion in x or y or both"
         (low_y <= high_x and: [high_x <= high_y]) or: [
         (low_y <= low_x  and: [low_x <= high_y]) or: [
         (low_x <= low_y  and: [high_y <= high_x]) ]]]]]
  )

  put: motion and: voxel into: voxelMap = (
    | array |
    array := voxelMap at: voxel.
    array isNil ifTrue: [
      array := Vector new.
      voxelMap at: voxel put: array ].
    array append: motion
  )

  recurse: voxelMap seen: seen voxel: nextVoxel motion: motion = (
    (self isInVoxel: nextVoxel motion: motion) ifFalse: [ ^ self ].
    (seen at: nextVoxel put: true) = true ifTrue: [ ^ self ].

    self put: motion and: nextVoxel into: voxelMap.

    self recurse: voxelMap seen: seen voxel: (nextVoxel minus: Constants horizontal) motion: motion.
    self recurse: voxelMap seen: seen voxel: (nextVoxel plus:  Constants horizontal) motion: motion.
    self recurse: voxelMap seen: seen voxel: (nextVoxel minus: Constants vertical)   motion: motion.
    self recurse: voxelMap seen: seen voxel: (nextVoxel plus:  Constants vertical)   motion: motion.
    self recurse: voxelMap seen: seen voxel: ((nextVoxel minus: Constants horizontal) minus: Constants vertical) motion: motion.
    self recurse: voxelMap seen: seen voxel: ((nextVoxel minus: Constants horizontal) plus:  Constants vertical) motion: motion.
    self recurse: voxelMap seen: seen voxel: ((nextVoxel plus:  Constants horizontal) minus: Constants vertical) motion: motion.
    self recurse: voxelMap seen: seen voxel: ((nextVoxel plus:  Constants horizontal) plus:  Constants vertical) motion: motion.
  )

  reduceCollisionSet: motions = (
    | voxelMap result |
    voxelMap := RedBlackTree new.
    motions forEach: [:motion | self draw: motion on: voxelMap ].

    result := Vector new.
```

```
    voxelMap forEach: [:e |
      e value size > 1 ifTrue: [ result append: e value ] ].
    ^ result
  )

  voxelHash: position = (
    | xDiv yDiv x y |
    xDiv := (position x // Constants GoodVoxelSize) asInteger.
    yDiv := (position y // Constants GoodVoxelSize) asInteger.

    x := Constants GoodVoxelSize * xDiv.
    y := Constants GoodVoxelSize * yDiv.

    position x < 0 ifTrue: [ x := x - Constants GoodVoxelSize ].
    position y < 0 ifTrue: [ y := y - Constants GoodVoxelSize ].

    ^ Vector2D x: x y: y
  )

  draw: motion on: voxelMap = (
    | seen |
    seen := RedBlackTree new.
    self recurse: voxelMap seen: seen voxel: (self voxelHash: motion posOne) motion: motion
  )

  ----

  new = ( ^ super new initialize )

)
```

Examples/AreWeFastYet/CD/Constants.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Constants = (
  ----
  | horizontal vertical |

  initialize = (
    horizontal := Vector2D x: self GoodVoxelSize y: 0.0.
    vertical   := Vector2D x: 0.0 y: self GoodVoxelSize.
  )

  MinX = ( ^ 0.0    )
  MinY = ( ^ 0.0    )
  MaxX = ( ^ 1000.0 )
  MaxY = ( ^ 1000.0 )
  MinZ = ( ^ 0.0    )
  MaxZ = ( ^ 10.0   )

  ProximityRadius = ( ^ 1.0 )
  GoodVoxelSize   = ( "ProximityRadius *" ^ 2.0 )

  horizontal = ( ^ horizontal )
  vertical   = ( ^ vertical   )
)
```

Examples/AreWeFastYet/CD/InsertResult.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
  * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.
  * Neither the name of the Purdue University nor the
    names of its contributors may be used to endorse or promote products
    derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
InsertResult  = (
  | isNewEntry newNode oldValue |

  init: aBool node: aNode value: val = (
    isNewEntry := aBool.
    newNode    := aNode.
    oldValue   := val.
  )

  isNewEntry = ( ^ isNewEntry )
  newNode    = ( ^ newNode    )
  oldValue   = ( ^ oldValue   )

  ----

  new: isNewEntry node: newNode value: oldValue = (
    ^ self new init: isNewEntry node: newNode value: oldValue
  )
)
```

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Motion = (
  | callsign posOne posTwo |

  init: aCallsign old: aPosOne new: aPosTwo = (
    callsign := aCallsign.
    posOne   := aPosOne.
    posTwo   := aPosTwo.
  )

  callsign = ( ^ callsign )
  posOne   = ( ^ posOne )
  posTwo   = ( ^ posTwo )

  delta = (
    ^ posTwo minus: posOne
  )

  findIntersection: other = (
    | init1 init2 vec1 vec2 radius a dist |
    init1 := posOne.
    init2 := other posOne.
    vec1  := self delta.
    vec2  := other delta.
    radius := Constants ProximityRadius.
```

```
    " this test is not geometrical 3-d intersection test, it takes the fact that the aircraft move
       into account ; so it is more like a 4d test
       (it assumes that both of the aircraft have a constant speed over the tested interval)

       we thus have two points, each of them moving on its line segment at constant speed ; we are looking
       for times when the distance between these two points is smaller than r

       vec1 is vector of aircraft 1
       vec2 is vector of aircraft 2

       a = (V2 - V1)^T * (V2 - V1)"
    a := (vec2 minus: vec1) squaredMagnitude.

    a <> 0.0 ifTrue: [
      | b c discr v1 v2 |
      "  we are first looking for instances of time when the planes are exactly r from each other
         at least one plane is moving ; if the planes are moving in parallel, they do not have constant speed

         if the planes are moving in parallel, then
           if the faster starts behind the slower, we can have 2, 1, or 0 solutions
           if the faster plane starts in front of the slower, we can have 0 or 1 solutions

         if the planes are not moving in parallel, then

         point P1 = I1 + vV1
         point P2 = I2 + vV2
           - looking for v, such that dist(P1,P2) = || P1 - P2 || = r

         it follows that || P1 - P2 || = sqrt( < P1-P2, P1-P2 > )
           0 = -r^2 + < P1 - P2, P1 - P2 >
          from properties of dot product
           0 = -r^2 + <I1-I2,I1-I2> + v * 2<I1-I2, V1-V2> + v^2 *<V1-V2,V1-V2>
           so we calculate a, b, c - and solve the quadratic equation
           0 = c + bv + av^2

       b = 2 * <I1-I2, V1-V2>"
      b := 2.0 * ((init1 minus: init2) dot: (vec1 minus: vec2)).

      "c = -r^2 + (I2 - I1)^T * (I2 - I1)"
      c := ((0.0 - radius) * radius) + ((init2 minus: init1) squaredMagnitude).

      discr := (b * b) - (4.0 * a * c).
      discr < 0.0 ifTrue: [ ^ nil ].

      v1 := ((0.0 - b) - discr sqrt) // (2.0 * a).
      v2 := ((0.0 - b) + discr sqrt) // (2.0 * a).

      (v1 <= v2 and: [((v1  <= 1.0 and: [1.0 <= v2]) or: [
                       (v1  <= 0.0 and: [0.0 <= v2]) or: [
                       (0.0 <= v1  and: [v2  <= 1.0])]])]) ifTrue: [
```

```
        "Pick a good 'time' at which to report the collision"
        | v result1 result2 result |
        v1 <= 0.0
          ifTrue: [
            "The collision started before this frame. Report it at the start of the frame"
            v := 0.0 ]
          ifFalse: [
            "The collision started during this frame. Report it at that moment"
            v := v1 ].

        result1 := init1 plus: (vec1 times: v).
        result2 := init2 plus: (vec2 times: v).

        result := (result1 plus: result2) times: 0.5.

        (result x >= Constants MinX and: [
         result x <= Constants MaxX and: [
         result y >= Constants MinY and: [
         result y <= Constants MaxY and: [
         result z >= Constants MinZ and: [
         result z <= Constants MaxZ ]]]]]) ifTrue: [ ^ result ] ].

      ^ nil ].

    "  the planes have the same speeds and are moving in parallel (or they are not moving at all)
       they  thus have the same distance all the time ; we calculate it from the initial point

       dist = || i2 - i1 || = sqrt(  ( i2 - i1 )^T * ( i2 - i1 ) )"
    dist := (init2 minus: init1) magnitude.
    dist <= radius ifTrue: [
      ^ (init1 plus: init2) times: 0.5 ].

    ^ nil
  )

  ----

  new: callsign old: posOne new: posTwo = (
    ^ self new init: callsign old: posOne new: posTwo
  )
)
```

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Node = (
  | key value left right parent color |

  init: aKey value: aValue = (
    key   := aKey.
    value := aValue.
    color := #red.
  )

  key   = ( ^ key )

  value = ( ^ value )
  value: val = ( value := val )

  left       = ( ^ left )
  left: n    = ( left := n )
  right      = ( ^ right )
  right: n   = ( right := n )
  parent     = ( ^ parent )
  parent: n  = ( parent := n )
  color      = ( ^ color )
  color: sym = ( color := sym )

  successor = (
    | x y |
    x := self.
```

```
    x right notNil ifTrue: [
      ^ RedBlackTree treeMinimum: x right ].

    y := x parent.
    [ y notNil and: [ x == y right ]] whileTrue: [
      x := y.
      y := y parent ].
    ^ y
  )

  ----

  key: key value: value = (
    ^ self new init: key value: value
  )
)
```

Examples/AreWeFastYet/CD/RbtEntry.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
RbtEntry = (
  | key value |

  init: aKey value: val = (
    key   := aKey.
    value := val.
  )

  key   = ( ^ key )
  value = ( ^ value )

  ----

  key: key value: value = (
    ^ self new init: key value: value
  )
)

```

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
RedBlackTree = (
  | root |

  at: key put: value = (
    | insertionResult x |
    insertionResult := self treeAt: key insert: value.
    insertionResult isNewEntry ifFalse: [
      ^ insertionResult oldValue ].

    x := insertionResult newNode.

    [ x ~= root and: [ x parent color = #red ]] whileTrue: [
      x parent == x parent parent left
        ifTrue: [
          | y |
          y := x parent parent right.
          (y notNil and: [y color = #red])
            ifTrue: [
              "Case 1"
              x parent color: #black.
              y color: #black.
              x parent parent color: #red.
              x := x parent parent ]
            ifFalse: [
```

```
            x == x parent right ifTrue: [
              "Case 2"
              x := x parent.
              self leftRotate: x ].

            "Case 3"
            x parent color: #black.
            x parent parent color: #red.
            self rightRotate: x parent parent ] ]
      ifFalse: [
        "Same as 'then' clause with 'right' and 'left' exchanged"
        | y |
        y := x parent parent left.
        (y notNil and: [ y color = #red ])
          ifTrue: [
            "Case 1"
            x parent color: #black.
            y color: #black.
            x parent parent color: #red.
            x := x parent parent ]
          ifFalse: [
            x == x parent left ifTrue: [
              "Case 2"
              x := x parent.
              self rightRotate: x ].

            "Case 3"
            x parent color: #black.
            x parent parent color: #red.
            self leftRotate: x parent parent ] ] ].

  root color: #black.
  ^ nil
)

remove: key = (
  | x y z xParent |
  z := self findNode: key.
  z isNil ifTrue: [ ^ nil ].

  "Y is the node to be unlinked from the tree."
  (z left isNil or: [ z right isNil ])
    ifTrue:  [ y := z ]
    ifFalse: [ y := z successor ].

  "Y is guaranteed to be non-null at this point."
  y left notNil
    ifTrue:  [ x := y left ]
    ifFalse: [ x := y right ].

  "X is the child of y which might potentially replace y in the tree.
   X might be null at this point."
  x notNil
    ifTrue: [
      x parent: y parent.
      xParent := x parent ]
    ifFalse: [
      xParent := y parent ].

  y parent isNil
    ifTrue:  [ root := x ]
```

```
      ifFalse: [
        y == y parent left
          ifTrue:  [ y parent left: x ]
          ifFalse: [ y parent right: x ] ].

    y ~= z ifTrue: [
      y color = #black ifTrue: [
        self remove: x andFixup: xParent ].

      y parent: z parent.
      y color:  z color.
      y left:   z left.
      y right:  z right.

      z left notNil ifTrue: [
        z left parent: y ].
      x right notNil ifTrue: [
        z right parent: y ].

      z parent notNil
        ifTrue: [
          z parent left == z
            ifTrue:  [ z parent left: y ]
            ifFalse: [ z parent right: y ] ]
        ifFalse: [ root := y ] ]
      ifFalse: [
        y color = #black ifTrue: [
          self remove: x andFixup: xParent ] ].

    ^ z value
  )

  at: key = (
    | node |
    node := self findNode: key.
    node isNil ifTrue: [ ^ nil ].
    ^ node value
  )

  forEach: block = (
    | current |
    root isNil ifTrue: [ ^ self ].
    current := RedBlackTree treeMinimum: root.
    [ current notNil ] whileTrue: [
      block value: (RbtEntry key: current key value: current value).
      current := current successor ].
  )

  findNode: key = (
    | current |
    current := root.
    [ current notNil ] whileTrue: [
      | comparisonResult |
      comparisonResult := key compareTo: current key.
      comparisonResult = 0 ifTrue: [ ^ current ].
      comparisonResult < 0
        ifTrue:  [ current := current left ]
        ifFalse: [ current := current right ] ].
    ^ nil
  )
```

```
treeAt: key insert: value = (
  | x y z |
  y := nil.
  x := root.

  [ x notNil ] whileTrue: [
    | comparisonResult |
    y := x.
    comparisonResult := key compareTo: x key.
    comparisonResult < 0
      ifTrue:  [ x := x left ]
      ifFalse: [
        comparisonResult > 0
          ifTrue: [ x := x right ]
          ifFalse: [
            | oldValue |
            oldValue := x value.
            x value: value.
            ^ InsertResult new: false node: nil value: oldValue ] ] ].

  z := Node key: key value: value.
  z parent: y.
  y isNil
    ifTrue: [ root := z ]
    ifFalse: [
      (key compareTo: y key) < 0
        ifTrue:  [ y left: z ]
        ifFalse: [ y right: z ] ].

  ^ InsertResult new: true node: z value: nil
)

leftRotate: x = (
  | y |
  y := x right.

  "Turn y's left subtree into x's right subtree"
  x right: y left.
  y left notNil ifTrue: [
    y left parent: x ].

  "Link x's parent to y"
  y parent: x parent.
  x parent isNil
    ifTrue: [ root := y ]
    ifFalse: [
      x == x parent left ifTrue:  [ x parent left: y ]
                         ifFalse: [ x parent right: y ] ].

  "Put x on y's left"
  y left: x.
  x parent: y.
  ^ y
)

rightRotate: y = (
  | x |
  x := y left.

  "Turn x's right subtree into y's left subtree"
  y left: x right.
```

```
  x right notNil ifTrue: [ x right parent: y ].

  "Link y's parent to x"
  x parent: y parent.
  y parent isNil
    ifTrue: [ root := x ]
    ifFalse: [
      y == y parent left
        ifTrue: [ y parent left: x ]
        ifFalse: [ y parent right: x ] ].

  x right: y.
  y parent: x.
  ^ x
)

remove: anX andFixup: anXParent = (
  | x xParent |
  x := anX.
  xParent := anXParent.

  x ~= root and: [ x isNil or: [ x color = #black ]] whileTrue: [
    x == xParent left
      ifTrue: [
        | w |
         "Note: the text points out that w cannot be null. The reason is not obvious from
          simply looking at the code; it comes about from the properties of the red-black
          tree."
        w := xParent right.
        w color = #red ifTrue: [
          "Case 1"
          w color: #black.
          xParent color: #red.
          self leftRotate: xParent.
          w := xParent right ].

        ((w left isNil or: [ w left color = #black ]) and: [
          w right isNil or: [ w right color = #black ]])
          ifTrue:  [
            "Case 2"
            w color: #red.
            x := xParent.
            xParent := x parent ]
          ifFalse: [
            (w right isNil or: [ w right color = #black ]) ifTrue: [
              "Case 3"
              w left color: #black.
              w color: #red.
              self rightRotate: w.
              w := xParent right ].

            "Case 4"
            w color: xParent color.
            xParent color: #black.
            w right notNil ifTrue: [
              w right color: #black ].

            self leftRotate: xParent.
            x := root.
```

```
              xParent := x parent ] ]
        ifFalse: [
          | w |
          "Same as 'then' clause with 'right' and 'left' exchanged"
          w := xParent left.
          w color = #red ifTrue: [
            "Case 1"
            w color: #black.
            xParent color: #red.
            self rightRotate: xParent.
            w := xParent left ].

          ((w right isNil or: [ w right color = #black ]) and: [
           (w left isNil or: [ w left color = #black ])])
            ifTrue: [
              "Case 2"
              w color: #red.
              x := xParent.
              xParent := x parent ]
            ifFalse: [
              w left isNil or: [ w left color = #black ] ifTrue: [
                "Case 3"
                w right color: #black.
                w color: #red.
                self leftRotate: w.
                w := xParent left ].

              "Case 4"
              w color: xParent color.
              xParent color: #black.
              w left notNil ifTrue: [
                w left color: #black ].

              self rightRotate: xParent.
              x := root.
              xParent = x parent ] ] ].

    x notNil ifTrue: [ x color: #black ]
  )

  ----

  treeMinimum: x = (
    | current |
    current := x.
    [ current left notNil ] whileTrue: [
      current := current left ].
    ^ current
  )
)
```

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
  * Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.
  * Neither the name of the Purdue University nor the
    names of its contributors may be used to endorse or promote products
    derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Simulator = (
  | aircrafts |

  init: numAircrafts = (
    aircrafts := Vector new.

    0 to: numAircrafts - 1 do: [:i |
      aircrafts append: (CallSign new: i)]
  )

  simulate: time = (
    | frame |
    frame := Vector new.
    0 to: aircrafts size - 2 by: 2 do: [:i |
      frame append: (Aircraft new: (aircrafts at: i + 1)
                              pos: (Vector3D x: time
                                             y: (time cos * 2.0) + (i * 3.0)
                                             z: 10.0)).
      frame append: (Aircraft new: (aircrafts at: i + 2)
                              pos: (Vector3D x: time
                                             y: (time sin * 2.0) + (i * 3.0)
                                             z: 10.0)) ].
    ^ frame
  )
```

```
  ----

  new: numAircrafts = (
  ^ self new init: numAircrafts
  )
)

```

Examples/AreWeFastYet/CD/Vector2D.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Vector2D = (
  | x y |

  x = ( ^ x )
  y = ( ^ y )

  initX: anX y: aY = (
    x := anX.
    y := aY
  )

  plus: other = (
    ^ Vector2D x: x + other x
               y: y + other y
  )

   minus: other = (
    ^ Vector2D x: x - other x
               y: y - other y
  )

  compareTo: other = (
    | result |
    result := self compare: x and: other x.
    result <> 0 ifTrue: [ ^ result ].
    ^ self compare: y and: other y
```

```
  )

  compare: a and: b = (
    a = b ifTrue: [ ^  0 ].
    a < b ifTrue: [ ^ -1 ].
    a > b ifTrue: [ ^  1 ].

    "We say that NaN is smaller than non-NaN."
    a = a ifTrue: [ ^ 1 ].
    ^ -1
  )

  ----

  x: anX y: aY = (
    ^ self new initX: anX y: aY
  )
)
```

Examples/AreWeFastYet/CD/Vector3D.som

```
"
Ported from the adapted JavaScript and Java versions.

Copyright (c) 2001-2010, Purdue University. All rights reserved.
Copyright (C) 2015 Apple Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
 * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
 * Neither the name of the Purdue University nor the
   names of its contributors may be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
"
Vector3D = (
  | x y z |

  x = ( ^ x )
  y = ( ^ y )
  z = ( ^ z )

  initX: anX y: aY z: aZ = (
    x := anX.
    y := aY.
    z := aZ
  )

  plus: other = (
    ^ Vector3D x: x + other x
               y: y + other y
               z: z + other z
  )

  minus: other = (
    ^ Vector3D x: x - other x
               y: y - other y
               z: z - other z
  )

  dot: other = (
```

```
    ^ (x * other x) + (y * other y) + (z * other z)
  )

  squaredMagnitude = (
    ^ self dot: self
  )

  magnitude = (
    ^ self squaredMagnitude sqrt
  )

  times: amount = (
    ^ Vector3D x: x * amount
               y: y * amount
               z: z * amount
  )

  ----

  x: x y: y z: z = (
    ^ self new initX: x y: y z: z
  )
)
```

```
"
Copyright (c) 2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

DictEntry = (
  | hash key value next |

  init: aHash key: aKey value: val next: anEntry = (
    hash  := aHash.
    key   := aKey.
    value := val.
    next  := anEntry.
  )

  hash  = ( ^ hash )
  key   = ( ^ key  )

  value = ( ^ value )
  value: val = ( value := val )

  next  = ( ^ next )
  next: e = ( next := e )

  match: aHash key: aKey = (
    ^ hash = aHash and: [key = aKey]
  )

  ----

  new: hash key: key value: val next: next = (
    ^ self new init: hash key: key value: val next: next
  )
)
```

Examples/AreWeFastYet/Core/DictIdEntry.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
DictIdEntry = DictEntry (

  match: aHash key: aKey = (
    ^ hash = aHash and: [key == aKey]
  )

  ----

  new: hash key: key value: val next: next = (
    ^ self new init: hash key: key value: val next: next
  )
)

```

Examples/AreWeFastYet/Core/Pair.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Pair = (
  | key value |

  initialize: aKey and: aValue = (
    key   := aKey.
    value := aValue.
  )

  key   = ( ^ key )
  value = ( ^ value )

  key:   aKey   = ( key   := aKey   )
  value: aValue = ( value := aValue )

  ----

  withKey: aKey andValue: aValue = (
    ^ self new initialize: aKey and: aValue
  )
)

```

Examples/AreWeFastYet/Core/SomDictionary.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SomDictionary = (
  | buckets size_ |

  initialize: size = (
    buckets := Array new: size.
    size_   := 0
  )

  hash: key = (
    | hash |
    key isNil ifTrue: [ ^ 0 ].
    hash := key customHash.
    ^ hash bitXor: (hash >>> 16)
  )

  bucketIdx: hash = (
    ^ 1 + ((buckets length - 1) & hash)
  )

  bucket: hash = (
    ^ buckets at: (self bucketIdx: hash)
  )

  at: aKey = (
    | hash e |
    hash := self hash: aKey.
    e    := self bucket: hash.

    [ e notNil ] whileTrue: [
      (e match: hash key: aKey)
        ifTrue: [ ^ e value ].
```

```
      e := e next ].
    ^ nil
  )

  containsKey: aKey = (
    | hash e |
    hash := self hash: aKey.
    e    := self bucket: hash.

    [ e notNil ] whileTrue: [
      (e match: hash key: aKey)
        ifTrue: [ ^ true ].
      e := e next ].
    ^ false
  )

  at: aKey put: aVal = (
    | hash i current |
    hash := self hash: aKey.
    i    := self bucketIdx: hash.
    current := buckets at: i.

    current isNil
      ifTrue: [
        buckets at: i put: (self newEntry: aKey value: aVal hash: hash).
        size_ := size_ + 1 ]
      ifFalse: [
        self insertBucketEntry: aKey value: aVal hash: hash head: current ].

    size_ > buckets length ifTrue: [ self resize ]
  )

  newEntry: aKey value: value hash: hash = (
    ^ DictEntry new: hash key: aKey value: value next: nil
  )

  insertBucketEntry: key value: value hash: hash head: head = (
    | current |
    current := head.

    [true] whileTrue: [
      (current match: hash key: key) ifTrue: [
        current value: value.
        ^ self ].
      current next isNil ifTrue: [
        size_ := size_ + 1.
        current next: (self newEntry: key value: value hash: hash).
        ^ self ].
      current := current next ]
  )

  resize = (
    | oldStorage |
    oldStorage := buckets.
    buckets := Array new: oldStorage length * 2.
    self transferEntries: oldStorage
  )

  transferEntries: oldStorage = (
    1 to: oldStorage length do: [:i |
```

```
    | current |
    current := oldStorage at: i.
    current notNil ifTrue: [
      oldStorage at: i put: nil.
      current next isNil
        ifTrue: [
          buckets at: 1 + (current hash & (buckets length - 1)) put: current ]
        ifFalse: [
          self splitBucket: oldStorage bucket: i head: current ] ] ]
)

splitBucket: oldStorage bucket: i head: head = (
  | loHead loTail hiHead hiTail current |
  loHead := nil. loTail := nil.
  hiHead := nil. hiTail := nil.
  current := head.

  [current notNil] whileTrue: [
    (current hash & oldStorage length) = 0
      ifTrue: [
        loTail isNil
          ifTrue:  [ loHead := current ]
          ifFalse: [ loTail next: current ].
        loTail := current ]
      ifFalse: [
        hiTail isNil
          ifTrue:  [ hiHead := current ]
          ifFalse: [ hiTail next: current ].
        hiTail := current ].
    current := current next ].

  loTail notNil ifTrue: [
    loTail next: nil.
    buckets at: i put: loHead ].
  hiTail notNil ifTrue: [
    hiTail next: nil.
    buckets at: i + oldStorage length put: hiHead ]
)

size      = ( ^ size_ )
isEmpty   = ( ^ size_ = 0 )
removeAll = (
  buckets := Array new: buckets length.
  size_ := 0.
)

keys = (
  | keys |
  keys := Vector new: size_.
  buckets do: [:b |
    | current |
    current := b.
    [ current notNil ] whileTrue: [
      keys append: current key.
      current := current next ] ].
  ^ keys
)

values = (
  | values |
```

```
    values := Vector new: size_.
    buckets do: [:b |
      | current |
      current := b.
      [ current notNil ] whileTrue: [
        values append: current value.
        current := current next ] ].
    ^ values
  )

  ----

  new: size = (
    ^ super new initialize: size
  )

  new = (
    ^ self new: 16
  )
)
```

Examples/AreWeFastYet/Core/SomIdentityDictionary.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SomIdentityDictionary = SomDictionary (

  newEntry: aKey value: value hash: hash = (
    ^ DictIdEntry new: hash key: aKey value: value next: nil
  )

  ----

  new: size = (
    ^ self new: size
  )

  new = ( ^ super new: 16 )
)

```

Examples/AreWeFastYet/Core/SomIdentitySet.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
SomIdentitySet = SomSet (
  contains: anObject = (
    ^ self hasSome: [ :it |  it == anObject ]
  )
  ----

  new: size = (
    ^ super new initialize: size.
  )
)

```

Examples/AreWeFastYet/Core/SomSet.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SomSet = (

  | items |

  initialize: size = (
    items := Vector new: size.
  )

  forEach: block = ( items forEach: block )
  hasSome: block = ( ^ items hasSome: block )
  getOne:  block = ( ^ items getOne:  block )

  add: anObject = (
    (self contains: anObject)
        ifFalse: [ items append: anObject ]
  )

  collect: block = ( | coll |
    coll := Vector new.
    self forEach: [ :e | coll append: (block value: e) ].
    ^ coll
  )

  contains: anObject = (
    ^ self hasSome: [ :it | it = anObject ]
  )

  size = ( ^ items size )
  removeAll = ( ^ items removeAll )

  ----

```

```
  new = (
    ^ super new initialize: 10.
  )
)

```

Examples/AreWeFastYet/Core/Vector.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Vector = (

  | first last storage |

  initialize: size = (
    first := 1.
    last  := 1.
    storage := Array new: size.
  )

  at: index = (
    index > storage length ifTrue: [ ^ nil ].
    ^ storage at: index
  )

  at: index put: val = (
    index > storage length ifTrue: [
      | newLength newStorage |
      newLength := storage length.
      [ newLength < index ] whileTrue: [
        newLength := newLength * 2 ].
      newStorage := Array new: newLength.
      storage doIndexes: [:i | newStorage at: i put: (storage at: i) ].
      storage := newStorage ].

    storage at: index put: val.
    last < (index + 1) ifTrue: [
      last := index + 1 ]
  )

  append: element = (
```

```
    (last > storage length) ifTrue: [
      "Need to expand capacity first"
      | newStorage |
      newStorage := Array new: (2 * storage length).
      storage doIndexes: [ :i | newStorage at: i put: (storage at: i) ].
      storage := newStorage. ].

    storage at: last put: element.
    last := last + 1.
    ^ self
  )

  isEmpty  = ( ^ last = first   )

  forEach: block = (
    first to: last - 1 do: [ :i | block value: (storage at: i) ]
  )

  hasSome: block = (
    first to: last - 1 do: [ :i |
      (block value: (storage at: i))
        ifTrue: [ ^ true ] ].
    ^ false
  )

  getOne: block = (
    first to: last - 1 do: [ :i |
      | e |
      e := storage at: i.
      (block value: e)
        ifTrue: [ ^ e ] ].
    ^ nil
  )

  removeFirst = (
    self isEmpty ifTrue: [ ^ nil ].
    first := first + 1.
    ^ storage at: first - 1
  )

  removeAll = (
    first := 1.
    last  := 1.
    storage  := Array new: storage length
  )

  remove: object = (
    | newArray newLast found |
    newArray := Array new: self capacity.
    newLast := 1.
    found := false.

    self forEach: [ :it |
      it == object
          ifTrue:  [ found := true ]
          ifFalse: [
              newArray at: newLast put: it.
              newLast := newLast + 1 ] ].

    storage := newArray.
    last := newLast.
```

```
  first := 1.
  ^ found
)

size     = ( ^ last - first   )
capacity = ( ^ storage length )

sort: aBlock = (
  " Make the argument, aBlock, be the criterion for ordering elements of
     the receiver.
     sortBlocks with side effects may not work right "
  self size > 0 ifTrue: [
    self sort: first
           to: last - 1
         with: aBlock ]
)

sort: i to: j with: sortBlock = (
  " Sort elements i through j of self to be non-descending according to
     sortBlock. "
  | di dij dj tt ij k l n |
  sortBlock isNil ifTrue: [ ^ self defaultSort: i to: j ].

  "The prefix d means the data at that index."
  (n := j + 1  - i) <= 1 ifTrue: [ ^ self ]. "Nothing to sort."
  " Sort di,dj. "
  di := storage at: i.
  dj := storage at: j.

  "i.e., should di precede dj?"
  (sortBlock value: di with: dj) ifFalse: [
    storage swap: i with: j.
    tt := di.
    di := dj.
    dj := tt
  ].

  n > 2 ifTrue: [ " More than two elements. "
    ij := (i + j) / 2.  " ij is the midpoint of i and j. "
    dij := storage at: ij.  " Sort di,dij,dj.  Make dij be their median. "
    (sortBlock value: di with: dij)
      ifTrue: [ " i.e. should di precede dij? "
        (sortBlock value: dij with: dj) ifFalse: [ " i.e., should dij precede dj? "
          storage swap: j with: ij.
          dij := dj]]
      ifFalse: [ " i.e. di should come after dij "
        storage swap: i with: ij.
        dij := di].

    n > 3 ifTrue: [
      " More than three elements.
        Find k>i and l<j such that dk,dij,dl are in reverse order.
        Swap k and l.  Repeat this procedure until k and l pass each other. "
      k := i.
      l := j.
      [ [l := l - 1.  k <= l and: [sortBlock value: dij with: (storage at: l)]]
          whileTrue.  " i.e. while dl succeeds dij "
```

```
          [k := k + 1.  k <= l and: [sortBlock value: (storage at: k) with: dij]]
            whileTrue.  " i.e. while dij succeeds dk "
          k <= l]
            whileTrue:
              [ storage swap: k with: l ].

        " Now l<k (either 1 or 2 less), and di through dl are all less than or equal to dk
          through dj.  Sort those two segments. "
        self sort: i to: l with: sortBlock.
        self sort: k to: j with: sortBlock
      ]
    ]
  )

  ----------------------------

  "Allocation"
  new              = ( ^ self new: 50 )
  new: initialSize = ( ^ super new initialize: initialSize )

  with: elem = (
    | newVector |
    newVector := self new: 1.
    newVector append: elem.
    ^ newVector
  )
)

```

Examples/AreWeFastYet/DeltaBlue/AbstractConstraint.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
AbstractConstraint = (
  "I am an abstract class representing a system-maintainable relationship (or
  'constraint') between a set of variables. I supply a strength instance
  variable; concrete subclasses provide a means of storing the constrained
  variables and other information required to represent a constraint.

  Instance variables:
      strength            the strength of this constraint <Strength>"
  | strength |

  initialize: strengthSymbol = (
    strength := Strength of: strengthSymbol
  )

  "accessing"

  strength = (
    "Answer my strength."
    ^ strength
  )

  "queries"
  isInput = (
    "Normal constraints are not input constraints. An input constraint is
     one that depends on external state, such as the mouse, the keyboard,
     a clock, or some arbitrary piece of imperative code."
    ^ false
  )

  isSatisfied = (
    "Answer true if this constraint is satisfied in the current solution."
    self subclassResponsibility
  )

  "add/remove"
  addConstraint: planner = (
    "Activate this constraint and attempt to satisfy it."

    self addToGraph.
    planner incrementalAdd: self.
  )

  addToGraph = (
    "Add myself to the constraint graph."
    self subclassResponsibility
  )

  destroyConstraint: planner = (
```

```
    "Deactivate this constraint, remove it from the constraint graph,
     possibly causing other constraints to be satisfied, and destroy it."

    self isSatisfied ifTrue: [planner incrementalRemove: self].
    self removeFromGraph.
  )

  removeFromGraph = (
    "Remove myself from the constraint graph."
    self subclassResponsibility
  )

  "planning"
  chooseMethod: mark = (
    "Decide if I can be satisfied and record that decision. The output of
     the chosen method must not have the given mark and must have a
     walkabout strength less than that of this constraint."
    self subclassResponsibility
  )

  execute = (
    "Enforce this constraint. Assume that it is satisfied."
    self subclassResponsibility
  )

  inputsDo: aBlock = (
    "Assume that I am satisfied. Evaluate the given block on all my current
     input variables."
    self subclassResponsibility
  )

  inputsKnown: mark = (
    "Assume that I am satisfied. Answer true if all my current inputs are
     known. A variable is known if either a) it is 'stay' (i.e. it is a
     constant at plan execution time), b) it has the given mark (indicating
     that it has been computed by a constraint appearing earlier in the
     plan), or c) it is not determined by any constraint."

    ^ (self inputsHasOne: [:v |
        (v mark = mark or: [v stay or: [v determinedBy == nil]]) not ]) not
  )

  markUnsatisfied = (
    "Record the fact that I am unsatisfied."
    self subclassResponsibility
  )

  output = (
    "Answer my current output variable. Raise an error if I am not
     currently satisfied."
    self subclassResponsibility
  )

  recalculate = (
    "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
     the value for the current output of this constraint. Assume this
     constraint is satisfied."
    self subclassResponsibility
```

```
  )

  satisfy: mark propagate: planner = (
    "Attempt to find a way to enforce this (still unsatisfied) constraint.
     If successful, record the solution, perhaps modifying the current
     dataflow graph.  Answer the constraint that this constraint overrides,
     if there is one, or nil, if there isn't."

    | overridden |
    self chooseMethod: mark.
    self isSatisfied
        ifTrue:         "constraint can be satisfied"
            ["mark inputs to allow cycle detection in addPropagate"
             | out |
             self inputsDo: [: in | in mark: mark].
             out := self output.
             overridden := out determinedBy.
             (overridden == nil) ifFalse: [overridden markUnsatisfied].
             out determinedBy: self.
             (planner addPropagate: self mark: mark) ifFalse:
                [self error: 'Cycle encountered adding:\tConstraint removed.'.
                 ^nil].
             out mark: mark]
        ifFalse:            "constraint cannot be satisfied"
            [overridden := nil.
             (strength sameAs: (Strength required)) ifTrue:
                [self error: 'Failed to satisfy a required constraint']].
    ^ overridden
  )
)
```

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
BinaryConstraint = AbstractConstraint (
  "I am an abstract superclass for constraints having two possible output
  variables.

  Instance variables:
      v1, v2      possible output variables <Variable>
      direction       one of:
                      #forward (v2 is output)
                      #backward ( v1 is output)
                      nil (not satisfied)"
  | v1 v2 direction |

  "initialize-release"

  initializeVar: variable1 var: variable2 strength: strengthSymbol addTo: planner = (
    "Initialize myself with the given variables and strength."
    super initialize: strengthSymbol.

    v1 := variable1.
    v2 := variable2.
    direction := nil.
  )

  "queries"
  isSatisfied = (
    "Answer true if this constraint is satisfied in the current solution."

    ^ direction notNil
  )

  "add/remove"
  addToGraph = (
    "Add myself to the constraint graph."
    v1 addConstraint: self.
    v2 addConstraint: self.
    direction := nil
  )

  removeFromGraph = (
    "Remove myself from the constraint graph."
    (v1 == nil) ifFalse: [v1 removeConstraint: self].
    (v2 == nil) ifFalse: [v2 removeConstraint: self].
    direction := nil.
  )

  "planning"
  chooseMethod: mark = (
    "Decide if I can be satisfied and which way I should flow based on
```

```
     the relative strength of the variables I relate, and record that
     decision."

    (v1 mark = mark) ifTrue:       "forward or nothing"
        [((v2 mark <> mark) and: [strength stronger: v2 walkStrength])
            ifTrue:  [ ^ direction := #forward ]
            ifFalse: [ ^ direction := nil ] ].

    (v2 mark = mark) ifTrue:       "backward or nothing"
        [((v1 mark <> mark) and: [strength stronger: v1 walkStrength])
            ifTrue:  [ ^ direction := #backward ]
            ifFalse: [ ^ direction := nil ] ].

    "if we get here, neither variable is marked, so we have choice"
    (v1 walkStrength weaker: v2 walkStrength)
        ifTrue:
            [(strength stronger: v1 walkStrength)
                ifTrue:  [ ^ direction := #backward ]
                ifFalse: [ ^ direction := nil ] ]
        ifFalse:
            [(strength stronger: v2 walkStrength)
                ifTrue:  [ ^ direction := #forward ]
                ifFalse: [ ^ direction := nil ] ].
  )

  execute = (
    "Enforce this constraint. Assume that it is satisfied."
    self subclassResponsibility
  )

  inputsDo: aBlock = (
    "Evaluate the given block on my current input variable."
    direction = #forward
      ifTrue:  [ aBlock value: v1 ]
      ifFalse: [ aBlock value: v2 ].
  )

  inputsHasOne: aBlock = (
    ^ direction = #forward
        ifTrue:  [ aBlock value: v1 ]
        ifFalse: [ aBlock value: v2 ]
  )

  markUnsatisfied = (
    "Record the fact that I am unsatisfied."
    direction := nil.
  )

  output = (
    "Answer my current output variable."
    (direction == #forward)
        ifTrue:  [ ^ v2 ]
        ifFalse: [ ^ v1 ]
  )

  recalculate = (
    "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
         the value for the current output of this constraint. Assume this
     constraint is satisfied."

```

```
    | in out |
    (direction = #forward)
        ifTrue:  [ in := v1. out := v2 ]
        ifFalse: [ in := v2. out := v1 ].
    out walkStrength: (strength weakest: in walkStrength).
    out stay: in stay.
    out stay ifTrue: [self execute].      "stay optimization"
  )
)
```

Examples/AreWeFastYet/DeltaBlue/DeltaBlue.som

```
"
This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html

Original comment.
”ä ÕY”FVÇF &ÇVR &Væ6†Ö &°
” UD„õ)”¦ö†â Ö ÆöæW’Â †V f–Ç’ ÖöF–f–VB '’ Ö &–ò vöÆ7¦¶ù (Mario.Wolczko@sun.com)
”eTä5D”ôéLanguage implementation benchmark
•5BÕdU%4”ôé4.1
• $U$U T•4•DU9
”4ôädÄ”5E9
”D•5E$”%UD”ôéworld
•dU%4”ôé“
”D DY “" ö7B  ““`
SUMMARY
This benchmark is an implementation of the DeltaBlue Constraint Solver
described in `The DeltaBlue Algorithm: An Incremental Constraint
Hierarchy Solver', by Bjorn N. Freeman-Benson and John Maloney,
Communications of the ACM, January 1990 (also as University of
Washington TR 89-08-06)
"
DeltaBlue = Benchmark (

  innerBenchmarkLoop: innerIterations = (
    Planner chainTest: innerIterations.
    Planner projectionTest: innerIterations.
    ^ true
  )

  ----

  new = (
    Strength initialize.
    ^ super new
  )
)
```

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
EditConstraint = UnaryConstraint (
  "I am a unary input constraint used to mark a variable that the client
   wishes to change."

  "queries"
  isInput = (
    "I indicate that a variable is to be changed by imperative code."
    ^ true
  )

  "execution"
  execute = (
    "Edit constraints do nothing."
  )

  ----

  "instance creation"

  var: aVariable strength: strengthSymbol addTo: planner = (
      "Install an edit constraint with the given strength on the given
       variable."

      ^ self new initializeVar: aVariable strength: strengthSymbol addTo: planner
  )
)

```

Examples/AreWeFastYet/DeltaBlue/EqualityConstraint.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
EqualityConstraint = BinaryConstraint (
  "I constrain two variables to have the same value: `v1 = v2`."

  initializeVar: variable1 var: variable2 strength: strengthSymbol addTo: planner = (
    super initializeVar: variable1 var: variable2 strength: strengthSymbol addTo: planner.
    self addConstraint: planner.
  )

  "execution"
  execute = (
    "Enforce this constraint. Assume that it is satisfied."
    direction = #forward
      ifTrue:  [ v2 value: v1 value ]
      ifFalse: [ v1 value: v2 value ].
  )

  ----

  "instance creation"

  var: variable1 var: variable2 strength: strengthSymbol addTo: planner = (
    "Install a constraint with the given strength equating the given
     variables."

    ^ self new initializeVar: variable1 var: variable2 strength: strengthSymbol addTo: planner
  )
)
```

Examples/AreWeFastYet/DeltaBlue/Plan.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
Plan = Vector (
  "A Plan is an ordered list of constraints to be executed in sequence to
  resatisfy all currently satisfiable constraints in the face of one or more
  changing inputs."

  "execution"
  execute = (
    "Execute my constraints in order."

    self forEach: [: c | c execute ]
  )

  ----

  new = ( ^ self new: 15 )
)
```

Examples/AreWeFastYet/DeltaBlue/Planner.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
Planner = (
  "This benchmark is an implementation of the DeltaBlue Constraint Solver
   described in `The DeltaBlue Algorithm: An Incremental Constraint
   Hierarchy Solver'', by Bjorn N. Freeman-Benson and John Maloney,
   Communications of the ACM, January 1990 (also as University of
   Washington TR 89-08-06).

   To run the benchmark, execute the expression `Planner standardBenchmark`."
  | currentMark |

  initialize = (
    currentMark := 1
  )

  "add/remove"

  incrementalAdd: c = (
    "Attempt to satisfy the given constraint and, if successful,
     incrementally update the dataflow graph.

     Details: If satifying the constraint is successful, it may override a
     weaker constraint on its output. The algorithm attempts to resatisfy
     that constraint using some other method. This process is repeated
     until either a) it reaches a variable that was not previously
     determined by any constraint or b) it reaches a constraint that
     is too weak to be satisfied using any of its methods. The variables
     of constraints that have been processed are marked with a unique mark
     value so that we know where we've been. This allows the algorithm to
     avoid getting into an infinite loop even if the constraint graph has
     an inadvertent cycle."

    | mark overridden |
    mark := self newMark.
    overridden := c satisfy: mark propagate: self.
    [overridden == nil] whileFalse:
        [overridden := overridden satisfy: mark propagate: self]
  )

  incrementalRemove: c = (
    "Entry point for retracting a constraint. Remove the given constraint,
     which should be satisfied, and incrementally update the dataflow
     graph.

     Details: Retracting the given constraint may allow some currently
     unsatisfiable downstream constraint be satisfied. We thus collect a
     list of unsatisfied downstream constraints and attempt to satisfy
     each one in turn. This list is sorted by constraint strength,
     strongest first, as a heuristic for avoiding unnecessarily adding
```

```
    and then overriding weak constraints."

   | out unsatisfied |
   out := c output.
   c markUnsatisfied.
   c removeFromGraph.
   unsatisfied := self removePropagateFrom: out.
   unsatisfied forEach: [: u | self incrementalAdd: u]
 )

 "planning/value propagation"
 extractPlanFromConstraints: constraints = (
   "Extract a plan for resatisfaction starting from the outputs of the
    given constraints, usually a set of input constraints."

   | sources |
   sources := Vector new.
   constraints forEach: [: c |
     (c isInput and: [c isSatisfied]) ifTrue: [sources append: c]].
   ^ self makePlan: sources
 )

 makePlan: sources = (
   "Extract a plan for resatisfaction starting from the given satisfied
    source constraints, usually a set of input constraints. This method
    assumes that stay optimization is desired; the plan will contain only
    constraints whose output variables are not stay. Constraints that do
    no computation, such as stay and edit constraints, are not included
    in the plan.

    Details: The outputs of a constraint are marked when it is added to
    the plan under construction. A constraint may be appended to the plan
    when all its input variables are known. A variable is known if either
    a) the variable is marked (indicating that has been computed by a
    constraint appearing earlier in the plan), b) the variable is 'stay'
    (i.e. it is a constant at plan execution time), or c) the variable
    is not determined by any constraint. The last provision is for past
    states of history variables, which are not stay but which are also
    not computed by any constraint."

   | mark plan todo c |
   mark := self newMark.
   plan := Plan new.
   todo := sources.
   [todo isEmpty] whileFalse:
       [c := todo removeFirst.
        ((c output mark <> mark) and:      "not in plan already and..."
         [c inputsKnown: mark]) ifTrue:    "eligible for inclusion"
           [plan append: c.
            c output mark: mark.
            self addConstraintsConsuming: c output to: todo]].
   ^ plan
 )

 propagateFrom: v = (
   "The given variable has changed. Propagate new values downstream."
   | todo c |
   todo := Vector new.
   self addConstraintsConsuming: v to: todo.
   [todo isEmpty] whileFalse:
     [c := todo removeFirst.
```

```
      c execute.
      self addConstraintsConsuming: c output to: todo].
)

"private"
addConstraintsConsuming: v to: aCollection = (
  | determiningC |
  determiningC := v determinedBy.
  v constraints forEach: [: c |
    ((c == determiningC) or: [c isSatisfied not]) ifFalse:
      [aCollection append: c]].
)

addPropagate: c mark: mark = (
  "Recompute the walkabout strengths and stay flags of all variables
   downstream of the given constraint and recompute the actual values
   of all variables whose stay flag is true. If a cycle is detected,
   remove the given constraint and answer false. Otherwise, answer true.

   Details: Cycles are detected when a marked variable is encountered
   downstream of the given constraint. The sender is assumed to have
   marked the inputs of the given constraint with the given mark. Thus,
   encountering a marked node downstream of the output constraint means
   that there is a path from the constraint's output to one of its
   inputs."

  | todo d |
  todo := Vector with: c.
  [todo isEmpty] whileFalse:
    [d := todo removeFirst.
     (d output mark = mark) ifTrue:
      [self incrementalRemove: c.
       ^ false].
     d recalculate.
     self addConstraintsConsuming: d output to: todo].
  ^ true
)

changeVar: aVariable newValue: newValue = (
  | editConstraint plan |
  editConstraint := EditConstraint var: aVariable strength: Strength SymPreferred addTo: self.
  plan := self extractPlanFromConstraints: (Vector with: editConstraint).
  10 timesRepeat: [
    aVariable value: newValue.
    plan execute ].
  editConstraint destroyConstraint: self.
)

constraintsConsuming: v do: aBlock = (
  | determiningC |
  determiningC := v determinedBy.
  v constraints forEach: [: c |
   (c == determiningC or: [c isSatisfied not]) ifFalse:
    [aBlock value: c]].
)

newMark = (
  "Select a previously unused mark value.

```

```
     Details: We just keep incrementing. If necessary, the counter will
     turn into a LargePositiveInteger. In that case, it will be a bit
     slower to compute the next mark but the algorithms will all behave
     correctly. We reserve the value '0' to mean 'unmarked'. Thus, this
     generator starts at '1' and will never produce '0' as a mark value."

    ^ currentMark := currentMark + 1
  )

  removePropagateFrom: out = (
    "Update the walkabout strengths and stay flags of all variables
     downstream of the given constraint. Answer a collection of unsatisfied
     constraints sorted in order of decreasing strength."

    | unsatisfied todo v |
    unsatisfied := Vector new.

    out determinedBy: nil.
    out walkStrength: Strength absoluteWeakest.
    out stay: true.
    todo := Vector with: out.
    [todo isEmpty] whileFalse: [
      v := todo removeFirst.
       v constraints forEach: [:c |
         c isSatisfied ifFalse: [unsatisfied append: c]].
       self constraintsConsuming: v do: [:c |
         c recalculate.
         todo append: c output]].

    unsatisfied sort: [:c1 :c2 | c1 strength stronger: c2 strength].
    ^ unsatisfied
  )

  ----

  "instance creation"
  new = (
    ^ super new initialize
  )

  "benchmarks"
  chainTest: n = (
      "Do chain-of-equality-constraints performance tests."
      | vars editConstraint plan planner |

      planner := Planner new.
      vars := Array new: n+1 withAll: [ Variable new ].

      "thread a chain of equality constraints through the variables"
      1 to: n do: [ :i |
        | v1 v2 |
        v1 := vars at: i.
        v2 := vars at: i + 1.
        EqualityConstraint var: v1 var: v2 strength: Strength SymRequired addTo: planner].

      StayConstraint var: vars last strength: Strength SymStrongDefault addTo: planner.
      editConstraint := EditConstraint var: vars first strength: Strength SymPreferred addTo: planner.
```

```
        plan := planner extractPlanFromConstraints: (Vector with: editConstraint).

        1 to: 100 do: [ :v |
          vars first value: v.
          plan execute.
          vars last value <> v ifTrue: [self error: 'Chain test failed!!']].

        editConstraint destroyConstraint: planner
    )

    projectionTest: n = (
        "This test constructs a two sets of variables related to each other by
         a simple linear transformation (scale and offset)."

        | scale offset src dst planner dests |
        planner := Planner new.
        dests   := Vector new.
        scale   := Variable value: 10.
        offset  := Variable value: 1000.

        1 to: n do: [ :i |
          src := Variable value: i.
          dst := Variable value: i.
          dests append: dst.
          StayConstraint  var: src strength: Strength SymDefault addTo: planner.
          ScaleConstraint var: src var: scale var: offset var: dst strength: Strength SymRequired addTo: planner
        ].

        planner changeVar: src newValue: 17.
        dst value <> 1170 ifTrue: [self error: 'Projection test 1 failed!!'].

        planner changeVar: dst newValue: 1050.
        src value <> 5 ifTrue: [self error: 'Projection test 2 failed!!'].

        planner changeVar: scale newValue: 5.
        1 to: n - 1 do: [ :i |
            (dests at: i) value <> (i*5 + 1000)
                ifTrue: [self error: 'Projection test 3 failed!!']].

        planner changeVar: offset newValue: 2000.
        1 to: n - 1 do: [ :i |
            (dests at: i) value <> (i*5 + 2000)
                ifTrue: [self error: 'Projection test 4 failed!!']]
    )
  )
```

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
ScaleConstraint = BinaryConstraint (
  "I relate two variables by the linear scaling relationship:
  `v2 = (v1 * scale) + offset`. Either v1 or v2 may be changed to maintain
  this relationship but the scale factor and offset are considered read-only.

  Instance variables:
      scale       scale factor input variable <Variable>
      offset      offset input variable <Variable>"
  | scale offset |

  "initialize-release"
  initializeSrc: srcVar scale: scaleVar offset: offsetVar dst: dstVar strength: strengthSymbol addTo: planner = (
    "Initialize myself with the given variables and strength."

    super initializeVar: srcVar var: dstVar strength: strengthSymbol addTo: planner.
    scale  := scaleVar.
    offset := offsetVar.

    self addConstraint: planner.
  )

  "add/remove"
  addToGraph = (
    "Add myself to the constraint graph."
    v1 addConstraint: self.
    v2 addConstraint: self.
    scale  addConstraint: self.
    offset addConstraint: self.
    direction := nil.
  )

  removeFromGraph = (
    "Remove myself from the constraint graph."
    v1 == nil ifFalse: [ v1 removeConstraint: self ].
    v2 == nil ifFalse: [ v2 removeConstraint: self ].
    scale  == nil ifFalse: [ scale removeConstraint: self ].
    offset == nil ifFalse: [ offset removeConstraint: self ].
    direction := nil.
  )

  "planning"
  execute = (
    "Enforce this constraint. Assume that it is satisfied."
    direction = #forward
        ifTrue:  [ v2 value: (v1 value * scale value) + offset value ]
        ifFalse: [ v1 value: (v2 value - offset value) / scale value ].
```

```
  )

  inputsDo: aBlock = (
    "Evaluate the given block on my current input variable."
    direction = #forward
        ifTrue: [aBlock value: v1.
                 aBlock value: scale.
                 aBlock value: offset]
        ifFalse: [aBlock value: v2.
                  aBlock value: scale.
                  aBlock value: offset].
  )

  recalculate = (
    "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
     the value for the current output of this constraint. Assume this
     constraint is satisfied."

    | in out |
    direction = #forward
        ifTrue:  [in  := v1. out := v2]
        ifFalse: [out := v1. in  := v2].
    out walkStrength: (strength weakest: in walkStrength).
    out stay: (in stay and: [scale stay and: [offset stay]]).
    out stay ifTrue: [self execute].      "stay optimization"
  )

  ----

  "instance creation"

  var: src var: scale var: offset var: dst strength: strengthSymbol addTo: planner = (
    "Install a scale constraint with the given strength on the given
     variables."
    ^ self new initializeSrc: src scale: scale offset: offset dst: dst strength: strengthSymbol addTo: planner
  )
)
```

Examples/AreWeFastYet/DeltaBlue/StayConstraint.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
StayConstraint = UnaryConstraint (
  "I mark variables that should, with some level of preference, stay the same.
   I have one method with zero inputs and one output, which does nothing.
   Planners may exploit the fact that, if I am satisfied, my output will not
   change during plan execution. This is called 'stay optimization.'"

  "execution"

  execute = (
    "Stay constraints do nothing."
  )

  ----

  "instance creation"
  var: aVariable strength: strengthSymbol addTo: planner = (
    "Install a stay constraint with the given strength on the given variable."

    ^ self new initializeVar: aVariable strength: strengthSymbol addTo: planner
  )
)

```

Examples/AreWeFastYet/DeltaBlue/Strength.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
Strength = (
  "Strengths are used to measure the relative importance of constraints. The
   hierarchy of available strengths is determined by the class variable
   StrengthTable (see my class initialization method). Because Strengths are
   invariant, references to Strength instances are shared (i.e. all references
   to `Strength of: #required` point to a single, shared instance). New
   strengths may be inserted in the strength hierarchy without disrupting
   current constraints.

   Instance variables:
       symbolicValue      symbolic strength name (e.g. #required) <Symbol>
       arithmeticValue    index of the constraint in the hierarchy, used for comparisons <Number>"
  | symbolicValue arithmeticValue |

  initializeWith: symVal = (
    symbolicValue   := symVal.
    arithmeticValue := Strength strengthTable at: symVal
  )

  "comparing"
  sameAs: aStrength = (
    "Answer true if I am the same strength as the given Strength."
    ^ arithmeticValue = aStrength arithmeticValue
  )

  stronger: aStrength = (
    "Answer true if I am stronger than the given Strength."
    ^ arithmeticValue < aStrength arithmeticValue
  )

  weaker: aStrength = (
    "Answer true if I am weaker than the given Strength."
    ^ arithmeticValue > aStrength arithmeticValue
  )

  "max/min"
  strongest: aStrength = (
    "Answer the stronger of myself and aStrength."

    (aStrength stronger: self)
      ifTrue:  [ ^ aStrength ]
      ifFalse: [ ^ self ].
  )

  weakest: aStrength = (
```

```
  "Answer the weaker of myself and aStrength."

  (aStrength weaker: self)
    ifTrue:  [ ^ aStrength ]
    ifFalse: [ ^ self ].
)

arithmeticValue = (
  "Answer my arithmetic value. Used for comparisons. Note that
   STRONGER constraints have SMALLER arithmetic values."

  ^ arithmeticValue
)

----
| AbsoluteStrongest AbsoluteWeakest Required StrengthConstants StrengthTable
  SymAbsoluteStrongest
  SymRequired
  SymStrongPreferred
  SymPreferred
  SymStrongDefault
  SymDefault
  SymWeakDefault
  SymAbsoluteWeakest |

new: symVal = (
  ^ self new initializeWith: symVal
)

strengthTable = (
    ^ StrengthTable
)

"class initialization"

createStrengthTable = (
  | table |
  table := SomIdentityDictionary new.
  table at: SymAbsoluteStrongest put: -10000.
  table at: SymRequired          put: -800.
  table at: SymStrongPreferred   put: -600.
  table at: SymPreferred         put: -400.
  table at: SymStrongDefault     put: -200.
  table at: SymDefault           put: 0.
  table at: SymWeakDefault       put: 500.
  table at: SymAbsoluteWeakest   put: 10000.
  ^ table
)

createStrengthConstants = (
  | constants |
  constants := SomIdentityDictionary new.
  StrengthTable keys forEach: [:strengthSymbol |
    constants
      at: strengthSymbol
      put: (self new: strengthSymbol)].
  ^ constants
)

initialize = (
```

```
    SymAbsoluteStrongest := Sym new: 0.
    SymRequired          := Sym new: 1.
    SymStrongPreferred   := Sym new: 2.
    SymPreferred         := Sym new: 3.
    SymStrongDefault     := Sym new: 4.
    SymDefault           := Sym new: 5.
    SymWeakDefault       := Sym new: 6.
    SymAbsoluteWeakest   := Sym new: 7.

    StrengthTable := self createStrengthTable.
    StrengthConstants := self createStrengthConstants.

    AbsoluteStrongest := Strength of: SymAbsoluteStrongest.
    AbsoluteWeakest := Strength of: SymAbsoluteWeakest.
    Required := Strength of: SymRequired.
  )

  "instance creation"
  of: aSymbol = (
    "Answer an instance with the specified strength."
    ^ StrengthConstants at: aSymbol
  )

  "constants"
  absoluteStrongest = (
    ^ AbsoluteStrongest
  )

  absoluteWeakest = (
    ^ AbsoluteWeakest
  )

  required = (
    ^ Required
  )

  SymAbsoluteStrongest = ( ^ SymAbsoluteStrongest )
  SymRequired          = ( ^ SymRequired )
  SymStrongPreferred   = ( ^ SymStrongPreferred )
  SymPreferred         = ( ^ SymPreferred )
  SymStrongDefault     = ( ^ SymStrongDefault )
  SymDefault           = ( ^ SymDefault )
  SymWeakDefault       = ( ^ SymWeakDefault )
  SymAbsoluteWeakest   = ( ^ SymAbsoluteWeakest )
)
```

```
"
Copyright (c) 2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Sym = (
  | hash |
  init: aHash = ( hash := aHash )

  customHash = ( ^ hash )
  ----

  new: aHash = (
    ^ self new init: aHash
  )
)
```

Examples/AreWeFastYet/DeltaBlue/UnaryConstraint.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
UnaryConstraint = AbstractConstraint (
  "I am an abstract superclass for constraints having a single possible output
  variable.

  Instance variables:
      output      possible output variable <Variable>
      satisfied       true if I am currently satisfied <Boolean>"
  | output satisfied |

  "initialize-release"

  initializeVar: aVariable strength: strengthSymbol addTo: planner = (
    "Initialize myself with the given variable and strength."
    super initialize: strengthSymbol.
    output := aVariable.
    satisfied := false.
    self addConstraint: planner.
  )

  "queries"
  isSatisfied = (
    "Answer true if this constraint is satisfied in the current solution."
    ^ satisfied
  )

  "add/remove"

  addToGraph = (
    "Add myself to the constraint graph."
    output addConstraint: self.
    satisfied := false.
  )

  removeFromGraph = (
    "Remove myself from the constraint graph."
    output == nil ifFalse: [output removeConstraint: self].
    satisfied := false.
  )

  "planning"
  chooseMethod: mark = (
    "Decide if I can be satisfied and record that decision."

    satisfied :=
        output mark <> mark and:
        [strength stronger: output walkStrength].
    ^ nil
  )
```

```
  execute = (
    "Enforce this constraint. Assume that it is satisfied."
    self subclassResponsibility
  )

  inputsDo: aBlock = (
    "I have no input variables."
  )

  inputsHasOne: aBlock = (
    ^ false
  )

  markUnsatisfied = (
    "Record the fact that I am unsatisfied."
    satisfied := false.
  )

  output = (
    "Answer my current output variable."
    ^ output
  )

  recalculate = (
    "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
     the value for the current output of this constraint. Assume this
     constraint is satisfied."

    output walkStrength: strength.
    output stay: self isInput not.
    output stay ifTrue: [self execute].   "stay optimization"
  )
)
```

Examples/AreWeFastYet/DeltaBlue/Variable.som

```
"This benchmark is derived from Mario Wolczko's Smalltalk version of DeltaBlue.

It is modified to use the SOM class library.
License details:

http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
Variable = (
  "I represent a constrained variable. In addition to my value, I maintain the
  structure of the constraint graph, the current dataflow graph, and various
  parameters of interest to the DeltaBlue incremental constraint solver.

  Instance variables:
      value           my value; changed by constraints, read by client <Object>
      constraints     normal constraints that reference me <Array of Constraint>
      determinedBy    the constraint that currently determines
                      my value (or nil if there isn't one) <Constraint>
      walkStrength        my walkabout strength <Strength>
      stay            true if I am a planning-time constant <Boolean>
      mark            used by the planner to mark constraints <Number>"
  | value constraints determinedBy walkStrength stay mark |

  "initialize-release"

  initialize = (
    value := 0.
    constraints  := Vector new: 2.
    determinedBy := nil.
    walkStrength := Strength absoluteWeakest.
    stay := true.
    mark := 0.
  )

  "access"
  addConstraint: aConstraint = (
    "Add the given constraint to the set of all constraints that refer
     to me."

    constraints append: aConstraint.
  )

  constraints = (
    "Answer the set of constraints that refer to me."
    ^ constraints
  )

  determinedBy = (
    "Answer the constraint that determines my value in the current
     dataflow."
    ^ determinedBy
  )

```

```
determinedBy: aConstraint = (
  "Record that the given constraint determines my value in the current
   data flow."
  determinedBy := aConstraint.
)

mark = (
  "Answer my mark value."
  ^ mark
)

mark: markValue = (
  "Set my mark value."
  mark := markValue.
)

removeConstraint: c = (
  "Remove all traces of c from this variable."
  constraints remove: c.
  determinedBy == c ifTrue: [ determinedBy := nil ].
)

stay = (
  "Answer my stay flag."
  ^ stay
)

stay: aBoolean = (
  "Set my stay flag."
  stay := aBoolean
)

value = (
  "Answer my value."
  ^ value
)

value: anObject = (
  "Set my value."
  value := anObject.
)

walkStrength = (
  "Answer my walkabout strength in the current dataflow."
  ^ walkStrength
)

walkStrength: aStrength = (
  "Set my walkabout strength in the current dataflow."
  walkStrength := aStrength.
)

----

"instance creation"

new = (
  ^ super new initialize
)

value: aValue = (
```

```
        | o |
        o := self new.
        o value: aValue.
        ^ o
    )
)

```

Examples/AreWeFastYet/Harness.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Harness = (

  processArguments: args= (
    | run i doGC |
    doGC := false.
    i := 2.

    ((args at: i) = '--gc-between-iterations' or: [(args at: i) = '--gc'])
      ifTrue: [
        doGC := true.
        i := i + 1 ].

    run := Run new: (args at: i).
    run doGC: doGC.

    args length > i ifTrue: [
      run numIterations: (args at: i + 1) asInteger.
      args length > (i + 1) ifTrue: [
        run innerIterations: (args at: i + 2) asInteger.
    ] ].
    ^ run
  )

  run: args = (
    | run |
    args length < 2 ifTrue: [
      self printUsage.
      system exit: 1
    ].

    run := self processArguments: args.

    run runBenchmark.
```

```
    run printTotal.
  )

  printUsage = (
    './som -cp Smalltalk Benchmarks/Harness.som [--gc-between-iterations|-gc] benchmark [num-iterations [inner-iter]]' println.
    '' println.
    '  --gc-between-iterations | --gc  - trigger a full GC between benchmark iteratons' println.
    '  benchmark      - benchmark class name' println.
    '  num-iterations - number of times to execute benchmark, default: 1' println.
    '  inner-iter     - number of times the benchmark is executed in an inner loop, ' println.
    '                   which is measured in total, default: 1' println.
  )
)
```

Examples/AreWeFastYet/Havlak/BasicBlock.som

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

BasicBlock = (
  | inEdges outEdges name |

  init: aName = (
    inEdges  := Vector new: 2.
    outEdges := Vector new: 2.
    name     := aName
  )

  inEdges  = ( ^ inEdges )
  outEdges = ( ^ outEdges )

  numPred = ( ^ inEdges size )

  addOutEdge: to = (
    outEdges append: to
  )

  addInEdge: from = (
    inEdges append: from
  )

  customHash = (
    ^ name
  )

  ----

  new: name = (
    ^ self new init: name
  )
)

```

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

BasicBlockEdge = (
  | from to |

  init: cfg from: fromName to: toName = (
    from := cfg createNode: fromName.
    to   := cfg createNode: toName.

    from addOutEdge: to.
    to   addInEdge:  from.
    cfg  addEdge:    self
  )

  ----

  for: cfg from: fromName to: toName = (
    ^ self new init: cfg from: fromName to: toName
  )
)

```

Examples/AreWeFastYet/Havlak/ControlFlowGraph.som

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

ControlFlowGraph = (
  | basicBlockMap startNode edgeList |

  initialize = (
    basicBlockMap := Vector new.
    edgeList := Vector new.
  )

  createNode: name = (
    | node |

    (basicBlockMap at: name) notNil
      ifTrue:  [ node := basicBlockMap at: name ]
      ifFalse: [
        node := BasicBlock new: name.
        basicBlockMap at: name put: node ].

    self numNodes = 1 ifTrue: [startNode := node].
    ^ node
  )

  addEdge: edge = (
    edgeList append: edge
  )

  numNodes = (
    ^ basicBlockMap size
  )

  startBasicBlock = (
    ^ startNode
  )

  basicBlocks = (
    ^ basicBlockMap
  )

  ----

  new = ( ^ super new initialize )
)
```

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

Havlak = Benchmark (
  innerBenchmarkLoop: innerIterations = (
    ^ self verifyResult:
       (LoopTesterApp new main: innerIterations loop: 50 p: 10 p: 10 p: 5)
      iterations: innerIterations
  )

  verifyResult: result iterations: innerIterations = (
    innerIterations = 15000 ifTrue: [ ^ (result at: 1) = 46602 and: [(result at: 2) = 5213] ].
    innerIterations =  1500 ifTrue: [ ^ (result at: 1) =  6102 and: [(result at: 2) = 5213] ].
    innerIterations =   150 ifTrue: [ ^ (result at: 1) =  2052 and: [(result at: 2) = 5213] ].
    innerIterations =    15 ifTrue: [ ^ (result at: 1) =  1647 and: [(result at: 2) = 5213] ].
    innerIterations =     1 ifTrue: [ ^ (result at: 1) =  1605 and: [(result at: 2) = 5213] ].

    ('No verification result for' + innerIterations + ' found') println.
    ('Result is ' + (result at: 1) + ', ' + (result at: 2)) println.
    ^ false
  )
)
```

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

HavlakLoopFinder = (
  | cfg lsg
    nonBackPreds backPreds number
    maxSize header type last nodes |

  init: aCfg lsg: aLsg = (
    cfg := aCfg.
    lsg := aLsg.

    nonBackPreds := Vector new.
    backPreds    := Vector new.
    number := SomIdentityDictionary new.

    maxSize := 0.
  )

  Unvisited = ( ^ 2147483647 )
  MaxNonBackPreds = ( ^ 32 * 1024 )

  isAncestor: w v: v = (
    ^ (w <= v) && (v <= (last at: w))
  )

  doDFS: currentNode current: current = (
    | lastId outerBlocks |

    (nodes at: current) initNode: currentNode dfs: current.
    number at: currentNode put: current.

    lastId := current.
    outerBlocks := currentNode outEdges.

    1 to: outerBlocks size do: [:i |
      | target |
      target := outerBlocks at: i.
      (number at: target) = self Unvisited ifTrue: [
        lastId := self doDFS: target current: lastId + 1 ] ].

    last at: current put: lastId.
    ^ lastId
  )

```

```
initAllNodes = (
  cfg basicBlocks forEach: [:bb |
    number at: bb put: self Unvisited ].

  self doDFS: cfg startBasicBlock current: 1
)

identifyEdges: size = (
  1 to: size do: [:w |
    | nodeW |
    header at: w put: 1.
    type   at: w put: #BBNonHeader.

    nodeW := (nodes at: w) bb.
    nodeW isNil
      ifTrue:  [ type at: w put: #BBDead ]
      ifFalse: [ self processEdges: nodeW w: w ] ]
)

processEdges: nodeW w: w = (
  nodeW numPred > 0 ifTrue: [
    nodeW inEdges forEach: [:nodeV |
      | v |
      v := number at: nodeV.
      v <> self Unvisited ifTrue: [
        (self isAncestor: w v: v)
          ifTrue:  [ (backPreds at: w) append: v ]
          ifFalse: [ (nonBackPreds at: w) add: v ] ] ] ]
)

findLoops = (
  | size |
  cfg startBasicBlock isNil ifTrue: [ ^ self ].

  size := cfg numNodes.

  nonBackPreds removeAll.
  backPreds removeAll.
  number removeAll.

  size > maxSize ifTrue: [
    header := Array new: size.
    type   := Array new: size.
    last   := Array new: size.
    nodes  := Array new: size.
    maxSize := size ].

  1 to: size do: [:i |
    nonBackPreds append: SomSet new.
    backPreds append: Vector new.
    nodes at: i put: UnionFindNode new ].

  self initAllNodes.
  self identifyEdges: size.
  header at: 1 put: 1.

  size downTo: 1 do: [:w |
    | nodePool nodeW |
    nodePool := Vector new.
    nodeW := (nodes at: w) bb.

```

```
      nodeW notNil ifTrue: [
        | workList |
        self stepD: w nodePool: nodePool.

        workList := Vector new.
        nodePool forEach: [:niter | workList append: niter ].

        nodePool size <> 0 ifTrue: [
          type at: w put: #BBReducible.
        ].

        [ workList isEmpty ] whileFalse: [
          | x nonBackSize |
          x := workList removeFirst.

          nonBackSize := (nonBackPreds at: x dfsNumber) size.
          nonBackSize > self MaxNonBackPreds ifTrue: [ ^ self ].
          self stepEProcessNonBackPreds: w nodePool: nodePool workList: workList x: x ].

        (nodePool size > 0 or: [(type at: w) = #BBSelf]) ifTrue: [
          | loop |
          loop := lsg createNewLoop: nodeW reducible: ((type at: w) ~= #BBIrreducible).
          self setLoopAttribute: w nodePool: nodePool loop: loop ] ] ]
  )

  stepEProcessNonBackPreds: w nodePool: nodePool workList: workList x: x = (
    (nonBackPreds at: x dfsNumber) forEach: [:iter |
      | y ydash |
      y := nodes at: iter.
      ydash := y findSet.

      (self isAncestor: w v: ydash dfsNumber) not
        ifTrue: [
          type at: w put: #BBIrreducible.
          (nonBackPreds at: w) add: ydash dfsNumber ]
        ifFalse: [
          ydash dfsNumber <> w ifTrue: [
            (nodePool hasSome: [:e | e == ydash]) ifFalse: [
              workList append: ydash.
              nodePool append: ydash ] ] ] ]
  )

  setLoopAttribute: w nodePool: nodePool loop: loop = (
    (nodes at: w) loop: loop.

    nodePool forEach: [:node |
      header at: node dfsNumber put: w.
      node union: (nodes at: w).

      node loop notNil
        ifTrue: [ node loop parent: loop ]
        ifFalse: [ loop addNode: node bb ] ]
  )

  stepD: w nodePool: nodePool = (
    (backPreds at: w) forEach: [:v |
      v <> w ifTrue: [ nodePool append: (nodes at: v) findSet ]
             ifFalse: [ type at: w put: #BBSelf ] ]
```

```
  )

  ----

  new: cfg lsg: lsg = (
    ^ self new init: cfg lsg: lsg
  )
)
```

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

LoopStructureGraph = (
  | root loops loopCounter |

  initialize = (
    root  := SimpleLoop basicBlock: nil reducible: false.
    loops := Vector new.
    loopCounter := 0.

    root nestingLevel: 0.
    root counter: loopCounter.
    loopCounter := loopCounter + 1.
    loops append: root
  )

  createNewLoop: bb reducible: isReducible = (
    | loop |
    loop := SimpleLoop basicBlock: bb reducible: isReducible.
    loop counter: loopCounter.
    loopCounter := loopCounter + 1.
    loops append: loop.
    ^ loop
  )

  calculateNestingLevel = (
    loops forEach: [:liter |
      liter isRoot ifFalse: [
        liter parent isNil ifTrue: [
          liter parent: root ] ] ].

    self calculateNestingLevelRec: root depth: 0
  )

  calculateNestingLevelRec: loop depth: depth = (
    loop depthLevel: depth.
    loop children forEach: [:liter |
      self calculateNestingLevelRec: liter depth: depth + 1.
      loop nestingLevel: (loop nestingLevel max: 1 + liter nestingLevel) ]
  )

  numLoops = (
    ^ loops size
  )
```

```
  ----

  new = (
    ^ super new initialize
  )
)
```

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

LoopTesterApp = (
  | cfg lsg |

  initialize = (
    cfg := ControlFlowGraph new.
    lsg := LoopStructureGraph new.
    cfg createNode: 1.
  )

  buildDiamond: start = (
    | bb0 |
    bb0 := start.
    BasicBlockEdge for: cfg from: bb0 to: bb0 + 1.
    BasicBlockEdge for: cfg from: bb0 to: bb0 + 2.
    BasicBlockEdge for: cfg from: bb0 + 1 to: bb0 + 3.
    BasicBlockEdge for: cfg from: bb0 + 2 to: bb0 + 3.
    ^ bb0 + 3
  )

  buildConnect: start end: end = (
    BasicBlockEdge for: cfg from: start to: end
  )

  buildStraight: start n: n = (
    0 to: n - 1 do: [:i |
      self buildConnect: start + i end: start + i + 1 ].
    ^ start + n
  )

  buildBaseLoop: from = (
    | header diamond1 d11 diamond2 footer |
    header   := self buildStraight: from n: 1.
    diamond1 := self buildDiamond: header.
    d11      := self buildStraight: diamond1 n: 1.
    diamond2 := self buildDiamond: d11.
    footer   := self buildStraight: diamond2 n: 1.

    self buildConnect: diamond2 end: d11.
    self buildConnect: diamond1 end: header.
    self buildConnect: footer   end: from.
    footer := self buildStraight: footer n: 1.
    ^ footer
```

```
  )

  main: numDummyLoops loop: findLoopIterations p: parLoop p: pparLoops p: ppparLoops = (
    self constructSimpleCFG.
    self addDummyLoops: numDummyLoops.
    self constructCFG: parLoop p: pparLoops p: ppparLoops.

    self findLoops: lsg.
    findLoopIterations timesRepeat: [
      self findLoops: LoopStructureGraph new ].

    lsg calculateNestingLevel.
    ^ Array with: lsg numLoops with: cfg numNodes
  )

  constructCFG: parLoops p: pparLoops p: ppparLoops = (
    | n |
    n := 3.

    parLoops timesRepeat: [
      cfg createNode: n + 1.
      self buildConnect: 2 end: n + 1.
      n := n + 1.

      pparLoops timesRepeat: [
        | top bottom |
        top := n.
        n := self buildStraight: n n:1.
        ppparLoops timesRepeat: [ n := self buildBaseLoop: n ].
        bottom := self buildStraight: n n: 1.
        self buildConnect: n end: top.
        n := bottom ].

      self buildConnect: n end: 1 ]
  )

  addDummyLoops: numDummyLoops = (
    numDummyLoops timesRepeat: [
      self findLoops: lsg ]
  )

  findLoops: loopStructure = (
    | finder |
    finder := HavlakLoopFinder new: cfg lsg: loopStructure.
    finder findLoops
  )

  constructSimpleCFG = (
    cfg createNode: 1.
    self buildBaseLoop: 1.
    cfg createNode: 2.
    BasicBlockEdge for: cfg from: 1 to:  3
  )

  ----

  new = ( ^ super new initialize )
)
```

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

SimpleLoop = (
  | counter depthLevel
    parent_ isRoot_ nestingLevel_
    header isReducible basicBlocks children |

  init: aBB reducible: aBool = (
    counter    := 0.
    depthLevel := 0.

    isRoot_ := false.
    nestingLevel_ := 0.
    header  := aBB.
    isReducible := aBool.
    basicBlocks := SomIdentitySet new.
    children    := SomIdentitySet new.

    aBB notNil ifTrue: [ basicBlocks add: aBB ]
  )

  counter = ( ^ counter )
  counter: val = ( counter := val )

  depthLevel = ( ^ depthLevel )
  depthLevel: val = ( depthLevel := val )

  children = ( ^ children )

  addNode: bb = (
    basicBlocks add: bb
  )

  addChildLoop: loop = (
    children add: loop
  )

  parent = ( ^ parent_ )
  parent: val = (
    parent_ := val.
    parent_ addChildLoop: self
  )

  isRoot    = ( ^ isRoot_ )
```

```
  setIsRoot = ( isRoot_ := true )

  nestingLevel = ( ^ nestingLevel_ )

  nestingLevel: level = (
    nestingLevel_ := level.
    level = 0 ifTrue: [ self setIsRoot ]
  )

  ----

  basicBlock: bb reducible: isReducible = (
    ^ self new init: bb reducible: isReducible
  )
)
```

```
"
Copyright 2011 Google Inc.

Licensed under the Apache License, Version 2.0 (the 'License');
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an 'AS IS' BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
"

UnionFindNode = (
  | parent_ bb_ dfsNumber_ loop |

  initialize = (
    dfsNumber_ := 0.
  )

  initNode: bb dfs: dfsNumber = (
    parent_ := self.
    bb_ := bb.
    dfsNumber_ := dfsNumber.
  )

  loop = ( ^ loop )
  loop: aLoop = ( loop := aLoop )

  findSet = (
    | nodeList node |
    nodeList := Vector new.

    node := self.

    [node ~= node parent] whileTrue: [
      ((node parent) ~= (node parent parent)) ifTrue: [
        nodeList append: node ].
      node := node parent ].

    nodeList forEach: [:iter | iter union: parent_ ].
    ^ node
  )

  union: basicBlock = (
    parent_ := basicBlock
  )

  parent    = ( ^ parent_ )
  bb        = ( ^ bb_ )
  dfsNumber = ( ^ dfsNumber_ )

  ----

  new = ( ^ super new initialize )
```

```
)
```

Examples/AreWeFastYet/Json/HashIndexTable.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
HashIndexTable = (
  | hashTable |

  initialize = (
    hashTable := Array new: 32 withAll: 0
  )

  at: name put: index = (
    | slot |
    slot := self hashSlotFor: name.

    index < 255
      ifTrue:  [ hashTable at: slot put: index + 1 ]
      ifFalse: [ hashTable at: slot put: 0 ]
  )

  at: name = (
    | slot |
    slot := self hashSlotFor: name.

    " subtract 1, 0 stands for empty "
    ^ ((hashTable at: slot) & 255) - 1
  )

  stringHash: s = (
    "this is not a proper hash, but sufficient for the benchmark,
     and very portable!"
```

```
     ^ s length * 1402589
  )

  hashSlotFor: element = (
    ^ ((self stringHash: element) & (hashTable length - 1)) + 1
  )

  ----

  new = ( ^ super new initialize )
)

```

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
Json = Benchmark (

  benchmark = (
    ^ (JsonParser with: Json RapBenchmarkMinified) parse.
  )

  verifyResult: result = (
    result isObject ifFalse: [ ^ false ].
    (result asObject at: 'head') isObject ifFalse: [ ^ false ].
    (result asObject at: 'operations') isArray ifFalse: [ ^ false ].
    ^ (result asObject at: 'operations') asArray size = 156
  )

  ----

  new = (
    JsonLiteral initialize.
    ^ super new
  )

  RapBenchmarkMinified = (
    ^ '{"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","indentionWidth":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"]]}]]}'
  )
)
```

Examples/AreWeFastYet/Json/JsonArray.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
JsonArray = JsonValue (
  | values |

  initialize = ( values := Vector new )

  add: value = (
    value ifNil: [ self error: 'value is null' ].
    values append: value
  )

  size = (
    ^ values size
  )

  at: index = (
    ^ values at: index
  )

  isArray = (
    ^ true
  )

  asArray = (
    ^ self
  )

  ----
```

```

  new = ( ^ super new initialize )
)

```

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
JsonLiteral = JsonValue (
  | value isNull isTrue isFalse |

  initializeWith: val = (
    value   := val.
    isNull  := 'null'  = val.
    isTrue  := 'true'  = val.
    isFalse := 'false' = val.
  )

  asString  = ( ^ value )
  isNull    = ( ^ isNull )
  isTrue    = ( ^ isTrue )
  isFalse   = ( ^ isFalse )
  isBoolean = ( ^ isTrue || isFalse )

  ----

  | NULL TRUE FALSE |

  initialize = (
    NULL  := self new initializeWith: 'null'.
    TRUE  := self new initializeWith: 'true'.
    FALSE := self new initializeWith: 'false'.
  )

  NULL  = ( ^ NULL )
```

```
  TRUE  = ( ^ TRUE )
  FALSE = ( ^ FALSE )
)

```

Examples/AreWeFastYet/Json/JsonNumber.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
JsonNumber = JsonValue (
  | string |

  initializeWith: str = ( string := str )

  asString = ( ^ string )
  isNumber = ( ^ true )

  ----

  new: string = (
    string ifNil: [ self error: 'string is null' ].
    ^ self new initializeWith: string
  )
)

```

Examples/AreWeFastYet/Json/JsonObject.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
JsonObject = JsonValue (
  | names values table |

  initialize = (
    names  := Vector new.
    values := Vector new.
    table  := HashIndexTable new
  )

  add: name with: aJsonValue = (
    name ifNil: [ self error: 'name is null'  ].
    aJsonValue ifNil: [ self error: 'aJsonValue is null' ].

    table at: name put: names size + 1. "+ 1 for 1-based indexing"
    names  append: name.
    values append: aJsonValue.
  )

  at: name = (
    | idx |
    name ifNil: [ self error: 'name is null'  ].
    idx := self indexOf: name.
    idx = 0
      ifTrue:  [ ^ nil ]
      ifFalse: [ ^ values at: idx ]
  )

```

```
  size = (
    ^ names size
  )

  isEmpty = (
    ^ names isEmpty
  )

  isObject = ( ^ true )
  asObject = ( ^ self )

  indexOf: name = (
    | idx |
    idx := table at: name.
    idx <> 0 && (name = (names at: idx)) ifTrue: [ ^ idx ].
    ^ self error: 'not implement'
  )

  ----

  new = ( ^ super new initialize )

  readFrom: string = (
    ^ (JsonValue readFrom: string) asObject
  )
)

```

Examples/AreWeFastYet/Json/JsonParser.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
JsonParser = (
  | input index line column current captureBuffer captureStart exceptionBlock |

  initializeWith: string = (
    input := string.
    index := 0.
    line  := 1.
    column := 0.
    current := nil.
    captureBuffer := ''.
    captureStart  := -1.
  )

  parse = (
    | result |
    exceptionBlock := [:ex | ^ ex ].
    self read.
    self skipWhiteSpace.
    result := self readValue.
    self skipWhiteSpace.
    self isEndOfText ifFalse: [ self error: 'Unexpected character'].
    ^ result
  )

  readValue = (
```

```
    current = 'n' ifTrue: [ ^ self readNull   ].
    current = 't' ifTrue: [ ^ self readTrue   ].
    current = 'f' ifTrue: [ ^ self readFalse  ].
    current = '"' ifTrue: [ ^ self readString ].
    current = '[' ifTrue: [ ^ self readArray  ].
    current = '{' ifTrue: [ ^ self readObject ].

    "Is this really the best way to write this?, or better #or:?,
     but with all the nesting, it's just ugly."
    current = '-' ifTrue: [ ^ self readNumber ].
    current = '0' ifTrue: [ ^ self readNumber ].
    current = '1' ifTrue: [ ^ self readNumber ].
    current = '2' ifTrue: [ ^ self readNumber ].
    current = '3' ifTrue: [ ^ self readNumber ].
    current = '4' ifTrue: [ ^ self readNumber ].
    current = '5' ifTrue: [ ^ self readNumber ].
    current = '6' ifTrue: [ ^ self readNumber ].
    current = '7' ifTrue: [ ^ self readNumber ].
    current = '8' ifTrue: [ ^ self readNumber ].
    current = '9' ifTrue: [ ^ self readNumber ].

    "else"
    self expected: 'value'
  )

  readArrayElement: array = (
    self skipWhiteSpace.
    array add: self readValue.
    self skipWhiteSpace.
  )

  readArray = (
    | array |
    self read.
    array := JsonArray new.

    "Array might be empty"
    self skipWhiteSpace.
    (self readChar: ']') ifTrue: [
      ^ array
    ].

    self readArrayElement: array.
    [self readChar: ','] whileTrue: [
      self readArrayElement: array.
    ].

    (self readChar: ']') ifFalse: [
      self expected: '"," or "]"'
    ].
    ^ array
  )

  readObjectKeyValuePair: object = (
    | name |
    self skipWhiteSpace.
    name := self readName.
    self skipWhiteSpace.

    (self readChar: ':') ifFalse: [ self expected: ':' ].

```

```
  self skipWhiteSpace.

  object add: name with: self readValue.

  self skipWhiteSpace.
)

readObject = (
  | object |
  self read.
  object := JsonObject new.
  self skipWhiteSpace.

  (self readChar: '}') ifTrue: [
    ^ object
  ].

  self readObjectKeyValuePair: object.
  [self readChar: ','] whileTrue: [
    self readObjectKeyValuePair: object.
  ].

  (self readChar: '}') ifFalse: [
    self expected: '"," or "}"'
  ].

  ^ object
)

readName = (
  current = '"' ifFalse: [ self expected: 'name' ].
  ^ self readStringInternal
)

readNull = (
  self read.
  self readRequiredChar: 'u'.
  self readRequiredChar: 'l'.
  self readRequiredChar: 'l'.
  ^ JsonLiteral NULL
)

readTrue = (
  self read.
  self readRequiredChar: 'r'.
  self readRequiredChar: 'u'.
  self readRequiredChar: 'e'.
  ^ JsonLiteral TRUE
)

readFalse = (
  self read.
  self readRequiredChar: 'a'.
  self readRequiredChar: 'l'.
  self readRequiredChar: 's'.
  self readRequiredChar: 'e'.
  ^ JsonLiteral FALSE
)

readRequiredChar: ch = (
  (self readChar: ch) ifFalse: [
```

```
      self expected: 'character: ' + ch
    ]
  )

  readString = (
    ^ JsonString new: self readStringInternal
  )

  readStringInternal = (
    | string |
    self read.
    self startCapture.

    [current = '"'] whileFalse: [
      current = '\\' ifTrue: [
        self pauseCapture.
        self readEscape.
        self startCapture.
      ] ifFalse: [
        "if (current < 0x20) { throw expected('valid string character'); }"
        "we currently don't have a way to get the ordinal value for a character"
        "} else {"
        self read.
      ]
    ].
    string := self endCapture.
    self read.
    ^ string
  )

  readEscapeChar = (
    current = '"'  ifTrue: [ ^ '"'  ].
    current = '/'  ifTrue: [ ^ '/'  ].
    current = '\\' ifTrue: [ ^ '\\' ].

    current = 'b'  ifTrue: [ ^ '\b'  ].
    current = 'f'  ifTrue: [ ^ '\f'  ].
    current = 'n'  ifTrue: [ ^ '\n'  ].
    current = 'r'  ifTrue: [ ^ '\r'  ].
    current = 't'  ifTrue: [ ^ '\t'  ].

    "TODO: SOM doesn't have a way to create unicode characters."
    self expected: 'valid escape sequence. note, some are not supported'
  )

  readEscape = (
    self read.
    captureBuffer := captureBuffer concatenate: self readEscapeChar.
    self read
  )

  readNumber = (
    | firstDigit |
    self startCapture.
    self readChar: '-'.
    firstDigit := current.

    self readDigit ifFalse: [ self expected: 'digit' ].
    firstDigit <> '0' ifTrue: [ [self readDigit] whileTrue: []].
```

```
    self readFraction.
    self readExponent.
    ^ JsonNumber new: self endCapture
  )

  readFraction = (
    (self readChar: '.') ifFalse: [ ^ false ].

    self readDigit ifFalse: [ self expected: 'digit' ].

    [self readDigit] whileTrue: [].

    ^ true
  )

  readExponent = (
    ((self readChar: 'e') not and: [
      (self readChar: 'E') not]) ifTrue: [ ^ false ].

    (self readChar: '+') ifFalse: [ self readChar: '-' ].

    self readDigit ifFalse: [ self expected: 'digit' ].

    [self readDigit] whileTrue: [].

    ^ true
  )

  readChar: ch = (
    current = ch ifFalse: [ ^ false ].
    self read.
    ^ true
  )

  readDigit = (
    self isDigit ifFalse: [ ^ false ].
    self read.
    ^ true
  )

  skipWhiteSpace = (
    [ self isWhiteSpace ]
      whileTrue:
        [ self read ].
  )

  read = (
    current = '\n' ifTrue: [
      line := line + 1.
      column := 0.
    ].

    index := index + 1.
    column := column + 1.

    input ifNil: [ self error:'input nil'].
    index <= input length
      ifTrue:  [ current := input charAt: index ]
      ifFalse: [ current := nil ]
  )
```

```

  startCapture = (
    captureStart := index
  )

  pauseCapture = (
    captureBuffer := captureBuffer concatenate: (
      input substringFrom: captureStart to: index - 1).
    captureStart := -1
  )

  endCapture = (
    | captured |
    '' = captureBuffer
      ifTrue:  [ captured := input substringFrom: captureStart to: index - 1 ]
      ifFalse: [
        self pauseCapture.
        captured := captureBuffer.
        captureBuffer := '' ].
    captureStart := -1.

    ^ captured
  )

  expected: expected = (
    self isEndOfText ifTrue: [
      self error: 'Unexpected end of input, expected ' + expected asString
    ].
    self error: 'Expected ' + expected
  )

  error: message = (
    "('error:' + message + ':' + line + ':' + column) println."
    exceptionBlock value: (ParseException with: message at: index
                                          line: line    column: column )
  )

  isWhiteSpace = (
    current = ' '  ifTrue: [^ true].
    current = '\t' ifTrue: [^ true].
    current = '\n' ifTrue: [^ true].
    current = '\r' ifTrue: [^ true].
    ^ false
  )

  isDigit = (
    current = '0' ifTrue: [^ true].
    current = '1' ifTrue: [^ true].
    current = '2' ifTrue: [^ true].
    current = '3' ifTrue: [^ true].
    current = '4' ifTrue: [^ true].
    current = '5' ifTrue: [^ true].
    current = '6' ifTrue: [^ true].
    current = '7' ifTrue: [^ true].
    current = '8' ifTrue: [^ true].
    current = '9' ifTrue: [^ true].
    ^ false
  )

  isEndOfText = (
```

```
    ^ current isNil
  )

  ----

  with: aJsonString = (
    ^ self new initializeWith: aJsonString
  )
)
```

Examples/AreWeFastYet/Json/JsonString.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
JsonString = JsonValue (
  | string |

  initializeWith: str = ( string := str )

  isString = (
    ^ true
  )

  ----

  new: str = ( ^ self new initializeWith: str )
)
```

Examples/AreWeFastYet/Json/JsonValue.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
JsonValue = (

  isObject  = ( ^ false )
  isArray   = ( ^ false )
  isNumber  = ( ^ false )
  isString  = ( ^ false )
  isBoolean = ( ^ false )

  isTrue  = ( ^ false )
  isFalse = ( ^ false )
  isNull  = ( ^ false )

  asObject = (
    self error: 'Unsupported operation, not an object: ' + self asString
  )

  asArray = (
    self error: 'Unsupported operation, not an array: ' + self asString
  )
)

```

Examples/AreWeFastYet/Json/ParseException.som

```
"
This benchmark is based on the minimal-json Java library maintained at:
https://github.com/ralfstx/minimal-json

Original copyright information:

Copyright (c) 2013, 2014 EclipseSource

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE."
ParseException = (
  | offset line column msg |

  initializeWith: message at: anOffset line: aLine column: aColumn = (
    msg    := message.
    offset := anOffset.
    line   := aLine.
    column := aColumn.
  )

  message = ( ^ msg )

  offset = ( ^ offset )
  line   = ( ^ line   )
  column = ( ^ column )

  asString = ( ^ msg + ':' + line + ':' + column )

  ----

  with: aMessageString at: offset line: line column: column = (
    ^ self new initializeWith: aMessageString
                           at: offset
                         line: line
                       column: column
  )
)
```

Examples/AreWeFastYet/List.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
List = Benchmark (

  benchmark = ( | result |
    result := self
        tailWithX: (self makeList: 15)
        withY:     (self makeList: 10)
        withZ:     (self makeList: 6).
    ^ result length
  )

  verifyResult: result = (
    ^ 10 = result
  )

  makeList: length = (
    (length = 0)
      ifTrue: [ ^ nil ]
      ifFalse: [
        | e |
        e := ListElement new: length.
        e next: (self makeList: (length - 1)).
        ^ e ]
  )

  isShorter: x than: y = (
    | xTail yTail |

    xTail := x. yTail := y.
    [ yTail isNil ]
      whileFalse: [
        xTail isNil ifTrue: [ ^ true ].
        xTail := xTail next.
        yTail := yTail next ].
```

```

    ^ false
  )

  tailWithX: x withY: y withZ: z = (
    (self isShorter: y than: x)
      ifTrue: [
        ^ (self
            tailWithX: (self tailWithX: x next withY: y withZ: z)
            withY: (self tailWithX: y next withY: z withZ: x)
            withZ: (self tailWithX: z next withY: x withZ: y)) ]
      ifFalse: [ ^ z ].
  )
)

```

Examples/AreWeFastYet/ListElement.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
ListElement = (
  | val next |

  length = ( next isNil ifTrue: [ ^ 1 ] ifFalse: [ ^ (1 + next length) ] )

  val    = ( ^ val )
  val: n = ( val := n )

  next          = ( ^ next )
  next: element = ( next := element )

  initialize: n = (
    val  := n.
    next := nil.
  )

  ------------------

  new: n = ( ^ self new initialize: n )
)
```

```
" This version is a transcription of the Ruby implementation mandelbrot.rb
  found with JRuby
  https://raw.githubusercontent.com/jruby/jruby/3e43676ee6dc3c13e70fe4a52cce685128c23b8e/bench/truffle/mandelbrot.rb

  Since then it has been modified in a number of ways by Stefan Marr.

  The original copyright statement reads as follows:

# Copyright © 2004-2013 Brent Fulgham
#
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
#   * Redistributions of source code must retain the above copyright notice,
#     this list of conditions and the following disclaimer.
#
#   * Redistributions in binary form must reproduce the above copyright notice,
#     this list of conditions and the following disclaimer in the documentation
#     and/or other materials provided with the distribution.
#
#   * Neither the name of 'The Computer Language Benchmarks Game' nor the name
#     of 'The Computer Language Shootout Benchmarks' nor the names of its
#     contributors may be used to endorse or promote products derived from this
#     software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS'
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
# OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# The Computer Language Benchmarks Game
# http://benchmarksgame.alioth.debian.org
#
#  contributed by Karl von Laudermann
```

```
#  modified by Jeremy Echols
#  modified by Detlef Reichl
#  modified by Joseph LaFata
#  modified by Peter Zotov
#
# http://benchmarksgame.alioth.debian.org/u64q/program.php?test=mandelbrot&lang=yarv&id=3
"
Mandelbrot = Benchmark (

  innerBenchmarkLoop: innerIterations = (
    ^ self verify: (self mandelbrot: innerIterations) inner: innerIterations
  )

  verify: result inner: innerIterations = (
    innerIterations = 500 ifTrue: [ ^ result = 191 ].
    innerIterations = 750 ifTrue: [ ^ result = 50  ].
    innerIterations = 1   ifTrue: [ ^ result = 128 ].

    ('No verification result for ' + innerIterations + ' found') println.
    ('Result is: ' + result asString) println.
    ^ false
  )

  mandelbrot: size = (
      | sum byteAcc bitNum y |
      sum     := 0.
      byteAcc := 0.
      bitNum  := 0.

      y := 0.

      [y < size] whileTrue: [
          | ci x |
          ci := (2.0 * y // size) - 1.0.
          x  := 0.

          [x < size] whileTrue: [
              | zrzr zi zizi cr escape z notDone |
              zrzr := 0.0.
              zizi := zi := 0.0.
              cr   := (2.0 * x // size) - 1.5.

              z := 0.
              notDone := true.
              escape := 0.
              [notDone and: [z < 50]] whileTrue: [
                  | zr |
                  zr := zrzr - zizi + cr.
                  zi := 2.0 * zr * zi + ci.

                  "preserve recalculation"
                  zrzr := zr * zr.
                  zizi := zi * zi.

                  (zrzr + zizi > 4.0) ifTrue: [
                      notDone := false.
                      escape  := 1.
                  ].
                  z := z + 1.
```

```
              ].

              byteAcc := (byteAcc << 1) + escape.
              bitNum  := bitNum + 1.

              " Code is very similar for these cases, but using separate blocks
                ensures we skip the shifting when it's unnecessary,
                which is most cases. "
              bitNum = 8
                  ifTrue: [
                    sum := sum bitXor: byteAcc.
                    byteAcc := 0.
                    bitNum  := 0. ]
                  ifFalse: [
                    (x = (size - 1)) ifTrue: [
                        byteAcc := byteAcc << (8 - bitNum).
                        sum := sum bitXor: byteAcc.
                        byteAcc := 0.
                        bitNum  := 0. ]].
              x := x + 1.
          ].
          y := y + 1.
      ].

      ^ sum
  )
)
```

```
" The Computer Language Benchmarks Game
  http://benchmarksgame.alioth.debian.org/

  contributed by Mark C. Lewis
  modified slightly by Chad Whipkey

  Based on nbody.java ported to SOM by Stefan Marr.
  See LICENSE.md file.
"
Body = (
  | x y z vx vy vz mass |

  initX: anX y: aY z: aZ vx: aVX vy: aVY vz: aVZ mass: aMass = (
    x := anX.
    y := aY.
    z := aZ.
    vx := aVX * Body DaysPerYear.
    vy := aVY * Body DaysPerYear.
    vz := aVZ * Body DaysPerYear.
    mass := aMass * Body SolarMass.
  )

  x = ( ^ x )
  y = ( ^ y )
  z = ( ^ z )

  vx = ( ^ vx )
  vy = ( ^ vy )
  vz = ( ^ vz )

  mass = ( ^ mass )

  x: val = ( x := val )
  y: val = ( y := val )
  z: val = ( z := val )

  vx: val = ( vx := val )
  vy: val = ( vy := val )
  vz: val = ( vz := val )

  mass: val = ( mass := val )

  offsetMomentumX: px y: py z: pz = (
    vx := 0.0 - (px // Body SolarMass).
    vy := 0.0 - (py // Body SolarMass).
    vz := 0.0 - (pz // Body SolarMass).
  )

  print = (
    'x:  ' print. x println.
    'y:  ' print. y println.
    'z:  ' print. z println.

    'vx: ' print. vx println.
    'vy: ' print. vy println.
    'vz: ' print. vz println.

    'mass: ' print. mass println.
```

```
)

----

| solarMass |

Pi          = ( ^ 3.141592653589793 )
SolarMass   = ( ^ solarMass )
DaysPerYear = ( ^ 365.24 )

initialize = (
  solarMass := 4 * self Pi * self Pi.
)

jupiter = (
   ^ super new
       initX:    4.84143144246472090
           y:   -1.16032004402742839
           z:   -0.103622044471123109
           vx:   0.00166007664274403694
           vy:   0.00769901118419740425
           vz:  -0.0000690460016972063023
           mass: 0.000954791938424326609
)

saturn = (
  ^ super new
      initX:    8.34336671824457987
          y:    4.12479856412430479
          z:   -0.403523417114321381
          vx:  -0.00276742510726862411
          vy:   0.00499852801234917238
          vz:   0.0000230417297573763929
          mass: 0.000285885980666130812
)

uranus = (
  ^ super new
      initX:   12.894369562139131
          y:  -15.111151401698631 2
          z:   -0.223307578892655734
          vx:   0.00296460137564761618
          vy:   0.0023784717395948095
          vz:  -0.0000296589568540237556
          mass: 0.0000436624404335156298
)

neptune = (
  ^ super new
      initX:   15.3796971148509165
          y:  -25.9193146099879641
          z:    0.179258772950371181
          vx:   0.00268067772490389322
          vy:   0.00162824170038242295
          vz:  -0.000095159225451971587
          mass: 0.0000515138902046611451
)

sun = (
  ^ super new initX: 0.0 y: 0.0 z: 0.0 vx: 0.0 vy: 0.0 vz: 0.0 mass: 1.0
)
```

120)

Examples/AreWeFastYet/NBody/NBody.som

```
" The Computer Language Benchmarks Game
  http://shootout.alioth.debian.org/

  contributed by Mark C. Lewis
  modified slightly by Chad Whipkey

  Based on nbody.java ported to SOM by Stefan Marr.
  See LICENSE.md file.
"
NBody = Benchmark (
  innerBenchmarkLoop: innerIterations = (
    | system |
    system := NBodySystem new.

    1 to: innerIterations do: [:i |
      system advance: 0.01.
    ].

    ^ self verify: system energy for: innerIterations.
  )

  verify: result for: innerIterations = (
    innerIterations = 250000 ifTrue: [ ^ result = -0.1690859889909308  ].
    innerIterations =      1 ifTrue: [ ^ result = -0.16907495402506745 ].

    ('No verification result for ' + innerIterations asString + ' found') println.
    ('Result is: ' + result asString) println.
    ^ false
  )

  ----

  new = (
    Body initialize.
      ^ super new
  )
)
```

```
" The Computer Language Benchmarks Game
  http://shootout.alioth.debian.org/

  contributed by Mark C. Lewis
  modified slightly by Chad Whipkey

  Based on nbody.java ported to SOM by Stefan Marr.
  See LICENSE.md file.
"
NBodySystem = (
  | bodies |

  initialize = (
    bodies := self createBodies
  )

  createBodies = (
    | px py pz bodies |

    bodies := Array new: 5.
    bodies at: 1 put: Body sun.
    bodies at: 2 put: Body jupiter.
    bodies at: 3 put: Body saturn.
    bodies at: 4 put: Body uranus.
    bodies at: 5 put: Body neptune.

    "bodies do: [:b | b print. '' println ]."

    px := 0.0.  py := 0.0.  pz := 0.0.

    bodies do: [:b |
        px := px + (b vx * b mass).
        py := py + (b vy * b mass).
        pz := pz + (b vz * b mass).
    ].

    (bodies at: 1) offsetMomentumX: px y: py z: pz.

    "bodies do: [:b | b print. '' println ]."
    ^ bodies
  )

  advance: dt = (
    1 to: bodies length do: [:i |
      | iBody |
      iBody := bodies at: i.

      i + 1 to: bodies length do: [:j |
        | dx dy dz jBody dSquared distance mag |
        jBody := bodies at: j.
        dx := iBody x - jBody x.
        dy := iBody y - jBody y.
        dz := iBody z - jBody z.

        dSquared := (dx * dx) + (dy * dy) + (dz * dz).
        distance := dSquared sqrt.
        mag      := dt // (dSquared * distance).

```

```
        iBody vx: iBody vx - (dx * jBody mass * mag).
        iBody vy: iBody vy - (dy * jBody mass * mag).
        iBody vz: iBody vz - (dz * jBody mass * mag).

        jBody vx: jBody vx + (dx * iBody mass * mag).
        jBody vy: jBody vy + (dy * iBody mass * mag).
        jBody vz: jBody vz + (dz * iBody mass * mag).
      ].
    ].

    bodies do: [:body |
       body x: body x + (dt * body vx).
       body y: body y + (dt * body vy).
       body z: body z + (dt * body vz).
    ].
  )

  energy = (
    | e |
    e := 0.0.

    1 to: bodies length do: [:i |
      | iBody |
      iBody := bodies at: i.

      e := e + (0.5 * iBody mass *
           ((iBody vx * iBody vx) +
            (iBody vy * iBody vy) +
            (iBody vz * iBody vz))).

      i + 1 to: bodies length do: [:j |
          | jBody dx dy dz distance |
          jBody := bodies at: j.

          dx := iBody x - jBody x.
          dy := iBody y - jBody y.
          dz := iBody z - jBody z.

          distance := ((dx*dx) + (dy*dy) + (dz*dz)) sqrt.
          e := e - ((iBody mass * jBody mass) // distance).
      ].
    ].
    ^ e
  )

  ----

  new = (
      ^ super new initialize
  )
)
```

Examples/AreWeFastYet/Permute.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Permute = Benchmark (

  | count v |

  benchmark = (
    count := 0.
    v     := Array new: 6 withAll: 0.
    self permute: 6.
    ^ count
  )

  verifyResult: result = (
    ^ 8660 = result
  )

  permute: n = (
    count := count + 1.
    (n <> 0)
      ifTrue: [
        self permute: n - 1.
        n downTo: 1 do: [ :i |
          self swap: n with: i.
          self permute: n - 1.
          self swap: n with: i ] ]
  )

  swap: i with: j = (
    | tmp |
    tmp := v at: i.
    v at: i put: (v at: j).
    v at: j put: tmp
  )
)
```

Examples/AreWeFastYet/Queens.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Queens = Benchmark (

  | freeMaxs freeRows freeMins queenRows |

  benchmark = (
    | result |
    result := true.
    1 to: 10 do: [ :i | result := result and: self queens ].
    ^ result
  )

  verifyResult: result = (
    ^ result
  )

  queens = (
    freeRows  := Array new:  8 withAll: true.
    freeMaxs  := Array new: 16 withAll: true.
    freeMins  := Array new: 16 withAll: true.
    queenRows := Array new:  8 withAll: -1.
    ^ self placeQueen: 1
  )

  placeQueen: c = (
    1 to: 8 do: [ :r |
      (self row: r column: c)
        ifTrue: [
          queenRows at: r put: c.
          self row: r column: c put: false.
          (c = 8) ifTrue: [ ^ true ].
          (self placeQueen: c + 1) ifTrue: [ ^ true ].
          self row: r column: c put: true ] ].
    ^ false
```

```
  )

  row: r column: c = (
    ^ (freeRows at: r) && (freeMaxs at: c + r) && (freeMins at: c - r + 8)
  )

  row: r column: c put: v = (
    freeRows at: r         put: v.
    freeMaxs at: c + r     put: v.
    freeMins at: c - r + 8 put: v.
  )
)
```

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
"
DeviceTaskDataRecord = RBObject (
  | pending |

  pending = ( ^ pending )
  pending: packet = ( pending := packet )

  create = ( pending := RBObject NoWork )
  ----
  create = (
    ^ super new create
  )
)

```

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
HandlerTaskDataRecord = RBObject (
  | workIn deviceIn |

  deviceIn = ( ^ deviceIn )

  deviceIn: aPacket = ( deviceIn := aPacket )

  deviceInAdd: packet = (
      deviceIn := self append: packet head: deviceIn
  )

  workIn = ( ^ workIn )

  workIn: aWorkQueue = ( workIn := aWorkQueue )

  workInAdd: packet = (
      workIn := self append: packet head: workIn
  )

  create = (
    workIn := deviceIn := RBObject NoWork
  )
  asString = (
      ^ 'HandlerTaskDataRecord(' + workIn asString + ', ' + deviceIn asString + ')'
  )
  ----
  create = ( ^ super new create )
)

```

Examples/AreWeFastYet/Richards/IdleTaskDataRecord.som

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
IdleTaskDataRecord = RBObject (
  | control count |

  control = ( ^ control )

  control: aNumber = (
    control := aNumber
  )

  count = ( ^ count )

  count: aCount = (
    count := aCount
  )

  create = (
    control := 1.
    count := 10000
  )

  ----

  create = ( ^ super new create )
)

```

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
Packet = RBObject (
  | link identity kind datum data |

  data  = ( ^ data )

  datum = ( ^ datum )
  datum: someData = ( datum := someData )

  identity = ( ^ identity )
  identity: anIdentity = ( identity := anIdentity )

  kind = ( ^ kind )
  link = ( ^ link )

  link: aWorkQueue = ( link := aWorkQueue )
  link: aLink identity: anIdentity kind: aKind = (
    link := aLink.
    kind := aKind.
    identity := anIdentity.
    datum := 1.
    data  := Array new: 4 withAll: 0
  )

  asString = (
      ^ 'Packet(' +
          link asString + ', ' +
          identity asString + ', ' +
          kind asString + ', ' +
          datum asString + ', ' +
          data asString +
        ')'
  )

  ----

  create: link identity: identity kind: kind = (
    ^ super new
        link: link
        identity: identity
        kind: kind
  )
)
```

Examples/AreWeFastYet/Richards/RBObject.som

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
RBObject = Object (

  append: packet head: queueHead = (
    | mouse link |
    packet link: RBObject NoWork.
    RBObject NoWork == queueHead ifTrue: [ ^ packet ].
    mouse := queueHead.
    [RBObject NoWork == (link := mouse link)]
            whileFalse: [mouse := link].
    mouse link: packet.
    ^ queueHead
  )

  ----

  NoTask  = ( ^ nil )
  Idler   = ( ^ 1   )
  NoWork  = ( ^ nil )
  Worker  = ( ^ 2 )
  WorkPacketKind = ( ^ 2 )
  HandlerA = ( ^ 3 )
  HandlerB = ( ^ 4 )
  DeviceA  = ( ^ 5 )
  DeviceB  = ( ^ 6 )
  DevicePacketKind = ( ^ 1 )

)

```

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
Richards = Benchmark (
  benchmark = (
    ^ Scheduler new start.
  )

  verifyResult: result = (
    ^ result
  )
)

```

Examples/AreWeFastYet/Richards/Scheduler.som

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
Scheduler = RBObject (
  |taskList currentTask currentTaskIdentity taskTable tracing layout queuePacketCount holdCount|

  initialize = (
    taskList    := RBObject NoTask.
    currentTask := RBObject NoTask.
    currentTaskIdentity := 0.
    taskTable   := Array new: 6 withAll: RBObject NoTask.
    layout      := 0.
    queuePacketCount := 0.
    holdCount   := 0.
  )

  tracing = (
    ^ false
  )

  createDevice: identity priority: priority work: work state: state = (
    | data |
    data := DeviceTaskDataRecord create.
    self
      createTask: identity
      priority: priority
      work: work
      state: state
      function:
        [:work :word |
        | data functionWork |
        data := word.
        functionWork := work.
        RBObject NoWork == functionWork ifTrue: [
          RBObject NoWork == (functionWork := data pending)
            ifTrue: [ self wait ]
            ifFalse: [
              data pending: RBObject NoWork.
              self queuePacket: functionWork]]
          ifFalse: [
            data pending: functionWork.
            self tracing ifTrue: [
              self trace: functionWork datum].
            self holdSelf]]
      data: data
  )

  createHandler: identity priority: priority work: work state: state = (
    | data |
    data := HandlerTaskDataRecord create.
    self createTask: identity
         priority:   priority
```

```
        work:       work
        state:      state
        function:   [:work :word |
          | data workPacket |
          data := word.
          RBObject NoWork == work ifFalse: [
            RBObject WorkPacketKind == work kind
              ifTrue:  [ data workInAdd: work ]
              ifFalse: [ data deviceInAdd: work ]].

          RBObject NoWork == (workPacket := data workIn)
            ifTrue: [ self wait ]
            ifFalse: [
              | count |
              count := workPacket datum.
              count > 4
                ifTrue: [
                  data workIn: workPacket link.
                  self queuePacket: workPacket]
                ifFalse: [
                  | devicePacket |
                  RBObject NoWork == (devicePacket := data deviceIn)
                    ifTrue: [ self wait ]
                    ifFalse: [
                      data deviceIn: devicePacket link.
                      devicePacket datum: (workPacket data at: count).
                      workPacket datum: count + 1.
                      self queuePacket: devicePacket]]]]
        data: data
)

createIdler: identity priority: priority work: work state: state = (
    | data |
    data := IdleTaskDataRecord create.
    self createTask: identity
         priority:   priority
         work:       work
         state:      state
         function:  [:work :word |
           | data |
           data := word.
           data count: data count - 1.
           0 = data count
             ifTrue:  [ self holdSelf ]
             ifFalse: [
               0 = (data control & 1 )
               ifTrue: [
                 data control: data control / 2.
                 self release: RBObject DeviceA]
               ifFalse: [
                 data control: (data control / 2 bitXor: 53256).
                 self release: RBObject DeviceB]]]
         data: data
)

createPacket: link identity: identity kind: kind = (
  ^ Packet create:   link
           identity: identity
           kind:     kind
)
```

```
createTask: identity priority: priority work: work state: state function: aBlock data: data = (
  | t |
  t := TaskControlBlock   link:             taskList
                          create:           identity
                          priority:         priority
                          initialWorkQueue: work
                          initialState:     state
                          function:         aBlock
                          privateData:      data.
  taskList := t.
  taskTable at: identity put: t
)

createWorker: identity priority: priority work: work state: state = (
    | data |
    data := WorkerTaskDataRecord create.
    self createTask: identity
         priority:   priority
         work:       work
         state:      state
         function: [:work :word |
           | data |
           data := word.
           RBObject NoWork == work
             ifTrue: [ self wait ]
             ifFalse: [
               data destination: (RBObject HandlerA = data destination
                                    ifTrue:  [RBObject HandlerB]
                                    ifFalse: [RBObject HandlerA]).
               work identity: data destination.
               work datum: 1.
               1 to: 4 do: [:i |
                 data count: data count + 1.
                 data count > 26 ifTrue: [data count: 1].
                 "work data at: i put: $A asInteger + data count - 1]."
                 work data at: i put: 65 + data count - 1].
               self queuePacket: work]]
        data: data
)

start = (
  | workQ |
  self
          createIdler: RBObject Idler
          priority: 0
          work: RBObject NoWork
          state: TaskState running.
  workQ := self
                          createPacket: RBObject NoWork
                          identity: RBObject Worker
                          kind: RBObject WorkPacketKind.
  workQ := self
                          createPacket: workQ
                          identity: RBObject Worker
                          kind: RBObject WorkPacketKind.
  self
          createWorker: RBObject Worker
          priority: 1000
          work: workQ
          state: TaskState waitingWithPacket.
```

```
  workQ := self
                          createPacket: RBObject NoWork
                          identity: RBObject DeviceA
                          kind: RBObject DevicePacketKind.
  workQ := self
                          createPacket: workQ
                          identity: RBObject DeviceA
                          kind: RBObject DevicePacketKind.
  workQ := self
                          createPacket: workQ
                          identity: RBObject DeviceA
                          kind: RBObject DevicePacketKind.
  self
          createHandler: RBObject HandlerA
          priority: 2000
          work: workQ
          state: TaskState waitingWithPacket.
  workQ := self
                          createPacket: RBObject NoWork
                          identity: RBObject DeviceB
                          kind: RBObject DevicePacketKind.
  workQ := self
                          createPacket: workQ
                          identity: RBObject DeviceB
                          kind: RBObject DevicePacketKind.
  workQ := self
                          createPacket: workQ
                          identity: RBObject DeviceB
                          kind: RBObject DevicePacketKind.
  self
          createHandler: RBObject HandlerB
          priority: 3000
          work: workQ
          state: TaskState waitingWithPacket.
  self
          createDevice: RBObject DeviceA
          priority: 4000
          work: RBObject NoWork
          state: TaskState waiting.
  self
          createDevice: RBObject DeviceB
          priority: 5000
          work: RBObject NoWork
          state: TaskState waiting.

  self schedule.

  ^ queuePacketCount = 23246 and: holdCount = 9297
)

findTask: identity = (
  | t |
  t := taskTable at: identity.
  RBObject NoTask == t ifTrue: [self error: 'findTask failed'].
  ^ t
)

holdSelf = (
  holdCount := holdCount + 1.
  currentTask taskHolding: true.
  ^ currentTask link
```

```
  )

  queuePacket: packet = (
    | t |
    t := self findTask: packet identity.
    RBObject NoTask == t ifTrue: [ ^ RBObject NoTask ].
    queuePacketCount := queuePacketCount + 1.
    packet link: RBObject NoWork.
    packet identity: currentTaskIdentity.
    ^ t addInput: packet checkPriority: currentTask
  )

  release: identity = (
      | t |
      t := self findTask: identity.
      RBObject NoTask == t ifTrue: [ ^ RBObject NoTask ].
      t taskHolding: false.
      t priority > currentTask priority
        ifTrue:  [ ^ t ]
        ifFalse: [ ^ currentTask ]
  )

  trace: id = (
    layout := layout - 1.
    0 >= layout ifTrue: [
      '' println.
      layout := 50 ].
    id print
  )

  wait = (
    currentTask taskWaiting: true.
    ^currentTask
  )

  schedule = (
    currentTask := taskList.
    [ RBObject NoTask == currentTask ] whileFalse: [
      currentTask isTaskHoldingOrWaiting
        ifTrue:  [ currentTask := currentTask link ]
        ifFalse: [
          currentTaskIdentity := currentTask identity.
          self tracing ifTrue: [ self trace: currentTaskIdentity ].
          currentTask := currentTask runTask ] ]
  )

  ----

  new = ( ^ super new initialize )
)
```

Examples/AreWeFastYet/Richards/TaskControlBlock.som

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
TaskControlBlock = TaskState (
  | link identity priority input function handle |

  identity = ( ^ identity )
  link     = ( ^ link )
  priority = ( ^ priority )

  link: aLink identity: anIdentity priority: aPriority initialWorkQueue: anInitialWorkQueue initialState: anInitialState function: aBlock privateData: aPrivateData = (
    link     := aLink.
    identity := anIdentity.
    function := aBlock.
    priority := aPriority.
    input    := anInitialWorkQueue.
    handle   := aPrivateData.
    self packetPending: anInitialState isPacketPending.
    self taskWaiting: anInitialState isTaskWaiting.
    self taskHolding: anInitialState isTaskHolding.
  )

  addInput: packet checkPriority: oldTask = (
    RBObject NoWork == input
      ifTrue: [
        input := packet.
        self packetPending: true.
        priority > oldTask priority ifTrue: [ ^ self ] ]
      ifFalse: [
        input := self append: packet head: input ].
    ^ oldTask
  )

  runTask = (
    | message |
    self isWaitingWithPacket
      ifTrue: [
        message := input.
        input := message link.
        RBObject NoWork == input
          ifTrue: [self running]
          ifFalse: [self packetPending]]
      ifFalse: [message := RBObject NoWork].
    ^ function value: message with: handle
  )

  ----

  link: link create: identity priority: priority initialWorkQueue: initialWorkQueue initialState: initialState function: aBlock privateData: privateData = (
```

```
		^super new
			link: link
			identity: identity
			priority: priority
			initialWorkQueue: initialWorkQueue
			initialState: initialState
			function: aBlock
			privateData: privateData
	)
)

```

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
    http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/mario/java_benchmarking/index.html
"
TaskState = RBObject (
    | packetPending taskWaiting taskHolding |

    isPacketPending = ( ^ packetPending )

    isTaskHolding = ( ^ taskHolding )

    isTaskWaiting = ( ^ taskWaiting )

    taskHolding: aBoolean   = ( taskHolding   := aBoolean )
    taskWaiting: aBoolean   = ( taskWaiting   := aBoolean )
    packetPending: aBoolean = ( packetPending := aBoolean )

    packetPending = (
        packetPending := true.
        taskWaiting := false.
        taskHolding := false
    )

    running = (
        packetPending := taskWaiting := taskHolding := false
    )

    waiting = (
        packetPending := taskHolding := false.
        taskWaiting := true
    )

    waitingWithPacket = (
        taskHolding := false.
        taskWaiting := packetPending := true
    )

    isTaskHoldingOrWaiting = ( ^ taskHolding or: [packetPending not and: [taskWaiting]] )

    isWaitingWithPacket = ( ^ packetPending and: [taskWaiting and: [taskHolding not]] )

    ----
    running = ( ^ super new running )

    waiting = ( ^ super new waiting )

    waitingWithPacket = ( ^ super new waitingWithPacket )
)
```

```
"
This benchmark is derived from richards.st, which is
part of Mario Wolczko's DeltaBlue and Richards collection.

License details:
  http://web.archive.org/web/20050825101121/http://www.sunlabs.com/people/
mario/java_benchmarking/index.html
"
WorkerTaskDataRecord = RBObject (
  |destination count|
  count = ( ^ count )

  count: aCount =( count := aCount )

  destination = ( ^ destination )

  destination: aHandler = ( destination := aHandler )

  create = (
      destination := RBObject HandlerA.
      count := 0
  )
  ----
  create = ( ^ super new create )
)

```

Examples/AreWeFastYet/Run.som

```
"
Copyright (c) 2011-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Run = (
  | total numIterations innerIterations benchmarkSuite name doGC |

  initialize: aName = (
    name            := aName.
    benchmarkSuite  := self loadBenchmarkSuite: aName.
    total           := 0.
    numIterations   := 1.
    innerIterations := 1.
  )

  loadBenchmarkSuite: className = (
    | sym cls |
    sym := className asSymbol.
    cls := system load: sym.
    cls ifNil: [
        self error: 'Failed loading benchmark: ', className ].
    ^ cls
  )

  name: aString = ( name := aString )
  benchmarkSuite: aSuite = ( benchmarkSuite  := aSuite )
  numIterations:   anInt = ( numIterations   := anInt )
  innerIterations: anInt = ( innerIterations := anInt )
  doGC: aBool            = ( doGC := aBool )

  runBenchmark = (
    ('Starting ' + name + ' benchmark ... ') println.

    self doRuns: benchmarkSuite new.
    self reportBenchmark.

```

```
    '' println
  )

  measure: bench = (
    | startTime endTime runTime endGcStats startGcStats startCompTime endCompTime |
    startGcStats  := system gcStats.
    startCompTime := system totalCompilationTime.
    startTime     := system ticks.

    (bench innerBenchmarkLoop: innerIterations) ifFalse: [
      self error: 'Benchmark failed with incorrect result'. ].

    endTime     := system ticks.
    endGcStats  := system gcStats.
    endCompTime := system totalCompilationTime.

    runTime := endTime - startTime.
    self printResult: runTime startGc: startGcStats endGc: endGcStats compileTime: endCompTime - startCompTime.

    total := total + runTime.

    doGC ifTrue: [
      system fullGC ]
  )

  doRuns: bench = (
    1 to: numIterations do: [:i |
      self measure: bench
    ]
  )

  reportBenchmark = (
    (name + ': iterations=' + numIterations +
     ' average: ' + (total / numIterations) + 'us total: ' + total +
     'us\n') println.
  )

  printResult: runTime startGc: startGc endGc: endGc compileTime: compTime = (
    name print. ': GC count:     ' print. (((endGc) at: 1) - ((startGc) at: 1)) print. 'n' println.
    name print. ': GC time:      ' print. (((endGc) at: 2) - ((startGc) at: 2)) print. 'ms' println.
    name print. ': Allocated:    ' print. (((endGc) at: 3) - ((startGc) at: 3)) print. 'bytes' println.
    name print. ': Compile time: ' print. compTime print. 'ms' println.

    (name + ': iterations=1 runtime: ' + runTime + 'us') println
  )

  printTotal = (
    ('Total Runtime: ' + total + 'us') println.
  )

  ----

  new: name = (
    ^ self new initialize: name
  )
```

109)
110

Examples/AreWeFastYet/Sieve.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Sieve = Benchmark (

  benchmark = (
    | flags |
    flags  := Array new: 5000 withAll: true.
    ^ self sieve: flags size: 5000.
  )

  verifyResult: result = (
    ^ 669 = result
  )

  sieve: flags size: size = (
    | primeCount |
    primeCount := 0.

    2 to: size do: [ :i |
      (flags at: i - 1)
        ifTrue: [
          | k |
          primeCount := primeCount + 1.
          k := i + i.
          [ k <= size ]
            whileTrue: [
              flags at: k - 1 put: false.
              k := k + i ]. ] ].
    ^ primeCount
  )
)
```

Examples/AreWeFastYet/SomRandom.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
SomRandom = (
  | seed |

  initialize = ( seed := 74755 )

  next = (
    seed := ((seed * 1309) + 13849) & 65535.
    ^ seed
  )

  ----

  new = ( ^ super new initialize )
)
```

Examples/AreWeFastYet/Storage.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
Storage = Benchmark (

  | count |

  initialize = (
    count := 0.
  )

  benchmark = (
    | random |
    random := SomRandom new.
    count := 0.
    self buildTreeDepth: 7 with: random.
    ^ count
  )

  verifyResult: result = (
    ^ 5461 = result
  )

  buildTreeDepth: depth with: random = (
    count := count + 1.
    ^ (depth = 1)
        ifTrue:  [ Array new: random next % 10 + 1 ]
        ifFalse: [
          Array new: 4 withAll: [ self buildTreeDepth: depth - 1 with: random ] ]
  )

  ----

  new = ( ^ super new initialize )
)
```

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

Mmm... Hanoi...
"
Towers = Benchmark (

  | piles movesdone |

  initialize = (
    piles     := nil.
    movesdone := 0.
  )

  pushDisk: disk onPile: pile = (
    | top |

    top := piles at: pile.
    top notNil && [ disk size >= top size ]
      ifTrue: [ self error: 'Cannot put a big disk on a smaller one' ].

    disk next: top.
    piles at: pile put: disk.
  )

  popDiskFrom: pile = (
    | top |

    top := piles at: pile.
    top isNil
      ifTrue: [
        self error: 'Attempting to remove a disk from an empty pile' ].

    piles at: pile put: top next.
    top next: nil.
    ^ top
```

```
  )

  moveTopDiskFrom: fromPile to: toPile = (
    self pushDisk: (self popDiskFrom: fromPile) onPile: toPile.
    movesdone := movesdone + 1.
  )

  buildTowerAt: pile disks: disks = (
    disks downTo: 0 do: [ :i |
      self pushDisk: (TowersDisk new: i) onPile: pile ]
  )

  move: disks disksFrom: fromPile to: toPile = (
    disks = 1
      ifTrue: [ self moveTopDiskFrom: fromPile to: toPile ]
      ifFalse: [ | otherPile |
        otherPile := 6 - fromPile - toPile.
        self move: disks - 1 disksFrom: fromPile to: otherPile.
        self moveTopDiskFrom: fromPile to: toPile.
        self move: disks - 1 disksFrom: otherPile to: toPile. ]
  )

  benchmark = (
    piles := Array new: 3.
    self buildTowerAt: 1 disks: 13.
    movesdone := 0.
    self move: 13 disksFrom: 1 to: 2.
    ^ movesdone
  )

  verifyResult: result = (
    ^ 8191 = result
  )

  ----

  new = ( ^ super new initialize )
)
```

Examples/AreWeFastYet/TowersDisk.som

```
"
Copyright (c) 2001-2016 see AUTHORS.md file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
TowersDisk = (

  | size next |

  initialize: anInt = (
    size := anInt
  )

  size        = ( ^ size        )
  next        = ( ^ next        )
  next: value = ( next := value )

  ----------------------------------

  new: size  = ( ^ super new initialize: size )
)

```

Examples/Benchmarks/All.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

All = BenchmarkHarness (
    | summedAverage |

    all = (
        ^ Fibonacci, Dispatch, Bounce, Loop, Permute, Queens, List, Recurse,
          Storage, Sieve, BubbleSort, QuickSort, Sum, Towers, TreeSort,
          IntegerLoop, FieldLoop
    )

    run: params = (
        params length < 2
            ifTrue:  [ self exec: 100 ]
            ifFalse: [ self exec: (params at: 2) asInteger ]
    )

    printUsage = (
      './som.sh -cp Smalltalk Examples/Benchmarks/All.som [number-of-iterations]' println.
      '' println.
      '  number-of-iterations - the number of time each benchmark is executed, default: 1' println.
    )

    initialize = (
        super initialize.
        summedAverage := 0.
    )

    exec: iterations = (
        'Start execution of all benchmarks. Iterations: ' print.
        iterations println.
```

```

        self all do: [:cls |
            self initialize.
            self benchmarkClass: cls.
            self printAll: false.
            self maxRuntime: 3. "seconds"
            self numIterations: iterations.
            self warmUp: 10.

            self runBenchmark.
        ].
        self printTotal.
    )

    reportBenchmark: bench result: total = (
        '' println.
        'Benchmark: ' print.
        bench name println.

        ('   Iterations: ' + numIterations + ' (elapsed time ' + (total // 1000) round
            + ' ms)') println.
        ('   AVERAGE: ' + ((total // numIterations) // 1000) round + ' ms') println.

        summedAverage := summedAverage + (total // numIterations).
    )

    printTotal = (
        ('Summed Average Runtime: ' + (summedAverage // 1000) round asString + ' ms') println.
    )

)

```

Examples/Benchmarks/Ball.som

```
"

$Id: Ball.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Ball = (

    | x y xVel yVel |

    bounce = (
        | xLimit yLimit bounced |
        xLimit  := yLimit := 500.
        bounced := false.

        x := x + xVel.
        y := y + yVel.
        (x > xLimit)
            ifTrue: [ x := xLimit. xVel := 0 - xVel abs. bounced := true ].
        (x < 0)
            ifTrue: [ x := 0.      xVel := xVel abs.     bounced := true ].
        (y > yLimit)
            ifTrue: [ y := yLimit. yVel := 0 - yVel abs. bounced := true ].
        (y < 0)
            ifTrue: [ y := 0.      yVel := yVel abs.     bounced := true ].
        ^bounced
    )

    initialize = (
        x := Random next % 500.
        y := Random next % 500.
        xVel := (Random next % 300) - 150.
        yVel := (Random next % 300) - 150.
    )

```

```
    -----------------

    new = ( ^super new initialize )

)

```

Examples/Benchmarks/Benchmark.som

```
"


Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Benchmark = (

    run = (
        | harness |
        harness := All new.
        harness initialize.
        harness benchmarkClass: self class.
        harness printAll: false.
        harness numIterations: 100.

        harness runBenchmark.
        harness printTotal.
    )

    oneTimeSetup = ()

    innerBenchmarkLoop: innerIterations = (
        | i |
        i := 0.
        [ i < innerIterations ] whileTrue: [
            (self verifyResult: self benchmark) ifFalse: [ ^ false ].
            i := i + 1.
        ].
        ^ true
    )

    benchmark  = ( self subclassResponsibility )
    verifyResult: result = ( self subclassResponsibility )
    name       = ( ^self class name asString )

```

```
    assert: expected equals: value = (
      expected = value ifFalse: [
        self error: 'Expected value (' + expected asString + ') differs from actual (' + value asString + ') benchmark result.'
      ].
      ^ true
    )
)

```

Examples/Benchmarks/BenchmarkHarness.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

BenchmarkHarness = (

    "The BenchmarkHarness can be invoked on the command line and should be
    passed a list of benchmarks to run (class names) as arguments. It will
    run each of them and output single as well as a total."

    | total benchmarkClass numIterations innerIterations printAll doGC |

    benchmarkClass: class = ( benchmarkClass := class )
    printAll: aBool = ( printAll := aBool )

    numIterations: anInt = (numIterations := anInt)

    total = ( ^ total )

    run: args = (
        args length < 2 ifTrue: [ ^ self printUsage ].

        self initialize.
        self processArguments: args.

        self runBenchmark.
        self printTotal.
    )

    initialize = (
        total         := 0.
        numIterations := 1.
        innerIterations := 1.
        printAll      := true.
        doGC          := false.
```

```
    )

    printUsage = (
      './som.sh -cp Smalltalk Examples/Benchmarks/BenchmarkHarness.som [--gc-between-iterations|-gc] benchmark [num-iterations [inner-iter]]' println.
      '' println.
      '  --gc-between-iterations | --gc  - trigger a full GC between benchmark iteratons' println.
      '  benchmark      - benchmark class name (e.g., Queens, Fibonacci, Dispatch)' println.
      '  num-iterations - number of times to execute benchmark, default: 1' println.
      '  inner-iter     - number of times the benchmark is executed in an inner loop, ' println.
      '                   which is measured in total, default: 1' println.
    )

    processArguments: args = (
        | v arg |
        v := Vector new: args length.
        v appendAll: args.

        "First argument is the BenchmarkHarness"
        v removeFirst.

        arg := v removeFirst.
        (arg = '--gc-between-iterations' or: [arg = '--gc'])
          ifTrue: [
            doGC := true.
            arg := v removeFirst ].

        self loadBenchmarkClass: arg.
        v size > 0 ifTrue: [
          arg := v removeFirst.
          numIterations := arg asInteger.
          v size > 0 ifTrue: [
            arg := v removeFirst.
            innerIterations := arg asInteger ] ]
    )

    loadBenchmarkClass: className = (
        | sym cls |
        sym := className asSymbol.
        cls := system load: sym.
        cls ifNil: [
            self error: 'Failed loading benchmark: ' + className ].
        benchmarkClass := cls.
    )

    runBenchmark = (
        | bench result |
        bench := benchmarkClass new.
        bench oneTimeSetup.

        ('Starting ' + bench name + ' benchmark.') println.
        result := self doRuns: bench.
        total := total + result.
        self reportBenchmark: bench result: result.

        '' println
    )
```

```
    doRuns: bench = (
        | i total |
        i := 0.
        total := 0.

        [ i < numIterations ] whileTrue: [
            | startTime endTime runTime endGcStats startGcStats startCompTime endCompTime |
            startGcStats  := system gcStats.
            startCompTime := system totalCompilationTime.
            startTime     := system ticks.

            (bench innerBenchmarkLoop: innerIterations) ifFalse: [
              self error: 'Benchmark failed with incorrect result'. ].

            endTime     := system ticks.
            endGcStats  := system gcStats.
            endCompTime := system totalCompilationTime.

            runTime := endTime - startTime.
            printAll ifTrue: [
              self print: bench run: runTime startGc: startGcStats endGc: endGcStats compileTime: endCompTime - startCompTime ].

            total := total + runTime.
            i := i + 1.

            doGC ifTrue: [
              system fullGC ] ].

        ^ total
    )

    reportBenchmark: bench result: result = (
        bench name print.
        ': iterations=' print.
        numIterations print.
        ' average: ' print.
        (result / numIterations) print.
        'us' print.
        ' total: ' print.
        result print.
        'us' println.
    )

    print: bench run: runTime startGc: startGc endGc: endGc compileTime: compTime = (
        bench name print. ': GC count:     ' print. (((endGc) at: 1) - ((startGc) at: 1)) print. 'n' println.
        bench name print. ': GC time:      ' print. (((endGc) at: 2) - ((startGc) at: 2)) print. 'ms' println.
        bench name print. ': Allocated:    ' print. (((endGc) at: 3) - ((startGc) at: 3)) print. 'bytes' println.
        bench name print. ': Compile time: ' print. compTime print. 'ms' println.

        bench name print. ': iterations=1 runtime: ' print. runTime print. 'us' println
    )

```

```
    printTotal = (
        ('Total Runtime: ' + total asString + 'us') println.
    )
)

```

Examples/Benchmarks/Bounce.som

```
"

$Id: Bounce.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Bounce = Benchmark (

    benchmark = (
        | ballCount balls bounces |

        Random initialize.

        ballCount := 100.
        bounces   := 0.
        balls     := Array new: ballCount withAll: [ Ball new ].

        1 to: 50 do: [ :i |
            balls do: [ :ball |
                (ball bounce) ifTrue: [ bounces := bounces + 1 ] ] ].

        ^ bounces
    )

    verifyResult: result = (
      ^ self assert: 1331 equals: result
    )

)
```

Examples/Benchmarks/BubbleSort.som

```
"

$Id: BubbleSort.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

BubbleSort = Sort (

    sort: array = (
        array length downTo: 1 do: [ :i |
            1 to: i - 1 do: [ :j |
                | current next |
                current := array at: j.
                next    := array at: j + 1.
                (current > next)
                    ifTrue: [
                        array at: j put: next.
                        array at: j + 1 put: current ] ] ].
        ^ array
    )

    dataSize = ( ^130 )

)
```

Examples/Benchmarks/DeltaBlue/AbstractConstraint.som

```
AbstractConstraint = (
    "I am an abstract class representing a system-maintainable relationship (or
    'constraint') between a set of variables. I supply a strength instance
    variable; concrete subclasses provide a means of storing the constrained
    variables and other information required to represent a constraint.

    Instance variables:
        strength            the strength of this constraint <Strength>"
    | strength |

    "accessing"

    strength = (
        "Answer my strength."
        ^strength
    )

    strength: strengthSymbol = (
        "Set my strength."
        strength := Strength of: strengthSymbol.
    )

    "queries"
    isInput = (
        "Normal constraints are not input constraints. An input constraint is
         one that depends on external state, such as the mouse, the keyboard,
         a clock, or some arbitrary piece of imperative code."
        ^false
    )

    isSatisfied = (
        "Answer true if this constraint is satisfied in the current solution."
        self subclassResponsibility
    )

    "add/remove"
    addConstraint = (
        "Activate this constraint and attempt to satisfy it."

        self addToGraph.
        Planner current incrementalAdd: self.
    )

    addToGraph = (
        "Add myself to the constraint graph."
        self subclassResponsibility
    )

    destroyConstraint = (
        "Deactivate this constraint, remove it from the constraint graph,
         possibly causing other constraints to be satisfied, and destroy it."

```

```
        (self isSatisfied) ifTrue: [Planner current incrementalRemove: self].
        self removeFromGraph.
        "self release."
    )

    removeFromGraph = (
        "Remove myself from the constraint graph."
        self subclassResponsibility
    )

    "planning"
    chooseMethod: mark = (
        "Decide if I can be satisfied and record that decision. The output of
         the choosen method must not have the given mark and must have a
         walkabout strength less than that of this constraint."
        self subclassResponsibility
    )

    execute = (
        "Enforce this constraint. Assume that it is satisfied."
        self subclassResponsibility
    )

    inputsDo: aBlock = (
        "Assume that I am satisfied. Evaluate the given block on all my current
         input variables."
        self subclassResponsibility
    )

    inputsKnown: mark = (
        "Assume that I am satisfied. Answer true if all my current inputs are
         known. A variable is known if either a) it is 'stay' (i.e. it is a
         constant at plan execution time), b) it has the given mark (indicating
         that it has been computed by a constraint appearing earlier in the
         plan), or c) it is not determined by any constraint."

        self inputsDo:
            [: v |
             ((v mark = mark) or: [(v stay) or: [v determinedBy == nil]]) ifFalse:
                 [^false]].
        ^true
    )

    markUnsatisfied = (
        "Record the fact that I am unsatisfied."
        self subclassResponsibility
    )

    output = (
        "Answer my current output variable. Raise an error if I am not
         currently satisfied."
        self subclassResponsibility
    )
```

```
    recalculate = (
        "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
         the value for the current output of this constraint. Assume this
         constraint is satisfied."
        self subclassResponsibility
    )

    satisfy: mark = (
        "Attempt to find a way to enforce this (still unsatisfied) constraint.
         If successful, record the solution, perhaps modifying the current
         dataflow graph.  Answer the constraint that this constraint overrides,
         if there is one, or nil, if there isn't."

        | overridden |
        self chooseMethod: mark.
        (self isSatisfied)
            ifTrue:         "constraint can be satisfied"
                ["mark inputs to allow cycle detection in addPropagate"
                 | out |
                 self inputsDo: [: in | in mark: mark].
                 out := self output.
                 overridden := out determinedBy.
                 (overridden == nil) ifFalse: [overridden markUnsatisfied].
                 out determinedBy: self.
                 (Planner current addPropagate: self mark: mark) ifFalse:
                    [self notify:
                        ('Cycle encountered adding:\n   ',
                         self printString,
                         '\nConstraint removed.') withCRs.
                     ^nil].
                 out mark: mark]
            ifFalse:            "constraint cannot be satisfied"
                [overridden := nil.
                 (strength sameAs: (Strength required)) ifTrue:
                    [self notify: 'Failed to satisfy a required constraint']].
        ^overridden
    )

    "printing"
    longPrintOn: aStream = (

        | bindings |
        aStream nextPut: '('.
        self shortPrintOn: aStream.
        aStream space.
        aStream nextPutAll: strength printString.
        (self isSatisfied)
            ifTrue:
                [aStream cr. aStream space. aStream space. aStream space.
                 self inputsDo:
                    [: in | aStream nextPutAll: 'v', in asOop printString, ' '].
                aStream nextPutAll: '-> '.
                aStream nextPutAll: 'v', self output asOop printString]
            ifFalse:
```

```
                [aStream space. aStream nextPutAll: 'UNSATISFIED'].
        aStream nextPut: ')'.
        aStream cr.
    )

    printOn: aStream = (
        self shortPrintOn: aStream
    )

    shortPrintOn: aStream = (
        aStream nextPutAll: self class name, '(', self asOop printString, ')'.
    )

)
```

Examples/Benchmarks/DeltaBlue/BinaryConstraint.som

```
BinaryConstraint = AbstractConstraint (
    "I am an abstract superclass for constraints having two possible output
    variables.

    Instance variables:
        v1, v2      possible output variables <Variable>
        direction       one of:
                        #forward (v2 is output)
                        #backward ( v1 is output)
                        nil (not satisfied)"
    | v1 v2 direction |

    "initialize-release"

    var: variable1 var: variable2 strength: strengthSymbol = (
        "Initialize myself with the given variables and strength."

        strength := Strength of: strengthSymbol.
        v1 := variable1.
        v2 := variable2.
        direction := nil.
        self addConstraint.
    )

    "queries"

    isSatisfied = (
        "Answer true if this constraint is satisfied in the current solution."

        ^direction notNil
    )

    "add/remove"

    addToGraph = (
        "Add myself to the constraint graph."

        v1 addConstraint: self.
        v2 addConstraint: self.
        direction := nil
    )

    removeFromGraph = (
        "Remove myself from the constraint graph."

        (v1 == nil) ifFalse: [v1 removeConstraint: self].
        (v2 == nil) ifFalse: [v2 removeConstraint: self].
        direction := nil.
    )

    "planning"

    chooseMethod: mark = (
        "Decide if I can be satisfied and which way I should flow based on
         the relative strength of the variables I relate, and record that
         decision."
```

```

        (v1 mark == mark) ifTrue:       "forward or nothing"
            [((v2 mark ~= mark) and: [strength stronger: v2 walkStrength])
                ifTrue: [^direction := #forward]
                ifFalse: [^direction := nil]].

        (v2 mark == mark) ifTrue:       "backward or nothing"
            [((v1 mark ~= mark) and: [strength stronger: v1 walkStrength])
                ifTrue: [^direction := #backward]
                ifFalse: [^direction := nil]].

        "if we get here, neither variable is marked, so we have choice"
        (v1 walkStrength weaker: v2 walkStrength)
            ifTrue:
                [(strength stronger: v1 walkStrength)
                    ifTrue: [^direction := #backward]
                    ifFalse: [^direction := nil]]
            ifFalse:
                [(strength stronger: v2 walkStrength)
                    ifTrue: [^direction := #forward]
                    ifFalse: [^direction := nil]].
    )

    execute = (
        "Enforce this constraint. Assume that it is satisfied."

        self subclassResponsibility
    )

    inputsDo: aBlock = (
        "Evaluate the given block on my current input variable."

        (direction == #forward)
            ifTrue: [aBlock value: v1]
            ifFalse: [aBlock value: v2].
    )

    markUnsatisfied = (
        "Record the fact that I am unsatisfied."

        direction := nil.
    )

    output = (
        "Answer my current output variable."

        (direction == #forward)
            ifTrue: [^v2]
            ifFalse: [^v1]
    )

    recalculate = (
        "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
             the value for the current output of this constraint. Assume this
         constraint is satisfied."

        | in out |
        (direction == #forward)
            ifTrue: [in := v1. out := v2]
```

```
                ifFalse: [in := v2. out := v1].
            out walkStrength: (strength weakest: in walkStrength).
            out stay: (in stay).
            (out stay) ifTrue: [self execute].      "stay optimization"
        )
    )
```

```
DeltaBlue = Benchmark (

    oneTimeSetup = ( Strength initialize )

    innerBenchmarkLoop: innerIterations = (
        Planner chainTest: innerIterations.
        Planner projectionTest: innerIterations.
        ^ true
    )
)
```

```
EditConstraint = UnaryConstraint (
    "I am a unary input constraint used to mark a variable that the client
     wishes to change."

    "queries"

    isInput = (
        "I indicate that a variable is to be changed by imperative code."

        ^true
    )

    "execution"

    execute = (
        "Edit constraints do nothing."
    )

    ----

    "instance creation"

    var: aVariable strength: strengthSymbol = (
        "Install an edit constraint with the given strength on the given
         variable."

        ^(self new) var: aVariable strength: strengthSymbol
    )
)
```

```
EqualityConstraint = BinaryConstraint (
    "I constrain two variables to have the same value: `v1 = v2`."

    "execution"

    execute = (
        "Enforce this constraint. Assume that it is satisfied."

        (direction == #forward)
            ifTrue:  [v2 value: v1 value]
            ifFalse: [v1 value: v2 value].
    )

    ----

    "instance creation"

    var: variable1 var: variable2 strength: strengthSymbol = (
        "Install a constraint with the given strength equating the given
         variables."

        ^(self new) var: variable1 var: variable2 strength: strengthSymbol
    )
)
```

Examples/Benchmarks/DeltaBlue/OrderedCollection.som

```
OrderedCollection = Vector (
    add: elem = (
        ^ self append: elem
    )

    addLast: elem = (
        ^ self append: elem
    )

    remove: obj ifAbsent: aBlock = (
        (self remove: obj) ifFalse: aBlock
    )

    " private "
    insert: anObject before: spot = (
        "spot is an index in the range firstIndex .. lastIndex,
         such an index is not known from outside the collection.
         Never use this method in your code, it is meant for private use by
         OrderedCollection only.
         The methods for use are:
            #add:before:   to insert an object before another object
            #add:beforeIndex:   to insert an object before a given position. "
        | delta spotIndex |
        spotIndex := spot.
        delta := spotIndex - first.
        first = 1 ifTrue: [
            self makeRoomAtFirst.
            spotIndex := first + delta].
        first := first - 1.
        storage
            replaceFrom: first
            to: spotIndex - 2
            with: storage
            startingAt: first + 1.
        storage at: spotIndex - 1 put: anObject.
        ^ anObject
    )

    makeRoomAtFirst = (
        "Make some empty slots at the front of the array.
         If we have more than 50% free space, then just move the elements,
         so that the first 50% of the slots are free, otherwise add new free
         slots to the front by growing. Precondition: firstIndex = 1"

        | tally newFirstIndex newLastIndex |
        tally := self size.
        tally * 2 >= storage length ifTrue: [ ^self growAtFirst ].
        tally = 0 ifTrue: [ ^self resetTo: storage length + 1 ].
        newFirstIndex := storage length // 2 + 1.
        newLastIndex := newFirstIndex - first + last - 1.
        0 to: tally - 1 do: [ :offset |
            storage at: newLastIndex - offset put: (storage at: last - offset - 1) ].
        storage from: first to: newFirstIndex - 1 put: nil.
        first := newFirstIndex.
```

```
        last := newLastIndex + 1
    )

    resetTo: index = (
        first := index.
        last  := first
    )

    sort: i to: j with: sortBlock = (
        "Sort elements i through j of self to be nondescending according to
        sortBlock."

        | di dij dj tt ij k l n |
        sortBlock ifNil: [^self defaultSort: i to: j].
        "The prefix d means the data at that index."
        (n := j + 1  - i) <= 1 ifTrue: [^self]. "Nothing to sort."
         "Sort di,dj."
        di := storage at: i.
        dj := storage at: j.
        (sortBlock value: di with: dj) "i.e., should di precede dj?"
            ifFalse:
                [storage swap: i with: j.
                 tt := di.
                 di := dj.
                 dj := tt].
        n > 2
            ifTrue:  "More than two elements."
                [ij := (i + j) / 2.  "ij is the midpoint of i and j."
                 dij := storage at: ij.  "Sort di,dij,dj.  Make dij be their median."
                 (sortBlock value: di with: dij) "i.e. should di precede dij?"
                   ifTrue:
                    [(sortBlock value: dij with: dj) "i.e., should dij precede dj?"
                      ifFalse:
                        [storage swap: j with: ij.
                         dij := dj]]
                   ifFalse:  "i.e. di should come after dij"
                    [storage swap: i with: ij.
                     dij := di].
                n > 3
                  ifTrue:  "More than three elements."
                    ["Find k>i and l<j such that dk,dij,dl are in reverse order.
                    Swap k and l.  Repeat this procedure until k and l pass each other."
                     k := i.
                     l := j.
                     [[l := l - 1.  k <= l and: [sortBlock value: dij with: (storage at: l)]]
                       whileTrue.  "i.e. while dl succeeds dij"
                      [k := k + 1.  k <= l and: [sortBlock value: (storage at: k) with: dij]]
                       whileTrue.  "i.e. while dij succeeds dk"
                      k <= l]
                       whileTrue:
                        [storage swap: k with: l].
        "Now l<k (either 1 or 2 less), and di through dl are all less than or equal to dk
```

```
        through dj.  Sort those two segments."
                    self sort: i to: l with: sortBlock.
                    self sort: k to: j with: sortBlock]] )

    sort: aBlock = (
        "Make the argument, aBlock, be the criterion for ordering
elements of the
        receiver."

        "sortBlocks with side effects may not work right"
        self size > 0 ifTrue: [
          self sort: first
                 to: last - 1
               with: aBlock
        ]
    )



    ----

    with: anElement = (
        | col |
        col := self new: 10.
        col append: anElement.
        ^ col
    )
)
```

Examples/Benchmarks/DeltaBlue/Plan.som

```
Plan = OrderedCollection (
    "A Plan is an ordered list of constraints to be executed in sequence to
    resatisfy all currently satisfiable constraints in the face of one or more
    changing inputs."

    "execution"
    execute = (
        "Execute my constraints in order."

        self do: [: c | c execute]
    )
)
```

Examples/Benchmarks/DeltaBlue/Planner.som

```
Planner = (
    "This benchmark is an implementation of the DeltaBlue Constraint Solver
     described in `The DeltaBlue Algorithm: An Incremental Constraint
     Hierarchy Solver'', by Bjorn N. Freeman-Benson and John Maloney,
     Communications of the ACM, January 1990 (also as University of
     Washington TR 89-08-06).

     To run the benchmark, execute the expression `Planner standardBenchmark`."
    | currentMark |

    "initialize"

    initialize = (
        "Planner initialize"

        currentMark := 1
    )

    "add/remove"

    incrementalAdd: c = (
        "Attempt to satisfy the given constraint and, if successful,
         incrementally update the dataflow graph.

         Details: If satifying the constraint is successful, it may override a
         weaker constraint on its output. The algorithm attempts to resatisfy
         that constraint using some other method. This process is repeated
         until either a) it reaches a variable that was not previously
         determined by any constraint or b) it reaches a constraint that
         is too weak to be satisfied using any of its methods. The variables
         of constraints that have been processed are marked with a unique mark
         value so that we know where we've been. This allows the algorithm to
         avoid getting into an infinite loop even if the constraint graph has
         an inadvertent cycle."

        | mark overridden |
        mark := self newMark.
        overridden := c satisfy: mark.
        [overridden == nil] whileFalse:
            [overridden := overridden satisfy: mark]
    )

    incrementalRemove: c = (
        "Entry point for retracting a constraint. Remove the given constraint,
         which should be satisfied, and incrementally update the dataflow
         graph.

         Details: Retracting the given constraint may allow some currently
         unsatisfiable downstream constraint be satisfied. We thus
```

```
collect a
         list of unsatisfied downstream constraints and attempt to satisfy
         each one in turn. This list is sorted by constraint strength,
         strongest first, as a heuristic for avoiding unnecessarily adding
         and then overriding weak constraints."

        | out unsatisfied |
        out := c output.
        c markUnsatisfied.
        c removeFromGraph.
        unsatisfied := self removePropagateFrom: out.
        unsatisfied do: [: u | self incrementalAdd: u]
    )

    "planning/value propagation"

    extractPlanFromConstraints: constraints = (
        "Extract a plan for resatisfaction starting from the outputs of the
         given constraints, usually a set of input constraints."

        | sources |
        sources := OrderedCollection new.
        constraints do:
            [: c | ((c isInput) and: [c isSatisfied]) ifTrue: [sources add: c]].
        ^self makePlan: sources
    )

    extractPlanFromVariables: variables = (
        "Extract a plan from the dataflow graph having the given variables. It
         is assumed that the given set of variables is complete, or at least
         that it contains all the input variables."

        | sources |
        sources := OrderedCollection new.
        variables do:
            [: v |
             (v constraints) do:
                [: c | ((c isInput) and: [c isSatisfied]) ifTrue: [sources add: c]]].
        ^self makePlan: sources
    )

    makePlan: sources = (
        "Extract a plan for resatisfaction starting from the given satisfied
         source constraints, usually a set of input constraints. This method
         assumes that stay optimization is desired; the plan will contain only
         constraints whose output variables are not stay. Constraints that do
         no computation, such as stay and edit constraints, are not included
         in the plan.

         Details: The outputs of a constraint are marked when it is added to
```

```
         the plan under construction. A constraint may be appended to the plan
         when all its input variables are known. A variable is known if either
         a) the variable is marked (indicating that has been computed by a
         constraint appearing earlier in the plan), b) the variable is 'stay'
         (i.e. it is a constant at plan execution time), or c) the variable
         is not determined by any constraint. The last provision is for past
         states of history variables, which are not stay but which are also
         not computed by any constraint."

        | mark plan todo c |
        mark := self newMark.
        plan := Plan new.
        todo := sources.
        [todo isEmpty] whileFalse:
            [c := todo removeFirst.
             ((c output mark ~= mark) and:      "not in plan already and..."
              [c inputsKnown: mark]) ifTrue:    "eligible for inclusion"
                [plan addLast: c.
                 c output mark: mark.
                 self addConstraintsConsuming: c output to: todo]].
        ^plan
    )

    propagateFrom: v = (
        "The given variable has changed. Propagate new values downstream."

        | todo c |
        todo := OrderedCollection new.
        self addConstraintsConsuming: v to: todo.
        [todo isEmpty] whileFalse:
            [c := todo removeFirst.
             c execute.
             self addConstraintsConsuming: c output to: todo].
    )

    "private"

    addConstraintsConsuming: v to: aCollection = (
        | determiningC |
        determiningC := v determinedBy.
        v constraints do:
            [: c |
             ((c == determiningC) or: [c isSatisfied not]) ifFalse:
                [aCollection add: c]].
    )

    addPropagate: c mark: mark = (
        "Recompute the walkabout strengths and stay flags of all variables
         downstream of the given constraint and recompute the actual values
         of all variables whose stay flag is true. If a cycle is detected,
         remove the given constraint and answer false. Otherwise, answer true.

```

```
         Details: Cycles are detected when a marked variable is encountered
         downstream of the given constraint. The sender is assumed to have
         marked the inputs of the given constraint with the given mark. Thus,
         encountering a marked node downstream of the output constraint means
         that there is a path from the constraint's output to one of its
         inputs."

        | todo d |
        todo := OrderedCollection with: c.
        [todo isEmpty] whileFalse:
            [d := todo removeFirst.
             (d output mark = mark) ifTrue:
                [self incrementalRemove: c.
                 ^false].
             d recalculate.
             self addConstraintsConsuming: d output to: todo].
        ^true
    )

    changeVar: aVariable newValue: newValue = (
        | editConstraint plan |
        editConstraint := EditConstraint var: aVariable strength: #preferred.
        plan := self extractPlanFromConstraints: (Array with: editConstraint).
        10 timesRepeat: [
            aVariable value: newValue.
            plan execute].
        editConstraint destroyConstraint.
    )

    constraintsConsuming: v do: aBlock = (

        | determiningC |
        determiningC := v determinedBy.
        v constraints do:
            [: c |
             ((c == determiningC) or: [c isSatisfied not]) ifFalse:
                [aBlock value: c]].
    )

    newMark = (
        "Select a previously unused mark value.

         Details: We just keep incrementing. If necessary, the counter will
         turn into a LargePositiveInteger. In that case, it will be a bit
         slower to compute the next mark but the algorithms will all behave
         correctly. We reserve the value '0' to mean 'unmarked'. Thus, this
         generator starts at '1' and will never produce '0' as a mark value."

        ^currentMark := currentMark + 1
    )

    removePropagateFrom: out = (
```

```
        "Update the walkabout strengths and stay flags of all variables
         downstream of the given constraint. Answer a collection of unsatisfied
         constraints sorted in order of decreasing strength."

        | unsatisfied todo v nextC  u2 |
        unsatisfied := OrderedCollection new.

        out determinedBy: nil.
        out walkStrength: Strength absoluteWeakest.
        out stay: true.
        todo := OrderedCollection with: out.
        [todo isEmpty] whileFalse: [
            v := todo removeFirst.
             v constraints do: [:c |
                 (c isSatisfied) ifFalse: [unsatisfied add: c]].
             self constraintsConsuming: v do:
                [:c |
                 c recalculate.
                 todo add: c output]].

        unsatisfied sort: [:c1 :c2 | c1 strength stronger: c2 strength].
        ^unsatisfied
    )

    run = (
        Planner standardBenchmark
    )

    ----

    | currentPlanner |

    "instance creation"
    new = (
        ^currentPlanner := super new initialize
    )

    "benchmarks"
    chainTest: n = (
        "Do chain-of-equality-constraints performance tests."
        | vars editConstraint plan planner |

        planner := Planner new.
        vars := Array new: n+1 withAll: [ Variable new ].

        "thread a chain of equality constraints through the variables"
        1 to: n do:
            [ :i | | v1 v2 |
             v1 := vars at: i.
             v2 := vars at: i + 1.
             EqualityConstraint var: v1 var: v2 strength: #required].

        StayConstraint var: vars last strength: #strongDefault.
        editConstraint := EditConstraint var: (vars first) strength: #preferred.
        plan := planner extractPlanFromConstraints: (Array with: editConstraint).
        1 to: 100 do: [ :v |
            vars first value: v.
            plan execute.
```

```
            vars last value ~= v ifTrue: [self error: 'Chain test failed!!']].
        editConstraint destroyConstraint
    )

    projectionTest: n = (
        "This test constructs a two sets of variables related to each other by
         a simple linear transformation (scale and offset)."

        | scale offset src dst planner dests |
        planner := Planner new.
        dests := OrderedCollection new.
        scale := Variable value: 10.
        offset := Variable value: 1000.
        1 to: n do:
            [ :i |
            src := Variable value: i.
            dst := Variable value: i.
            dests add: dst.
            StayConstraint var: src strength: #default.
            ScaleConstraint var: src var: scale var: offset var: dst strength: #required].

        planner changeVar: src newValue: 17.
        dst value ~= 1170 ifTrue: [self error: 'Projection test 1 failed!!'].

        planner changeVar: dst newValue: 1050.
        src value ~= 5 ifTrue: [self error: 'Projection test 2 failed!!'].

        planner changeVar: scale newValue: 5.
        1 to: n - 1 do: [ :i |
            (dests at: i) value ~= (i*5 + 1000)
                ifTrue: [self error: 'Projection test 3 failed!!']].

        planner changeVar: offset newValue: 2000.
        1 to: n - 1 do: [ :i |
            (dests at: i) value ~= (i*5 + 2000)
                ifTrue: [self error: 'Projection test 4 failed!!']]
    )

    report: string times: count run: aBlock = (
        "Report the time required to execute the given block."

        | startTime endTime |
        startTime := system ticks.
        count timesRepeat: aBlock.
        endTime := system ticks.

        (string + ' ' + ((endTime - startTime) / count / 1000) asString + ' milliseconds') println.
    )

    standardBenchmark = (
        "This the combined benchmark."
        "Planner standardBenchmark"

        Strength initialize.

        self report: 'Chain and projection tests' times: 100 run: [
```

```
            self chainTest: 100.
            self projectionTest: 100
        ]
    )

    "accessing"
    current = (
        ^currentPlanner
    )
)
```

Examples/Benchmarks/DeltaBlue/ScaleConstraint.som

```
ScaleConstraint = BinaryConstraint (
    "I relate two variables by the linear scaling relationship:
    `v2 = (v1 * scale) + offset`. Either v1 or v2 may be changed to maintain
    this relationship but the scale factor and offset are considered read-only.

    Instance variables:
        scale       scale factor input variable <Variable>
        offset      offset input variable <Variable>"
    | scale offset |

    "initialize-release"

    src: srcVar scale: scaleVar offset: offsetVar dst: dstVar strength: strengthSymbol = (
        "Initialize myself with the given variables and strength."

        strength := Strength of: strengthSymbol.
        v1 := srcVar.
        v2 := dstVar.
        scale := scaleVar.
        offset := offsetVar.
        direction := nil.
        self addConstraint.
    )

    "add/remove"

    addToGraph = (
        "Add myself to the constraint graph."

        v1 addConstraint: self.
        v2 addConstraint: self.
        scale addConstraint: self.
        offset addConstraint: self.
        direction := nil.
    )

    removeFromGraph = (
        "Remove myself from the constraint graph."

        (v1 == nil) ifFalse: [v1 removeConstraint: self].
        (v2 == nil) ifFalse: [v2 removeConstraint: self].
        (scale == nil) ifFalse: [scale removeConstraint: self].
        (offset == nil) ifFalse: [offset removeConstraint: self].
        direction := nil.
    )

    "planning"

    execute = (
        "Enforce this constraint. Assume that it is satisfied."

        (direction == #forward)
            ifTrue:  [v2 value: (v1 value * scale value) + offset value]
            ifFalse: [v1 value: (v2 value - offset value) / scale value].
    )
```

```

    inputsDo: aBlock = (
        "Evaluate the given block on my current input variable."

        (direction == #forward)
            ifTrue: [aBlock value: v1.
                     aBlock value: scale.
                     aBlock value: offset]
            ifFalse: [aBlock value: v2.
                      aBlock value: scale.
                      aBlock value: offset].
    )

    recalculate = (
        "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
         the value for the current output of this constraint. Assume this
         constraint is satisfied."

        | in out |
        (direction == #forward)
            ifTrue:  [in  := v1. out := v2]
            ifFalse: [out := v1. in  := v2].
        out walkStrength: (strength weakest: in walkStrength).
        out stay: ((in stay) and: [(scale stay) and: [offset stay]]).
        (out stay) ifTrue: [self execute].      "stay optimization"
    )

    ----

    "instance creation"

    var: src var: scale var: offset var: dst strength: strengthSymbol = (
        "Install a scale constraint with the given strength on the given
         variables."

        ^(self new) src: src scale: scale offset: offset dst: dst strength: strengthSymbol
    )
)
```

Examples/Benchmarks/DeltaBlue/SortedCollection.som

```
SortedCollection = OrderedCollection (

    | sortBlock |

    indexForInserting: newObject = (
        | index low high |
        low  := first.
        high := last - 1.

        sortBlock isNil
            ifTrue: [[index := high + low / 2.  low > high]
                whileFalse:
                    [((storage at: index) <= newObject)
                        ifTrue: [low := index + 1]
                        ifFalse: [high := index - 1]]]
            ifFalse: [[index := high + low / 2.  low > high]
                whileFalse:
                    [(sortBlock value: (storage at: index) with: newObject)
                        ifTrue: [low := index + 1]
                        ifFalse: [high := index - 1]]].
        ^low )

    sort: i to: j = (
        "Sort elements i through j of self to be nondescending according to
        sortBlock."

        | di dij dj tt ij k l n |
        sortBlock ifNil: [^self defaultSort: i to: j].
        "The prefix d means the data at that index."
        (n := j + 1  - i) <= 1 ifTrue: [^self]. "Nothing to sort."
         "Sort di,dj."
        di := storage at: i.
        dj := storage at: j.
        (sortBlock value: di with: dj) "i.e., should di precede dj?"
            ifFalse:
                [storage swap: i with: j.
                 tt := di.
                 di := dj.
                 dj := tt].
        n > 2
            ifTrue:  "More than two elements."
                [ij := (i + j) / 2.  "ij is the midpoint of i and j."
                 dij := storage at: ij.  "Sort di,dij,dj.  Make dij be their median."
                 (sortBlock value: di with: dij) "i.e. should di precede dij?"
                   ifTrue:
                    [(sortBlock value: dij with: dj) "i.e., should dij precede dj?"
                      ifFalse:
                        [storage swap: j with: ij.
                         dij := dj]]
                   ifFalse:  "i.e. di should come after dij"
                    [storage swap: i with: ij.
                     dij := di].
                n > 3
```

```
                    ifTrue:  "More than three elements."
                      ["Find k>i and l<j such that dk,dij,dl are in reverse
order.
                      Swap k and l.  Repeat this procedure until k and l
pass each other."
                       k := i.
                       l := j.
                       [[l := l - 1.  k <= l and: [sortBlock value: dij
with: (storage at: l)]]
                         whileTrue.  "i.e. while dl succeeds dij"
                        [k := k + 1.  k <= l and: [sortBlock value:
(storage at: k) with: dij]]
                         whileTrue.  "i.e. while dij succeeds dk"
                        k <= l]
                         whileTrue:
                          [storage swap: k with: l].
          "Now l<k (either 1 or 2 less), and di through dl are all less
than or equal to dk
          through dj.  Sort those two segments."
                      self sort: i to: l.
                      self sort: k to: j]] )

      addAll: aCollection = (
          aCollection size > (self size / 3)
              ifTrue:
                  [aCollection do: [:each | self addLast: each].
                  self reSort]
              ifFalse: [aCollection do: [:each | self add: each]].
          ^ aCollection )

      reSort = (
          self sort: first
                 to: last - 1
      )

      sortBlock: aBlock = (
          "Make the argument, aBlock, be the criterion for ordering
elements of the
          receiver."

          sortBlock := aBlock.
          "sortBlocks with side effects may not work right"
          self size > 0 ifTrue: [self reSort] )

      copyEmpty = (
          "Answer a copy of the receiver without any of the receiver's
elements."

          ^self species sortBlock: sortBlock )

      addFirst: newObject = (
          self shouldNotImplement )

      insert: anObject before: spot = (
          self shouldNotImplement )

      defaultSort: i to: j = (
          "Sort elements i through j of self to be nondescending according
to
          sortBlock." "Assume the default sort block ([:x :y | x <= y])."
```

```
        | di dij dj tt ij k l n |
        "The prefix d means the data at that index."
        (n := j + 1  - i) <= 1 ifTrue: [^self]. "Nothing to sort."
         "Sort di,dj."
        di := storage at: i.
        dj := storage at: j.
        (di <= dj) "i.e., should di precede dj?"
            ifFalse:
                [storage swap: i with: j.
                 tt := di.
                 di := dj.
                 dj := tt].
        n > 2
            ifTrue:  "More than two elements."
                [ij := (i + j) / 2.  "ij is the midpoint of i and j."
                 dij := storage at: ij.  "Sort di,dij,dj.  Make dij be their median."
                 (di <= dij) "i.e. should di precede dij?"
                   ifTrue:
                    [(dij <= dj) "i.e., should dij precede dj?"
                      ifFalse:
                        [storage swap: j with: ij.
                         dij := dj]]
                   ifFalse:  "i.e. di should come after dij"
                    [storage swap: i with: ij.
                     dij := di].
                n > 3
                  ifTrue:  "More than three elements."
                    ["Find k>i and l<j such that dk,dij,dl are in reverse order.
                    Swap k and l.  Repeat this procedure until k and l pass each other."
                     k := i.
                     l := j.
                     [[l := l - 1.  k <= l and: [dij <= (storage at: l)]]
                       whileTrue.  "i.e. while dl succeeds dij"
                      [k := k + 1.  k <= l and: [(storage at: k) <= dij]]
                       whileTrue.  "i.e. while dij succeeds dk"
                      k <= l]
                       whileTrue:
                        [storage swap: k with: l].
        "Now l<k (either 1 or 2 less), and di through dl are all less than or equal to dk
        through dj.  Sort those two segments."
                    self defaultSort: i to: l.
                    self defaultSort: k to: j]] )

    should: a precede: b = (
        ^ sortBlock ifNil: [a <= b] ifNotNil: [sortBlock value: a with: b] )

    median = (
        "Return the middle element, or as close as we can get."

        ^ self at: self size + 1 / 2 )

    at: anInteger put: anObject = (
        self shouldNotImplement )

    add: newObject = (
        ^ super insert: newObject before: (self indexForInserting:
```

```
newObject) )

    = aSortedCollection = (
        "Answer true if my and aSortedCollection's species are the same,
        and if our blocks are the same, and if our elements are the same."

        self species = aSortedCollection species ifFalse: [^ false].
        sortBlock = aSortedCollection sortBlock
            ifTrue: [^ super = aSortedCollection]
            ifFalse: [^ false] )

    collect: aBlock = (
        "Evaluate aBlock with each of my elements as the argument. Collect the
        resulting values into an OrderedCollection. Answer the new collection.
        Override the superclass in order to produce an OrderedCollection instead
        of a SortedCollection."

        | newCollection |
        newCollection := OrderedCollection new: self size.
        self do: [:each | newCollection addLast: (aBlock value: each)].
        ^ newCollection )

    sort: aSortBlock = (
        "Sort this storage using aSortBlock. The block should take two arguments
        and return true if the first element should preceed the second one."

        super sort: aSortBlock.
        sortBlock := aSortBlock )

    join: aCollection = (
        | result |
        "Curiously addAllLast: does not trigger a reSort, so we must do it here."
        result := super join: aCollection.
        result reSort.
        ^ result
    )

    sortTopologically = (
        "Plenty of room for increased efficiency in this one."

        | remaining result pick |
        remaining := self asOrderedCollection.
        result := OrderedCollection new.
        [remaining isEmpty] whileFalse: [
            pick := remaining select: [:item |
                remaining allSatisfy: [:anotherItem |
                    item == anotherItem or: [self should: item precede: anotherItem]]].
            pick isEmpty ifTrue: [self error: 'bad topological ordering'].
            result addAll: pick.
            remaining removeAll: pick].
        ^self copySameFrom: result )

    sortBlock = (
        "Answer the blockContext which is the criterion for sorting
```

```
elements of
        the receiver."

        ^sortBlock )

    ----

    sortBlock: aBlock = (
        "Answer an instance of me such that its elements are sorted according to
        the criterion specified in aBlock."

        ^(super new: 10) sortBlock: aBlock )

)

```

Examples/Benchmarks/DeltaBlue/StayConstraint.som

```
StayConstraint = UnaryConstraint (
    "I mark variables that should, with some level of preference, stay the same.
     I have one method with zero inputs and one output, which does nothing.
     Planners may exploit the fact that, if I am satisfied, my output will not
     change during plan execution. This is called 'stay optimization.'"


    "execution"

    execute = (
        "Stay constraints do nothing."
    )

    ----

    "instance creation"

    var: aVariable strength: strengthSymbol = (
        "Install a stay constraint with the given strength on the given variable."

        ^(self new) var: aVariable strength: strengthSymbol
    )
)
```

Examples/Benchmarks/DeltaBlue/Strength.som

```
Strength = (
    "Strengths are used to measure the relative importance of constraints. The
     hierarchy of available strengths is determined by the class variable
     StrengthTable (see my class initialization method). Because Strengths are
     invariant, references to Strength instances are shared (i.e. all references
     to `Strength of: #required` point to a single, shared instance). New
     strengths may be inserted in the strength hierarchy without disrupting
     current constraints.

     Instance variables:
         symbolicValue      symbolic strength name (e.g. #required) <Symbol>
         arithmeticValue    index of the constraint in the hierarchy, used for comparisons <Number>"
    | symbolicValue arithmeticValue |

    "comparing"

    sameAs: aStrength = (
        "Answer true if I am the same strength as the given Strength."

        ^arithmeticValue = aStrength arithmeticValue
    )

    stronger: aStrength = (
        "Answer true if I am stronger than the given Strength."
        ^arithmeticValue < aStrength arithmeticValue
    )

    weaker: aStrength = (
        "Answer true if I am weaker than the given Strength."
        ^arithmeticValue > aStrength arithmeticValue
    )

    "max/min"

    strongest: aStrength = (
        "Answer the stronger of myself and aStrength."

        (aStrength stronger: self)
            ifTrue: [^aStrength]
            ifFalse: [^self].
    )

    weakest: aStrength = (
        "Answer the weaker of myself and aStrength."

        (aStrength weaker: self)
            ifTrue: [^aStrength]
            ifFalse: [^self].
    )

    "printing"

```

```
    printOn: aStream = (
        "Append a string which represents my strength onto aStream."

        aStream nextPutAll: '%', symbolicValue, '%'.
    )

    "private"

    arithmeticValue = (
        "Answer my arithmetic value. Used for comparisons. Note that
         STRONGER constraints have SMALLER arithmetic values."

        ^arithmeticValue
    )

    initializeWith: symVal = (
        "Record my symbolic value and reset my arithmetic value."

        symbolicValue := symVal.
        arithmeticValue := Strength strengthTable at: symbolicValue.
    )

    ----
    | AbsoluteStrongest AbsoluteWeakest Required StrengthConstants StrengthTable |

    strengthTable = (
        ^ StrengthTable
    )

    "class initialization"

    initialize = (
        "Initialize the symbolic strength table. Fix the internally caches
         values of all existing instances."
        "Strength initialize"

        StrengthTable := Dictionary new.
        StrengthTable at: #absoluteStrongest put: -10000.
        StrengthTable at: #required put: -800.
        StrengthTable at: #strongPreferred put: -600.
        StrengthTable at: #preferred put: -400.
        StrengthTable at: #strongDefault put: -200.
        StrengthTable at: #default put: 0.
        StrengthTable at: #weakDefault put: 500.
        StrengthTable at: #absoluteWeakest put: 10000.

        StrengthConstants := Dictionary new.
        StrengthTable keys do:
            [: strengthSymbol |
                StrengthConstants
                    at: strengthSymbol
                    put: ((super new) initializeWith: strengthSymbol)].

        AbsoluteStrongest := Strength of: #absoluteStrongest.
        AbsoluteWeakest := Strength of: #absoluteWeakest.
        Required := Strength of: #required.
    )

    "instance creation"
    of: aSymbol = (
```

```
        "Answer an instance with the specified strength."
        ^StrengthConstants at: aSymbol
    )

    "constants"
    absoluteStrongest = (
        ^AbsoluteStrongest
    )

    absoluteWeakest = (
        ^AbsoluteWeakest
    )

    required = (
        ^Required
    )
)
```

Examples/Benchmarks/DeltaBlue/UnaryConstraint.som

```
UnaryConstraint = AbstractConstraint (
    "I am an abstract superclass for constraints having a single possible output
    variable.

    Instance variables:
        output      possible output variable <Variable>
        satisfied       true if I am currently satisfied <Boolean>"
    | output satisfied |

    "initialize-release"

    var: aVariable strength: strengthSymbol = (
        "Initialize myself with the given variable and strength."

        strength := Strength of: strengthSymbol.
        output := aVariable.
        satisfied := false.
        self addConstraint.
    )

    "queries"
    isSatisfied = (
        "Answer true if this constraint is satisfied in the current solution."

        ^satisfied
    )

    "add/remove"

    addToGraph = (
        "Add myself to the constraint graph."

        output addConstraint: self.
        satisfied := false.
    )

    removeFromGraph = (
        "Remove myself from the constraint graph."

        (output == nil) ifFalse: [output removeConstraint: self].
        satisfied := false.
    )

    "planning"

    chooseMethod: mark = (
        "Decide if I can be satisfied and record that decision."

        satisfied :=
            (output mark ~= mark) and:
            [strength stronger: output walkStrength].
    )

    execute = (
        "Enforce this constraint. Assume that it is satisfied."
        self subclassResponsibility
```

```
    )

    inputsDo: aBlock = (
        "I have no input variables."
    )

    markUnsatisfied = (
        "Record the fact that I am unsatisfied."
        satisfied := false.
    )

    output = (
        "Answer my current output variable."
        ^ output
    )

    recalculate = (
        "Calculate the walkabout strength, the stay flag, and, if it is 'stay',
         the value for the current output of this constraint. Assume this
         constraint is satisfied."

        output walkStrength: strength.
        output stay: (self isInput not).
        (output stay) ifTrue: [self execute].   "stay optimization"
    )
)
```

Examples/Benchmarks/DeltaBlue/Variable.som

```
Variable = (
    "I represent a constrained variable. In addition to my value, I maintain the
    structure of the constraint graph, the current dataflow graph, and various
    parameters of interest to the DeltaBlue incremental constraint solver.

    Instance variables:
        value           my value; changed by constraints, read by client <Object>
        constraints     normal constraints that reference me <Array of Constraint>
        determinedBy    the constraint that currently determines
                        my value (or nil if there isn't one) <Constraint>
        walkStrength        my walkabout strength <Strength>
        stay            true if I am a planning-time constant <Boolean>
        mark            used by the planner to mark constraints <Number>"
    | value constraints determinedBy walkStrength stay mark |

    "initialize-release"

    initialize = (
        value := 0.
        constraints  := OrderedCollection new: 2.
        determinedBy := nil.
        walkStrength := Strength absoluteWeakest.
        stay := true.
        mark := 0.
    )

    "access"
    addConstraint: aConstraint = (
        "Add the given constraint to the set of all constraints that refer
         to me."

        constraints add: aConstraint.
    )

    constraints = (
        "Answer the set of constraints that refer to me."

        ^constraints
    )

    determinedBy = (
        "Answer the constraint that determines my value in the current
         dataflow."

        ^determinedBy
    )

    determinedBy: aConstraint = (
        "Record that the given constraint determines my value in the current
         data flow."

        determinedBy := aConstraint.
    )
```

```
    mark = (
        "Answer my mark value."

        ^mark
    )

    mark: markValue = (
        "Set my mark value."

        mark := markValue.
    )

    removeConstraint: c = (
        "Remove all traces of c from this variable."

        constraints remove: c ifAbsent: [].
        (determinedBy == c) ifTrue: [determinedBy := nil].
    )

    stay = (
        "Answer my stay flag."

        ^stay
    )

    stay: aBoolean = (
        "Set my stay flag."

        stay := aBoolean
    )

    value = (
        "Answer my value."

        ^value
    )

    value: anObject = (
        "Set my value."

        value := anObject.
    )

    walkStrength = (
        "Answer my walkabout strength in the current dataflow."

        ^walkStrength
    )

    walkStrength: aStrength = (
        "Set my walkabout strength in the current dataflow."

        walkStrength := aStrength.
    )

    "printing"
    longPrintOn: aStream  = (

        self shortPrintOn: aStream.
        aStream nextPutAll: '   Constraints: '.
```

```
        (constraints isEmpty)
            ifTrue: [aStream cr. aStream tab. aStream nextPutAll: 'none']
            ifFalse:
                [constraints do:
                    [: c | aStream cr. aStream tab. c shortPrintOn: aStream]].
        (determinedBy isNil) ifFalse:
            [aStream cr. aStream nextPutAll: '   Determined by: '.
             aStream cr. aStream tab. determinedBy shortPrintOn: aStream].
        aStream cr.
    )

    printOn: aStream = (
        self shortPrintOn: aStream
    )

    shortPrintOn: aStream = (

        aStream nextPutAll: 'V(', self asOop printString, ', '.
        aStream nextPutAll: walkStrength printString, ', '.
        (stay isNil) ifFalse:
            [aStream nextPutAll: (stay ifTrue: ['stay, '] ifFalse: ['changing, '])].
        aStream nextPutAll: value printString.
        aStream nextPutAll: ')'.
        aStream cr.
    )

    ----

    "instance creation"

    new = (
        ^super new initialize
    )

    value: aValue = (
        | o |
        o := super new.
        o initialize.
        ^ o value: aValue
    )
)
```

Examples/Benchmarks/Fannkuch.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"Pfannkuchen are somehow known in Benchmarking. i.e., there is a Lisp story"

" Original written for VisualWorks
* The Computer Language Benchmarks Game
   http://shootout.alioth.debian.org/
   contributed by Paolo Bonzini *"

Fannkuch = Benchmark (

    | timesRotated perm atEnd |

    pfannkuchen: anArray = (
        | first complement a b k |
        k := 0.
        [ (first := anArray at: 1) = 1 ] whileFalse: [
            k := k + 1.
            complement := first + 1.

            1 to: first / 2 do: [ :i |
                a := anArray at: i.
                b := anArray at: complement - i.
                anArray at: i put: b.
                anArray at: complement - i put: a.
            ]
        ].
        ^k
    )

    initialize: size = (
        perm := (1 to: size).
```

```
        timesRotated := Array new: size withAll: 0.
        atEnd := false
    )

    makeNext = (
        | temp remainder |
        "* Generate the next permutation. *"
        2 to: perm length do: [ :r |
            "* Rotate the first r items to the left. *"
            temp := perm at: 1.
            1 to: r - 1 do: [ :i | perm at: i put: (perm at: i + 1) ].
            perm at: r put: temp.

            timesRotated at: r put: ((timesRotated at: r) + 1) % r.
            remainder := timesRotated at: r.
            remainder = 0 ifFalse: [ ^self ].

            "* After r rotations, the first r items are in their original positions.
            Go on rotating the first r+1 items. *"
        ].

        "* We are past the final permutation. *"
        atEnd := true.
    )


    maxPfannkuchen = (
        | max permutation |
        max := 0.
        [self atEnd] whileFalse: [
            permutation := self next.
            max := max max: (self pfannkuchen: permutation)].
        ^max
    )

    atEnd = (
        ^atEnd
    )

    next = (
        | result |
        result := perm copy.
        self makeNext.
        ^result
    )

    innerBenchmarkLoop: innerIterations = (
        | result |
        self initialize: innerIterations.
        result := self maxPfannkuchen.
        ^ result = (Fannkuch expectedResult: innerIterations)
    )

    ----
    | results |

    expectedResult: problemSize = (
      results isNil ifTrue: [
        results := Array new: 12.
        results at:  1 put:  0.
```

```
        results at:  2 put:  1.
        results at:  3 put:  2.
        results at:  4 put:  4.
        results at:  5 put:  7.
        results at:  6 put: 10.
        results at:  7 put: 16.
        results at:  8 put: 22.
        results at:  9 put: 30.
        results at: 10 put: 38.
        results at: 11 put: 51.
        results at: 12 put: 65.
      ].
      ^ results at: problemSize
    )
)

```

```
" Ported GCBench from PyPy Project: https://bitbucket.org/pypy/pypy/src/02ea09544fc9/pypy/translator/goal/gcbench.py
  Original comment was:

# Ported from a Java benchmark whose history is :
#  This is adapted from a benchmark written by John Ellis and Pete Kovac
#  of Post Communications.
#  It was modified by Hans Boehm of Silicon Graphics.
#
#       This is no substitute for real applications.  No actual application
#       is likely to behave in exactly this way.  However, this benchmark was
#       designed to be more representative of real applications than other
#       Java GC benchmarks of which we are aware.
#       It attempts to model those properties of allocation requests that
#       are important to current GC techniques.
#       It is designed to be used either to obtain a single overall performance
#       number, or to give a more detailed estimate of how collector
#       performance varies with object lifetimes.  It prints the time
#       required to allocate and collect balanced binary trees of various
#       sizes.  Smaller trees result in shorter object lifetimes.  Each cycle
#       allocates roughly the same amount of memory.
#       Two data structures are kept around during the entire process, so
#       that the measured performance is representative of applications
#       that maintain some live in-memory data.  One of these is a tree
#       containing many pointers.  The other is a large array containing
#       double precision floating point numbers.  Both should be of comparable
#       size.
#
#       The results are only really meaningful together with a specification
#       of how much memory was used.  It is possible to trade memory for
#       better time performance.  This benchmark should be run in a 32 MB
#       heap, though we don't currently know how to enforce that uniformly.
#
#       Unlike the original Ellis and Kovac benchmark, we do not attempt
#       measure pause times.  This facility should eventually be added back
#       in.  There are several reasons for omitting it for now.  The original
#       implementation depended on assumptions about the thread scheduler
#       that don't hold uniformly.  The results really measure both the
#       scheduler and GC.  Pause time measurements tend to not fit well with
#       current benchmark suites.  As far as we know, none of the current
#       commercial Java implementations seriously attempt to minimize GC pause
#       times.
#
#       Known deficiencies:
#               - No way to check on memory use
#               - No cyclic data structures
#               - No attempt to measure variation with object size
```

```
#                - Results are sensitive to locking cost, but we dont
#                  check for proper locking
"
GCBench = (
    |kStretchTreeDepth kLongLivedTreeDepth kArraySize kMaxTreeDepth
    kMinTreeDepth cur_depth|

    run = (
        |temp_tree long_lived_tree array depths cur_depth t_start t_finish|
        kStretchTreeDepth := 18.
        kLongLivedTreeDepth := 16.
        kArraySize := 500000.
        kMaxTreeDepth := 16.
        kMinTreeDepth := 4.
        'Garbage Collector Test' println.
        ('Stretching memory with a binary tree of depth ' + kStretchTreeDepth) println.
        t_start := system time.
        temp_tree := self make_tree: kStretchTreeDepth.
        temp_tree := nil.

        ('Creating a long-lived binary tree of depth ' + kLongLivedTreeDepth) println.
        long_lived_tree := Node create.
        self populate: kLongLivedTreeDepth tree: long_lived_tree.

        ('Creating a long-lived array of ' + kArraySize + ' doubles') println.
        array := Array new: kArraySize withAll: [(0//1)].
        1 to: (kArraySize/2) do: [:value | array at: value put: (1 // value)].
        depths := Array new: ((kMaxTreeDepth - kMinTreeDepth)/ 2 + 1).
        cur_depth := kMinTreeDepth.
        depths doIndexes: [:value |
            depths at: value put: cur_depth.
            cur_depth := cur_depth + 2.
            ].
        self time_constructions: depths.

        ((long_lived_tree == nil))
            ifTrue: ["test failed" println.].
        t_finish := system time.
        ('Completed in ' + (t_finish - t_start) + ' ms.') println.
    )

    make_tree: depth = (
        depth <= 0
            ifTrue:
                [^(Node create)]
            ifFalse:
                [^(Node create: (self make_tree: (depth - 1)) with: (self
                make_tree: (depth - 1)))]
    )

    time_constructions: depths = (
        depths do: [:value | self time_construction: value].
    )

    time_construction: depth = ( |niters t_start t_finish temp_tree|
        niters := self num_iters: depth.
```

```
        ('Creating ' + niters + ' trees of depth ' + depth) println.
        t_start := system time.
        0 to: (niters-1) do: [:i |
            temp_tree := Node create.
            self populate: depth tree: temp_tree.
            temp_tree := nil.].
        t_finish := system time.
        ('    Top down constrution took ' + (t_finish - t_start) + ' ms.') println.
        t_start := system time.
        0 to: (niters-1) do: [:i |
            temp_tree := self make_tree: depth.
            temp_tree := nil.].
        t_finish := system time.
        ('    Bottom up constrution took ' + (t_finish - t_start) + ' ms.') println.

    )

    num_iters: i = (
        ^(2 * (self tree_size: kStretchTreeDepth) / (self tree_size: i)).
    )

    populate: depth tree: node = (
        depth <= 0
            ifFalse: [
                depth := depth - 1.
                node left: Node create.
                node right: Node create.
                self populate: depth tree: (node left).
                self populate: depth tree: (node right).
                ]
    )

    tree_size: i = ( |val|
        val := 2.
        i timesRepeat: [val := val * 2].
        ^(val - 1).
    )
)
```

```
Node = (
    |left right|

    left = (^left)
    left: val = (
        left := val.
    )
    right = (^right)
    right: val = (
        right := val.
    )
    ----
    create = (
        ^(Node create: nil with: nil).
    )
    create: l with: r = (
        |n|
        n := Node new.
        n left: l.
        n right: r.
        ^n.
    )
)
```

```
Edge = (
  | dest weight |
  dest   = ( ^ dest )
  weight = ( ^ weight )

  initializeWith: destination and: w = (
    dest   := destination.
    weight := w.
  )

  ----

  newWith: dest and: weight = (
    ^ self new initializeWith: dest and: weight
  )

)
```

Examples/Benchmarks/GraphSearch/GraphSearch.som

```
GraphSearch = Benchmark (
  | graphNodes graphMask updatingGraphMask graphVisited cost graphEdges k firstCost |

  initializeGraph: noOfNodes = (
    | source graph totalEdges |
    graphNodes        := Array new: noOfNodes.
    graphMask         := Array new: noOfNodes withAll: false.
    updatingGraphMask := Array new: noOfNodes withAll: false.
    graphVisited      := Array new: noOfNodes withAll: false.
    cost              := Array new: noOfNodes withAll: -1.

    source := 1.

    graph := Array new: noOfNodes withAll: [Vector new].

    graph doIndexes: [:i |
      | noOfEdges |
      noOfEdges := (JenkinsRandom random
                       rem: GraphSearch MaxInitEdges - GraphSearch MinEdges + 1) abs
                     + GraphSearch MinEdges.
      1 to: noOfEdges do: [:j |
        | nodeId weight |
        nodeId := (JenkinsRandom random rem: noOfNodes) abs + 1.
        weight := (JenkinsRandom random rem: (
                    GraphSearch MaxWeight - GraphSearch MinWeight + 1)) abs
                   + GraphSearch MinWeight.
        (graph at: i)      append: (Edge newWith: nodeId and: weight).
        (graph at: nodeId) append: (Edge newWith:      i and: weight).
      ]
    ].

    totalEdges := 0.
    graph doIndexes: [:i |
      | noOfEdges |
      noOfEdges := (graph at: i) size.
      graphNodes at: i put: (Node newWith: totalEdges + 1 and: noOfEdges).
      totalEdges := totalEdges + noOfEdges
    ].

    graphMask    at: source put: true.
    graphVisited at: source put: true.

    graphEdges := Array new: totalEdges withAll: 0.

    k := 1.
    graph do: [:i |
      i do: [:j |
        graphEdges at: k put: j dest.
        k := k + 1.
      ]
    ].
    cost at: source put: 0.
  )

  innerBenchmarkLoop: innerIterations = (
```

```
    | noOfNodes |
    JenkinsRandom seed: 49734321.

    noOfNodes := GraphSearch ExpectedNoOfNodes / 900000 * innerIterations * innerIterations * innerIterations.

    self initializeGraph: noOfNodes.
    self breadthFirstSearch: noOfNodes.
    ^ self verify: cost inner: innerIterations
  )

  breadthFirstSearch: noOfNodes = (
    | stop |
    stop := true.
    [stop] whileTrue: [
      stop := false.

      1 to: noOfNodes do: [:tid |
        (graphMask at: tid) ifTrue: [
          graphMask at: tid put: false.
          (graphNodes at: tid) starting
              to: ((graphNodes at: tid) noOfEdges + (graphNodes at: tid) starting) - 1
              do: [:i |
            | id |
            id := graphEdges at: i.
            (graphVisited at: id) ifFalse: [
              cost at: id put: (cost at: tid) + 1.
              updatingGraphMask at: id put: true.
            ]
          ]
        ]
      ].

      1 to: noOfNodes do: [:tid |
        (updatingGraphMask at: tid) ifTrue: [
          graphMask    at: tid put: true.
          graphVisited at: tid put: true.
          stop := true.
          updatingGraphMask at: tid put: false
        ]
      ]
    ]
  )

  verify: result inner: innerIterations = (
    | totalCost |
    cost length = (GraphSearch ExpectedNoOfNodes / 900000 * innerIterations * innerIterations * innerIterations)
      ifFalse: [ ^ false ].

    totalCost := 0.
    cost do: [:c | totalCost := totalCost + c].

    cost length = GraphSearch ExpectedNoOfNodes
      ifTrue:  [
        totalCost = GraphSearch ExpectedTotalCost ifFalse: [
          self error: 'ERROR: the total cost obtained for ' + cost length asString
              + ' nodes is ' + totalCost asString + ' while the expected cost is '
```

```
              + GraphSearch ExpectedTotalCost
        ]]
      ifFalse: [
        firstCost == nil
          ifTrue: [
            firstCost := totalCost.
            ^ true ]
          ifFalse: [
            ^ firstCost = totalCost
          ]
      ].

    ^ true
  )

  ----

  MinEdges = ( ^ 2 )
  MaxInitEdges = ( ^ 4 )
  MinWeight = ( ^ 1 )
  MaxWeight = ( ^ 1 )

  ExpectedNoOfNodes = ( ^ 3000000 )
  ExpectedTotalCost = ( ^ 26321966 )

)
```

```
Node = (
  | starting noOfEdges |

  starting  = ( ^ starting )
  noOfEdges = ( ^ noOfEdges )

  initializeWith: start and: edges = (
    starting  := start.
    noOfEdges := edges.
  )

  ----

  newWith: starting and: noOfEdges = (
    ^ self new initializeWith: starting and: noOfEdges
  )
)
```

Examples/Benchmarks/JenkinsRandom.som

```
JenkinsRandom = (
  ----
  | seed |

  seed: val = ( seed := val )

  "Robert Jenkins' 32 bit integer hash function."
  random = (
    seed := ((seed       + 2127912214 "0x7ed55d16")       + (seed as32BitUnsignedValue  << 12) as32BitSignedValue) as32BitSignedValue.
    seed := ((seed bitXor: 3345072700 "0xc761c23c") bitXor: (seed as32BitUnsignedValue >>> 19)) as32BitSignedValue.
    seed := ((seed       +  374761393 "0x165667B1")       + (seed as32BitUnsignedValue  <<  5) as32BitSignedValue) as32BitSignedValue.
    seed := ((seed       + 3550635116 "0xd3a2646c") bitXor: (seed as32BitUnsignedValue  <<  9) as32BitSignedValue) as32BitSignedValue.
    seed := ((seed       + 4251993797 "0xfd7046c5")       + (seed as32BitUnsignedValue  <<  3) as32BitSignedValue) as32BitSignedValue.
    seed := ((seed bitXor: 3042594569 "0xb55a4f09") bitXor: (seed as32BitUnsignedValue >>> 16)) as32BitSignedValue.
    ^ seed
  )
)
```

```
HashIndexTable = (
  | hashTable |

  initialize = (
    hashTable := Array new: 32 withAll: 0
  )

  at: name put: index = (
    | slot |
    slot := self hashSlotFor: name.

    index < 255
      ifTrue:  [ hashTable at: slot put: index + 1 ]
      ifFalse: [ hashTable at: slot put: 0 ]
  )

  remove: index = (
    hashTable doIndexes: [:i |
      index + 1 = (hashTable at: i)
        ifTrue:  [ hashTable at: i put: 0 ]
        ifFalse: [
          (hashTable at: i) > index + 1 ifTrue: [
            hashTable at: i put: (hashTable at: i) - 1]]]
  )

  at: name = (
    | slot |
    slot := self hashSlotFor: name.

    " subtract 1, 0 stands for empty "
    ^ ((hashTable at: slot) & 255) - 1
  )

  hashSlotFor: element = (
    ^ (element hashcode & 31) + 1
  )

  ----

  new = ( ^ super new initialize )
)
```

Examples/Benchmarks/Json/Json.som

```
Json = Benchmark (

    oneTimeSetup = (
      JsonLiteral initialize.
    )

    benchmark = (
      ^ (JsonParser with: Json RapBenchmarkMinified) parse.
    )

    verifyResult: result = (
      result class = ParseException ifTrue: [ result println ].
      result isObject ifFalse: [^ false].
      (result asObject at: 'head') isObject ifFalse: [^ false].
      (result asObject at: 'operations') isArray ifFalse: [^ false].
      ^ (result asObject at: 'operations') asArray size = 156
    )

    ----

    RapBenchmarkMinified = (
      ^ '{"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","indentionWidth":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"]]}]]]}'
23   )
24)

```
JsonArray = JsonValue (
  | values |

  initialize = ( values := Vector new )

  addInteger: value = (
    values append: (JsonValue integer: value)
  )

  addDouble: value = (
    values append: (JsonValue double: value)
  )

  addBoolean: value = (
    values append: (JsonValue boolean: value)
  )

  addString: value = (
    values append: (JsonValue string: value)
  )

  add: value = (
    value ifNil: [ self error: 'value is null' ].
    values append: value
  )

  at: index putInteger: value = (
    values at: index put: (JsonValue integer: value)
  )

  at: index putDouble: value = (
    values at: index put: (JsonValue double: value)
  )

  at: index putBoolean: value = (
    values at: index put: (JsonValue boolean: value)
  )

  at: index putString: value = (
    values at: index put: (JsonValue string: value)
  )

  at: index put: value = (
    value ifNil: [ self error: 'value is null' ].
    values at: index put: value
  )

  removeAt: index = (
    values removeAt: index
  )

  size = (
    ^ values size
  )

  isEmpty = (
    ^ values isEmpty
  )
```

```

  at: index = (
    ^ values at: index
  )

  values = (
    ^ values
  )

  isArray = (
    ^ true
  )

  asArray = (
    ^ self
  )

  hashcode = (
    ^ values hashcode
  )

  = other = (
    self == other ifTrue: [ ^ true ].
    other == nil ifTrue: [ ^ false ].
    self class == other class ifFalse: [ ^ false ].

    ^ values = other values
  )

  ----

  new = ( ^ super new initialize )

  readFrom: string = (
    ^ (JsonValue readFrom: string) asArray
  )
)

```

```
JsonLiteral = JsonValue (
  | value isNull isTrue isFalse |

  initializeWith: val = (
    value   := val.
    isNull  := 'null'  = val.
    isTrue  := 'true'  = val.
    isFalse := 'false' = val.
  )

  asString  = ( ^ value )
  hashcode  = ( ^ value hashcode )
  isNull    = ( ^ isNull )
  isTrue    = ( ^ isTrue )
  isFalse   = ( ^ isFalse )
  isBoolean = ( ^ isTrue || isFalse )

  asBoolean = (
    isNull ifTrue:  [ ^ super asBoolean ]
           ifFalse: [ ^ isTrue ]
  )

  = other = (
    self == other ifTrue: [ ^ true ].
    other == nil ifTrue: [ ^ false ].
    self class == other class ifFalse: [ ^ false ].

    ^ value = other asString
  )

  ----

  | NULL TRUE FALSE |

  initialize = (
    NULL  := self new initializeWith: 'null'.
    TRUE  := self new initializeWith: 'true'.
    FALSE := self new initializeWith: 'false'.
  )

  NULL  = ( ^ NULL )
  TRUE  = ( ^ TRUE )
  FALSE = ( ^ FALSE )
)
```

```
JsonNumber = JsonValue (
  | string |

  initializeWith: str = ( string := str )

  asString = ( ^ string )
  isNumber = ( ^ true )

  asInteger = (
    ^ Integer fromString: string
  )

  asDouble = (
    ^ Double fromString: string
  )

  hashcode = (
    ^ string hashcode
  )

  = other = (
    self == other ifTrue: [ ^ true ].
    other == nil ifTrue: [ ^ false ].

    self class == other class ifFalse: [ ^ false ].
    ^ string = other asString
  )

  ----

  new: string = (
    string ifNil: [ self error: 'string is null' ].
    ^ self new initializeWith: string
  )
)

```

Examples/Benchmarks/Json/JsonObject.som

```
JsonObject = JsonValue (
  | names values table |

  initialize = (
    names  := Vector new.
    values := Vector new.
    table  := HashIndexTable new
  )

  add: name withNumber: value = (
    self add: name with: (JsonValue number: value)
  )

  add: name withBoolean: value = (
    self add: name with: (JsonValue boolean: value)
  )

  add: name withString: value = (
    self add: name with: (JsonValue string: value)
  )

  add: name with: aJsonValue = (
    name ifNil: [ self error: 'name is null'  ].
    aJsonValue ifNil: [ self error: 'aJsonValue is null' ].

    names  append: name.
    values append: aJsonValue.
    table at: name put: names size.
  )

  at: name putNumber: value = (
    self at: name put: (JsonValue number: value)
  )

  at: name putBoolean: value = (
    self at: name put: (JsonValue boolean: value)
  )

  at: name putString: value = (
    self at: name put: (JsonValue string: value)
  )

  at: name put: aJsonValue = (
    | idx |
    name ifNil: [ self error: 'name is null'  ].
    aJsonValue ifNil: [ self error: 'aJsonValue is null' ].

    idx := self indexOf: name.
    idx <> -1
      ifTrue:  [values at: idx put: aJsonValue]
      ifFalse: [
        names  append: name.
        values append: aJsonValue.
        table at: name put: names size.
      ].
  )

  remove: name = (
```

```
  | idx |
  name ifNil: [ self error: 'name is null' ].
  idx := self indexOf: name.
  idx = -1 ifFalse: [
    table  remove: name.
    names  remove: idx.
    values remove: idx.
  ].
)

at: name = (
  | idx |
  name ifNil: [ self error: 'name is null'  ].
  idx := self indexOf: name.
  idx = -1
    ifTrue:  [ ^ nil ]
    ifFalse: [ ^ values at: idx ]
)

at: name asIntegerWith: default = (
  | value |
  value := self at: name.
  value
    ifNil:    [ ^ default ]
    ifNotNil: [ ^ value asInteger ]
)

at: name asDoubleWith: default = (
  | value |
  value := self at: name.
  value
    ifNil:    [ ^ default ]
    ifNotNil: [ ^ value asDouble ]
)

at: name asBooleanWith: default = (
  | value |
  value := self at: name.
  value
    ifNil:    [ ^ default ]
    ifNotNil: [ ^ value asBoolean ]
)

at: name asStringWith: default = (
  | value |
  value := self at: name.
  value
    ifNil:    [ ^ default ]
    ifNotNil: [ ^ value asString ]
)

size = (
  ^ names size
)

isEmpty = (
  ^ names isEmpty
)

names = (
  ^ names
```

```
  )

  "TODO:
  @Override
  void write( final JsonWriter writer ) throws IOException {
    writer.writeObjectOpen();
    Iterator<String> namesIterator = names.iterator();
    Iterator<JsonValue> valuesIterator = values.iterator();
    boolean first = true;
    while( namesIterator.hasNext() ) {
      if( !first ) {
        writer.writeObjectSeparator();
      }
      writer.writeMemberName( namesIterator.next() );
      writer.writeMemberSeparator();
      valuesIterator.next().write( writer );
      first = false;
    }
    writer.writeObjectClose();
  }"

  isObject = ( ^ true )
  asObject = ( ^ self )

  hashcode = (
    | result |
    result := 1.
    result := 31 * result + names hashcode.
    result := 31 * result + values hashcode.
    ^ result
  )

  = other = (
    self == other ifTrue: [ ^ true  ].
    self == nil   ifTrue: [ ^ false ].
    self class == other class ifFalse: [ ^ false ].

    ^ names = other names && values = other values
  )

  indexOf: name = (
    | idx |
    idx := table at: name.
    idx <> -1 && (name = (names at: idx)) ifTrue: [ ^ idx ].
    ^ names lastIndexOf: name
  )

  updateHashIndex = (
    names doIndexes: [: i |
      table add: (names at: i) put: i
    ]
  )

  ----

  new = ( ^ super new initialize )

  readFrom: string = (
    ^ (JsonValue readFrom: string) asObject
  )
)
```

Examples/Benchmarks/Json/JsonParser.som

```
" This is a rough port of minijson from Java.
  TODO: add details (com.eclipsesource.json) "

JsonParser = (
  | input index line column current captureBuffer captureStart exceptionBlock |

  initializeWith: string = (
    input := string.
    index := 0.
    line  := 1.
    column := 0.
    current := nil.
    captureBuffer := ''.
  )

  parse = (
    | result |
    exceptionBlock := [:ex | ^ ex ].
    self read.
    self skipWhiteSpace.
    result := self readValue.
    self skipWhiteSpace.
    self isEndOfText ifFalse: [ self error: 'Unexpected character'].
    ^ result
  )

  readValue = (
    current = 'n' ifTrue: [ ^ self readNull   ].
    current = 't' ifTrue: [ ^ self readTrue   ].
    current = 'f' ifTrue: [ ^ self readFalse  ].
    current = '"' ifTrue: [ ^ self readString ].
    current = '[' ifTrue: [ ^ self readArray  ].
    current = '{' ifTrue: [ ^ self readObject ].

    "Is this really the best way to write this?, or better #or:?,
     but with all the nesting, it's just ugly."
    current = '-' ifTrue: [ ^ self readNumber ].
    current = '0' ifTrue: [ ^ self readNumber ].
    current = '1' ifTrue: [ ^ self readNumber ].
    current = '2' ifTrue: [ ^ self readNumber ].
    current = '3' ifTrue: [ ^ self readNumber ].
    current = '4' ifTrue: [ ^ self readNumber ].
    current = '5' ifTrue: [ ^ self readNumber ].
    current = '6' ifTrue: [ ^ self readNumber ].
    current = '7' ifTrue: [ ^ self readNumber ].
    current = '8' ifTrue: [ ^ self readNumber ].
    current = '9' ifTrue: [ ^ self readNumber ].

    "else"
    self expected: 'value'
  )

  readArrayElement: array = (
    self skipWhiteSpace.
    array add: self readValue.
    self skipWhiteSpace.
  )
```

```
readArray = (
  | array |
  self read.
  array := JsonArray new.

  "Array might be empty"
  self skipWhiteSpace.
  (self readChar: ']') ifTrue: [
    ^ array
  ].

  self readArrayElement: array.
  [self readChar: ','] whileTrue: [
    self readArrayElement: array.
  ].

  (self readChar: ']') ifFalse: [
    self expected: '"," or "]"'
  ].
  ^ array
)

readObjectKeyValuePair: object = (
  | name |
  self skipWhiteSpace.
  name := self readName.
  self skipWhiteSpace.

  (self readChar: ':') ifFalse: [ self expected: ':' ].

  self skipWhiteSpace.

  object add: name with: self readValue.

  self skipWhiteSpace.
)

readObject = (
  | object |
  self read.
  object := JsonObject new.
  self skipWhiteSpace.

  (self readChar: '}') ifTrue: [
    ^ object
  ].

  self readObjectKeyValuePair: object.
  [self readChar: ','] whileTrue: [
    self readObjectKeyValuePair: object.
  ].

  (self readChar: '}') ifFalse: [
    self expected: '"," or "}"'
  ].

  ^ object
)

readName = (
```

```
    current = '"' ifFalse: [ self expected: 'name' ].
    ^ self readStringInternal
  )

  readNull = (
    self read.
    self readRequiredChar: 'u'.
    self readRequiredChar: 'l'.
    self readRequiredChar: 'l'.
    ^ JsonLiteral NULL
  )

  readTrue = (
    self read.
    self readRequiredChar: 'r'.
    self readRequiredChar: 'u'.
    self readRequiredChar: 'e'.
    ^ JsonLiteral TRUE
  )

  readFalse = (
    self read.
    self readRequiredChar: 'a'.
    self readRequiredChar: 'l'.
    self readRequiredChar: 's'.
    self readRequiredChar: 'e'.
    ^ JsonLiteral FALSE
  )

  readRequiredChar: ch = (
    (self readChar: ch) ifFalse: [
      self expected: 'character: ' + ch
    ]
  )

  readString = (
    ^ JsonString new: self readStringInternal
  )

  readStringInternal = (
    | string |
    self read.
    self startCapture.

    [current = '"'] whileFalse: [
      current = '\\' ifTrue: [
        self pauseCapture.
        self readEscape.
        self startCapture.
      ] ifFalse: [
        "if (current < 0x20) { throw expected('valid string character'); }"
        "we currently don't have a way to get the ordinal value for a character"
        "} else {"
        self read.
      ]
    ].
    string := self endCapture.
    self read.
    ^ string
```

```
  )

  readEscapeChar = (
    current = '"'  ifTrue: [ ^ '"'  ].
    current = '/'  ifTrue: [ ^ '/'  ].
    current = '\\' ifTrue: [ ^ '\\' ].

    current = 'b'  ifTrue: [ ^ '\b'  ].
    current = 'f'  ifTrue: [ ^ '\f'  ].
    current = 'n'  ifTrue: [ ^ '\n'  ].
    current = 'r'  ifTrue: [ ^ '\r'  ].
    current = 't'  ifTrue: [ ^ '\t'  ].

    "TODO: SOM doesn't have a way to create unicode characters."
    self expected: 'valid escape sequence. note, some are not supported'
  )

  readEscape = (
    self read.
    captureBuffer := captureBuffer concatenate: self readEscapeChar.
    self read
  )

  readNumber = (
    | firstDigit |
    self startCapture.
    self readChar: '-'.
    firstDigit := current.

    self readDigit ifFalse: [ self expected: 'digit' ].
    firstDigit <> '0' ifTrue: [ [self readDigit] whileTrue: []].

    self readFraction.
    self readExponent.
    ^ JsonNumber new: self endCapture
  )

  readFraction = (
    (self readChar: '.') ifFalse: [ ^ false ].

    self readDigit ifFalse: [ self expected: 'digit' ].

    [self readDigit] whileTrue: [].

    ^ true
  )

  readExponent = (
    ((self readChar: 'e') not and: [
      (self readChar: 'E') not]) ifTrue: [ ^ false ].


    (self readChar: '+') ifFalse: [ self readChar: '-' ].

    self readDigit ifFalse: [ self expected: 'digit' ].

    [self readDigit] whileTrue: [].

    ^ true
  )

```

```
  readChar: ch = (
    current = ch ifFalse: [ ^ false ].
    self read.
    ^ true
  )

  readDigit = (
    self isDigit ifFalse: [ ^ false ].
    self read.
    ^ true
  )

  skipWhiteSpace = (
    [ self isWhiteSpace ]
      whileTrue:
        [ self read ].
  )

  read = ("TODO: this is probably broken"
    current = '\n' ifTrue: [
      line := line + 1.
      column := 0.
    ].

    index := index + 1.
    column := column + 1.

    input ifNil: [ self error:'input nil'].
    index <= input length
      ifTrue:  [ current := input charAt: index ]
      ifFalse: [ current := nil ]
  )

  startCapture = (
    captureStart := index
  )

  pauseCapture = (
    captureBuffer := captureBuffer concatenate: (
      input substringFrom: captureStart to: index - 1).
    captureStart := -1
  )

  endCapture = (
    | captured |
    '' = captureBuffer
      ifTrue:  [ captured := input substringFrom: captureStart to: index - 1 ]
      ifFalse: [
        self pauseCapture.
        captured := captureBuffer.
        captureBuffer := '' ].
    captureStart := -1.

    ^ captured
  )

  expected: expected = (
    self isEndOfText ifTrue: [
      self error: 'Unexpected end of input, expected ' + expected asString
    ].
```

```
    self error: 'Expected ' + expected
  )

  error: message = (
    'error:' print.
    message print.
    ':' print.
    line print.
    column println.
    exceptionBlock value: (ParseException with: message at: index
                                        line: line    column: column )
  )

  isWhiteSpace = (
    current = ' '  ifTrue: [^ true].
    current = '\t' ifTrue: [^ true].
    current = '\n' ifTrue: [^ true].
    current = '\r' ifTrue: [^ true].
    ^ false
  )

  isDigit = (
    current = '0' ifTrue: [^ true].
    current = '1' ifTrue: [^ true].
    current = '2' ifTrue: [^ true].
    current = '3' ifTrue: [^ true].
    current = '4' ifTrue: [^ true].
    current = '5' ifTrue: [^ true].
    current = '6' ifTrue: [^ true].
    current = '7' ifTrue: [^ true].
    current = '8' ifTrue: [^ true].
    current = '9' ifTrue: [^ true].
    ^ false
  )

  isEndOfText = (
    ^ current isNil
  )

  ----

  with: aJsonString = (
    ^ self new initializeWith: aJsonString
  )
)
```

Examples/Benchmarks/Json/JsonSmall.som

```
JsonSmall = Benchmark (

    oneTimeSetup = (
      JsonLiteral initialize.
    )

    benchmark = (
      ^ (JsonParser with: JsonSmall RapBenchmark) parse.
    )

    verifyResult: result = (
      result class = ParseException ifTrue: [ result println ].
      result isObject ifFalse: [^ false].
      (result asObject at: 'head') isObject ifFalse: [^ false].
      (result asObject at: 'operations') isArray ifFalse: [^ false].
      ^ (result asObject at: 'operations') asArray size = 41
    )

    ----

    RapBenchmark = (
      ^ '{
            "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":"TableViewer"}],
              ["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","indentionWidth":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,"focusItem":"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","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"]]}]]}'
    )
)
```

```
JsonString = JsonValue (
  | string |

  initializeWith: str = ( string := str )

  isString = (
    ^ true
  )

  asString = (
    ^ string
  )

  hashcode = (
    ^ string hashcode
  )

  = other = (
    self == other ifTrue: [ ^ true ].
    other == nil ifTrue: [ ^ false ].

    self class == other class ifFalse: [ ^ false ].

    ^ string =  other asString
  )

  ----

  new: str = ( ^ self new initializeWith: str )

)
```

Examples/Benchmarks/Json/JsonValue.som

```
JsonValue = (

  isObject  = ( ^ false )
  isArray   = ( ^ false )
  isNumber  = ( ^ false )
  isString  = ( ^ false )
  isBoolean = ( ^ false )

  isTrue  = ( ^ false )
  isFalse = ( ^ false )
  isNull  = ( ^ false )

  asObject = (
    self error: 'Unsupported operation, not an object: ' + self asString
  )

  asArray = (
    self error: 'Unsupported operation, not an array: ' + self asString
  )

  asInteger = (
    self error: 'Unsupported operation, not a number: ' + self asString
  )

  asDouble = (
    self error: 'Unsupported operation, not a number: ' + self asString
  )

  asString = (
    self error: 'Unsupported operation, not a string: ' + self asString
  )

  asBoolean = (
    self error: 'Unsupported operation, not a boolean: ' + self asString
  )

  writeTo: writer = (
    self writeTo: writer with: nil
  )

" TODO:
    public void writeTo( final Writer writer, final WriterConfig config ) throws IOException {
    WritingBuffer buffer = new WritingBuffer( writer, 128 );
    write( config == null ? new JsonWriter( buffer ) : config.createWriter( buffer ) );
    buffer.flush();
  }"

  "TODO:
  @Override
  public String toString() {
    return toString( null );
  }

  public String toString( final WriterConfig config ) {
    StringWriter writer = new StringWriter();
    try {
```

```
      writeTo( writer, config );
    } catch( IOException exception ) {
      // StringWriter does not throw IOExceptions
      throw new RuntimeException( exception );
    }
    return writer.toString();
  }"


  writeOn: writer = (
    ^ self subclassResponsibility
  )

  ----

  readFrom: str = (
    ^ (JsonParser with: str) parse
  )

  integer: anInteger = (
    ^ JsonNumber new: anInteger asString
  )

  double: aDouble = (
    aDouble ifNil: [self error: 'aDouble nil'].
    ^ JsonNumber new: (self cutOffPointZero: aDouble asString)
  )

  string: aString = (
    ^ aString == nil
      ifTrue:  [ JsonLiteral NULL ]
      ifFalse: [ JsonString new: aString ]
  )

  boolean: aBoolean = (
    ^ aBoolean
      ifTrue:  [ JsonLiteral TRUE  ]
      ifFalse: [ JsonLiteral FALSE ]
  )

  cutOffPointZero: str = (
    (str endsWith: '.0') ifTrue: [ ^ str substringFrom: 1 to: str length - 2].
    ^ str
  )
)
```

Examples/Benchmarks/Json/ParseException.som

```
ParseException = (
  | offset line column msg |

  initializeWith: message at: anOffset line: aLine column: aColumn = (
    msg    := message.
    offset := anOffset.
    line   := aLine.
    column := aColumn.
  )

  message = ( ^ msg )

  offset = ( ^ offset )
  line   = ( ^ line   )
  column = ( ^ column )

  asString = ( ^ msg + ':' + line + ':' + column )

  ----

  with: aMessageString at: offset line: line column: column = (
    ^ self new initializeWith: aMessageString
                           at: offset
                         line: line
                       column: column
  )
)
```

Examples/Benchmarks/LanguageFeatures/Dispatch.som

```
"

$Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Dispatch = Benchmark (

    benchmark = (
        | cnt |
        cnt := 0.
        1 to: 20000 do: [ :i | cnt := cnt + (self method: i) ].
        ^ cnt
    )

    method: argument = ( ^argument )

    verifyResult: result = (
      ^ 200010000 = result
    )

)
```

```
"

$Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

DispatchNoArg = Benchmark (

    benchmark = (
        1 to: 20000 do: [ :i | self method ]
    )

    method = ( ^ 1 )

)

```

```
"

$Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

DispatchPerform = Benchmark (

    benchmark = (
        1 to: 20000 do: [ :i | self perform: #method: withArguments: (Array with: i)]
    )

    method: argument = ( ^ argument )

)
```

Examples/Benchmarks/LanguageFeatures/DispatchPerformNoArg.som

```
"

$Id: Dispatch.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

DispatchPerformNoArg = Benchmark (

    benchmark = (
        1 to: 20000 do: [ :i | self perform: #method ]
    )

    method = ( ^ 1 )

)

```

```
Calculator = (
    | a |

    initializeWith: anInt = (
        a := anInt
    )

    inc: aSymbol = (
        aSymbol = #once ifTrue: [ a := a + 1 ]
    )

    a = ( ^ a )

    doesNotUnderstand: selector arguments: arguments = (
        ^ self inc: #once
    )

    ----

    new: a = (
        | calc |
        calc := self new.
        calc initializeWith: a.
        ^ calc
    )
)
```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/CalculatorDelegate.som

```
CalculatorDelegate = (
    | target |

    initializeWith: aCalculator = (
        target := aCalculator
    )

    inc: aSymbol = (
        target inc: aSymbol
    )

    ----

    new: a = (
        | calc |
        calc := self new.
        calc initializeWith: a.
        ^ calc
    )
)
```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/CalculatorDnuPerform.som

```
CalculatorDnuPerform = (
    | a |

    initializeWith: anInt = (
        a := anInt
    )

    inc: aSymbol = (
        aSymbol = #once ifTrue: [ a := a + 1 ]
    )

    a = ( ^ a )

    doesNotUnderstand: selector arguments: arguments = (
        ^ self perform: #inc: withArguments: arguments
    )

    ----

    new: a = (
        | calc |
        calc := self new.
        calc initializeWith: a.
        ^ calc
    )
)
```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/DirectAdd.som

```
DirectAdd = Benchmark (
    | calc |

    initialize = (
        calc := Calculator new.
    )

    benchmark = (
        calc initializeWith: 5.

        1 to: 20000 do: [ :i |
            calc inc: #once
        ].

        calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result' println. calc a println. ]
    )

    ----

    new = (
        ^ super new initialize
    )
)

```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/DnuAdd.som

```
DnuAdd = Benchmark (
    | calc |

    initialize = (
        calc := Calculator new.
    )

    benchmark = (
        calc initializeWith: 5.

        1 to: 20000 do: [ :i |
            calc incDNU: #once
        ].

        calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result' println. calc a println. ]
    )

    ----

    new = (
        ^ super new initialize
    )
)
```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/DnuPerformAdd.som

```
DnuPerformAdd = Benchmark (
    | calc |

    initialize = (
        calc := CalculatorDnuPerform new.
    )

    benchmark = (
        calc initializeWith: 5.

        1 to: 20000 do: [ :i |
            calc incDNU: #once
        ].

        calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result' println. calc a println. ]
    )

    ----

    new = (
        ^ super new initialize
    )
)

```

```
IndirectAdd = Benchmark (
    | delegate calc |

    initialize = (
        calc := Calculator new.
        delegate := CalculatorDelegate new: calc
    )

    benchmark = (
        calc initializeWith: 5.

        1 to: 20000 do: [ :i |
            delegate inc: #once
        ].

        ^ calc a
    )

    verifyResult: result = (
      ^ 20005 = result
    )

    ----

    new = (
        ^ super new initialize
    )
)

```

Examples/Benchmarks/LanguageFeatures/DoesNotUnderstand/PerformAdd.som

```
PerformAdd = Benchmark (
    | calc |

    initialize = (
        calc := Calculator new.
    )

    benchmark = (
        calc initializeWith: 5.

        1 to: 20000 do: [ :i |
            calc perform: #inc: withArguments: (Array with: #once)
        ].

        calc a = 20005 ifFalse: [ 'Benchmark failed with wrong result' println. calc a println. ]
    )

    ----

    new = (
        ^ super new initialize
    )
)

```

```
Proxy = (
    | target |

    initializeWith: anObj = (
        target := anObj
    )

    doesNotUnderstand: selector arguments: arguments = (
        ^ target perform: selector withArguments: arguments
    )

    ----

    new: target = (
        | proxy |
        proxy := self new.
        proxy initializeWith: target.
        ^ proxy
    )
)
```

```
ProxyAdd = Benchmark (
    | proxy calc |

    initialize = (
        calc := Calculator new.
        proxy := Proxy new: calc
    )

    benchmark = (
        calc initializeWith: 5.

        1 to: 20000 do: [ :i |
            proxy inc: #once
        ].

        ^ calc a
    )

    verifyResult: result = (
      ^ 20005 = result
    )

    ----

    new = (
        ^ super new initialize
    )
)
```

Examples/Benchmarks/LanguageFeatures/Fibonacci.som

```
"

$Id: Fibonacci.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Fibonacci = Benchmark (

    benchmark = ( | result |
        result := self fibonacci: 20.
        ^ result
    )

    fibonacci: n = (
        ^(n <= 1)
            ifTrue:  1
            ifFalse: [ (self fibonacci: (n - 1)) + (self fibonacci: (n - 2)) ]
    )

    verifyResult: result = (
      ^ 10946 = result
    )

)
```

Examples/Benchmarks/LanguageFeatures/FieldLoop.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

FieldLoop = Benchmark (
    | counter |

    benchmark = ( | iter |
        counter := 0.
        iter := 20000.

        [ iter > 0 ] whileTrue: [
          iter := iter - 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.

          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.

          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.

          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
```

```
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.

          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.
          counter := counter + 1.

        ].

        ^ counter
    )

    verifyResult: result = (
      ^ 600000 = result
    )

)
```

```
FieldWrite = Benchmark (
    | counter |

    benchmark = (
        | bar |
        bar := 1234.

        1 to: 20000 do: [:i |
          counter := 2122.
        ].
        ^ counter
    )

    verifyResult: result = (
      ^ 2122 = result and: [counter = result]
    )

)

```

```
"

$Id: IntegerLoop.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

IntegerLoop = Benchmark (

    benchmark = ( | bounds a |
        bounds := 20000.
        bounds negated to: bounds by: 1 do: [:value | a := value-value].
        ^ a
    )

    verifyResult: result = (
      ^ 0 = result
    )
)

```

```
"

$Id: Loop.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Loop = Benchmark (

    singleRun = (
        | sum |
        sum := 0.
        1 to: 100 do: [ :j | sum := sum + 1 ].
        (sum = 100)
            ifFalse: [
                self error: 'Wrong result: ' + sum + ' should be: 100' ].
        ^ sum
    )

    benchmark = (
        | sum |
        sum := 0.
        1 to: 200 do: [ :i | sum := sum + self singleRun ].
        ^ sum
    )

    verifyResult: result = (
      ^ 20000 = result
    )
)

```

```
NonLocalReturn = Benchmark (

    first:  a = ( ^ self second: a )
    second: a = ( ^ self third: a  )
    third:  a = ( a value )

    nlr = (
        self first: [^ 1]
    )

    benchmark = (
        | sum |
        sum := 0.
        1 to: 200 do: [ :i | sum := sum + self nlr ].
        ^ sum
    )

    verifyResult: result = (
      ^ 200 = result
    )

)
```

Examples/Benchmarks/LanguageFeatures/Recurse.som

```
"

$Id: Recurse.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Recurse = Benchmark (

    benchmark = (
        ^ self recurse: 13
    )

    recurse: n = (
        (n > 0) ifTrue: [ self recurse: n - 1. self recurse: n - 1 ].
        ^ n
    )

    verifyResult: result = (
      ^ 13 = result
    )
)
```

Examples/Benchmarks/LanguageFeatures/SlopStone.som

```
"
modified for SOM by Stefan Marr.

modified for Squeak by nishis@urban.ne.jp
I added one class methods.
and modified execute method.  block variables are renamed for Squeak.
Thanx to Mr. Tim Olson, Mr. Kohler Markus, Mr. Tim Rowledge, Mr. John Maloney, Mr. Ian Piumarta.

original comments
    NAME            STones80
    AUTHOR          bruce@utafll.uta.edu (Bruce Samuelson)
    FUNCTION        low and medium level benchmarks for ST80 and ST/V
    ST-VERSIONS     pre R4.0, R4.0, R4.1, ST/V
    PREREQUISITES   need floating point hardware or emulation
    CONFLICTS       none
    DISTRIBUTION    world
    VERSION         1.0
    DATE            April 16, 1993

SUMMARY The filein includes two classes: Slopstones (Smalltalk Low level
OPeration Stones) and Smopstones (Smalltalk Medium level OPeration Stones).
Each includes seven cpu intensive benchmarks. They work equally well with
16-bit and 32-bit implementations and are designed to be portable to all
Smalltalk versions from ParcPlace and Digitalk. They are normalized to the
author's 486/33 Windows 3.1 machine running ParcPlace VisualWorks 1.0.
Results have been posted to the Usenet group comp.lang.smalltalk and form the
basis of an article that is scheduled to be published in the June issue of The
Smalltalk Report.

The only difference between the ST80 (STones80) and ST/V (STonesV) filein is
in the messages that define the classes in the first few lines of code. The
ST80 messages specify the class category and message protocol, which are not
used in ST/V.

Bruce Samuelson
"

SlopStone = Benchmark (
  | o obj |

  innerBenchmarkLoop: innerIterations = (
    "Using the SlopStone benchmarks, but not doing the
     old style Stone performance number reporting"
    obj := Object.
    o := obj new.

    1 to: innerIterations do: [:i | self doIntAdd         ].
    1 to: innerIterations do: [:i | self doFloatAdd       ].
    "1 to: innerIterations do: [:i | self doStringAccess   ]." "NOT SUPPORTED"
    1 to: innerIterations do: [:i | self doObjectCreation ].
```

```
    "1 to: innerIterations do: [:i | self doObjectCopy     ]." "NOT SUPPORTED"
    1 to: innerIterations do: [:i | self doPerform        ].
    1 to: innerIterations do: [:i | self doBlockValue     ].
  )


  "STEFAN: use the original benchmarks, but do not use original way
           of reporting results. Rely on the SMark reporting instead."

    "INTRODUCTION

    Slopstone: Smalltalk Low level OPeration Stones
    Portable Low Level Benchmarks for ST80 and ST/V (using 16-bit
    SmallIntegers) Placed in public domain January 1993  (c) Bruce
    Samuelson Permission is given to place this in public Smalltalk archives

    Use monospaced fonts if possible to view the methods in this class.

    (1) Collect garbage if supported (2) do 'SlopstoneBenchmark new
    runBenchmark'. Results are printed in the Transcript window.
    Post results for your machines to comp.lang.smalltalk or
    mail them to bruce@ling.uta.edu or bruce@utafll.uta.edu.

    DISCUSSION

    This readme method would normally be in the class comment for ST80.
    ST/V-DOS doesn't support class comments.

    The benchmarks test strictly low level operations. They do not test
    higher level operations such as forming sets, sorting, or streaming, nor
    do they test
    applications. They also do not test user interface operations because of
    the non-portability of this area of Smalltalk and its sensitivity to the
    performance of the video subsystem. The tests are cpu bound. They do
    not access files and should not cause disk paging.

    The benchmarks use loop counts of 16000 because SmallIntegers cannot
    exceed 16383 for ST/V-DOS. 16-bit implementions would perform worse
    with large loop
    counts. The benchmarks are also suitable for testing 32-bit versions of
    Smalltalk.

    DEFINITION OF REFERENCE MACHINE (ONE SLOPSTONE)

    The following machine is the one on which I developed these
    benchmarks. By
    convention it is defined to operate at one slopstone. It's a mid range
    performer for current ParcPlace versions of Smalltalk.

    Hardware: Amax 486DX/33 (includes internal floating point processor
    and internal 8K cache), 256K external cache, 16MB RAM.

    Software: ParcPlace VisualWorks 1.0, Windows 3.1, DOS 5.0 (plain vanilla
```

```
  setup).

  COMPARISON TO XEROX DORADO

  For reference, the machine runs at 649% of a Dorado on ParcPlace
  benchmarks for ST80 4.1. Its fast video card helps on these PPS
  benchmarks. I didn't run
  them for VisualWorks 1.0. It would be somewhat slower because there
  are vastly
  more classes.

  EXAMPLE RESULTS FOR REFERENCE MACHINE

  1000s    time    1000s of
  itera-   sec-    iterations   slop-
  tions    onds    per sec      stones   explanation

  3808     0.577   6600         1.0      add integers
   544     2.262    240         1.0      add floats
   960     1.088    882         1.0      access strings
   320     0.908    352         1.0      create objects
   160     1.49     107         1.0      copy objects
   480     1.129    425         1.0      perform selectors
   896     1.237    724         1.0      evaluate blocks

   640     1.151    555         1.0      geometric mean"


doBlockValue = (
  [] value. [] value. [] value. [] value. [] value. [] value. [] value.
  [] value. [] value. [] value. [] value. [] value. [] value. [] value.
  [] value. [] value. [] value. [] value. [] value. [] value. [] value.
  [] value. [] value. [] value. [] value. [] value. [] value. [] value.
  [] value. [] value. [] value. [] value. [] value. [] value. [] value.
  [] value. [] value. [] value. [] value. [] value. [] value. [] value.
  [] value. [] value. [] value. [] value. [] value. [] value. [] value.
  [] value. [] value. [] value. [] value. [] value. [] value. [] value
)


doFloatAdd = (
  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+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+1.0
)

doIntAdd = (
  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+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+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+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+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+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+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+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+1+1+1+1+1+1+1+1+1"
)

doObjectCopy = (
  o copy copy copy copy copy copy copy copy copy copy
)

doObjectCreation = (
```

```
    obj new. obj new. obj new. obj new. obj new.
    obj new. obj new. obj new. obj new. obj new.
    obj new. obj new. obj new. obj new. obj new.
    obj new. obj new. obj new. obj new. obj new
  )

  doPerform = (
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value.
    o perform: #value. o perform: #value. o perform: #value
  )

  doStringAccess = (
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1.
    'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1. 'a' at: 1
  )
)
```

Examples/Benchmarks/LanguageFeatures/Sum.som

```
"

$Id: Sum.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Sum = Benchmark (

    benchmark = (
        | result |
        1 to: 2 do: [ :i | result := self sumFrom: 1 to: 10000 ].
        ^ result
    )

    sumFrom: start to: end = (
        | sum |
        sum := 0.
        start to: end do: [ :i | sum := sum + i ].
        ^sum
    )

    verifyResult: result = (
      ^ 50005000 = result
    )

)

```

Examples/Benchmarks/LanguageFeatures/WhileLoop.som

```
"
Copyright (c) 2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

WhileLoop = Benchmark (

    singleRun = (
        | sum |
        sum := 0.
        [sum < 1000]
            whileTrue:
                [sum := sum + 1].
        ^ sum
    )

    benchmark = (
        | sum |
        sum := 0.
        [sum < 20000]
            whileTrue:
                [sum := sum + self singleRun].
        ^ sum
    )

    verifyResult: result = (
      ^ 20000 = result
    )
)
```

```
"
Copyright (c) 2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

WhileLoopPoly = Benchmark (

    singleRun = (
        | sum poly b |
        sum := 0.
        [sum < 1000]
            whileTrue:
                [sum := sum + 1.
                ((sum % 4) = 0) ifTrue:  [poly := 1].
                ((sum % 4) = 1) ifTrue:  [poly := 'abc'].
                ((sum % 4) = 2) ifTrue:  [poly := 2222222222222222].
                ((sum % 4) = 3) ifTrue:  [poly := 1//2].
                b := poly
                ].
        b := b + b.
        ^ sum
    )

    benchmark = (
        | sum |
        sum := 0.
        [sum < 20000]
            whileTrue:
                [sum := sum + self singleRun].
        ^ sum
    )

    verifyResult: result = (
      ^ 20000 = result
    )
)
```

Examples/Benchmarks/LanguageFeatures/WhileLoopVAt.som

```
"
Copyright (c) 2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

WhileLoopVAt = Benchmark (

    singleRun = (
        | sum v |
        v := Vector new.
        v at: 1 put: 1.
        sum := 0.
        [sum < 1000]
            whileTrue:
                [sum := sum + (v at: 1)].
        ^ sum
    )

    benchmark = (
        | sum |
        sum := 0.
        [sum < 20000]
            whileTrue:
                [sum := sum + self singleRun].
        ^ sum
    )

    verifyResult: result = (
      ^ 20000 = result
    )
)
```

Examples/Benchmarks/List.som

```
"

$Id: List.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

List = Benchmark (

    benchmark = ( | result |
        result := self
            tailWithX: (self makeList: 15)
            withY: (self makeList: 10)
            withZ: (self makeList: 6).
        ^ result length
    )

    verifyResult: result = (
      ^ self assert: 10 equals: result
    )

    makeList: length = (
        (length = 0)
            ifTrue: [ ^nil ]
            ifFalse: [
                ^(ListElement new: length)
                    next: (self makeList: (length - 1)) ]
    )

    isShorter: x than: y = (
        | xTail yTail |

        xTail := x. yTail := y.
        [ yTail isNil ]
            whileFalse: [
                xTail isNil ifTrue: [ ^true ].
```

```
                xTail := xTail next.
                yTail := yTail next ].

        ^false
    )

    tailWithX: x withY: y withZ: z = (
        (self isShorter: y than: x)
            ifTrue: [
                ^(self
                    tailWithX: (self tailWithX: x next withY: y withZ: z)
                    withY: (self tailWithX: y next withY: z withZ: x)
                    withZ: (self tailWithX: z next withY: x withZ: y)) ]
            ifFalse: [ ^z ].
    )

)
```

Examples/Benchmarks/ListElement.som

```
"

$Id: ListElement.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

ListElement = (

    | val next |

    length = ( next isNil ifTrue: [ ^1 ] ifFalse: [ ^(1 + next length) ] )

    val    = ( ^val )
    val: n = ( val := n )

    next          = ( ^next )
    next: element = ( next := element )

    ------------------

    new: n = ( ^super new val: n )

)


```

```
" This version is a transcription of the Ruby implementation mandelbrot.rb
  found with JRuby
  https://raw.githubusercontent.com/jruby/jruby/3e43676ee6dc3c13e70fe4a52cce685128c23b8e/bench/truffle/mandelbrot.rb

  The original copyright statement reads as follows:

# Copyright © 2004-2013 Brent Fulgham
#
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
#   * Redistributions of source code must retain the above copyright notice,
#     this list of conditions and the following disclaimer.
#
#   * Redistributions in binary form must reproduce the above copyright notice,
#     this list of conditions and the following disclaimer in the documentation
#     and/or other materials provided with the distribution.
#
#   * Neither the name of 'The Computer Language Benchmarks Game' nor the name
#     of 'The Computer Language Shootout Benchmarks' nor the names of its
#     contributors may be used to endorse or promote products derived from this
#     software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS'
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
# OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
# OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
# The Computer Language Benchmarks Game
# http://benchmarksgame.alioth.debian.org
#
#  contributed by Karl von Laudermann
#  modified by Jeremy Echols
#  modified by Detlef Reichl
```

```
#  modified by Joseph LaFata
#  modified by Peter Zotov
#
# http://benchmarksgame.alioth.debian.org/u64q/program.php?test=mandelbrot&lang=yarv&id=3
"

Mandelbrot = Benchmark (
    | firstResult |

    innerBenchmarkLoop: innerIterations = (
      ^ self verify: (self mandelbrot: innerIterations) inner: innerIterations
    )

    verify: result inner: innerIterations = (
      innerIterations = 750 ifTrue: [
        ^ result = 50
      ].
      firstResult isNil ifTrue: [
        firstResult := result.
      ].
      ^ result = firstResult
    )

    mandelbrot: size = (
        | sum byteAcc bitNum y |
        sum     := 0.
        byteAcc := 0.
        bitNum  := 0.

        y := 0.

        [y < size] whileTrue: [
            | ci x |
            ci := (2.0 * y // size) - 1.0.
            x  := 0.

            [x < size] whileTrue: [
                | zr zrzr zi zizi cr escape z notDone |
                zrzr := zr := 0.0.
                zizi := zi := 0.0.
                cr   := (2.0 * x // size) - 1.5.

                z := 0.
                notDone := true.
                escape := 0.
                [notDone and: [z < 50]] whileTrue: [
                    zr := zrzr - zizi + cr.
                    zi := 2.0 * zr * zi + ci.

                    "preserve recalculation"
                    zrzr := zr * zr.
                    zizi := zi * zi.

                    (zrzr + zizi > 4.0) ifTrue: [
                        notDone := false.
                        escape  := 1.
                    ].
                    z := z + 1.
                ].
```

```

                byteAcc := (byteAcc << 1) + escape.
                bitNum  := bitNum + 1.

                " Code is very similar for these cases, but using separate blocks
                  ensures we skip the shifting when it's unnecessary,
                  which is most cases. "
                bitNum = 8
                    ifTrue: [
                      sum := sum bitXor: byteAcc.
                      byteAcc := 0.
                      bitNum  := 0. ]
                    ifFalse: [
                      (x = (size - 1)) ifTrue: [
                          byteAcc := byteAcc << (8 - bitNum).
                          sum := sum bitXor: byteAcc.
                          byteAcc := 0.
                          bitNum  := 0. ]].
                x := x + 1.
            ].
            y := y + 1.
        ].

        ^ sum
    )
)

```

Examples/Benchmarks/NBody/Body.som

```
" The Computer Language Benchmarks Game
  http://shootout.alioth.debian.org/

  contributed by Mark C. Lewis
  modified slightly by Chad Whipkey

  Based on nbody.java ported to SOM by Stefan Marr.
"
Body = (
    | x y z vx vy vz mass |

    x = ( ^ x )
    y = ( ^ y )
    z = ( ^ z )

    vx = ( ^ vx )
    vy = ( ^ vy )
    vz = ( ^ vz )

    mass = ( ^ mass )

    x: val = ( x := val )
    y: val = ( y := val )
    z: val = ( z := val )

    vx: val = ( vx := val )
    vy: val = ( vy := val )
    vz: val = ( vz := val )

    mass: val = ( mass := val )

    offsetMomentumX: px y: py z: pz = (
       vx := 0.0 - (px // Body SolarMass).
       vy := 0.0 - (py // Body SolarMass).
       vz := 0.0 - (pz // Body SolarMass).
    )

    print = (
        'x:  ' print. x println.
        'y:  ' print. y println.
        'z:  ' print. z println.

        'vx: ' print. vx println.
        'vy: ' print. vy println.
        'vz: ' print. vz println.

        'mass: ' print. mass println.
    )

    ----

    | solarMass |

    Pi          = ( ^ 3.141592653589793 )
    SolarMass   = ( ^ solarMass )
    DaysPerYear = ( ^ 365.24 )

    initialize = (
```

```
        solarMass := 4 * self Pi * self Pi.
    )

    new = ( | b |
        b := super new.
        b x: 0.0.  b vx: 0.0.
        b y: 0.0.  b vy: 0.0.
        b z: 0.0.  b vz: 0.0.
        b mass: 0.0.
        ^ b
    )

    jupiter = ( | b |
       b := super new.
       b x:    4.8414314424647209.
       b y:   -1.16032004402742839.
       b z:   -0.103622044471123109.
       b vx:   0.00166007664274403694   * self DaysPerYear.
       b vy:   0.00769901118419740425   * self DaysPerYear.
       b vz:  -0.0000690460016972063023 * self DaysPerYear.
       b mass: 0.000954791938424326609  * self SolarMass.
       ^ b
    )

    saturn = ( | b |
       b := super new.
       b x:    8.34336671824457987.
       b y:    4.12479856412430479.
       b z:   -0.403523417114321381.
       b vx:  -0.00276742510726862411   * self DaysPerYear.
       b vy:   0.00499852801234917238   * self DaysPerYear.
       b vz:   0.0000230417297573763929 * self DaysPerYear.
       b mass: 0.000285885980666130812  * self SolarMass.
       ^ b
    )

    uranus = ( | b |
       b := super new.
       b x:   12.894369562139131.
       b y:  -15.1111514016986312.
       b z:   -0.223307578892655734.
       b vx:   0.00296460137564761618   * self DaysPerYear.
       b vy:   0.0023784717395948095    * self DaysPerYear.
       b vz:  -0.0000296589568540237556 * self DaysPerYear.
       b mass: 0.0000436624404335156298 * self SolarMass.
       ^ b
    )

    neptune = ( | b |
       b := super new.
       b x:   15.3796971148509165.
       b y:  -25.9193146099879641.
       b z:    0.179258772950371181.
       b vx:   0.00268067772490389322   * self DaysPerYear.
       b vy:   0.00162824170038242295   * self DaysPerYear.
       b vz:  -0.000095159225451971587  * self DaysPerYear.
       b mass: 0.0000515138902046611451 * self SolarMass.
       ^ b
    )

    sun = ( | b |
```

```
        b := self new.
        b mass: self SolarMass.
        ^ b
    )
)
```

```
NBody = Benchmark (
    | expectedEnergy |

    initialize = (
        Body initialize.
    )

    innerBenchmarkLoop: innerIterations = (
        | bodies |
        bodies := NBodySystem new.

        1 to: innerIterations do: [:i |
            bodies advance: 0.01.
        ].

        innerIterations = 250000 ifTrue: [
          ^ bodies energy = -0.1690859889909308
        ].
        expectedEnergy isNil ifTrue: [
          expectedEnergy := bodies energy.
          ^ true
        ].

        ^ expectedEnergy = bodies energy.
    )

    ----

    new = (
        ^ super new initialize
    )
)
```

```
" The Computer Language Benchmarks Game
  http://shootout.alioth.debian.org/

  contributed by Mark C. Lewis
  modified slightly by Chad Whipkey

  Based on nbody.java ported to SOM by Stefan Marr.
"
NBodyBench = (
    run: args = (
        | n bodies |
        n := (args at: 2) asInteger.
        Body initialize.

        bodies := NBodySystem new.

        bodies energy println.

        n timesRepeat: [ bodies advance: 0.01 ].

        bodies energy println.
    )
)

```

```
" The Computer Language Benchmarks Game
  http://shootout.alioth.debian.org/

  contributed by Mark C. Lewis
  modified slightly by Chad Whipkey

  Based on nbody.java ported to SOM by Stefan Marr.
"
NBodySystem = (
    | bodies |

    initialize = (
        | px py pz |

        bodies := Array new: 5.
        bodies at: 1 put: Body sun.
        bodies at: 2 put: Body jupiter.
        bodies at: 3 put: Body saturn.
        bodies at: 4 put: Body uranus.
        bodies at: 5 put: Body neptune.

        "bodies do: [:b | b print. '' println ]."

        px := 0.0.  py := 0.0.  pz := 0.0.

        bodies do: [:b |
            px := px + (b vx * b mass).
            py := py + (b vy * b mass).
            pz := pz + (b vz * b mass).
        ].

        (bodies at: 1) offsetMomentumX: px y: py z: pz.

        "bodies do: [:b | b print. '' println ]."
    )

    advance: dt = (
        1 to: bodies length do: [:i |
            | iBody |
            iBody := bodies at: i.

            i + 1 to: bodies length do: [:j |
                | dx dy dz jBody dSquared distance mag |
                jBody := bodies at: j.
                dx := iBody x - jBody x.
                dy := iBody y - jBody y.
                dz := iBody z - jBody z.

                dSquared := (dx * dx) + (dy * dy) + (dz * dz).
                distance := dSquared sqrt.
                mag      := dt // (dSquared * distance).

                iBody vx: iBody vx - (dx * jBody mass * mag).
                iBody vy: iBody vy - (dy * jBody mass * mag).
                iBody vz: iBody vz - (dz * jBody mass * mag).

                jBody vx: jBody vx + (dx * iBody mass * mag).
                jBody vy: jBody vy + (dy * iBody mass * mag).
```

```
                jBody vz: jBody vz + (dz * iBody mass * mag).
            ].
        ].

        bodies do: [:body |
            body x: body x + (dt * body vx).
            body y: body y + (dt * body vy).
            body z: body z + (dt * body vz).
        ].
    )

    energy = (
        | dx dy dz distance e |
        e := 0.0.

        1 to: bodies length do: [:i |
            | iBody |
            iBody := bodies at: i.

            e := e + (0.5 * iBody mass *
                 ((iBody vx * iBody vx) +
                  (iBody vy * iBody vy) +
                  (iBody vz * iBody vz))).

            i + 1 to: bodies length do: [:j |
                | jBody |
                jBody := bodies at: j.

                dx := iBody x - jBody x.
                dy := iBody y - jBody y.
                dz := iBody z - jBody z.

                distance := ((dx*dx) + (dy*dy) + (dz*dz)) sqrt.
                e := e - ((iBody mass * jBody mass) // distance).
            ].
        ].
        ^ e
    )

    ----

    new = (
        ^ super new initialize
    )
)
```

Examples/Benchmarks/PageRank.som

```
"https://github.com/Sable/Ostrich/blob/master/map-reduce/page-rank/js/pagerank.js"
PageRank = Benchmark (
  | firstResult |

  " generates an array of random pages and their links "
  generateRandomPagesN: n outLinks: outLinks divisor: divisor  = (
    | pages |
    " matrix cell i,j means link from j->i "
    pages := Array new: n * n withAll: 0.

    0 to: n - 1 do: [:i |
      outLinks at: i + 1 put: 0.

      0 to: n - 1 do: [:j |
        (i <> j and: [(JenkinsRandom random abs % divisor) = 0]) ifTrue: [
          pages at: (i * n + j) + 1 put: 1.
          outLinks at: i + 1 put: (outLinks at: i + 1) + 1
        ]
      ].

      " the case with no outlinks is afunctioned "
    (outLinks at: i + 1) = 0 ifTrue: [
        | k |
        k := JenkinsRandom random abs % n.
        [i = k] whileTrue: [
          k := JenkinsRandom random abs % n.
        ].

        pages at: (i * n + k) + 1 put: 1.
        outLinks at: i + 1 put: 1
      ]
    ].
    ^ pages
  )

  mapPageRanks: pages pageRanks: pageRanks maps: maps outLinks: outLinks n: n = (
    0 to: n - 1 do: [:i |
      | outboundRank |
      outboundRank := (pageRanks at: i + 1) // (outLinks at: i + 1).

      0 to: n - 1 do: [:j |
        maps at: (i * n + j) + 1
            put: ((pages at: (i * n + j) + 1) = 0
                   ifTrue:  [0.0]
                   ifFalse: [(pages at: (i * n + j) + 1)  * outboundRank])
      ]
    ]
  )

  reducePageRanks: pageRanks maps: maps n: n = (
    | dif |
    dif := 0.0.

    0 to: n - 1 do: [:j |
```

```
      | oldRank newRank |
      oldRank := pageRanks at: j + 1.
      newRank := 0.0.

      0 to: n - 1 do: [:i |
        newRank := newRank + (maps at: (i * n + j) + 1)
      ].

      newRank := ((1 - PageRank DFactor) // n) + (PageRank DFactor * newRank).
      dif := (newRank - oldRank) abs > dif
               ifTrue: [(newRank - oldRank) abs] ifFalse: [ dif ].
      pageRanks at: j + 1 put: newRank
    ].
    ^ dif
  )

  innerBenchmarkLoop: innerIterations = (
    innerIterations < 2 ifTrue: [
      self error: 'innerIterations has to be 2 or more, but was ' + innerIterations ].

    JenkinsRandom seed: 49734321.
    "standard for verification"
    "n := 5000
     iter    := 10.
     thresh  := 0.00000001.
     divisor := 100000."
    ^ self verify: (self runPageRankN: innerIterations iter: 10 thresh: 0.00000001 divisor: 100000)
           inner:  innerIterations.
  )

  runPageRankN: n iter: iter thresh: thresh divisor: divisor = (
    | nbLinks maxDiff pageRanks maps outLinks t pages |
    maxDiff   := Double PositiveInfinity.
    pageRanks := Array new: n     withAll: 1.0 // n.
    maps      := Array new: n * n withAll: 0.0.
    outLinks  := Array new: n     withAll: 0.

    pages := self generateRandomPagesN: n outLinks: outLinks divisor: divisor.

    nbLinks := 0.
    0 to: n - 1 do: [:i |
      1 to: n do: [:j |
        nbLinks := nbLinks + (pages at: i * n + j)
      ]
    ].

    t := 1.
    [t <= iter and: [maxDiff >= thresh]] whileTrue: [
      self mapPageRanks: pages pageRanks: pageRanks maps: maps outLinks: outLinks n: n.
      maxDiff := self reducePageRanks: pageRanks maps: maps n: n.
      t := t + 1
    ].

    ^ pageRanks
  )

```

```
verify: pageRanks inner: innerIterations = (
  innerIterations = 5000 " and: [iter = 10 and: [thresh = 0.00000001 and: [divisor = 100000]]]"
    ifTrue: [
      | expected |
      expected := PageRank ExpectedPageRanks.
      pageRanks length = expected length ifFalse: [
        self error: 'Invalid length of page_ranks array'
      ].

      pageRanks doIndexes: [:i |
        (pageRanks at: i) = (expected at: i) ifFalse: [
          self error: 'ERROR: page_ranks[' + i asString + ']=' +
                      (pageRanks at: i) + ' differs from the expected value: ' +
                      (expected at: i)
        ]
      ] ]
    ifFalse: [
      ^ self checkBasedOnFirstResult: pageRanks inner: innerIterations
    ].
  ^ true
)

checkBasedOnFirstResult: pageRanks inner: innerIterations = (
  pageRanks length = innerIterations ifFalse: [ ^ false ].

  firstResult == nil ifTrue: [
    firstResult := pageRanks.
    ^ true
  ] ifFalse: [
    1 to: innerIterations do: [:i |
      (firstResult at: i) = (pageRanks at: i) ifFalse: [
        ^ false
      ]
    ]
  ].

  ^ true
)

----
| expected |

DFactor = ( ^ 0.85 ) " damping factor"
ExpectedPageRanks = (
  | prevSize |
  expected ifNotNil: [ ^ expected ].

  expected := Vector new: 5000.
  prevSize := expected size.

  1 to: 53 do: [:i |
    self perform: ('pageRanks' + i + ':') asSymbol withArguments: (Array with: expected).
    expected size = prevSize ifTrue: [ self error: 'The method ' + ('pageRanks' + i) asSymbol + ' did not add elements to expected' ].
    prevSize := expected size ].

  expected := expected asArray.

```

```
    ^ expected
  )

  pageRanks1: expected = (
    expected,
       0.000055500000000000001, 0.000030000000000000004, 0.000030000000000000004,
       0.000098850000000000002, 0.000030000000000000004, 0.000055500000000000001,
       0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
       0.0001937753607433594, 0.000030000000000000004, 0.000030000000000000004,
       0.00026861193950589846, 0.0011626495265046287, 0.00033125968546875005,
       0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
       0.002419297210359863, 0.000055500000000000001, 0.000055500000000000001,
       0.000030000000000000004, 0.000030000000000000004, 0.000081000000000000002,
       0.000030000000000000004, 0.00028382917108593755, 0.000055500000000000001,
       0.00022373208234375002, 0.0009500131460436332, 0.0004673045821320703,
       0.000055500000000000001, 0.000030000000000000004, 0.000055500000000000001,
       0.000030000000000000004, 0.000095598750000000003, 0.00029348190124218755,
       0.000095598750000000003, 0.000030000000000000004, 0.000030000000000000004,
       0.0002522648462121094, 0.0002007225, 0.0005721072315653125,
       0.0006549038427949217, 0.00027208502253146485, 0.000030000000000000004,
       0.000095598750000000003, 0.00022515500625, 0.000030000000000000004.
  )

  pageRanks2: expected = (
    expected,
       0.000030000000000000004, 0.00020761099425802734, 0.000077175000000000002,
       0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
       0.0014816283407223048, 0.000030000000000000004, 0.000077175000000000002,
       0.00018305958234375, 0.00014299384687500004, 0.00011125893750000003,
       0.00015412125000000001, 0.000055500000000000001, 0.0005866682663756834,
       0.0009038160999648827, 0.000055500000000000001, 0.000055500000000000001,
       0.000030000000000000004, 0.00014299384687500004, 0.000030000000000000004,
       0.000055500000000000001, 0.000055500000000000001, 0.000030000000000000004,
       0.00012968268750000001, 0.000030000000000000004, 0.000030000000000000004,
       0.000214412971875, 0.000030000000000000004, 0.000081000000000000002,
       0.00016827375000000002, 0.000055500000000000001, 0.0007612028507183201,
```

```
        0.00005550000000000001, 0.000030000000000000004, 0.00005550000000000001,
        0.0017911603607849024, 0.00012435000000000001, 0.0002074907896875,
        0.000030000000000000004, 0.000620176162217578, 0.0016205421327844336,
        0.000030000000000000004, 0.000030000000000000004, 0.0001772368043671875.
  )

  pageRanks3: expected = (
    expected,
        0.0002207814375, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.0009383260308160549, 0.000030000000000000004,
        0.00007717500000000002, 0.000030000000000000004, 0.00005550000000000001,
        0.00007717500000000002, 0.00011727375000000003, 0.00025645517495238287,
        0.00046964726999218764, 0.000030000000000000004, 0.00005550000000000001,
        0.000030000000000000004, 0.00007717500000000002, 0.000030000000000000004,
        0.00048092971362552716, 0.00022518875783285154, 0.000232088372179687 5,
        0.00014550189499218754, 0.000030000000000000004, 0.000030000000000000004,
        0.00007717500000000002, 0.0001801089375, 0.00032185403437500 01,
        0.000030000000000000004, 0.00024264642949335937, 0.00014659875000000002,
        0.00009885000000000002, 0.00005550000000000001, 0.000030000000000000004,
        0.00012435000000000001, 0.00005550000000000001, 0.00009885000000000002,
        0.00009559875000000003, 0.00007717500000000002, 0.00020376888060937495,
        0.00005550000000000001, 0.000030000000000000004, 0.00005550000000000001,
        0.00005550000000000001, 0.00011727375000000003, 0.00005550000000000001,
        0.00023148805436718755, 0.00005550000000000001, 0.0003663848267566407,
        0.000030000000000000004, 0.00008100000000000002, 0.000030000000000000004,
        0.00005550000000000001, 0.00020837250000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.000128175, 0.00005550000000000001,
        0.0007321965646351562, 0.00005550000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.00009559875000000003, 0.00029182125000000001,
        0.00005550000000000001, 0.00007717500000000002, 0.00018287250000000003,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.00011727375000000003, 0.00005550000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.0005117003058394531, 0.000030000000000000004,
        0.000030000000000000004, 0.0005327917869365431, 0.000030000000000000004,
        0.00013675893750000003, 0.00007717500000000002,
```

```
0.000030000000000000004,
        0.000077175000000000002, 0.000030000000000000004,
0.00005550000000000001,
        0.0008628989988383205, 0.00005550000000000001,
0.000030000000000000004,
        0.00010267500000000002, 0.000030000000000000004,
0.000030000000000000004,
        0.000126919125, 0.00010267500000000002, 0.000030000000000000004,
        0.00011125893750000003, 0.000030000000000000004,
0.000030000000000000004,
        0.000077175000000000002, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.00008100000000000002,
0.000030000000000000004,
        0.00027766269375, 0.00005550000000000001, 0.000030000000000000004.
  )

  pageRanks4: expected = (
    expected,
        0.00018287250000000003, 0.000030000000000000004, 0.00006825,
        0.000030000000000000004, 0.00012052500000000002,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.00008100000000000002, 0.000077175000000000002,
0.00005550000000000001,
        0.0005574534602777341, 0.00011402250000000001,
0.00009885000000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.000540919912277207, 0.00008100000000000002,
0.00005550000000000001,
        0.000077175000000000002, 0.000030000000000000004,
0.00005550000000000001,
        0.00013588458234375003, 0.000030000000000000004,
0.00020486333264062499,
        0.0010616644951779884, 0.00005550000000000001,
0.00012109875000000002,
        0.00013894875000000002, 0.0006629172614616794,
0.0015453146662926173,
        0.000030000000000000004, 0.0002057215678125,
0.00014277375000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.00012457009687500003,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.00022662231875, 0.0002800117218750001,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.00012457009687500003,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.0009413663007857225,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.00005550000000000001,
        0.00025216269375, 0.000030000000000000004, 0.00005550000000000001,
        0.00014299384687500004, 0.000030000000000000004,
0.0004204378892502149,
        0.00022315833234375003, 0.001512352057544629,
```

```
0.00040963230479023444,
        0.00026866422187500004, 0.00009885000000000002, 0.000030000000000000004,
        0.0004487230533789453, 0.000030000000000000004, 0.0001568163804609375,
        0.00014985, 0.00007717500000000002, 0.00037312038758925784,
        0.00010267500000000002, 0.00007717500000000002, 0.000935572065964551,
        0.000030000000000000004, 0.000030000000000000004, 0.0006538908974685352,
        0.00005550000000000001, 0.000030000000000000004, 0.0006925555784296876,
        0.00009559875000000003, 0.000030000000000000004, 0.00005550000000000001,
        0.000030000000000000004, 0.000146025, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00004275000000000001, 0.0002412005625, 0.00005550000000000001,
        0.00005550000000000001, 0.000351187703481504, 0.00010267500000000002,
        0.000030000000000000004, 0.00019815573560605466, 0.0001356975,
        0.00007717500000000002, 0.00013588458234375003, 0.0019199758003062113,
        0.000030000000000000004, 0.000030000000000000004, 0.0006042323629701756,
        0.000030000000000000004, 0.0008090710683361913, 0.000030000000000000004.
  )

  pageRanks5: expected = (
    expected,
        0.00021544875, 0.000030000000000000004, 0.0003507081512421876,
        0.000030000000000000004, 0.000030000000000000004, 0.00005550000000000001,
        0.00028123619308710943, 0.00007717500000000002, 0.00022293583249218752,
        0.000030000000000000004, 0.00028254900584279304, 0.0004448098933681447,
        0.000030000000000000004, 0.000030000000000000004, 0.00010267500000000002,
        0.00005550000000000001, 0.000406985079798047, 0.000030000000000000004,
        0.00007717500000000002, 0.000030000000000000004, 0.00005550000000000001,
        0.00020730370734375002, 0.000030000000000000004, 0.00008100000000000002,
        0.00009559875000000003, 0.000030000000000000004, 0.0006841020911458787,
        0.000030000000000000004, 0.000030000000000000004, 0.00019508699999999998,
        0.00023373894375, 0.000269050764124043, 0.00007717500000000002,
        0.0012665335189376758, 0.0003229170826183594, 0.00005550000000000001,
        0.00010267500000000002, 0.00007717500000000002, 0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004, 0.00026877685218750007,
        0.00007717500000000002, 0.000030000000000000004, 0.00005550000000000001,
        0.000030000000000000004, 0.00035339283501253916, 0.00007717500000000002,
```

```
        0.00005550000000000001, 0.000030000000000000004, 0.00007717500000000002,
        0.000220427784375, 0.000030000000000000004, 0.000132, 0.00025887973491310547,
        0.0010559972458375393, 0.00007717500000000002, 0.000030000000000000004,
        0.00005550000000000001, 0.00005550000000000001, 0.00008100000000000002,
        0.0007952039924323437, 0.00009559875000000003, 0.0007619092408695311,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.00008100000000000002, 0.00005550000000000001,
        0.0004741191615037304, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001, 0.00005550000000000001,
        0.00044242997182148444, 0.00014277375000000002, 0.000030000000000000004,
        0.00007717500000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.0008103090240004296, 0.000030000000000000004, 0.0008127957153562891.
  )

  pageRanks6: expected = (
    expected,
        0.00007717500000000002, 0.000030000000000000004, 0.0005670410421640625,
        0.00012435000000000001, 0.0009996132445867678, 0.00005550000000000001,
        0.000190168846875, 0.00019637074171875002, 0.00005550000000000001,
        0.0006413808275615429, 0.0008067955708537303, 0.0007001598184784763,
        0.00007717500000000002, 0.00005550000000000001, 0.00007717500000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.00019364019374999998, 0.000030000000000000004,
        0.000030000000000000004, 0.00024570768750000003, 0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001, 0.000816114721648008,
        0.00016996851984375, 0.000251748159375, 0.00007717500000000002,
        0.00005550000000000001, 0.00005550000000000001, 0.00005550000000000001,
        0.00045582617944294916, 0.000030000000000000004, 0.00007717500000000002,
        0.00009885000000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004, 0.0004087417901358985,
        0.000030000000000000004, 0.00011125893750000003,
```

```
0.0007096632942357614,
        0.0004450802223830273, 0.000030000000000000004,
0.00013788125625000002,
        0.000030000000000000004, 0.000030000000000000004,
0.0007781633581341993,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.0007831720377038671,
        0.00007717500000000002, 0.000146025, 0.000030000000000000004,
        0.0004440335706074609, 0.0002797916250000001,
0.00014534287500000002,
        0.000161417596875, 0.0009436753876707029, 0.00026090069906250005,
        0.000661540765294531, 0.00029711647746210945,
0.0002448333323437501,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.00021968458249218753, 0.00007717500000000002,
0.00018669750000000003,
        0.000030000000000000004, 0.000030000000000000004,
0.0004852117315546876,
        0.0003194149033965235, 0.000030000000000000004,
0.000770995975586406,
        0.000582465439638496, 0.001229452233367832,
0.00007717500000000002,
        0.000030000000000000004, 0.000166433693109375,
0.000969294173404297,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.0001513576875, 0.00009559875000000003, 0.00005550000000000001,
        0.00007717500000000002, 0.000030000000000000004,
0.00027891726984375005,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004.
  )

  pageRanks7: expected = (
    expected,
        0.0003005925113290822, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.0005980535795580271,
        0.000030000000000000004, 0.00005550000000000001,
0.0006150758829812889,
        0.000030000000000000004, 0.00011125893750000003,
0.000030000000000000004,
        0.00005550000000000001, 0.00007717500000000002,
0.00015007009687500002,
        0.000030000000000000004, 0.00005550000000000001,
0.00063777321360625,
        0.000030000000000000004, 0.00041378858300664085,
0.0002291377273359375,
        0.00008100000000000002, 0.0007787174847144531,
0.00011125893750000003,
        0.000319367505031465, 0.00037540608941236337,
0.000030000000000000004,
        0.0008391853330165822, 0.000030000000000000004, 0.0001952814375,
        0.00008100000000000002, 0.00005550000000000001,
0.00009885000000000002,
        0.000030000000000000004, 0.0004302220856250001,
0.00005550000000000001,
```

```
        0.00007717500000000002, 0.00022148922187500001,
0.00008100000000000002,
        0.000168493846875, 0.00005550000000000001, 0.00016444875000000002,
        0.00022096851984374999, 0.0004387228943070313,
0.000030000000000000004,
        0.00018994875000000002, 0.00043396908058710945,
0.000030000000000000004,
        0.000030000000000000004, 0.00119860742323042,
0.00013675893750000003,
        0.00005550000000000001, 0.00007717500000000002,
0.00008100000000000002,
        0.00010267500000000002, 0.000030000000000000004,
0.0002644775398359375,
        0.0007738206560650585, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.0007043146423551758, 0.00005550000000000001,
0.000753936463079121,
        0.00014277375000000002, 0.00005550000000000001,
0.000030000000000000004,
        0.0007755696951406249, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004.
  )

  pageRanks8: expected = (
    expected,
        0.000030000000000000004, 0.00010267500000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00008100000000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.0007849164249744726,
0.00019180253983593749,
        0.00013675893750000003, 0.000030000000000000004,
0.00017895411089179688,
        0.00013293393750000003, 0.000030000000000000004,
0.0009846043482525976,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.00033701304328125O1,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.0006119601669933592,
        0.0005546408271638474, 0.00035661147136054695,
0.000030000000000000004,
        0.00008100000000000002, 0.000030000000000000004,
0.00018305958234375,
        0.00005550000000000001, 0.000030000000000000004,
0.0004162326721509962,
        0.000030000000000000004, 0.00005550000000000001,
0.000140230284375,
        0.00007717500000000002, 0.000030000000000000004,
0.00012435000000000001,
```

```
     0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
     0.000030000000000000004, 0.0001752401304609375, 0.000124350000000000001,
     0.000055500000000000001, 0.0009916779323594143, 0.000030000000000000004,
     0.0001356975, 0.0005381491251261719, 0.000594984439271953,
     0.000030000000000000004, 0.000081000000000000002, 0.0009741378132010936,
     0.0010289986178938673, 0.000098850000000000002, 0.000055500000000000001,
     0.000055500000000000001, 0.000030000000000000004, 0.000077175000000000002,
     0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
     0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
     0.000030000000000000004, 0.000030000000000000004, 0.0003377064978837697,
     0.000030000000000000004, 0.000081000000000000002, 0.00014985,
     0.000077175000000000002, 0.000030000000000000004, 0.000055500000000000001,
     0.000030000000000000004, 0.0004793621620409764, 0.000121098750000000002,
     0.000944214872861504, 0.000055500000000000001, 0.000077175000000000002,
     0.000102675000000000002, 0.000030000000000000004, 0.000030000000000000004.
)

pageRanks9: expected = (
  expected,
     0.000055500000000000001, 0.0003911262409296876, 0.000164448750000000002,
     0.0004658189423039455, 0.0006191960454895896, 0.000121098750000000002,
     0.000081000000000000002, 0.00018994875, 0.0003503022762421876,
     0.000077175000000000002, 0.000146598750000000002, 0.000030000000000000004,
     0.000294136229749043, 0.0001546089375, 0.000030000000000000004,
     0.0004851203751484375 6, 0.000030000000000000004, 0.000102675000000000002,
     0.000055500000000000001, 0.000030000000000000004, 0.000030000000000000004,
     0.000030000000000000004, 0.0009038993515082423, 0.000055500000000000001,
     0.000030000000000000004, 0.000167920096875000002, 0.0004732979309689648 6,
     0.001073084842359102, 0.000227504034375, 0.000077175000000000002,
     0.0004113979149621094, 0.000030000000000000004, 0.000055500000000000001,
     0.000030000000000000004, 0.000030000000000000004, 0.000994461009736816,
     0.000030000000000000004, 0.000030000000000000004, 0.000055500000000000001,
     0.000095598750000000003, 0.00022421976984375, 0.0004580116392566408,
     0.0004266859737324610 3, 0.000030000000000000004, 0.0005213088860603709,
     0.000292395000000000004, 0.000030000000000000004, 0.000055500000000000001,
```

```
        0.000030000000000000004, 0.00009559875000000003, 0.00011727375000000003,
        0.00010267500000000002, 0.000030000000000000004, 0.0004510455397947461,
        0.00014277375000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.002259585483123663, 0.0010017437556216994, 0.00032997952022560553,
        0.000030000000000000004, 0.000030000000000000004, 0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.0007017341088929103,
        0.00005550000000000001, 0.0005571497378849411, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00022170931875,
        0.0006954863535522461, 0.00005550000000000001, 0.0012527846540595113,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.0005184859828281249, 0.000030000000000000004, 0.000030000000000000004,
        0.00013293393750000003, 0.00007717500000000002, 0.0008999596475350778,
        0.00031492419618537123, 0.00008100000000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.0012884279283853029, 0.000030000000000000004,
        0.000030000000000000004, 0.0007648574488383199, 0.0008473654450535153.
  )

  pageRanks10: expected = (
    expected,
        0.00008100000000000002, 0.000030000000000000004, 0.000180329034375,
        0.000170842875, 0.00005550000000000001, 0.000401158314638496,
        0.0006134211817580272, 0.00015881305436718754, 0.000030000000000000004,
        0.00005550000000000001, 0.00007717500000000002, 0.00008100000000000002,
        0.000030000000000000004, 0.00007717500000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001, 0.00007717500000000002,
        0.0005238088503046875, 0.00005550000000000001, 0.00012109875000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00012109875000000002, 0.00005550000000000001, 0.00005550000000000001,
        0.00010267500000000002, 0.0001605102271875, 0.0005087270201025195,
        0.0009967971475125393, 0.00007717500000000002, 0.00009559875000000003,
        0.00008100000000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.00005550000000000001, 0.00005550000000000001,
        0.0006039423526103709, 0.0009677339486394532, 0.000030000000000000004,
        0.0007675520473719141, 0.0007538680901701758,
```

```
0.00008100000000000002,
        0.0006169301652431443, 0.00009885000000000002,
0.0005449327357591993,
        0.00011125893750000003, 0.00012435000000000001,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.00007717500000000002,
        0.00011125893750000003, 0.00021132074186718748,
0.00007717500000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.00007717500000000002, 0.000254291625, 0.00005550000000000001,
        0.000140230284375, 0.00005550000000000001, 0.0005397573664441406,
        0.000030000000000000004, 0.0002286781021875,
0.00011727375000000003,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.00008100000000000002, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001,
0.00012109875000000002,
        0.00014985, 0.0003482766015246095, 0.00007717500000000002,
        0.00015843393750000003, 0.000030000000000000004,
0.00007717500000000002,
        0.000030000000000000004, 0.00011125893750000003,
0.0004241038692580274З,
        0.000030000000000000004, 0.00007717500000000002,
0.000030000000000000004,
        0.00007717500000000002, 0.0003711726986806446б,
0.00005550000000000001,
        0.0001806826875, 0.00005550000000000001, 0.000030000000000000004,
        0.00008100000000000002, 0.00009559875000000003,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00025942602609375004.
  )

  pageRanks11: expected = (
    expected,
        0.000030000000000000004, 0.000030000000000000004,
0.00010267500000000002,
        0.00005550000000000001, 0.00007717500000000002, 0.000245927784375,
        0.00005550000000000001, 0.0001768576875, 0.000030000000000000004,
        0.00005550000000000001, 0.00012435000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001,
0.0003816914234140626З,
        0.000030000000000000004, 0.000030000000000000004,
0.00016978143750000002,
        0.00012435000000000001, 0.000030000000000000004, 0.000218017875,
        0.0003311542154954884, 0.0023047579029954004,
0.0005710822497378906,
        0.00007717500000000002, 0.0003164610867890626,
0.00005550000000000001,
        0.000030000000000000004, 0.0003918354568640626З,
0.000030000000000000004,
        0.000030000000000000004, 0.00010267500000000002,
0.00009885000000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.00005550000000000001, 0.0006218280803472657,
```

```
0.00027158680436718753,
        0.0005329226223534762, 0.000030000000000000004, 0.0001546089375,
        0.000055500000000000001, 0.000030000000000000004,
0.00012109875000000002,
        0.0013023196241036914, 0.000030000000000000004,
0.0005814199422881052,
        0.000030000000000000004, 0.0010607720917741798,
0.00011125893750000003,
        0.000030000000000000004, 0.000030000000000000004,
0.0003044889569777344,
        0.000077175000000000002, 0.000055500000000000001,
0.0008152654201896286,
        0.000055500000000000001, 0.000140230284375, 0.0007082080488527344,
        0.0007379601076354103, 0.000408482320607461,
0.00012968268750000001,
        0.0010253114960741213, 0.000055500000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001,
0.000030000000000000004.
  )

  pageRanks12: expected = (
    expected,
        0.000077175000000000002, 0.000095598750000000003,
0.000030000000000000004,
        0.000146025, 0.00020560893750000002, 0.000030000000000000004,
        0.000030000000000000004, 0.00016119750000000003,
0.000055500000000000001,
        0.000030000000000000004, 0.00031838485247324223,
0.000030000000000000004,
        0.000055500000000000001, 0.0014520803604533988,
0.000030000000000000004,
        0.000055500000000000001, 0.000098850000000000002,
0.000364157824359375,
        0.000055500000000000001, 0.0013332881297266016,
0.0005557404117852929,
        0.000030000000000000004, 0.00010267500000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001,
0.0009105278934790038,
        0.000030000000000000004, 0.00010267500000000002,
0.000030000000000000004,
        0.000095598750000000003, 0.000055500000000000001,
0.000030000000000000004,
        0.0007129847634453126, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001,
0.00013675893750000003,
        0.000950365240418213, 0.000055500000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001,
0.00014919574171875,
        0.000030000000000000004, 0.00010267500000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.0001768576875,
        0.00043941458861267575, 0.000055500000000000001,
0.00028618575000000004,
        0.00020968190109375, 0.00012968268750000001,
0.00012109875000000002,
```

```
        0.0005675717694396287, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.0002826102781975489, 0.00003000000000000004,
        0.00009559875000000003, 0.00005550000000000001, 0.00003000000000000004.
  )

  pageRanks13: expected = (
    expected,
        0.00009559875000000003, 0.000345610471875, 0.00003000000000000004,
        0.0006355078163115429, 0.00005550000000000001, 0.00007717500000000002,
        0.0003739622193515626, 0.00014299384687500004, 0.0002244118043671874 8,
        0.00003000000000000004, 0.0004435998599412695 4, 0.0002641154176562501,
        0.00003000000000000004, 0.000206951353125, 0.00005550000000000001,
        0.00004275000000000001, 0.00003000000000000004, 0.00008100000000000002,
        0.00003000000000000004, 0.00008100000000000002, 0.00003000000000000004,
        0.00003000000000000004, 0.00009559875000000003, 0.00005550000000000001,
        0.00005550000000000001, 0.00035032903437500005, 0.00008100000000000002,
        0.0008822237984673634, 0.0007134373373503516, 0.00003000000000000004,
        0.0008928574627982618, 0.00018287250000000003, 0.00003000000000000004,
        0.00005550000000000001, 0.00005550000000000001, 0.00016827375000000002,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.00015865403437500001, 0.00008100000000000002, 0.00013293393750000003,
        0.00003000000000000004, 0.00003000000000000004, 0.000511693358096836,
        0.00003000000000000004, 0.00003000000000000004, 0.0008382409215566016,
        0.00003000000000000004, 0.00011402250000000001, 0.00005550000000000001,
        0.00003000000000000004, 0.00014277375000000002, 0.000590603701889238,
        0.00011402250000000001, 0.00003000000000000004, 0.0003913641587121094,
        0.00003000000000000004, 0.00003000000000000004, 0.00012435000000000001,
        0.00003000000000000004, 0.0005719241044744334, 0.00005550000000000001,
        0.00003000000000000004, 0.00009559875000000003, 0.00003000000000000004,
        0.00018612375, 0.0004199043139756054 7, 0.000128175, 0.00005550000000000001,
        0.00003000000000000004, 0.00008100000000000002, 0.00056602218509332,
        0.00008100000000000002, 0.00003000000000000004, 0.00003000000000000004,
        0.00014919574171875, 0.00005550000000000001,
```

```
0.000030000000000000004,
        0.000030000000000000004, 0.0007010716836625781,
0.000030000000000000004,
        0.00027271537500000005, 0.00009559875000000003,
0.000030000000000000004,
        0.00015851355328125002, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.0017785457481467384,
0.00035147874703125013,
        0.000030000000000000004, 0.000030000000000000004,
0.00012109875000000002,
        0.00017087074171875, 0.00005550000000000001,
0.00029875788750000006,
        0.00007717500000000002, 0.00005550000000000001,
0.00018287250000000003,
        0.00020730370734374997, 0.000030000000000000004,
0.00046078811633689466,
        0.00012435000000000001, 0.000030000000000000004,
0.00010267500000000002.
    )

  pageRanks14: expected = (
    expected,
        0.0011980866815906645, 0.0001897543125, 0.00007717500000000002,
        0.000030000000000000004, 0.00007717500000000002,
0.00024398593546875,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.00009559875000000003,
        0.00005550000000000001, 0.000030000000000000004,
0.0005523386141103321,
        0.000030000000000000004, 0.0011369696555575588,
0.00044302649175994153,
        0.00016225893750000003, 0.000030000000000000004,
0.00005550000000000001,
        0.00005550000000000001, 0.00009885000000000002,
0.000030000000000000004,
        0.00005550000000000001, 0.00010267500000000002,
0.00034647523371210935,
        0.00005550000000000001, 0.00005550000000000001,
0.00008100000000000002,
        0.000030000000000000004, 0.0006266011364933593,
0.0007460569827269042,
        0.000152419125, 0.0005574828435624999, 0.00045754291748888653,
        0.00005550000000000001, 0.00007717500000000002,
0.00012435000000000001,
        0.0006514320518563474, 0.0007781550062722654,
0.000030000000000000004,
        0.00011727375000000003, 0.000030000000000000004,
0.00016978143750000002,
        0.00005550000000000001, 0.00005550000000000001,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.00010267500000000002, 0.000030000000000000004,
0.00021112870734374997,
        0.000030000000000000004, 0.0007241425462692384,
0.000030000000000000004,
```

```
        0.00043975632939673826, 0.00026377172812500006, 0.0003192690972632813,
        0.00007717500000000002, 0.00005550000000000001, 0.001060316398582637,
        0.000030000000000000004, 0.00016827375, 0.000030000000000000004,
        0.00008100000000000002, 0.00010267500000000002, 0.000030000000000000004,
        0.0002944702747961329, 0.000030000000000000004, 0.00005550000000000001,
        0.000348277539984375, 0.0001546089375, 0.0016954224484599999,
        0.000030000000000000004, 0.00005550000000000001, 0.000180329034375.
  )

  pageRanks15: expected = (
    expected,
        0.00005550000000000001, 0.00024112459341996093, 0.000030000000000000004,
        0.00041604693152460934, 0.00007717500000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.0005496041375411911, 0.00019251787500000001,
        0.0005949961275392772, 0.0001551826875, 0.000254291625,
        0.00011727375000000003, 0.00007717500000000002, 0.00012109875000000002,
        0.00005550000000000001, 0.00015354144375, 0.00015007009687500002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00008100000000000002, 0.000030000000000000004, 0.00027656824171875004,
        0.00022955245734375, 0.00005550000000000001, 0.000030000000000000004,
        0.00032534906796093756, 0.00012457009687500003, 0.000215668846875,
        0.00013894875000000002, 0.000205829034375, 0.0002736906531408399,
        0.00010267500000000002, 0.00005550000000000001, 0.0007959996360539453,
        0.00005550000000000001, 0.00005550000000000001, 0.00012052500000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.00005550000000000001,
        0.000224252784375, 0.00005550000000000001, 0.00017087074171875,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.0003375011800228516, 0.000030000000000000004, 0.00005550000000000001,
        0.00010267500000000002, 0.000030000000000000004, 0.000030000000000000004.
  )

  pageRanks16: expected = (
    expected,
        0.000030000000000000004, 0.00014624509687500003, 0.00005550000000000001,
        0.00017152500000000004, 0.000030000000000000004, 0.0003609669341552931,
        0.00007717500000000002, 0.000030000000000000004, 0.00007717500000000002,
```

```
        0.00005550000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001, 0.0002042114773359375,
        0.0008721219106832425, 0.0006369682754531251, 0.0001356975,
        0.00005550000000000001, 0.00024838187461171876, 0.000713347958461465,
        0.00007717500000000002, 0.00017303268749999998, 0.00008100000000000002,
        0.0005128700965591992, 0.00019470905663185548, 0.00009885000000000002,
        0.00008100000000000002, 0.00026435153437500005, 0.000030000000000000004,
        0.00028910833249218753, 0.00005550000000000001, 0.00024367850573554694,
        0.00015843393750000003, 0.000030000000000000004, 0.0003614906134310743,
        0.000030000000000000004, 0.00035817465068654306, 0.00019702661074335936,
        0.000030000000000000004, 0.000030000000000000004, 0.00031682620734375006,
        0.000030000000000000004, 0.00009559875000000003, 0.0009339972378849416,
        0.000030000000000000004, 0.000030000000000000004, 0.00037407066244818355,
        0.000030000000000000004, 0.000030000000000000004, 0.00005550000000000001,
        0.00028621361671875, 0.000030000000000000004, 0.00017557009687500002,
        0.000030000000000000004, 0.00017598333234375, 0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.0005464359416179294,
        0.00012968268750000001, 0.000030000000000000004, 0.000030000000000000004,
        0.00009885000000000002, 0.000454723037081924, 0.000030000000000000004,
        0.00009885000000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.00026667045761835943, 0.0010615728509209819, 0.00043192417839472676,
        0.00014277375000000002, 0.0016763097454050194, 0.0004163827734375001,
        0.00017174509687500002, 0.00007717500000000002, 0.0005071352788982225,
        0.000030000000000000004, 0.0004724088400752538, 0.000030000000000000004,
        0.00005550000000000001, 0.00005550000000000001, 0.0013976675910372072,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.00010267500000000002, 0.000030000000000000004,
        0.00010267500000000002, 0.00016827375, 0.0009015184824459963.
  )

  pageRanks17: expected = (
    expected,
        0.00011402250000000001, 0.000030000000000000004, 0.00020620815124218746,
        0.0006429872559689648, 0.00009559875000000003,
```

```
0.000030000000000000004,
        0.000055500000000000001, 0.000030000000000000004,
0.00027610687500000006,
        0.000030000000000000004, 0.00030289569021933597,
0.0006080530205542576,
        0.00011727375000000003, 0.000030000000000000004,
0.000030000000000000004,
        0.0010484356591858599, 0.000030000000000000004,
0.0004212905427699219,
        0.00007717500000000002, 0.000030000000000000004,
0.00043143789891640633,
        0.00014659875000000002, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.0002032035485179687,
0.0007551751218261719,
        0.000030000000000000004, 0.000055500000000000001,
0.0003794698669933595,
        0.00014659875000000002, 0.00007717500000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.00011727375000000003,
0.000030000000000000004,
        0.000030000000000000004, 0.0004934760543513475,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000128175,
        0.000055500000000000001, 0.0008547118390340233,
0.000030000000000000004,
        0.0014346082973754881, 0.00021695643750000001,
0.000055500000000000001,
        0.0015211104302111527, 0.00011727375000000003,
0.000055500000000000001,
        0.0004392343555509962, 0.000030000000000000004,
0.000030000000000000004,
        0.00010267500000000002, 0.000055500000000000001,
0.0009416894236694923,
        0.00007717500000000002, 0.000055500000000000001,
0.00022873191724904294,
        0.0008242737845741408, 0.00025836458859375004,
0.00009559875000000003,
        0.000030000000000000004, 0.00031687600301942386,
0.000030000000000000004,
        0.000030000000000000004, 0.00007717500000000002,
0.000055500000000000001,
        0.00015843393750000003, 0.000390462103125,
0.000030000000000000004,
        0.000055500000000000001, 0.0004362053242109375,
0.00019035592921875,
        0.000030000000000000004, 0.0005769638100933201, 0.00006825,
        0.000055500000000000001, 0.000030000000000000004,
0.00022241513046093752,
        0.000030000000000000004, 0.000055500000000000001,
0.001006261613162637.
  )

  pageRanks18: expected = (
    expected,
        0.0007857088789003512, 0.000030000000000000004,
0.00025498721718750006,
        0.000055500000000000001, 0.0011432981682779491,
0.000030000000000000004,
        0.000030000000000000004, 0.00007717500000000002,
0.000030000000000000004,
```

720 0.00005550000000000001, 0.00007717500000000002,
0.0006757280227431446,
721 0.000600739772793574, 0.00014919574171875,
0.000030000000000000004,
722 0.0004217083662265626, 0.000030000000000000004,
0.0021483529248317776,
723 0.00005550000000000001, 0.0006842449449634961,
0.00005550000000000001,
724 0.000030000000000000004, 0.00013588458234375003,
0.000030000000000000004,
725 0.000030000000000000004, 0.00008100000000000002,
0.000030000000000000004,
726 0.0003539867951484376, 0.000030000000000000004,
0.000030000000000000004,
727 0.00005550000000000001, 0.00005550000000000001,
0.000030000000000000004,
728 0.000030000000000000004, 0.00009559875000000003,
0.0005256094171827733,
729 0.000030000000000000004, 0.00014985, 0.00007717500000000002,
730 0.00008100000000000002, 0.000030000000000000004,
0.000030000000000000004,
731 0.0006064650511017775, 0.000030000000000000004,
0.00005550000000000001,
732 0.000030000000000000004, 0.00023480038124999998,
0.000030000000000000004,
733 0.000386993224139668, 0.0007025938102384176,
0.00007717500000000002,
734 0.000030000000000000004, 0.000030000000000000004,
0.0008015986250978318,
735 0.0011856587198157474, 0.001220542597781563,
0.00005550000000000001,
736 0.001103845676174756, 0.000030000000000000004,
0.0005001217209945313,
737 0.000164668846875, 0.000030000000000000004,
0.00021766422187500002,
738 0.00009559875000000003, 0.0002708821616396681,
0.0008902189805322072,
739 0.0002728723524621094, 0.0001928969918671875,
0.000030000000000000004,
740 0.00005550000000000001, 0.00008100000000000002,
0.00012435000000000001,
741 0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
742 0.00020673125625000002, 0.000030000000000000004,
0.00009559875000000003,
743 0.00012109875000000002, 0.000030000000000000004,
0.0001795858324921875,
744 0.00014659875000000002, 0.000030000000000000004,
0.000030000000000000004,
745 0.000030000000000000004, 0.000030000000000000004,
0.0006738107530478319,
746 0.00012435000000000001, 0.000030000000000000004,
0.000030000000000000004,
747 0.00005550000000000001, 0.00005550000000000001,
0.0005546722821167576,
748 0.00013675893750000003, 0.00005550000000000001,
0.00007717500000000002,
749 0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
750 0.0007719215799593356, 0.00009559875000000003,
0.00021162375000000001,

```
        0.00005550000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004, 0.00012968268750000001.
  )

  pageRanks19: expected = (
    expected,
        0.00005550000000000001, 0.00027164788125, 0.00005550000000000001,
        0.0014103401638943945, 0.00015154476984375004, 0.0005737931960558594,
        0.0008436621533431052, 0.00012457009687500003, 0.00019145643750000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00010267500000000002, 0.000030000000000000004, 0.0002807485055871094,
        0.0005178079962573242, 0.000030000000000000004, 0.00042718429824335937,
        0.000030000000000000004, 0.000030000000000000004, 0.00009885000000000002,
        0.00008100000000000002, 0.00007717500000000002, 0.000030000000000000004,
        0.0009962234120309377, 0.000030000000000000004, 0.0002712076875,
        0.0007484966368463475, 0.000030000000000000004, 0.000030000000000000004,
        0.00021782324186718748, 0.0007975849074420704, 0.00005550000000000001,
        0.0014440971395321093, 0.0004298739782575976, 0.000030000000000000004,
        0.00016827375, 0.000030000000000000004, 0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.0010549562700050393,
        0.000030000000000000004, 0.00009559875000000003, 0.000030000000000000004,
        0.000126919125, 0.000183092596875, 0.00045689434595333993,
        0.000030000000000000004, 0.0006202840925404099, 0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004, 0.00005550000000000001,
        0.00041851013033476574, 0.0009673196422284764, 0.00011402250000000001,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.00024139500000000002, 0.00015881305436718754,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.0005228328479950585, 0.000030000000000000004,
        0.00021283083234375, 0.000030000000000000004, 0.0007436318898950389,
        0.0004801294072429297, 0.00010267500000000002, 0.000502228642298047,
        0.000030000000000000004, 0.000572949911904863, 0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001,
```

```
0.00007717500000000002,
        0.00005550000000000001, 0.00005550000000000001, 0.00003000000000000004,
        0.00003000000000000004, 0.000269925119280293, 0.0007819586751550391,
        0.00005550000000000001, 0.00003000000000000004, 0.0007033901825952341,
        0.00007717500000000002, 0.000165730284375, 0.00008100000000000002,
        0.0010263752493400197, 0.00003000000000000004, 0.00014299384687500004,
        0.00003000000000000004, 0.0002240326875, 0.00003000000000000004,
        0.00003000000000000004, 0.0001584339375, 0.00045708025931367194,
        0.00007717500000000002, 0.00047436583249218747, 0.00003000000000000004,
        0.0002730541168671876, 0.00003000000000000004, 0.00016225893750000003,
        0.00003000000000000004, 0.00040197340380373055, 0.00003000000000000004.
   )

   pageRanks20: expected = (
     expected,
        0.000261367875, 0.00003000000000000004, 0.00003000000000000004,
        0.00005550000000000001, 0.00013894875000000002, 0.00003000000000000004,
        0.0002513864773359376, 0.00005550000000000001, 0.00003000000000000004,
        0.00047612431834921886, 0.00007717500000000002, 0.00007717500000000002,
        0.00003000000000000004, 0.0006952168221796875, 0.00012109875000000002,
        0.00005550000000000001, 0.00007717500000000002, 0.00003000000000000004,
        0.001107573809079551, 0.00003000000000000004, 0.00007717500000000002,
        0.00003000000000000004, 0.00003000000000000004, 0.00005550000000000001,
        0.00018431305436718753, 0.00003000000000000004, 0.000165730284375,
        0.00030354806250000013, 0.00040802521613302745, 0.0007408627669822461,
        0.0003570662890816407, 0.00028091394375000006, 0.00005550000000000001,
        0.00003000000000000004, 0.00003000000000000004, 0.00007717500000000002,
        0.0001356975, 0.00043968573371853352, 0.00003000000000000004,
        0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
        0.00011125893750000003, 0.00003000000000000004, 0.00005550000000000001,
        0.00008100000000000002, 0.00003000000000000004, 0.00008100000000000002,
        0.0003847838695078125 7, 0.00003000000000000004, 0.0007459325985840232,
        0.0003567720534166603, 0.00003000000000000004, 0.00008100000000000002,
        0.0001356975, 0.00005550000000000001, 0.00003000000000000004,
        0.00003000000000000004, 0.00017798000625, 0.0014359547638245706,
        0.00009559875000000003, 0.00003000000000000004, 0.00003000000000000004,
        0.00046789477015955094, 0.00003000000000000004,
```

```
0.00010650000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.000226919852484375,
        0.00022297125, 0.000596758121936328, 0.000030000000000000004,
        0.0006366992817095507, 0.000030000000000000004,
0.000030000000000000004,
        0.00010267500000000002, 0.00007717500000000002, 0.00017196519375,
        0.00007717500000000002, 0.000030000000000000004, 0.0002011018125,
        0.000030000000000000004, 0.0002167318951183594,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00012109875000000002,
        0.0008290133266141014, 0.000030000000000000004, 0.000183580284375,
        0.00023110893750000005, 0.00005550000000000001,
0.000030000000000000004,
        0.00028930645988302735, 0.0001649910962121094,
0.00005550000000000001,
        0.000030000000000000004, 0.0003763601051910157,
0.00010267500000000002,
        0.000030000000000000004, 0.0001573725, 0.00007717500000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.00007717500000000002,
        0.000030000000000000004, 0.00007717500000000002,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.0004523801286034764, 0.00005550000000000001,
0.000030000000000000004,
        0.0004951815598534765, 0.00008100000000000002,
0.00010267500000000002.
  )

  pageRanks21: expected = (
    expected,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.00010267500000000002,
0.00009559875000000003,
        0.000030000000000000004, 0.001175440344576426,
0.000030000000000000004,
        0.000030000000000000004, 0.00035443770990195323,
0.000030000000000000004,
        0.0003747032963148926, 0.00005550000000000001,
0.0005116082614710938,
        0.000030000000000000004, 0.000030000000000000004,
0.00021331851984375,
        0.00007717500000000002, 0.0008077201393309957,
0.000030000000000000004,
        0.000030000000000000004, 0.000994534956437754,
0.000030000000000000004,
        0.00011727375000000003, 0.000030000000000000004,
0.0009117984778980079,
        0.00005550000000000001, 0.00007717500000000002,
0.00009559875000000003,
        0.000030000000000000004, 0.000030000000000000004,
0.00036226694577816426,
        0.000030000000000000004, 0.000030000000000000004,
0.00032663461605585945,
        0.000030000000000000004, 0.000030000000000000004,
```

```
0.000030000000000000004,
        0.000055500000000000001, 0.000030000000000000004, 0.00009559875000000003,
        0.0006578930123802537, 0.00013293393750000003, 0.000030000000000000004,
        0.000030000000000000004, 0.000146025, 0.0006895069584232615,
        0.00012968268750000001, 0.000030000000000000004, 0.00007717500000000002,
        0.00034956950625, 0.000055500000000000001, 0.00007717500000000002,
        0.000030000000000000004, 0.00047335328654826193, 0.000030000000000000004,
        0.000055500000000000001, 0.0003485537326737892, 0.000055500000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.00020124834327152344,
        0.000030000000000000004, 0.0005462430679701755, 0.0010715045568435355,
        0.00007717500000000002, 0.0001356975, 0.00008100000000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00009559875000000003, 0.000055500000000000001, 0.001050337302045796,
        0.0005798148140722265, 0.000030000000000000004, 0.000055500000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.000055500000000000001,
        0.00008100000000000002, 0.000030000000000000004, 0.0005860187924074219,
        0.000695229028248086, 0.00036895237500000005, 0.000055500000000000001,
        0.00009559875000000003, 0.000030000000000000004, 0.00010267500000000002,
        0.00022698333234375003, 0.000030000000000000004, 0.00018994875000000002,
        0.00030363530370429697, 0.000055500000000000001, 0.000030000000000000004.
  )

  pageRanks22: expected = (
    expected,
        0.000030000000000000004, 0.0012613613445382034, 0.0009061000272912988,
        0.000030000000000000004, 0.000030000000000000004, 0.00044870342936718753,
        0.000030000000000000004, 0.000030000000000000004, 0.000055500000000000001,
        0.000055500000000000001, 0.000030000000000000004, 0.00030216397748437506,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.0003228767387781642, 0.00009885000000000002, 0.00016338125625,
        0.000030000000000000004, 0.00007717500000000002, 0.000030000000000000004,
        0.000055500000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.0005762014488482618, 0.000055500000000000001,
        0.000252516346875, 0.000030000000000000004, 0.00007717500000000002,
```

```
        0.00043890087460225595, 0.000030000000000000004,
0.000030000000000000004,
        0.00008100000000000002, 0.00016701787500000002,
0.00007717500000000002,
        0.000030000000000000004, 0.0009819741968800392,
0.00047788861671875,
        0.00005550000000000001, 0.00008100000000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.001424334953569922,
0.000030000000000000004,
        0.000030000000000000004, 0.0003072850223830275,
0.00019267689499218754,
        0.0002809656928125, 0.000174094125, 0.0001650225,
0.00007717500000000002,
        0.000030000000000000004, 0.0011707606836439263,
0.000030000000000000004,
        0.00008100000000000002, 0.00014277375000000002,
0.0011823230263219532,
        0.000030000000000000004, 0.000030000000000000004,
0.00008100000000000002,
        0.000030000000000000004, 0.00013244625000000002,
0.0006514792329670507,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.0003684399653628907.
   )

   pageRanks23: expected = (
     expected,
        0.000030000000000000004, 0.00038739278765625003,
0.000030000000000000004,
        0.00022297125000000003, 0.00028660922160039077,
0.00022708709554921871,
        0.00005550000000000001, 0.000030000000000000004,
0.0006567602159249608,
        0.00005550000000000001, 0.00005550000000000001,
0.0005183511222142771,
        0.00201874207261627, 0.00037233539805243184,
0.00005550000000000001,
        0.0004911306645736522, 0.000030000000000000004,
0.000030000000000000004,
        0.0005789639421744336, 0.000030000000000000004,
0.000030000000000000004,
        0.00023881851984375, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000258336721875,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.0004664699360l609374,
        0.00005550000000000001, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.00015865403437500001, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
```

```
0.00017174509687500002,
        0.000055500000000000001, 0.00019164351984375002, 0.000030000000000000004,
        0.000030000000000000004, 0.0004902321882890625, 0.00011402250000000001,
        0.00012968268750000001, 0.000030000000000000004, 0.000030000000000000004,
        0.00021589500000000002, 0.000030000000000000004, 0.000055500000000000001,
        0.0010836219592849413, 0.0010891681019760942, 0.000030000000000000004,
        0.00012435000000000001, 0.000055500000000000001, 0.0007079486145866796,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004.
  )

  pageRanks24: expected = (
    expected,
        0.000077175000000000002, 0.0003419444918671875, 0.000030000000000000004,
        0.000055500000000000001, 0.001505804961977207, 0.000030000000000000004,
        0.000081000000000000002, 0.00032973896826562506, 0.00013675893750000003,
        0.000077175000000000002, 0.0007428987645248242, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000095598750000000003, 0.000030000000000000004, 0.0004649452599635351,
        0.000128175, 0.000030000000000000004, 0.0012289260041186134,
        0.0005547659253248241, 0.0012735475711920707, 0.000030000000000000004,
        0.000030000000000000004, 0.0006304702346261715, 0.000055500000000000001,
        0.000030000000000000004, 0.00010267500000000002, 0.0006962683298203127,
        0.0015407160072108007, 0.000030000000000000004, 0.000055500000000000001,
        0.000030000000000000004, 0.000077175000000000002, 0.000055500000000000001,
        0.000575346237916621, 0.0008075947471042188, 0.000055500000000000001,
        0.000095598750000000003, 0.000030000000000000004, 0.0001513576875,
        0.000077175000000000002, 0.00016444875, 0.000030000000000000004,
        0.000030000000000000004, 0.00012109875000000002, 0.000030000000000000004,
        0.0012133833776340236, 0.00016701787500000002, 0.0006743516353132419,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00028358187446328134, 0.0005657072281472655, 0.000128175,
        0.000055500000000000001, 0.000030000000000000004, 0.00010267500000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000055500000000000001, 0.000030000000000000004, 0.000077175000000000002,
        0.000055500000000000001, 0.00010267500000000002, 0.00010267500000000002,
```

```
        0.0005714230341351562, 0.0003404881216429688, 0.0005203748888714452,
        0.000030000000000000004, 0.00039713555163551777, 0.0003546995317031251,
        0.0004783518928479491, 0.000030000000000000004, 0.00012052500000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.00044235231887617196, 0.000030000000000000004,
        0.00016444875, 0.00019609428398671873, 0.000126919125,
        0.000030000000000000004, 0.000030000000000000004, 0.00007717500000000002,
        0.00005550000000000001, 0.00005550000000000001, 0.000030000000000000004,
        0.00008100000000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.0001677, 0.000030000000000000004, 0.00005550000000000001.
  )

  pageRanks25: expected = (
    expected,
        0.000030000000000000004, 0.000030000000000000004, 0.00034297342150898445,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.00007717500000000002, 0.00016792009687500002,
        0.000030000000000000004, 0.000030000000000000004, 0.00010267500000000002,
        0.000128175, 0.00040291446016097664, 0.00007717500000000002,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.00007717500000000002, 0.00037065834674240237, 0.000030000000000000004,
        0.0006004148802589451, 0.000030000000000000004, 0.00005550000000000001,
        0.00010267500000000002, 0.00019016884687500003, 0.000030000000000000004,
        0.00007717500000000002, 0.00008100000000000002, 0.0006253264456486521,
        0.00005550000000000001, 0.000030000000000000004, 0.0010081750473936526,
        0.0001584339375, 0.00015007009687500002, 0.000425316405686543,
        0.00032775693480585947, 0.00016792009687500002, 0.00005550000000000001,
        0.000030000000000000004, 0.000770613605453125, 0.000030000000000000004,
        0.000030000000000000004, 0.0010328207671150588, 0.0001801089375,
        0.000030000000000000004, 0.00005550000000000001, 0.0005880984259152342,
        0.00009885000000000002, 0.00005550000000000001, 0.0002693696017992188,
        0.000030000000000000004, 0.000030000000000000004, 0.0005425831398122849,
        0.000030000000000000004, 0.00028088092921875, 0.00010267500000000002,
        0.0006764355701484372, 0.00008100000000000002, 0.0003437430909375001,
        0.0009308938220033006, 0.0009178192984247847, 0.0007719799011481251,
```

```
        0.00007717500000000002, 0.00013675893750000003, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00009885000000000002,
        0.00014810643750000002, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.0006161561722175781,
        0.002341625584141152, 0.0002463233892566407, 0.0014496255159246877,
        0.00039820903892660153, 0.000030000000000000004, 0.00007717500000000002,
        0.000055500000000000001, 0.0004494569376072462, 0.000055500000000000001.
  )

  pageRanks26: expected = (
    expected,
        0.0007375958831105469, 0.000055500000000000001, 0.000030000000000000004,
        0.00019691513046093753, 0.000030000000000000004, 0.0006960112701089452,
        0.000030000000000000004, 0.0001356975, 0.00010267500000000002,
        0.00012457009687500003, 0.00009559875000000003, 0.00028669488604804693,
        0.0005773723220977929, 0.0025455574471148447, 0.000055500000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.0008289915908865819,
        0.00012435000000000001, 0.000254291625, 0.000030000000000000004,
        0.000030000000000000004, 0.0003034902985243946, 0.00031878986671875003,
        0.0006200913580648829, 0.00089866348489084, 0.000030000000000000004,
        0.00024847125000000003, 0.000030000000000000004, 0.00044653863884943365,
        0.0009588353839884179, 0.000030000000000000004, 0.000055500000000000001,
        0.0004802001794933595, 0.000030000000000000004, 0.00007717500000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.0005993897213728708, 0.000030000000000000004, 0.000030000000000000004,
        0.00012435000000000001, 0.0006565863834296874, 0.0004414859974607228,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00010267500000000002, 0.000030000000000000004, 0.00011125893750000003,
        0.000030000000000000004, 0.000030000000000000004, 0.00046621688976466807,
        0.000030000000000000004, 0.00007717500000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.0003956887727812501, 0.000187405284375, 0.00008100000000000002,
```

```
        0.00008100000000000002, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000322784383514668,
0.00005550000000000001,
        0.00013588458234375003, 0.00005550000000000001,
0.000030000000000000004,
        0.00013293393750000003, 0.00012109875000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.0002027230162402734,
0.00007717500000000002,
        0.000030000000000000004, 0.00005550000000000001,
0.00011727375000000003,
        0.000030000000000000004, 0.0002919278240625,
0.000030000000000000004,
        0.000030000000000000004, 0.00007717500000000002,
0.000030000000000000004.
   )

   pageRanks27: expected = (
     expected,
        0.00005550000000000001, 0.0011767052391037309,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.00005550000000000001, 0.00007717500000000002,
0.00007717500000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00015361125,
        0.0017282723083981836, 0.00005550000000000001,
0.000030000000000000004,
        0.0007795432040711617, 0.00007717500000000002,
0.00008100000000000002,
        0.00014985, 0.00005550000000000001, 0.000030000000000000004,
        0.0003018932018828127, 0.00035541894550021494,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.0007141020085688866, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00011125893750000003,
        0.001132409535121133, 0.00024089480953125002,
0.00022514327545312499,
        0.000030000000000000004, 0.00005550000000000001,
0.0004311059424464845,
        0.000128175, 0.00019035592921875, 0.000030000000000000004,
0.0001768576875,
        0.0009238328865187112, 0.000030000000000000004,
0.00005550000000000001,
        0.00008100000000000002, 0.00012435000000000001,
0.00012109875000000002,
        0.00005550000000000001, 0.000030000000000000004,
0.0001965018949921875,
        0.00007717500000000002, 0.00005550000000000001,
```

```
0.000340432358860547,
        0.000030000000000000004, 0.00009885000000000002,
0.00009885000000000002,
        0.000030000000000000004, 0.00008100000000000002,
0.0006406874962545115,
        0.00047452478709895517, 0.0010634564139578517,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000164668846875,
        0.00007717500000000002, 0.00008100000000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001,
0.00014076403125000002,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004.
  )

  pageRanks28: expected = (
    expected,
        0.000030000000000000004, 0.00031425467936718754,
0.000030000000000000004,
        0.000030000000000000004, 0.00013244625000000002,
0.00011125893750000003,
        0.00019459416468749997, 0.001184899122552129,
0.0006817205711949704,
        0.00005550000000000001, 0.00007717500000000002,
0.00005550000000000001,
        0.0005271594965196095, 0.0010400870586408399,
0.0004752608268892381,
        0.000030000000000000004, 0.00020129625, 0.000030000000000000004,
        0.00064671966671541, 0.0002982153750000001,
0.000030000000000000004,
        0.000030000000000000004, 0.0003997243869443556,
0.00048734821359374986,
        0.0004759093776406251, 0.00011727375000000003,
0.000030000000000000004,
        0.0005440875227589844, 0.00005550000000000001,
0.00019582394999999997,
        0.00037308581183710947, 0.00005550000000000001,
0.00005550000000000001,
        0.00008100000000000002, 0.00005550000000000001,
0.000030000000000000004,
        0.0004962196445816407, 0.00011727375000000003, 0.000164668846875,
        0.00005550000000000001, 0.00013588458234375003,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.00012109875000000002,
0.00005550000000000001,
        0.000030000000000000004, 0.00007717500000000002,
0.000030000000000000004,
        0.00020380225558710936, 0.00010267500000000002,
0.000030000000000000004,
        0.00009559875000000003, 0.000030000000000000004,
0.00022728393750000005,
        0.00011402250000000001, 0.000030000000000000004,
0.0006183836193906247,
        0.0001730326875, 0.00008100000000000002, 0.0012089610287725784,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.0010155726653473343, 0.000030000000000000004,
```

```
0.000030000000000000004,
        0.00016119750000000003, 0.000030000000000000004, 0.000030000000000000004,
        0.000055500000000000001, 0.000055500000000000001, 0.000192737971875,
        0.000030000000000000004, 0.0006722068841278708, 0.0010718436839313866,
        0.000030000000000000004, 0.001128313168464766, 0.002193723830047618,
        0.00022640958234375002, 0.000030000000000000004, 0.00025480590886054686,
        0.0008077473013638865, 0.000030000000000000004, 0.000055500000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.000081000000000000002,
        0.0005865207417187499, 0.000030000000000000004, 0.00017579625,
        0.000081000000000000002, 0.000030000000000000004, 0.000055500000000000001,
        0.00022315833234375003, 0.00014277375000000002, 0.000055500000000000001,
        0.00010267500000000002, 0.0003722072014756055, 0.000030000000000000004,
        0.000030000000000000004, 0.00017598333234375, 0.0006452517980099412,
        0.000174094125, 0.00024649398358593755, 0.000030000000000000004,
        0.00011125893750000003, 0.00044401822041662117, 0.000030000000000000004.
    )

    pageRanks29: expected = (
      expected,
        0.00012109875000000002, 0.000081000000000000002, 0.0004345341423233984,
        0.0016198246023303518, 0.000077175000000000002, 0.00024390100558710937,
        0.00023174226984375002, 0.00016827375000000002, 0.000030000000000000004,
        0.00023969287499999998, 0.000030000000000000004, 0.00029862255421875006,
        0.0002884303875000001, 0.000030000000000000004, 0.000030000000000000004,
        0.000055500000000000001, 0.0012822416701354098, 0.00011727375000000003,
        0.00019875278437500002, 0.000055500000000000001, 0.000030000000000000004,
        0.000055500000000000001, 0.00013293393750000003, 0.000030000000000000004,
        0.000030000000000000004, 0.000231320375, 0.00011727375000000003,
        0.000030000000000000004, 0.00013293393750000003, 0.000030000000000000004,
        0.000077175000000000002, 0.000030000000000000004, 0.0006375577699699217,
        0.00013293393750000003, 0.0005119848858394532, 0.00012968268750000001,
        0.000055500000000000001, 0.000098850000000000002, 0.0001974726191318554 5,
        0.000055500000000000001, 0.000030000000000000004, 0.000055500000000000001,
        0.0002687487898359375, 0.000030000000000000004, 0.000055500000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.000055500000000000001,
```

```
        0.000055500000000000001, 0.000077175000000000002,
0.000055500000000000001,
        0.000055500000000000001, 0.00010267500000000002,
0.00012052500000000002,
        0.00018287250000000003, 0.000030000000000000004,
0.00027437842921875004,
        0.00025469880421875006, 0.00021870000000000003,
0.001054292033974219,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000081000000000000002, 0.000055500000000000001,
0.0002522608324921875,
        0.00017904144375, 0.0000880125, 0.000081000000000000002,
        0.0008665907698721879, 0.000095598750000000003,
0.00011727375000000003,
        0.000055500000000000001, 0.000055500000000000001,
0.000195405039984375,
        0.000030000000000000004, 0.00015154476984375004,
0.000030000000000000004,
        0.000055500000000000001, 0.000055500000000000001,
0.000030000000000000004,
        0.000055500000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.000095598750000000003, 0.000030000000000000004,
0.0004193817696888867,
        0.000030000000000000004, 0.000055500000000000001,
0.000030000000000000004,
        0.000077175000000000002, 0.00020620815124218752,
0.00011402250000000001,
        0.00034031658501562504, 0.000081000000000000002,
0.00016225893750000003,
        0.000030000000000000004, 0.00016444875000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001,
0.000030000000000000004,
        0.00012457009687500003, 0.000030000000000000004,
0.000030000000000000004,
        0.0013239555484927151, 0.000030000000000000004,
0.000055500000000000001,
        0.000030000000000000004, 0.00032496215883828l4,
0.000030000000000000004,
        0.000030000000000000004, 0.002044314609656622,
0.000030000000000000004.
   )

   pageRanks30: expected = (
     expected,
        0.000030000000000000004, 0.000030000000000000004,
0.0002445528608917969,
        0.000030000000000000004, 0.0005265423747665818, 0.000140230284375,
        0.000055500000000000001, 0.00011125893750000003,
0.000030000000000000004,
        0.000095598750000000003, 0.000030000000000000004,
0.00035159226984375006,
        0.000098850000000000002, 0.000030000000000000004,
0.000030000000000000004,
        0.00012435000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.0006776933182608983, 0.000030000000000000004,
```

```
0.00003000000000000004,
        0.00003000000000000004, 0.00011402250000000001, 0.00003000000000000004,
        0.00011125893750000003, 0.00022071992883046875, 0.00003000000000000004,
        0.00003000000000000004, 0.00015412125000000001, 0.00003000000000000004,
        0.00003000000000000004, 0.00003000000000000004, 0.00005550000000000001,
        0.00005550000000000001, 0.0003944472271875, 0.00014277375000000002,
        0.00003000000000000004, 0.0003893728637121095, 0.00005550000000000001,
        0.0005825809400868945, 0.00003000000000000004, 0.00005550000000000001,
        0.00019546851984375, 0.00003000000000000004, 0.00005550000000000001,
        0.00003000000000000004, 0.00022476706823554686, 0.00009559875000000003,
        0.00007717500000000002, 0.00003000000000000004, 0.0004692964517755665,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00008100000000000002, 0.0006707026050272654,
        0.00003000000000000004, 0.000128175, 0.0010464823826843753,
        0.00013293393750000003, 0.00005550000000000001, 0.00009885000000000002,
        0.00003000000000000004, 0.00003000000000000004, 0.00005550000000000001,
        0.00005550000000000001, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
        0.00009885000000000002, 0.00009559875000000003, 0.00011125893750000003,
        0.00005550000000000001, 0.0011688491673253515, 0.0003834167234990431 5,
        0.00033088877587617195, 0.00003000000000000004, 0.00040050305411792956,
        0.0003604310280146681, 0.00016827375000000002, 0.00005550000000000001,
        0.00005550000000000001, 0.00005550000000000001, 0.00005550000000000001,
        0.00003000000000000004, 0.00009559875000000003, 0.00003000000000000004.
   )

   pageRanks31: expected = (
     expected,
        0.00060983427609375, 0.00003000000000000004, 0.00005550000000000001,
        0.00014985, 0.00003000000000000004, 0.00048470548947291976,
        0.00003000000000000004, 0.00005550000000000001, 0.00016827375,
        0.00007717500000000002, 0.00005550000000000001, 0.00020129625,
        0.00008100000000000002, 0.00018820518749999997, 0.00022042778437499996,
        0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
        0.00005550000000000001, 0.00007717500000000002, 0.0007755538548972265,
```

```
        0.00003000000000000004, 0.0006086423521652343, 0.00010267500000000002,
        0.0006501657417822068, 0.0002758531021875, 0.00005550000000000001,
        0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
        0.0009579667596042187, 0.00003000000000000004, 0.00007717500000000002,
        0.00003000000000000004, 0.00003000000000000004, 0.00008100000000000002,
        0.00021845292093749998, 0.00040177039710841806, 0.00043681785095626963,
        0.0003784534574986133, 0.00003000000000000004, 0.00026525375625,
        0.00003000000000000004, 0.0002016704146875, 0.0016676698321384962,
        0.0016026092253459765, 0.00005550000000000001, 0.00008100000000000002,
        0.00005550000000000001, 0.00011125893750000003, 0.00003000000000000004,
        0.00003000000000000004, 0.000268820838890625, 0.00003000000000000004,
        0.00005550000000000001, 0.00005550000000000001, 0.00040044269215550797,
        0.00013788125625000002, 0.0010699961603220117, 0.00003000000000000004,
        0.000146025, 0.00008100000000000002, 0.0001801089375, 0.000927249592168359,
        0.00025420596890625007, 0.00030216799120429697, 0.0003564819935984766,
        0.00003000000000000004, 0.00003000000000000004, 0.0005913996994322457,
        0.00003000000000000004, 0.00003000000000000004, 0.0003126232639756O547,
        0.0008478974921164453, 0.00005550000000000001, 0.00003000000000000004,
        0.00024811154062500003, 0.00003000000000000004, 0.00005550000000000001,
        0.00009559875000000003, 0.00003000000000000004, 0.00005550000000000001,
        0.00003000000000000004, 0.00003000000000000004, 0.00007717500000000002,
        0.00008100000000000002, 0.00046281358580802727, 0.00005550000000000001,
        0.00003000000000000004, 0.00003000000000000004, 0.0004620340538085939,
        0.00003000000000000004, 0.00008100000000000002, 0.00010267500000000002,
        0.00003000000000000004, 0.0004924743943451367, 0.00003000000000000004,
        0.00003000000000000004, 0.00003000000000000004, 0.0006910466454180467,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.0008784156453262499, 0.00003000000000000004.
  )

  pageRanks32: expected = (
    expected,
        0.00009559875000000003, 0.00005550000000000001, 0.00003000000000000004,
        0.0003290759292187501, 0.00007717500000000002,
```

0.00003000000000000004,
1231 0.00009885000000000002, 0.0002467391346093751,
0.000770629669361074,
1232 0.00003000000000000004, 0.00005550000000000001,
0.0003563719741396804,
1233 0.0006510181400392773, 0.0001932, 0.00003000000000000004,
1234 0.00013293393750000003, 0.00012109875000000002,
0.00003000000000000004,
1235 0.00003000000000000004, 0.0005881109660193553,
0.0006697311012689257,
1236 0.00003000000000000004, 0.00003000000000000004,
0.0019048124731323435,
1237 0.00003000000000000004, 0.0012570390195532622,
0.000817076816654082,
1238 0.00003000000000000004, 0.00003000000000000004,
0.00005550000000000001,
1239 0.00037593731782833994, 0.0003176617982433595,
0.00005550000000000001,
1240 0.00005550000000000001, 0.0009558905795487891, 0.0001356975,
1241 0.00007717500000000002, 0.002363161256569453,
0.00003000000000000004,
1242 0.00008100000000000002, 0.0003379288264756056,
0.00003000000000000004,
1243 0.00003000000000000004, 0.00003000000000000004,
0.00005550000000000001,
1244 0.00010267500000000002, 0.00003000000000000004,
0.0002601929146875,
1245 0.0002584977130570313, 0.00003000000000000004,
0.00003000000000000004,
1246 0.00003000000000000004, 0.0006118566301355858,
0.00013675893750000003,
1247 0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
1248 0.00003000000000000004, 0.00019853268750000003,
0.00003000000000000004,
1249 0.00005550000000000001, 0.00005550000000000001,
0.00003000000000000004,
1250 0.00008100000000000002, 0.00008100000000000002,
0.00003000000000000004,
1251 0.00005550000000000001, 0.00005550000000000001,
0.00003000000000000004,
1252 0.00003000000000000004, 0.00005550000000000001,
0.00003000000000000004,
1253 0.00007717500000000002, 0.00003000000000000004,
0.00003000000000000004,
1254 0.00003000000000000004, 0.00005550000000000001,
0.00011125893750000003,
1255 0.00007717500000000002, 0.00017174509687500002,
0.00003000000000000004,
1256 0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
1257 0.00003000000000000004, 0.00015843393750000003,
0.00005550000000000001,
1258 0.00003000000000000004, 0.00012968268750000001,
0.00005550000000000001,
1259 0.00007717500000000002, 0.0003116525398359375 5,
0.0004989153507442577,
1260 0.0002587445625, 0.00005550000000000001, 0.00038396702839960947,
1261 0.00045063995922013693, 0.0009222551727412307,
0.00003000000000000004,
1262 0.00005550000000000001, 0.00012457009687500003,

```
0.00003000000000000004,
        0.00008100000000000002, 0.00008100000000000002, 0.00003000000000000004,
        0.00014330527412167969, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00047010862846427726, 0.00035199944906250005,
        0.0003710352612187501, 0.00011727375000000003, 0.0012424059491165434.
  )

  pageRanks33: expected = (
    expected,
        0.00003000000000000004, 0.00007717500000000002, 0.00041257311675994l5,
        0.00008100000000000002, 0.0002748134751562501, 0.00003000000000000004,
        0.00003000000000000004, 0.00023476058810156252, 0.00003000000000000004,
        0.00005550000000000001, 0.00003000000000000004, 0.00009559875000000003,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.000332562522531465, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
        0.00007717500000000002, 0.00003000000000000004, 0.00003000000000000004,
        0.00005550000000000001, 0.00003000000000000004, 0.00015354144375,
        0.00012109875000000002, 0.0002386314375, 0.00003000000000000004,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.00007717500000000002, 0.00003000000000000004, 0.00009885000000000002,
        0.00003000000000000004, 0.0009038323984436134, 0.00007717500000000002,
        0.00005550000000000001, 0.0009495415982433593, 0.0003980020569064649 5,
        0.00003000000000000004, 0.00009559875000000003, 0.00003000000000000004,
        0.00003000000000000004, 0.00003000000000000004, 0.00039053699136197274,
        0.00005550000000000001, 0.00012968268750000001, 0.00003000000000000004,
        0.0006424087503536033, 0.00011727375000000003, 0.0007974356282954882,
        0.00005550000000000001, 0.00013293393750000003, 0.00005550000000000001,
        0.00003000000000000004, 0.002784270476832207, 0.00007717500000000002,
        0.0010764892763057425, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.0002833907783996095, 0.0007036298547836328,
        0.00005550000000000001, 0.00046026382205075194, 0.0013872956926727733,
        0.00007717500000000002, 0.00009559875000000003, 0.00008100000000000002,
        0.00005550000000000001, 0.00007717500000000002, 0.00003000000000000004,
```

```
        0.0003395341507054688, 0.00003000000000000004, 0.0005002082528993947,
        0.00003000000000000004, 0.00007717500000000002, 0.0007778580435433202,
        0.00003000000000000004, 0.00005550000000000001, 0.0007483684404247458,
        0.00011727375000000003, 0.00009559875000000003, 0.00005550000000000001,
        0.00003000000000000004, 0.00003000000000000004, 0.00012109875000000002,
        0.0006657233743404492, 0.00003000000000000004, 0.000548018287406992,
        0.00042246069565507797, 0.001355053843909375, 0.0008567852761221873,
        0.00016119750000000003, 0.00016701787500000002, 0.00007717500000000002,
        0.00007717500000000002, 0.00020443397718749996, 0.00011402250000000001,
        0.00009559875000000003, 0.00014277375000000002, 0.00005550000000000001,
        0.00003000000000000004, 0.00009559875000000003, 0.00003000000000000004,
        0.00003000000000000004, 0.00005550000000000001, 0.00005550000000000001,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.00008100000000000002, 0.00003000000000000004, 0.00009559875000000003.
  )

  pageRanks34: expected = (
    expected,
        0.00003000000000000004, 0.00003000000000000004, 0.00005550000000000001,
        0.00017152500000000004, 0.00003000000000000004, 0.0005834868354039061,
        0.00003000000000000004, 0.00008100000000000002, 0.00005550000000000001,
        0.00003000000000000004, 0.00003000000000000004, 0.00028394420437500005,
        0.00010267500000000002, 0.00005550000000000001, 0.00003000000000000004,
        0.00003000000000000004, 0.0003260878243371094, 0.00005550000000000001,
        0.00003000000000000004, 0.00003000000000000004, 0.000128175,
        0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
        0.00003000000000000004, 0.00003000000000000004, 0.000192737971875,
        0.00009885000000000002, 0.0005593131134316015, 0.00007717500000000002,
        0.0003434709373674024, 0.00003000000000000004, 0.00003000000000000004,
        0.00033521453983593764, 0.00010267500000000002, 0.00011727375000000003,
        0.0010148410295049998, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00003000000000000004, 0.00007717500000000002,
```

```
        0.00009885000000000002, 0.000030000000000000004,
0.000030000000000000004,
        0.00019744738492187498, 0.00019875278437500002,
0.0001303106250000001,
        0.000030000000000000004, 0.00020403750000000002,
0.0005788974922679687,
        0.000030000000000000004, 0.000030000000000000004,
0.00017521644374999997,
        0.0003113064375, 0.000030000000000000004, 0.00007717500000000002,
        0.000030000000000000004, 0.00040593012538185554,
0.00015119241562500002,
        0.0020183828489022167, 0.000030000000000000004,
0.000030000000000000004,
        0.0010626563778712694, 0.00005550000000000001,
0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.00010267500000000002,
0.000030000000000000004,
        0.0010527559032889554, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.00011727375000000003, 0.00014985,
        0.000030000000000000004, 0.000030000000000000004,
0.0008434553828328515,
        0.0004977945566349413, 0.00005550000000000001,
0.0006722223484265624,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.0006747054431943555,
        0.000210941625, 0.0008963466727096389, 0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.0001605102271875, 0.00005550000000000001,
0.000030000000000000004,
        0.00009559875000000003, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.00023775102609375, 0.0001932,
        0.00005550000000000001, 0.000030000000000000004,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.00012109875000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.0013960562834876764,
        0.00009885000000000002, 0.000030000000000000004,
0.000030000000000000004,
        0.00007717500000000002, 0.00010267500000000002,
0.00005550000000000001.
  )

  pageRanks35: expected = (
    expected,
        0.00005550000000000001, 0.00009885000000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.00043246640566427745, 0.0002664178528851563,
0.000030000000000000004,
        0.000030000000000000004, 0.0005632257407151952,
```

```
0.00047896735528576165,
        0.000632508809124043, 0.00003000000000000004,
0.0016506434779794729,
        0.00024059509687500002, 0.00003000000000000004, 0.000192517875,
        0.00003000000000000004, 0.00003000000000000004,
0.00019853268750000003,
        0.00005550000000000001, 0.0012807454091226561,
0.0006797589345724413,
        0.00019668014138871093, 0.00038820125014843763,
0.00009885000000000002,
        0.00005550000000000001, 0.00018010893750000003,
0.0012861574391040038,
        0.00019382727609374998, 0.000803065772465078,
0.00003000000000000004,
        0.00003000000000000004, 0.00048038040350685555,
0.00003000000000000004,
        0.0004718426704453124, 0.00005550000000000001,
0.00010267500000000002,
        0.0004208408546006056, 0.0001828725, 0.001204916585841094,
        0.000687985769325869, 0.00021094162500000002, 0.00016827375,
        0.00008100000000000002, 0.00003000000000000004,
0.00011727375000000003,
        0.0003829687037402735, 0.00003000000000000004, 0.0002555475,
        0.00003000000000000004, 0.00003000000000000004,
0.00005550000000000001,
        0.00011402250000000001, 0.00027699795761835936,
0.00010267500000000002,
        0.00007717500000000002, 0.00003000000000000004,
0.00003000000000000004,
        0.00003000000000000004, 0.00012457009687500003,
0.00003000000000000004,
        0.000181390471875, 0.00007717500000000002, 0.00026633888046093747,
        0.00003000000000000004, 0.00003000000000000004,
0.0002201722699921875,
        0.00003000000000000004, 0.00003000000000000004,
0.00003000000000000004,
        0.00008100000000000002, 0.00005550000000000001,
0.00007717500000000002,
        0.00030823986686718754, 0.00007717500000000002,
0.00009559875000000003,
        0.00003000000000000004, 0.00007717500000000002,
0.00011125893750000003,
        0.00005550000000000001, 0.00021162375000000001,
0.00003000000000000004,
        0.00003000000000000004, 0.00010650000000000001,
0.00019303215886054686,
        0.00011125893750000003, 0.00003000000000000004,
0.00009885000000000002,
        0.00003000000000000004, 0.00005550000000000001,
0.00005550000000000001,
        0.00026469084375000003, 0.00003000000000000004,
0.00003000000000000004.
    )

    pageRanks36: expected = (
      expected,
        0.00005550000000000001, 0.00003000000000000004,
0.00019486807662404297,
        0.00003000000000000004, 0.00005550000000000001,
0.00005550000000000001,
        0.00123809428599271 5, 0.00005550000000000001,
```

```
0.000030000000000000004,
        0.00007717500000000002, 0.0009582389058128515,
0.000030000000000000004,
        0.00005550000000000001, 0.00009559875000000003,
0.000030000000000000004,
        0.0002495326874999997, 0.000030000000000000004,
0.000030000000000000004,
        0.00005550000000000001, 0.00010267500000000002,
0.000030000000000000004,
        0.00011125893750000003, 0.0001513576875, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00010267500000000002,
        0.00014257931250000002, 0.000030000000000000004,
0.001362380613464414,
        0.000030000000000000004, 0.00007717500000000002,
0.00023734384687500003,
        0.00031125545949474615, 0.00016887406781249998,
0.000030000000000000004,
        0.00017962125, 0.000030000000000000004, 0.00032175329238164074,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.001066234109780723, 0.0001395225,
        0.00010267500000000002, 0.00013293393750000003,
0.0004844613860603709,
        0.0002699217076183594, 0.000030000000000000004,
0.00009559875000000003,
        0.000030000000000000004, 0.000030000000000000004,
0.0005309502408943944,
        0.000030000000000000004, 0.000030000000000000004, 0.0001356975,
        0.00011402250000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.0010622049395273834,
0.00023004750000000002,
        0.000030000000000000004, 0.0002491371986964844,
0.000030000000000000004,
        0.00012435000000000001, 0.00005550000000000001,
0.00022138175531249998,
        0.00007717500000000002, 0.0017936928663086918,
0.00042966960194765636,
        0.000030000000000000004, 0.000030000000000000004,
0.0009841252989410156,
        0.000261941499592793, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.00007717500000000002,
0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.0001932, 0.000030000000000000004, 0.000030000000000000004.
    )

    pageRanks37: expected = (
      expected,
        0.00010267500000000002, 0.00043106509355255866,
0.0010437505899590234,
        0.0002792762970273438, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.00018047532249609375,
0.00005550000000000001,
        0.000030000000000000004, 0.00048641498056345683,
0.00024958621359375,
        0.0003643005830224805, 0.000030000000000000004,
```

```
0.000030000000000000004,
      0.0007504075031519922, 0.000030000000000000004, 0.0002680099635937501,
      0.000055500000000000001, 0.0003932948035558594, 0.000030000000000000004,
      0.000172545, 0.0003614907562500001, 0.000030000000000000004,
      0.00022164824186718748, 0.0006228411419443552, 0.000030000000000000004,
      0.000030000000000000004, 0.0005299478300855074, 0.000030000000000000004,
      0.000077175000000000002, 0.000055500000000000001, 0.000081000000000000002,
      0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
      0.00012968268750000001, 0.0003085340538527344, 0.00019853268750000003,
      0.000055500000000000001, 0.0009653476063435351, 0.000030000000000000004,
      0.0011345384625932814, 0.0008455575109275976, 0.000081000000000000002,
      0.000030000000000000004, 0.000030000000000000004, 0.000055500000000000001,
      0.000128175, 0.000055500000000000001, 0.000055500000000000001,
      0.000055500000000000001, 0.0004085092553125001, 0.00010267500000000002,
      0.000055500000000000001, 0.00019645035749999996, 0.000030000000000000004,
      0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
      0.0003811410199921876, 0.000077175000000000002, 0.0012464304642064259,
      0.00020837250000000002, 0.0007725661289884177, 0.0005878685803759568,
      0.00015865403437500001, 0.00011402250000000001, 0.00017303268749999998,
      0.0001828725, 0.000098850000000000002, 0.00012052500000000002,
      0.00010267500000000002, 0.0002678228812500001, 0.000030000000000000004,
      0.000030000000000000004, 0.000077175000000000002, 0.000030000000000000004,
      0.00023969287500000004, 0.0008607545337244333, 0.000055500000000000001,
      0.000030000000000000004, 0.000077175000000000002, 0.000030000000000000004,
      0.000030000000000000004, 0.000098850000000000002, 0.000055500000000000001,
      0.000030000000000000004, 0.00012109875000000002, 0.000055500000000000001,
      0.000077175000000000002, 0.000055500000000000001, 0.0002632662317968750 5,
      0.000030000000000000004, 0.000055500000000000001, 0.000055500000000000001,
      0.000030000000000000004, 0.0007833576583764941, 0.000030000000000000004,
      0.000030000000000000004, 0.000055500000000000001, 0.000055500000000000001.
  )

  pageRanks38: expected = (
    expected,
      0.000055500000000000001, 0.00010267500000000002,
```

0.0005110525401105469,
1468 0.00019871976984375, 0.000030000000000000004,
0.000030000000000000004,
1469 0.0003483386168671875, 0.0000555000000000001,
0.0000555000000000001,
1470 0.000124350000000000001, 0.000030000000000000004,
0.000030000000000000004,
1471 0.000030000000000000004, 0.000030000000000000004,
0.00042438202159986323,
1472 0.00019396244308710937, 0.000030000000000000004,
0.000030000000000000004,
1473 0.000030000000000000004, 0.00019698280387499996,
0.000030000000000000004,
1474 0.000077175000000000002, 0.00011125893750000003,
0.000030000000000000004,
1475 0.000081000000000000002, 0.000030000000000000004,
0.0000555000000000001,
1476 0.0000555000000000001, 0.00014299384687500004,
0.000030000000000000004,
1477 0.000030000000000000004, 0.000030000000000000004,
0.00047081371679001953,
1478 0.000030000000000000004, 0.0009089884096401953,
0.000030000000000000004,
1479 0.000224376386859375, 0.00011402250000000001,
0.0000555000000000001,
1480 0.0009667603782073827, 0.000030000000000000004,
0.00011125893750000003,
1481 0.000030000000000000004, 0.000030000000000000004,
0.0000555000000000001,
1482 0.000030000000000000004, 0.0000555000000000001,
0.001241030682654688,
1483 0.000030000000000000004, 0.0004823641678046876,
0.000095598750000000003,
1484 0.00021509509687500002, 0.00034904545746992195,
0.0008725268335975779,
1485 0.000030000000000000004, 0.0004102900824509962,
0.0002089040241216797,
1486 0.000030000000000000004, 0.0003740019745820314,
0.0012997076386943947,
1487 0.0000555000000000001, 0.00018994875000000002,
0.0006884943091336133,
1488 0.0000555000000000001, 0.00010267500000000002,
0.000124350000000000001,
1489 0.000030000000000000004, 0.0000555000000000001,
0.0011576753198899027,
1490 0.000030000000000000004, 0.000649733662027949,
0.000030000000000000004,
1491 0.000030000000000000004, 0.000492883398434199,
0.0007212162063932225,
1492 0.000128175, 0.000030000000000000004, 0.000030000000000000004,
1493 0.00036528014310937504, 0.0009957260430857229,
0.000030000000000000004,
1494 0.001329195085524004, 0.0006927975738414062,
0.000030000000000000004,
1495 0.000030000000000000004, 0.000030000000000000004,
0.000081000000000000002,
1496 0.000030000000000000004, 0.001100317863523008,
0.0000555000000000001,
1497 0.00010267500000000002, 0.000030000000000000004, 0.000146025,
1498 0.000081000000000000002, 0.000030000000000000004,
0.0006030464054154491.

```
)

pageRanks39: expected = (
  expected,
      0.00007717500000000002, 0.00014277375000000002, 0.00011727375000000003,
      0.000030000000000000004, 0.0008181163542995311, 0.000030000000000000004,
      0.00014277375000000002, 0.000030000000000000004, 0.000030000000000000004,
      0.0014100132311993948, 0.00010267500000000002, 0.00009559875000000003,
      0.0006551366364775194, 0.00007717500000000002, 0.00009559875000000003,
      0.00008100000000000002, 0.00010267500000000002, 0.00005550000000000001,
      0.0007480018415248242, 0.0005355432135937501, 0.00007717500000000002,
      0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
      0.00027271537500000005, 0.000030000000000000004, 0.000030000000000000004,
      0.00046871357410369156, 0.000030000000000000004, 0.00005550000000000001,
      0.000128175, 0.000030000000000000004, 0.00005550000000000001,
      0.000030000000000000004, 0.00009559875000000003, 0.00012968268750000001,
      0.00013675893750000003, 0.0006451623310560742, 0.000030000000000000004,
      0.00020930038125, 0.000030000000000000004, 0.000030000000000000004,
      0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
      0.00025474342050342774, 0.000030000000000000004, 0.000030000000000000004,
      0.0006523670799398827, 0.000030000000000000004, 0.00014919574171875,
      0.000030000000000000004, 0.00009559875000000003, 0.00009559875000000003,
      0.000204767596875, 0.000030000000000000004, 0.00009559875000000003,
      0.000030000000000000004, 0.00009885000000000002, 0.00007717500000000002,
      0.000030000000000000004, 0.00007717500000000002, 0.0004906871529635349,
      0.000030000000000000004, 0.00014277375000000002, 0.000030000000000000004,
      0.00030763942364748055, 0.000030000000000000004, 0.000030000000000000004,
      0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
      0.0001677, 0.00005550000000000001, 0.000030000000000000004,
      0.000030000000000000004, 0.000030000000000000004, 0.00005550000000000001,
      0.00007717500000000002, 0.00005550000000000001, 0.000030000000000000004,
      0.00008100000000000002, 0.00008100000000000002, 0.00014810643750000002,
      0.00008100000000000002, 0.000030000000000000004, 0.0004704629513872852,
      0.000205829034375, 0.00009885000000000002, 0.00019853268750000003,
```

```
        0.00012968268750000001, 0.000030000000000000004, 0.000030000000000000004,
        0.00013588458234375003, 0.000030000000000000004, 0.0007767642435570312,
        0.000030000000000000004, 0.0006168027873406638, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004.
  )

  pageRanks40: expected = (
    expected,
        0.000081000000000000002, 0.000030000000000000004, 0.00023329875,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000077175000000000002, 0.000030000000000000004,
        0.0005132322826406251, 0.0005522987924888865, 0.000098850000000000002,
        0.000098850000000000002, 0.0002266018125, 0.0015834187107174223,
        0.00033898444402460947, 0.0008324012817950196, 0.000030000000000000004,
        0.000180329034375, 0.0008978864941825587, 0.000055500000000000001,
        0.00014299384687500004, 0.0001801089375, 0.000055500000000000001,
        0.000030000000000000004, 0.0005692153571463084, 0.0005368253732539843,
        0.000055500000000000001, 0.000055500000000000001, 0.000030000000000000004,
        0.0006468409577320312, 0.000030000000000000004, 0.0003211811310781251,
        0.000030000000000000004, 0.0006582275594396678, 0.0007064921391304686,
        0.000030000000000000004, 0.000030000000000000004, 0.00010650000000000001,
        0.001123349208692012, 0.00013894875000000002, 0.00017174509687500002,
        0.000768254994630996, 0.000030000000000000004, 0.000077175000000000002,
        0.000030000000000000004, 0.001407102531453867, 0.000030000000000000004,
        0.0009750694000490038, 0.000055500000000000001, 0.000055500000000000001,
        0.00024731642339179686, 0.00015154476984375004, 0.000030000000000000004,
        0.00035790707901488293, 0.00022954649988632815, 0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
        0.00027896925010724617, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00037515426984375004,
        0.000030000000000000004, 0.00010267500000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.000077175000000000002, 0.000030000000000000004,
        0.0009195100252817186, 0.00010267500000000002, 0.0011678366105352931,
        0.0004596703178347658, 0.000030000000000000004, 0.000030000000000000004,
        0.0004559106924523828, 0.000030000000000000004,
```

```
0.0001823163804609375,
        0.000030000000000000004, 0.000030000000000000004, 0.001121086223968906,
        0.000030000000000000004, 0.00005550000000000001, 0.00004275000000000001,
        0.00007717500000000002, 0.00011125893750000003, 0.000030000000000000004,
        0.00007717500000000002, 0.000030000000000000004, 0.00014534287500000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004.
  )

  pageRanks41: expected = (
    expected,
        0.0017556940398932423, 0.00015007009687500002, 0.000030000000000000004,
        0.00011727375000000003, 0.000030000000000000004, 0.000030000000000000004,
        0.00005550000000000001, 0.00005550000000000001, 0.000030000000000000004,
        0.000168493846875, 0.000030000000000000004, 0.000030000000000000004,
        0.000636605465767539, 0.0006222834791867967, 0.000030000000000000004,
        0.00016225893750000003, 0.00020495467921875, 0.00019540503998437497,
        0.000030000000000000004, 0.00008100000000000002, 0.00009559875000000003,
        0.00008100000000000002, 0.000163766625, 0.0004007745255792969,
        0.00005550000000000001, 0.00005550000000000001, 0.00007717500000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.00023427967921875,
        0.00005550000000000001, 0.00005550000000000001, 0.00040445321618064444,
        0.0002949046588605469, 0.00052845754028125, 0.000030000000000000004,
        0.0004532404760482618, 0.00028161129623437505, 0.00023373894375000001,
        0.000030000000000000004, 0.00005550000000000001, 0.0012445951359051468,
        0.000030000000000000004, 0.000168493846875, 0.00057090017059375,
        0.000030000000000000004, 0.0005946650262327733, 0.00005550000000000001,
        0.00007717500000000002, 0.0001973778608917969, 0.0004028958548752149,
        0.00010267500000000002, 0.000030000000000000004, 0.00012109875000000002,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.0004724473638984376, 0.00007717500000000002, 0.00025497271615529296,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.0003436367774209181, 0.0006515472775947068, 0.000030000000000000004,
        0.0007925057115101563, 0.00005550000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00025811662500000007,
```

```
        0.000055500000000000001, 0.000030000000000000004, 0.0003093791396384961,
        0.000030000000000000004, 0.0004234529288368946, 0.0005038604170281249,
        0.00012109875000000002, 0.00010267500000000002, 0.0001784913804609375,
        0.0006189891913622458, 0.000055500000000000001, 0.000055500000000000001,
        0.000055500000000000001, 0.000030000000000000004, 0.000081000000000000002,
        0.00010650000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001, 0.0010110427299035549,
        0.000055500000000000001, 0.000077175000000000002, 0.0012826222617107622,
        0.0009419121841931444, 0.00010650000000000001, 0.000055500000000000001,
        0.00013293393750000003, 0.0005449953795705271, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000095598750000000003,
        0.000055500000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.000055500000000000001, 0.000098850000000000002, 0.0012708980017750392,
        0.000055500000000000001, 0.00020881269375, 0.0003530600866864356 3,
        0.000030000000000000004, 0.0005407651450046874, 0.0001356975.
  )

  pageRanks42: expected = (
    expected,
        0.000055500000000000001, 0.000055500000000000001, 0.00022511452490337893,
        0.000030000000000000004, 0.00021046892339179687, 0.00043287358488302733,
        0.000030000000000000004, 0.00019783027718359373, 0.000030000000000000004,
        0.0003161108721796875 6, 0.000055500000000000001, 0.0003520324635937501,
        0.00011125893750000003, 0.000030000000000000004, 0.000055500000000000001,
        0.0004311003574218751, 0.00010267500000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.0003175084025693555, 0.0011122127446247852,
        0.00012435000000000001, 0.000030000000000000004, 0.000030000000000000004,
        0.0009516672924987891, 0.0007951100024036912, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.0004405921018214845,
        0.0006293696852640232, 0.000030000000000000004, 0.000055500000000000001,
        0.000030000000000000004, 0.00014277375000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.000081000000000000002, 0.00011125893750000003,
        0.000221269125, 0.0002654379938572461, 0.000055500000000000001,
        0.00010650000000000001, 0.000030000000000000004, 0.00015154476984375004,
```

```
        0.00010267500000000002, 0.0003013663127651465,
0.000030000000000000004,
        0.00009559875000000003, 0.00005550000000000001,
0.00009885000000000002,
        0.000030000000000000004, 0.00028801585312500014,
0.00051896699171875,
        0.000030000000000000004, 0.000030000000000000004,
0.00008100000000000002,
        0.00005550000000000001, 0.00010267500000000002,
0.00007717500000000002,
        0.00027813865045312505, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.0002295803240625, 0.000591690735639453,
        0.00040644273358593757, 0.000030000000000000004,
0.00005550000000000001,
        0.0004418071279275587, 0.00005550000000000001,
0.0004835387763812108,
        0.001059485402328809, 0.000998035093129512,
0.00012052500000000002,
        0.00010267500000000002, 0.0007237203894777343,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.00010267500000000002,
        0.00005550000000000001, 0.00021695643750000001,
0.00007717500000000002,
        0.00012968268750000001, 0.00012109875000000002,
0.000030000000000000004,
        0.00005550000000000001, 0.0022970199819720605,
0.00005550000000000001,
        0.00047827658859375005, 0.00005550000000000001, 0.00006825,
        0.0005510104412360739, 0.0010187398686616798,
0.00011125893750000003,
        0.00005550000000000001, 0.000030000000000000004,
0.00008100000000000002,
        0.000030000000000000004, 0.00012968268750000001,
0.000030000000000000004,
        0.00005550000000000001, 0.00013293393750000003,
0.00017321976984375,
        0.00016225893750000003, 0.00019145643750000002,
0.00007717500000000002,
        0.00031651012275117196, 0.00014659875000000002,
0.00039644005107951184.
  )

  pageRanks43: expected = (
    expected,
        0.0002660132896875, 0.00005550000000000001,
0.00005550000000000001,
        0.000152419125, 0.0016891438247597268, 0.00012457009687500003,
        0.00016119750000000003, 0.00010267500000000002,
0.000030000000000000004,
        0.00010267500000000002, 0.000933136792756053,
0.000030000000000000004,
        0.00019528143750000004, 0.00009885000000000002,
0.00005550000000000001,
        0.0003461528180871094, 0.000030000000000000004,
0.00039424190589769544,
        0.00033577812302578136, 0.000030000000000000004,
0.00005550000000000001,
```

```
        0.00007717500000000002, 0.0002877893380792969, 0.00003000000000000004,
        0.00003000000000000004, 0.0002240326875, 0.00003000000000000004,
        0.00007717500000000002, 0.00003000000000000004, 0.00008100000000000002,
        0.00003000000000000004, 0.0005661583932553319, 0.00003000000000000004,
        0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
        0.0010921575166542774, 0.0014712415030663285, 0.00003000000000000004,
        0.00003000000000000004, 0.00023186717123437498, 0.0011153109768279107,
        0.00037087997203593765, 0.00005550000000000001, 0.00008100000000000002,
        0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00005550000000000001, 0.0005527790258255663,
        0.00008100000000000002, 0.00003000000000000004, 0.00003000000000000004,
        0.00003000000000000004, 0.00009559875000000003, 0.000128175,
        0.00005550000000000001, 0.00009559875000000003, 0.0003304736308681447,
        0.00003000000000000004, 0.00011125893750000003, 0.0018335073900723833,
        0.00003000000000000004, 0.00003000000000000004, 0.00007717500000000002,
        0.00003000000000000004, 0.00003000000000000004, 0.00005550000000000001,
        0.00007717500000000002, 0.00010267500000000002, 0.00003000000000000004,
        0.0001513576875, 0.00032466091776349617, 0.00003000000000000004,
        0.00003000000000000004, 0.00022403268749999998, 0.0006712320912878907,
        0.00024917903437500006, 0.00003000000000000004, 0.00025780846905468756,
        0.00003000000000000004, 0.0008138743728837696, 0.00003000000000000004,
        0.00003000000000000004, 0.0005576755996828122, 0.0006588155804808006,
        0.00007717500000000002, 0.0005866007615345508, 0.00005550000000000001,
        0.00029290500012617196, 0.0007023281972303903, 0.00003000000000000004,
        0.0015980741978907033, 0.00003000000000000004, 0.00007717500000000002,
        0.00005550000000000001, 0.00016827375000000002, 0.00003000000000000004,
        0.00010267500000000002, 0.00003000000000000004, 0.00003000000000000004.
  )

  pageRanks44: expected = (
    expected,
        0.00003000000000000004, 0.00012109875000000002, 0.00003000000000000004,
        0.00005550000000000001, 0.00003000000000000004, 0.0009396548305022267,
        0.00012109875000000002, 0.00003000000000000004, 0.00003000000000000004,
```

```
        0.000095598750000000003, 0.001814793173989063, 0.000030000000000000004,
        0.000095598750000000003, 0.00010267500000000002, 0.000030000000000000004,
        0.002758934391622441, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.0003839836037885841, 0.000030000000000000004,
        0.0008133136555286917, 0.000055500000000000001, 0.000077175000000000002,
        0.000030000000000000004, 0.00010267500000000002, 0.0003875349344324235,
        0.000030000000000000004, 0.00019609428398671875, 0.001827727958064629,
        0.0007766777549231443, 0.002192770461927129, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.0006964689514820309,
        0.000030000000000000004, 0.000095598750000000003, 0.0001356975,
        0.000789890542953125, 0.000055500000000000001, 0.000030000000000000004,
        0.00021036787500000001, 0.000030000000000000004, 0.00016444875000000002,
        0.000081000000000000002, 0.000055500000000000001, 0.00017146863914296875,
        0.000095598750000000003, 0.0004347056705968360б, 0.000209654034375,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.0002679983892566408, 0.000030000000000000004,
        0.000172545, 0.00012109875000000002, 0.00024865833234375003,
        0.000030000000000000004, 0.000077175000000000002, 0.000030000000000000004,
        0.000081000000000000002, 0.00015007009687500002, 0.000055500000000000001,
        0.0005220628684277733, 0.0007966927069299414, 0.00018691759687499999,
        0.0008192211742001366, 0.000030000000000000004, 0.0011461627584204495,
        0.00012435000000000001, 0.000055500000000000001, 0.000030000000000000004,
        0.00017303268749999998, 0.0005645964837343749, 0.0003273738042187501,
        0.00013293393750000003, 0.000030000000000000004, 0.0011958616266875002,
        0.00011727375000000003, 0.0006563778608917968, 0.00038789903437500006,
        0.000030000000000000004, 0.000077175000000000002, 0.000055500000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.00014277375000000002,
        0.000030000000000000004, 0.000095598750000000003, 0.000030000000000000004,
        0.0003000125285068556, 0.000098850000000000002, 0.0001945941646875,
        0.000077175000000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.0001611975, 0.000030000000000000004, 0.00019422, 0.000055500000000000001.
    )

    pageRanks45: expected = (
```

```
    expected,
        0.000030000000000000004, 0.000030000000000000004, 0.0003215274082826368,
        0.00009559875000000003, 0.00008100000000000002, 0.0005578175908965235,
        0.00007717500000000002, 0.00010650000000000001, 0.00007717500000000002,
        0.0002650830054609375, 0.000030000000000000004, 0.00012109875000000002,
        0.0003952191616555079, 0.00007717500000000002, 0.00009559875000000003,
        0.00010650000000000001, 0.00030403575000000006, 0.00020817806250000002,
        0.00011125893750000003, 0.0007993951805930079, 0.0009647913134157617,
        0.000030000000000000004, 0.00007717500000000002, 0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000140230284375, 0.00015843393750000003,
        0.0001573725, 0.00022963383274453124, 0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004, 0.00010267500000000002,
        0.00005550000000000001, 0.00012109875000000002, 0.0005682106623220115,
        0.000030000000000000004, 0.00007717500000000002, 0.00048354904214099615,
        0.000171525, 0.000030000000000000004, 0.00008100000000000002,
        0.00005550000000000001, 0.000030000000000000004, 0.0011088292365873393,
        0.00005550000000000001, 0.00019399384687500003, 0.000030000000000000004,
        0.00007717500000000002, 0.00010267500000000002, 0.000030000000000000004,
        0.000478586195888496, 0.000030000000000000004, 0.00027036634687500004,
        0.00013293393750000003, 0.000985747305325, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00007717500000000002, 0.00005550000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001, 0.000030000000000000004,
        0.00005550000000000001, 0.00005550000000000001, 0.000030000000000000004,
        0.000187405284375, 0.00009559875000000003, 0.00037878833859375006,
        0.000030000000000000004, 0.00025573217936718755, 0.000030000000000000004,
        0.0004222222140789454, 0.00005550000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00007717500000000002,
        0.00009885000000000002, 0.000030000000000000004, 0.000030000000000000004,
        0.0008828566442090232, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00007717500000000002,
        0.000030000000000000004, 0.000030000000000000004,
```

```
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
        0.000126919125, 0.000030000000000000004, 0.000030000000000000004,
        0.0008706173606547655, 0.000030000000000000004,
0.000030000000000000004.
  )

  pageRanks46: expected = (
    expected,
        0.000030000000000000004, 0.000030000000000000004,
0.001097905835669102,
        0.000030000000000000004, 0.000030000000000000004,
0.0012911886927233205,
        0.00010267500000000002, 0.000030000000000000004,
0.000030000000000000004,
        0.0014471317232076367, 0.00007717500000000002,
0.00032919842311718756,
        0.0006736807611742187, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.0005083643571588086,
0.00020877967921875,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.00005550000000000001, 0.00010267500000000002,
0.00005550000000000001,
        0.000030000000000000004, 0.000243164221875, 0.000267602784375,
        0.00005550000000000001, 0.000030000000000000004,
0.00047918890923906253,
        0.000030000000000000004, 0.0004517866978943946,
0.0015112480414817186,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.00021090620749218748, 0.0005817136509740037,
0.000030000000000000004,
        0.00005550000000000001, 0.0006902496070234765,
0.00005550000000000001,
        0.00005550000000000001, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.00023004750000000002,
0.000030000000000000004,
        0.00014810643750000002, 0.000030000000000000004,
0.000030000000000000004,
        0.0006170154330910936, 0.000165730284375, 0.00020927027636835936,
        0.00042570324346896486, 0.0005386219151164453,
0.00007717500000000002,
        0.0013819125144174025, 0.000030000000000000004,
0.0010520113375331644,
        0.00009559875000000003, 0.0005797662982845508,
0.00015007009687500002,
        0.00005550000000000001, 0.00005550000000000001,
0.00026522826441796886,
        0.0001356975, 0.00009885000000000002, 0.0012027005989846294,
        0.00005550000000000001, 0.00017685768750000004,
0.00005550000000000001,
        0.000030000000000000004, 0.00016119750000000003,
0.000030000000000000004,
        0.00011727375000000003, 0.00015755958234375,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.000030000000000000004,
```

```
        0.000030000000000000004, 0.000030000000000000004,
0.0010036956054313869,
        0.0001866975, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.0004468573575634569,
0.0007568851183328513,
        0.00042272894573730486, 0.000030000000000000004,
0.00005550000000000001,
        0.0003257730341351564, 0.0005692704913527343,
0.00019777118175802734,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.0002544981050078126,
        0.000030000000000000004, 0.000030000000000000004,
0.00005550000000000001,
        0.0009998740718150587, 0.000030000000000000004,
0.000030000000000000004,
        0.00028684279252365243, 0.000030000000000000004,
0.00005550000000000001,
        0.00022644259687500004, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.00011125893750000003,
0.00029678490030468754.
   )

   pageRanks47: expected = (
     expected,
        0.000030000000000000004, 0.000030000000000000004,
0.00007717500000000002,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.00005550000000000001, 0.000030000000000000004,
0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001,
0.00090625993512791,
        0.00007717500000000002, 0.000030000000000000004, 0.00021531519375,
        0.0004893327048039453, 0.00008100000000000002,
0.00028449085474904296,
        0.00011402250000000001, 0.000030000000000000004,
0.0004426083352713087,
        0.000030000000000000004, 0.000168493846875, 0.000225540375,
        0.00024176661889201175, 0.00010650000000000001,
0.00038301536889201183,
        0.0003264409174306447, 0.000030000000000000004,
0.0007187389780083399,
        0.0006355851256347263, 0.000030000000000000004,
0.000030000000000000004,
        0.000944545681565801, 0.00044651261407503907,
0.00011727375000000003,
        0.00008100000000000002, 0.0005137126510087695,
0.00007717500000000002,
        0.000030000000000000004, 0.0001392675, 0.000030000000000000004,
        0.00008100000000000002, 0.00005550000000000001,
0.00008100000000000002,
        0.000030000000000000004, 0.000030000000000000004,
0.000030000000000000004,
        0.000030000000000000004, 0.00014277375000000002,
0.00020859259687499998,
        0.0001806826875, 0.00009559875000000003, 0.00012109875000000002,
        0.000030000000000000004, 0.00006825, 0.00011125893750000003,
        0.00011402250000000001, 0.00021219750000000002,
```

```
0.00005550000000000001,
        0.0005975487986158006, 0.00005550000000000001,
0.000030000000000000004,
        0.0006294703299214255, 0.00010267500000000002,
0.00005550000000000001,
        0.000030000000000000004, 0.00024502556250000004,
0.000030000000000000004,
        0.00021736361671874997, 0.000030000000000000004,
0.000478267900330039,
        0.000030000000000000004, 0.000030000000000000004,
0.0005636191341747459,
        0.000030000000000000004, 0.000030000000000000004, 0.0001610030625,
        0.0003478678864488282, 0.000030000000000000004,
0.0008226126343261718,
        0.00044772343749357427, 0.0010844268495790823,
0.00027328157421093753,
        0.000030000000000000004, 0.00007717500000000002,
0.0004377152188212696,
        0.00005550000000000001, 0.00007717500000000002,
0.0003810599095457033,
        0.000030000000000000004, 0.0007371604510937501,
0.0010461534389142773,
        0.00024611486671875007, 0.0008474182917279883,
0.00038903171229613285.
  )

  pageRanks48: expected = (
    expected,
        0.00005550000000000001, 0.000030000000000000004,
0.00007717500000000002,
        0.0007478200571626757, 0.000030000000000000004,
0.00013293393750000003,
        0.0003705205402589845, 0.000030000000000000004,
0.00009559875000000003,
        0.00015007009687500002, 0.00026467378371210936,
0.00013330032249609376,
        0.00047262199203455097, 0.000168493846875, 0.00009559875000000003,
        0.000030000000000000004, 0.00005550000000000001,
0.00019035592921875,
        0.00005550000000000001, 0.0004023563457193946,
0.000792128087533164,
        0.00005550000000000001, 0.000030000000000000004,
0.0004883296733917968,
        0.00005550000000000001, 0.0004559052944855469,
0.000497331361295918,
        0.00005550000000000001, 0.000400097954671875,
0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001,
0.00007717500000000002,
        0.0001513576875, 0.000030000000000000004, 0.00011402250000000001,
        0.00005550000000000001, 0.00010267500000000002,
0.0005869410960447462,
        0.00005550000000000001, 0.00011727375000000003, 0.00014985,
        0.00046397884894101556, 0.00013293393750000003,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.0004985444978679295,
        0.00009559875000000003, 0.00029019763671093757, 0.00024614788125,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.00004275000000000001, 0.00005550000000000001,
```

```
0.0002890909435165822,
        0.000030000000000000004, 0.0009232520230490042, 0.000077175000000000002,
        0.000077175000000000002, 0.000030000000000000004, 0.0002699518125,
        0.000030000000000000004, 0.000095598750000000003, 0.000030000000000000004.
)

pageRanks49: expected = (
  expected,
        0.0004820662698785156, 0.000055500000000000001, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.0001584339375,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.00011727375000000003, 0.0004580961996339844, 0.000055500000000000001,
        0.0034371926123786135, 0.0007171233431230858, 0.000030000000000000004,
        0.000030000000000000004, 0.0004940070602542576, 0.000030000000000000004,
        0.000055500000000000001, 0.0020155847563304292, 0.0002634430606171875 7,
        0.00025103522718749997, 0.0004731140913068163 4, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.000055500000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.0005740974556124608, 0.000055500000000000001, 0.000055500000000000001,
        0.000030000000000000004, 0.00012109875000000002, 0.000055500000000000001,
        0.000030000000000000004, 0.0003655747524131056, 0.0003691095187500001,
        0.000128175, 0.0002538039375000001, 0.000185441625, 0.0004509094982433595,
        0.000030000000000000004, 0.000095598750000000003, 0.000030000000000000004,
        0.00012968268750000001, 0.000055500000000000001, 0.0013434953179166213,
        0.00019364019375, 0.000081000000000000002, 0.000152419125,
        0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
        0.00036471634687500003, 0.000030000000000000004, 0.00017557009687500002,
        0.000055500000000000001, 0.000030000000000000004, 0.000095598750000000003,
        0.00016225893750000003, 0.000140230284375, 0.0010686188597835936,
        0.0005728017023122849, 0.000030000000000000004, 0.0002805243949921875 4,
        0.0003154114045719142, 0.0003410776875000001, 0.000030000000000000004,
        0.000077175000000000002, 0.000077175000000000002, 0.000585874040565371,
        0.0008652517462578516, 0.0006117752634263864, 0.0006415838093136718,
        0.000055500000000000001, 0.000987794997580957, 0.000077175000000000002,
        0.0003370707326484376, 0.000077175000000000002, 0.000077175000000000002,
```

```
        0.0011201223724816604, 0.000030000000000000004, 0.000030000000000000004,
        0.000055500000000000001, 0.000030000000000000004, 0.00041637400038828116,
        0.00030097138671093755, 0.0005901878716493943, 0.0009725682330679693,
        0.0005054563544739061, 0.000081000000000000002, 0.0001744732418671875,
        0.000030000000000000004, 0.000055500000000000001, 0.0011511606224924999,
        0.000055500000000000001, 0.00017469574171875, 0.000030000000000000004,
        0.000055500000000000001, 0.000055500000000000001, 0.000081000000000000002.
    )

    pageRanks50: expected = (
      expected,
        0.000077175000000000002, 0.00017174509687500002, 0.0005585642724306445,
        0.0004331871046195312, 0.000030000000000000004, 0.000077175000000000002,
        0.0006780410787286913, 0.00019145643750000002, 0.0006755674858472655,
        0.0004212642249228322, 0.000055500000000000001, 0.000030000000000000004,
        0.00020085, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.00011727375000000003, 0.000030000000000000004,
        0.000055500000000000001, 0.00017730948300342773, 0.000077175000000000002,
        0.000030000000000000004, 0.000081000000000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.00012109875000000002, 0.000030000000000000004,
        0.0009763612694554688, 0.00024463889484375003, 0.00021184384687500003,
        0.000030000000000000004, 0.00013675893750000003, 0.000077175000000000002,
        0.000030000000000000004, 0.00036788708859375, 0.0012347261757795606,
        0.000030000000000000004, 0.00023885153437500003, 0.0002056089375,
        0.00020476759687499999, 0.000030000000000000004, 0.0009038579981077738,
        0.00020235768750000003, 0.000030000000000000004, 0.00020859259687500004,
        0.000055500000000000001, 0.00036820402530468756, 0.000055500000000000001,
        0.00066419930620291, 0.00010267500000000002, 0.0003713825254372853,
        0.000055500000000000001, 0.00010267500000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.000055500000000000001, 0.000030000000000000004,
        0.000055500000000000001, 0.0004132730735068556, 0.00011402250000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.000610537969854375,
        0.000055500000000000001, 0.000055500000000000001, 0.000030000000000000004,
        0.000055500000000000001, 0.000055500000000000001,
```

```
0.00231323575590872,
        0.000030000000000000004, 0.00010267500000000002,
0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004,
0.0007635236087597262,
        0.00005550000000000001, 0.000030000000000000004,
0.00012109875000000002,
        0.00008100000000000002, 0.00007717500000000002,
0.000030000000000000004,
        0.000030000000000000004, 0.0006835465769273435,
0.00010267500000000002,
        0.000030000000000000004, 0.00011125893750000003,
0.0002674281846796875,
        0.00089338520952625, 0.00007717500000000002,
0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.00010267500000000002, 0.001359938627840293,
0.00008100000000000002,
        0.00014534287500000002, 0.00005550000000000001,
0.00010267500000000002,
        0.000030000000000000004, 0.00008100000000000002,
0.00030859339458498054.
  )

  pageRanks51: expected = (
    expected,
        0.00025607358488302737, 0.000030000000000000004,
0.00040411500814865246,
        0.000030000000000000004, 0.00012968268750000001,
0.000030000000000000004,
        0.00007717500000000002, 0.000030000000000000004,
0.00034456086915078134,
        0.000030000000000000004, 0.000030000000000000004,
0.00010267500000000002,
        0.00011727375000000003, 0.00009559875000000003,
0.000030000000000000004,
        0.000030000000000000004, 0.00005550000000000001,
0.000030000000000000004,
        0.00012109875000000002, 0.00013293393750000003,
0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004,
0.00011125893750000003,
        0.000030000000000000004, 0.00005550000000000001,
0.0007807470633976951,
        0.00005550000000000001, 0.000030000000000000004,
0.00005550000000000001,
        0.00007717500000000002, 0.00005550000000000001,
0.00005550000000000001,
        0.000030000000000000004, 0.0003173643442031251,
0.00009559875000000003,
        0.000030000000000000004, 0.00010650000000000001,
0.000030000000000000004,
        0.0006441722416652344, 0.00024267653437499997,
0.000030000000000000004,
        0.00008100000000000002, 0.00005550000000000001,
0.000030000000000000004,
        0.000030000000000000004, 0.00011125893750000003,
0.000030000000000000004,
        0.00016827375000000002, 0.0002450773350314649,
0.000030000000000000004,
```

```
          0.00003000000000000004, 0.00003000000000000004, 0.0005768196484567382,
          0.00003000000000000004, 0.00007717500000000002, 0.00003000000000000004,
          0.00003000000000000004, 0.00003000000000000004, 0.0014809880060556444,
          0.0003982600444914454, 0.00003000000000000004, 0.00005550000000000001,
          0.00007717500000000002, 0.0006242420395359766, 0.00003000000000000004,
          0.00016444875, 0.00003000000000000004, 0.00033119181562500001,
          0.00003000000000000004, 0.00010267500000000002, 0.00003000000000000004,
          0.00003000000000000004, 0.0008059152801275586, 0.0012004930497145903,
          0.00003000000000000004, 0.0004951419985664452, 0.0004380695932937892,
          0.00003000000000000004, 0.00010267500000000002, 0.00011727375000000003,
          0.00003000000000000004, 0.00003000000000000004, 0.00005550000000000001,
          0.00003000000000000004, 0.0005296988262009961, 0.00003000000000000004,
          0.00003000000000000004, 0.00003000000000000004, 0.00003000000000000004,
          0.00010650000000000001, 0.00003000000000000004, 0.0005620807610697853,
          0.0005286613671070311, 0.00003000000000000004, 0.00003000000000000004,
          0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004.
    )

    pageRanks52: expected = (
      expected,
          0.00011125893750000003, 0.00003000000000000004, 0.00005550000000000001,
          0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
          0.00076081716755199 2, 0.00003000000000000004, 0.00010650000000000001,
          0.00003000000000000004, 0.00003000000000000004, 0.0003138477416337697,
          0.00003000000000000004, 0.00005550000000000001, 0.00003000000000000004,
          0.00003000000000000004, 0.0008371103585068553, 0.00007717500000000002,
          0.0004518288427982617, 0.00003000000000000004, 0.00003000000000000004,
          0.00003000000000000004, 0.00003000000000000004, 0.0002608801875,
          0.00005550000000000001, 0.00003000000000000004, 0.00005550000000000001,
          0.00003000000000000004, 0.00003000000000000004, 0.00005550000000000001,
          0.000164668846875, 0.00011727375000000003, 0.0003858458493481542,
          0.00007717500000000002, 0.00003000000000000004, 0.00003000000000000004,
          0.00003000000000000004, 0.001094358015352618, 0.00035263408031250003,
          0.000370735670154336, 0.00008100000000000002, 0.0007597855260648826,
```

```
        0.000030000000000000004, 0.000030000000000000004, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00009885000000000002,
        0.00005550000000000001, 0.00014277375000000002, 0.00012457009687500003,
        0.00005550000000000001, 0.00016444875000000002, 0.000030000000000000004,
        0.000030000000000000004, 0.00007717500000000002, 0.00009559875000000003,
        0.0004551034628453516, 0.00017174509687500002, 0.00016119750000000003,
        0.00005550000000000001, 0.000030000000000000004, 0.0003727185005080275,
        0.00012435000000000001, 0.00048311825031271483, 0.000030000000000000004,
        0.0012598353609723147, 0.000030000000000000004, 0.000030000000000000004,
        0.00038769586070216803, 0.00005550000000000001, 0.00010267500000000002,
        0.00013588458234375003, 0.000568914242375215, 0.0006243928897521678,
        0.000030000000000000004, 0.000030000000000000004, 0.00007717500000000002,
        0.0010377028924124902, 0.00008100000000000002, 0.00005550000000000001.
  )

  pageRanks53: expected = (
    expected,
        0.00010267500000000002, 0.000030000000000000004, 0.0005503465372175779,
        0.000030000000000000004, 0.001355548606467051, 0.0003556893808617188,
        0.000030000000000000004, 0.00016827375, 0.000030000000000000004,
        0.0008246139931943555, 0.000030000000000000004, 0.0010154740938932223,
        0.00040518822771779304, 0.00045871095761835937, 0.00005550000000000001,
        0.00016701787500000002, 0.002071953365629297, 0.000030000000000000004,
        0.000030000000000000004, 0.000030000000000000004, 0.00005550000000000001,
        0.000030000000000000004, 0.000030000000000000004, 0.0005470457547568554,
        0.0005157095790148827, 0.000030000000000000004, 0.000688324568213281,
        0.0006465114741332422, 0.0010292594947507033, 0.00005550000000000001,
        0.000030000000000000004, 0.00005550000000000001, 0.0001649910962121094,
        0.000030000000000000004, 0.00009559875000000003, 0.00014277375000000002,
        0.00010267500000000002, 0.000030000000000000004, 0.00031039666483593755,
        0.0005295671026200583, 0.000030000000000000004, 0.00008100000000000002,
        0.0004951876449921876, 0.000146025, 0.0005186043726941406,
        0.00005550000000000001, 0.00007717500000000002, 0.00011727375000000003,
        0.0004914393382277344, 0.00030981865964960947,
```

```
0.0005172251503359373,
        0.00007717500000000002, 0.00014624509687500003,
0.0009398635399813478,
        0.0014674101670753515, 0.000055500000000000001,
0.000030000000000000004,
        0.000030000000000000004.
    )
)

```

Examples/Benchmarks/Permute.som

```
"

$Id: Permute.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Permute = Benchmark (

    | count v |

    benchmark = (
        count := 0.
        v     := Array new: 7.
        self permute: 6.
        ^ count
    )

    verifyResult: result = (
      ^ self assert: 8660 equals: result
    )

    permute: n = (
        count := count + 1.
        (n <> 0)
            ifTrue: [
                self permute: n - 1.
                n downTo: 1 do: [ :i |
                    self swap: n with: i.
                    self permute: n - 1.
                    self swap: n with: i ] ]
    )

    swap: i with: j = (
        | tmp |
        tmp := v at: i.
```

```
        v at: i put: (v at: j).
        v at: j put: tmp
    )

)

```

```
Poly1 = (
  | sub |
  initializeWith: anObject = (
    sub := anObject
  )

  score = (
    ^ sub score
  )
  ----
  with: anObject = (
    ^ super new initializeWith: anObject
  )

)
```

Examples/Benchmarks/Polymorphism/Polymorphism.som

```
"Port of  https://github.com/dropbox/pyston/blob/master/microbenchmarks/polymorphism.py
# This microbenchmark is inspired by the ICBD type checker, which has a 'scoring' phase at
# the end of the analysis.
# I'm not sure how representative this file is of the type checker, or even if
# this is really a polymorphism test or maybe just a tree test, or some cross of the two.
"

Polymorphism = Benchmark (
  | d |
  rand = (
    d := (d * 1.24591 + 0.195) % 1.0.
    ^ d
  )

  makeRandom: x = (
    self rand > x   ifTrue: [^ Simple new].
    self rand < 0.3 ifTrue: [
      ^ Union with: (
          Array with: (self makeRandom: 0.5 * (x - 1))
                with: (self makeRandom: 0.5 * (x - 1)))].
    ^ Poly1 with: (self makeRandom: x - 1)
  )

  benchmark = (
    | r |
    d := 0.0.
    r := self makeRandom: 10000.
    1000 timesRepeat: [r score]
  )
)
```

Examples/Benchmarks/Polymorphism/Simple.som

```
Simple = (
  score = ( ^ 1.0 )
)
```

Examples/Benchmarks/Polymorphism/Union.som

```
Union = (
  | subs |

  initializeWith: anArray = ( subs := anArray )
  score = (
    | t |
    t := 0.0.
    subs do: [:s | t := t + s score].
    t := t // (subs length * subs length).
    ^ t
  )

  ----
  with: anArray = (
    ^ super new initializeWith: anArray
  )
)
```

Examples/Benchmarks/Queens.som

```
"

$Id: Queens.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Queens = Benchmark (

    | freeMaxs freeRows freeMins queenRows |

    benchmark = (
        | result |
        result := true.
        1 to: 10 do: [ :i | result := result and: self queens ].
        ^ result
    )

    verifyResult: result = (
      ^ result
    )

    queens = (
        freeRows  := Array new:  8 withAll: true.
        freeMaxs  := Array new: 16 withAll: true.
        freeMins  := Array new: 16 withAll: true.
        queenRows := Array new:  8 withAll: -1.
        ^ self placeQueen: 1
    )

    placeQueen: c = (
        1 to: 8 do: [ :r |
            (self row: r column: c)
                ifTrue: [
                    queenRows at: r put: c.
                    self row: r column: c put: false.
```

```
                    (c = 8) ifTrue: [ ^true ].
                    (self placeQueen: c + 1) ifTrue: [ ^true ].
                    self row: r column: c put: true ] ].
        ^false
    )

    row: r column: c = (
        ^(freeRows at: r) && (freeMaxs at: c + r) && (freeMins at: c - r + 8)
    )

    row: r column: c put: v = (
        freeRows at: r         put: v.
        freeMaxs at: c + r     put: v.
        freeMins at: c - r + 8 put: v.
    )

)
```

Examples/Benchmarks/QuickSort.som

```
"

$Id: QuickSort.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

QuickSort = Sort (

    sort: array = (
        self sort: array low: 1 high: self dataSize.
        ^ array
    )

    sort: array low: low high: high = (
        | pivot i j |

        pivot := array at: (low + high) / 2.
        i := low.
        j := high.
        [ i <= j ]
            whileTrue: [
                [ (array at: i) < pivot ] whileTrue: [ i := i + 1 ].
                [ pivot < (array at: j) ] whileTrue: [ j := j - 1 ].
                ( i <= j )
                    ifTrue: [
                        | tmp |
                        tmp := array at: i.
                        array at: i put: (array at: j).
                        array at: j put: tmp.
                        i := i + 1.
                        j := j - 1. ] ].

        (low < j)  ifTrue: [ self sort: array low: low high: j  ].
        (i < high) ifTrue: [ self sort: array low: i high: high ]
    )
```

```

    dataSize = ( ^800 )

)

```

Examples/Benchmarks/Random.som

```
"

$Id: Random.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Random = (

    | seed |

    initialize = (
        seed := 74755
    )

    next = (
        seed := ((seed * 1309) + 13849) & 65535.
        ^seed
    )

    run = (
        | fail |
        'Testing random number generator ... ' print.
        fail := [ 'FAILED:' println. ^nil ].
        (self next <> 22896) ifTrue: fail.
        (self next <> 34761) ifTrue: fail.
        (self next <> 34014) ifTrue: fail.
        (self next <> 39231) ifTrue: fail.
        (self next <> 52540) ifTrue: fail.
        (self next <> 41445) ifTrue: fail.
        (self next <>  1546) ifTrue: fail.
        (self next <>  5947) ifTrue: fail.
        (self next <> 65224) ifTrue: fail.
        'PASSED' println
    )

```

```
    -------------------

    | random |

    new        = ( ^super new initialize )
    next       = ( ^self random next )
    initialize = ( ^random := Random new )

    random = ( ^random )

)
```

```
DeviceTaskDataRecord = RBObject (
    |pending|

    pending = (^pending)

    pending: packet = (pending := packet)

    create = (pending := RBObject NoWork)
----
    create = (
        ^super new create
    )
)

```

Examples/Benchmarks/Richards/HandlerTaskDataRecord.som

```
HandlerTaskDataRecord = RBObject (
    |workIn deviceIn|

    deviceIn = (^deviceIn)

    deviceIn: aPacket = (deviceIn := aPacket)

    deviceInAdd: packet = (
        deviceIn := self append: packet head: deviceIn
    )

    workIn = (^workIn)

    workIn: aWorkQueue = (workIn := aWorkQueue)

    workInAdd: packet = (
        workIn := self append: packet head: workIn
    )

    create = (
        workIn := deviceIn := RBObject NoWork
    )
    asString = (
        ^ 'HandlerTaskDataRecord(' + workIn asString + ', ' + deviceIn asString + ')'
    )
----
    create = (^super new create)
)
```

```
IdleTaskDataRecord = RBObject (

    |control count|

    control = (^control)

    control: aNumber = (
        control := aNumber
    )

    count = (^count)

    count: aCount = (
        count := aCount
    )

    create = (
        control := 1.
        count := 10000
    )

    ----

    create = (^super new create)
)
```

```
Packet = RBObject (
    | link identity kind datum data |

    data  = ( ^data)

    datum = ( ^datum)
    datum: someData = (datum := someData)

    identity = ( ^identity)
    identity: anIdentity = ( identity := anIdentity)

    kind = ( ^kind)
    link = ( ^link)

    link: aWorkQueue = ( link := aWorkQueue )
    link: aLink identity: anIdentity kind: aKind = (
        link := aLink.
        identity := anIdentity.
        kind := aKind.
        datum := 1.
        data := Array new: 4 withAll: 0
    )

    asString = (
        ^ 'Packet(' +
            link asString + ', ' +
            identity asString + ', ' +
            kind asString + ', ' +
            datum asString + ', ' +
            data asString +
          ')'
    )

    ----

    create: link identity: identity kind: kind = (
        ^super new
            link: link
            identity: identity
            kind: kind
    )
)
```

Examples/Benchmarks/Richards/RBObject.som

```
RBObject = Object (

    append: packet head: queueHead = (
        | mouse link |
        packet link: RBObject NoWork.
        RBObject NoWork == queueHead ifTrue: [^packet].
        mouse := queueHead.
        [RBObject NoWork == (link := mouse link)]
                whileFalse: [mouse := link].
        mouse link: packet.
        ^queueHead
’  •

    ----

    NoTask  = ( ^ nil )
    Idler   = ( ^ 1   )
    NoWork  = ( ^ nil )
    Worker  = ( ^ 2 )
    WorkPacketKind = ( ^ 2 )
    HandlerA = ( ^ 3 )
    HandlerB = ( ^ 4 )
    DeviceA  = ( ^ 5 )
    DeviceB  = ( ^ 6 )
    DevicePacketKind = ( ^ 1 )
)

```

```
Richards = Benchmark (
    benchmark = (
        ^ RichardsBenchmarks new reBenchStart.
    )
    verifyResult: result = (
      ^ result
    )
)
```

Examples/Benchmarks/Richards/RichardsBenchmarks.som

```
RichardsBenchmarks = RBObject (
    |taskList currentTask currentTaskIdentity taskTable tracing layout queuePacketCount holdCount|

    createDevice: identity priority: priority work: work state: state = (
        | data |
        data := DeviceTaskDataRecord create.
        self
                createTask: identity
                priority: priority
                work: work
                state: state
                function:
                        [:work :word |
                        | data functionWork |
                        data := word.
                        functionWork := work.
                        RBObject NoWork == functionWork
                                ifTrue:
                                        [RBObject NoWork == (functionWork := data pending)
                                                ifTrue: [self wait]
                                                ifFalse:
                                                        [data pending: RBObject NoWork.
                                                        self queuePacket: functionWork]]
                                ifFalse:
                                        [data pending: functionWork.
                                        tracing
                                                ifTrue:
                                                        [self trace: functionWork datum].
                                        self holdSelf]]
                data: data
    )

    createHandler: identity priority: priority work: work state: state = (
        | data |
        data := HandlerTaskDataRecord create.
        self
                createTask: identity
                priority: priority
                work: work
                state: state
                function:
                        [:work :word |
                        | data workPacket |
                        data := word.
                        RBObject NoWork == work
                                ifFalse: [RBObject WorkPacketKind == work kind
                                        ifTrue: [data workInAdd: work]
                                        ifFalse: [data deviceInAdd: work]].
                        RBObject NoWork == (workPacket := data workIn)
                                ifTrue: [self wait]
                                ifFalse:
```

```
                                        [ |count|
                                          count := workPacket datum.
                                        count > 4
                                                ifTrue:
                                                        [data workIn: workPacket link.
                                                        self queuePacket: workPacket]
                                                ifFalse:
                                                        [ | devicePacket |
                                                          RBObject NoWork == (devicePacket := data deviceIn)
                                                                ifTrue: [self wait]
                                                                ifFalse:
                                                                        [data deviceIn: devicePacket link.
                                                                        devicePacket datum: (workPacket data at: count).
                                                                        workPacket datum: count + 1.
                                                                        self queuePacket: devicePacket]]]]
                    data: data
        )

        createIdler: identity priority: priority work: work state: state = (
            | data |
            data := IdleTaskDataRecord create.
            self
                    createTask: identity
                    priority: priority
                    work: work
                    state: state
                    function:
                            [:work :word |
                            | data |
                            data := word.
                            data count: data count - 1.
                            0 = data count
                                    ifTrue: [self holdSelf]
                                    ifFalse:
                                            [0 = (data control & 1)
                                                    ifTrue:
                                                            [data control: data control / 2.
                                                            self release: RBObject DeviceA]
                                                    ifFalse:
                                                            [data control: (data control / 2 bitXor: 53256).
                                                            self release: RBObject DeviceB]]]
                    data: data
        )

        createPacket: link identity: identity kind: kind = (
            ^Packet
                    create: link
                    identity: identity
                    kind: kind
```

```
    )

    createTask: identity priority: priority work: work state: state function: aBlock data: data = (
        | t |

        t := TaskControlBlock
                                link: taskList
                                create: identity
                                priority: priority
                                initialWorkQueue: work
                                initialState: state
                                function: aBlock
                                privateData: data.
        taskList := t.
        taskTable at: identity put: t
    )

    createWorker: identity priority: priority work: work state: state = (
        | data |
        data := WorkerTaskDataRecord create.
        self
                createTask: identity
                priority: priority
                work: work
                state: state
                function:
                        [:work :word |
                        | data |
                        data := word.
                        RBObject NoWork == work
                                ifTrue: [self wait]
                                ifFalse:
                                        [data destination: (RBObject HandlerA = data destination
                                                                ifTrue: [RBObject HandlerB]
                                                                ifFalse: [RBObject HandlerA]).
                                        work identity: data destination.
                                        work datum: 1.
                                        1 to: 4 do:
                                                [:i |
                                                data count: data count + 1.
                                                data count > 26 ifTrue: [data count: 1].
                                                "work data at: i put: $A asInteger + data count - 1]."
                                                work data at: i put: 65 + data count - 1].
                                        self queuePacket: work]]
                data: data
    )

    run = (
        RBObject initialize.
        self start.
    )

    start = (
```

```
        | workQ mark1 mark2 mark3 mark4|
        self initTrace.
        self initScheduler.
        mark1 := Time millisecondClockValue.
        Transcript show: 'Bench mark starting'.
        Transcript cr.
        self
                createIdler: RBObject Idler
                priority: 0
                work: RBObject NoWork
                state: TaskState running.
        workQ := self
                                createPacket: RBObject NoWork
                                identity: RBObject Worker
                                kind: RBObject WorkPacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject Worker
                                kind: RBObject WorkPacketKind.
        self
                createWorker: RBObject Worker
                priority: 1000
                work: workQ
                state: TaskState waitingWithPacket.
        workQ := self
                                createPacket: RBObject NoWork
                                identity: RBObject DeviceA
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceA
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceA
                                kind: RBObject DevicePacketKind.
        self
                createHandler: RBObject HandlerA
                priority: 2000
                work: workQ
                state: TaskState waitingWithPacket.
        workQ := self
                                createPacket: RBObject NoWork
                                identity: RBObject DeviceB
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceB
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceB
                                kind: RBObject DevicePacketKind.
        self
                createHandler: RBObject HandlerB
                priority: 3000
                work: workQ
                state: TaskState waitingWithPacket.
        self
                createDevice: RBObject DeviceA
                priority: 4000
```

```
                work: RBObject NoWork
                state: TaskState waiting.
        self
                createDevice: RBObject DeviceB
                priority: 5000
                work: RBObject NoWork
                state: TaskState waiting.
        Transcript show: 'Starting'.
        Transcript cr.
        mark2 := Time millisecondClockValue.
        self schedule.
        mark3 := Time millisecondClockValue.
        Transcript show: 'Finished'.
        Transcript cr.
        Transcript show: 'QueuePacket count = '.
        Transcript show: queuePacketCount asString.
        Transcript show: ' HoldCount = '.
        Transcript show: holdCount asString.
        Transcript cr.
        Transcript show: 'These results are '.
        (((queuePacketCount = 23246) and: (holdCount = 9297))
            ifTrue: [Transcript show: 'correct']
            ifFalse: [Transcript show: 'wrong']).
        Transcript cr.
        Transcript show: 'End of run'.
        Transcript cr.
        mark4 := Time millisecondClockValue.
        Transcript show: '*****Scheduler time = '.
        Transcript show: (mark3 - mark2) asString.
        Transcript show: ' Total time = '.
        Transcript show: (mark4 - mark1) asString.
        Transcript cr
    )

    "This is start simply duplicated, removing all output and making the
     correctness check exiting with an error."
    reBenchStart = (
        | workQ |
        self initTrace.
        self initScheduler.

        self
                createIdler: RBObject Idler
                priority: 0
                work: RBObject NoWork
                state: TaskState running.
        workQ := self
                                createPacket: RBObject NoWork
                                identity: RBObject Worker
                                kind: RBObject WorkPacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject Worker
                                kind: RBObject WorkPacketKind.
        self
                createWorker: RBObject Worker
                priority: 1000
                work: workQ
                state: TaskState waitingWithPacket.
        workQ := self
                                createPacket: RBObject NoWork
```

```
                                identity: RBObject DeviceA
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceA
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceA
                                kind: RBObject DevicePacketKind.
        self
                createHandler: RBObject HandlerA
                priority: 2000
                work: workQ
                state: TaskState waitingWithPacket.
        workQ := self
                                createPacket: RBObject NoWork
                                identity: RBObject DeviceB
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceB
                                kind: RBObject DevicePacketKind.
        workQ := self
                                createPacket: workQ
                                identity: RBObject DeviceB
                                kind: RBObject DevicePacketKind.
        self
                createHandler: RBObject HandlerB
                priority: 3000
                work: workQ
                state: TaskState waitingWithPacket.
        self
                createDevice: RBObject DeviceA
                priority: 4000
                work: RBObject NoWork
                state: TaskState waiting.
        self
                createDevice: RBObject DeviceB
                priority: 5000
                work: RBObject NoWork
                state: TaskState waiting.

        self schedule.

        ^ ((queuePacketCount = 23246) and: (holdCount = 9297))
    )


    findTask: identity = (
        | t |
        t := taskTable at: identity.
        RBObject NoTask == t ifTrue: [self error: 'findTask failed'].
        ^t
    )

    holdSelf = (
        holdCount := holdCount + 1.
        currentTask taskHolding: true.
        ^currentTask link
    )
```

```

    initScheduler = (
        queuePacketCount := 0.
        holdCount := 0.
        taskTable := Array new: 6 withAll: RBObject NoTask.
        taskList := RBObject NoTask
    )

    initTrace = (
        "îâ    &–æ '”6†ö–6R
        message: 'Trace?'
        displayAt: Sensor mousePoint
        centered: true
        ifTrue: [tracing := true]
        ifFalse: [tracing := false].
        "
        "does not work in V. 4"

        tracing := false.
        layout := 0
    )

    queuePacket: packet = (
        | t |
        t := self findTask: packet identity.
        RBObject NoTask == t ifTrue: [^RBObject NoTask].
        queuePacketCount := queuePacketCount + 1.
        packet link: RBObject NoWork.
        packet identity: currentTaskIdentity.
        ^t addInput: packet checkPriority: currentTask
    )

    release: identity = (
        | t |
        t := self findTask: identity.
        RBObject NoTask == t ifTrue: [^RBObject NoTask].
        t taskHolding: false.
        t priority > currentTask priority
                ifTrue: [^t]
                ifFalse: [^currentTask]
    )

    trace: id = (
        layout := layout - 1.
        0 >= layout
                ifTrue:
                        [Transcript cr.
                        layout := 50].
        Transcript show: id asString
    )

    wait = (
        currentTask taskWaiting: true.
        ^currentTask
    )

    schedule = (
        currentTask := taskList.
        [RBObject NoTask == currentTask]
                whileFalse:
                        [currentTask isTaskHoldingOrWaiting
```

```
                                    ifTrue: [currentTask := currentTask link]
                                    ifFalse:
                                            [currentTaskIdentity := currentTask identity.
                                            tracing ifTrue: [self trace: currentTaskIdentity].
                                            currentTask := currentTask runTask]]
        )

        ----

        start = (
            super new start
        )
)

```

Examples/Benchmarks/Richards/TaskControlBlock.som

```

   TaskControlBlock = TaskState (
    |link identity priority input function handle|

    identity = (^identity)

    link = (^link)

    priority = (^priority)

    link: aLink identity: anIdentity priority: aPriority initialWorkQueue: anInitialWorkQueue initialState: anInitialState function: aBlock privateData: aPrivateData = (
        link := aLink.
        identity := anIdentity.
        priority := aPriority.
        input := anInitialWorkQueue.
        self packetPending: anInitialState isPacketPending.
        self taskWaiting: anInitialState isTaskWaiting.
        self taskHolding: anInitialState isTaskHolding.
        function := aBlock.
        handle := aPrivateData.
    )

    addInput: packet checkPriority: oldTask = (
        RBObject NoWork == input
                ifTrue:
                        [input := packet.
                        self packetPending: true.
                        priority > oldTask priority ifTrue: [^self]]
                ifFalse:
                        [input := self append: packet head: input].
        ^oldTask
    )

    runTask = (
     | message |
     self isWaitingWithPacket
         ifTrue:
             [message := input.
             input := message link.
             RBObject NoWork == input
                 ifTrue: [self running]
                 ifFalse: [self packetPending]]
         ifFalse: [message := RBObject NoWork].
        ^function value: message with: handle
    )

----

    link: link create: identity priority: priority initialWorkQueue: initialWorkQueue initialState: initialState function: aBlock privateData: privateData = (
        ^super new
                link: link
                identity: identity
                priority: priority
                initialWorkQueue: initialWorkQueue
```

```
                initialState: initialState
                function: aBlock
                privateData: privateData
    )
)

```

Examples/Benchmarks/Richards/TaskState.som

```
TaskState = RBObject (
    |packetPending taskWaiting taskHolding|

    isPacketPending = (^packetPending)

    isTaskHolding = (^taskHolding)

    isTaskWaiting = (^taskWaiting)

    taskHolding: aBoolean   = (taskHolding   := aBoolean)
    taskWaiting: aBoolean   = (taskWaiting   := aBoolean)
    packetPending: aBoolean = (packetPending := aBoolean)

    packetPending = (
        packetPending := true.
        taskWaiting := false.
        taskHolding := false
    )

    running = (
        packetPending := taskWaiting := taskHolding := false
    )

    waiting = (
        packetPending := taskHolding := false.
        taskWaiting := true
    )

    waitingWithPacket = (
        taskHolding := false.
        taskWaiting := packetPending := true
    )

    isRunning = (^packetPending not and: [taskWaiting not and: [taskHolding not]])

    isTaskHoldingOrWaiting = (^taskHolding or: [packetPending not and: [taskWaiting]])

    isWaiting = (^packetPending not and: [taskWaiting and: [taskHolding not]])

    isWaitingWithPacket = (^packetPending and: [taskWaiting and: [taskHolding not]])


    ----
    packetPending = (^super new packetPending)

    running = (
        ^super new running
    )

    waiting = (^super new waiting)

    waitingWithPacket = (^super new waitingWithPacket)
)
```

Examples/Benchmarks/Richards/Time.som

```
Time = Object (
    ----
    millisecondClockValue = (^system time)
)
```

```
Transcript = (
    ----
    cr = (system printNewline)
    show: text = (system printString: text)
)
```

```
WorkerTaskDataRecord = RBObject (
    |destination count|
    count = (^count)

    count: aCount =(count := aCount)

    destination = (^destination)

    destination: aHandler = (destination := aHandler)

    create = (
        destination := RBObject HandlerA.
        count := 0
    )
----
create = (^super new create)
)
```

Examples/Benchmarks/Sieve.som

```
"

$Id: Sieve.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Sieve = Benchmark (

    benchmark = (
        | flags |
        flags  := Array new: 5000.
        ^ self sieve: flags size: 5000.
    )

    verifyResult: result = (
      ^ self assert: 669 equals: result
    )

    sieve: flags size: size = (
        | primeCount |
        primeCount := 0.
        flags putAll: true.
        2 to: size do: [ :i |
            (flags at: i - 1)
                ifTrue: [
                    | k |
                    primeCount := primeCount + 1.
                    k := i + i.
                    [ k <= size ]
                        whileTrue: [
                            flags at: k - 1 put: false. k := k + i ]. ] ].
        ^primeCount
    )

)
```

Examples/Benchmarks/SomInit.som

```
"
Copyright (c) 2022 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
SomInit = Benchmark (
  | rnd vec benchHarn planner jsonLit jsonNum jsonStr jsonP parEx edge
    lexer sblock sclass sdouble sinteger smethod sprimitive sstring |

  oneTimeSetup = (
    rnd     := Random new.
    vec     := Vector new.
    benchHarn := BenchmarkHarness new.
    planner := Planner new.
    jsonLit := JsonLiteral new.
    jsonNum := JsonNumber new.
    jsonStr := JsonString new.
    jsonP   := JsonParser new.
    parEx   := ParseException new.
    edge    := Edge new.

    lexer  := Lexer new.
    sblock := SBlock new.
    sclass := SClass new.

    sdouble    := SDouble new.
    sinteger   := SInteger new.
    smethod    := SMethod new.
    sprimitive := SPrimitive new.
    sstring    := SString new.
  )

  benchmark = (
    "Fannkuch new initialize: 1."
    rnd initialize.
    vec initialize: 1.
    benchHarn initialize.
    planner initialize.
```

```

    jsonLit initializeWith: 'null'.
    jsonLit initializeWith: 'true'.
    jsonLit initializeWith: 'false'.
    jsonNum initializeWith: '123'.
    jsonStr initializeWith: '123'.
    jsonP initializeWith: '123'.

    parEx initializeWith: 'msg' at: 1 line: 3 column: 55.

    edge initializeWith: self and: self.
    lexer initialize: ''.

    sblock initialize: self in: self with: self.
    sclass initialize: self.
    sdouble initialize: 0.0.
    sinteger initialize: 10.
    smethod initializeWith: self bc: self literals: self numLocals: 1 maxStack: 1.
    sprimitive initialize: #sym with: self.
    sstring initializeWith: 'www'.
  )

  verifyResult: result = (
    ^ true
  )
)
```

```
"
Copyright (c) 2021 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"
SomParse = Benchmark (
    | classNames first |
    oneTimeSetup = (
      | v |
      first := true.

      v := Vector new: 100.
      self classNames1: v.
      self classNames2: v.
      classNames := v asArray
    )

    classNames1: v = (
      v, #Echo
       , #Hello
       , #QuickSort
       , #Ball
       , #ListElement
       , #JenkinsRandom
       , #NBodyBench
       , #NBody
       , #Body
       , #NBodySystem
       , #Queens
       , #List
       , #BubbleSort
       , #TestHarness
       , #SuperTest
       , #ReflectionTest
       , #ClassStructureTest
       , #BlockTest
       , #SuperTestSuperClass
```

```
       , #EmptyTest
       , #StringTest
       , #ClassLoadingTest
       , #CompilerReturnTest
       , #ClassC
       , #ClassB

       , #SelfBlockTest
       , #HashTest
       , #ClassA
       , #SetTest
       , #GlobalTest
       , #ClosureTest
       , #DoesNotUnderstandMessage
       , #SpecialSelectorsTest
       , #PreliminaryTest
       , #TestRunner
       , #DictionaryTest
       , #VectorTest
       , #TestCase
       , #DoesNotUnderstandTest
       , #CoercionTest
       , #ArrayTest
       , #SystemTest
       , #DoubleTest
       , #BooleanTest
    )

    classNames2: v = (
      v, #SString
       , #SObject
       , #SAbstractObject
       , #SSymbol
       , #SBlock
       , #SDouble
       , #SArray
       , #SPrimitive
       , #SMethod
       , #SClass
       , #SInteger
       , #SystemPrimitives
       , #ClassPrimitives
       , #DoublePrimitives
       , #Primitives
       , #IntegerPrimitives
       , #PrimitivePrimitives
       , #SymbolPrimitives
       , #MethodPrimitives
       , #StringPrimitives
       , #BlockPrimitives
       , #ObjectPrimitives
       , #ArrayPrimitives

       , #Parser
       , #BytecodeGenerator
       , #ClassGenerationContext
       , #Lexer
       , #SourcecodeCompiler
       , #Disassembler
       , #MethodGenerationContext
       , #BasicInterpreterTests
```

```
        , #SomSomTests
        , #SomTests
        , #FrameTests
        , #LexerTests
        , #ParserWithError
        , #ParserTests
    )

    benchmark = (
      classNames do: [:name |
        system load: name ].
      ^ first
    )

    verifyResult: result = (
      first := false.
      "Can only be execute once"
      ^ result
    )
)

```

Examples/Benchmarks/Sort.som

```
"

$Id: Sort.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Sort = Benchmark (

    | smallest largest |

    benchmark = (
        | array |
        array := self randomArray: self dataSize.
        ^ self sort: array.
    )

    dataSize    = ( self subclassResponsibility )
    sort: array = ( self subclassResponsibility )

    verifyResult: array = (
        ((array at: 1) <> smallest)
            || ((array at: (array length)) <> largest)
                ifTrue: [ self error: 'Array is not sorted. smallest: ' + smallest asString + ' largest: ' + largest asString + ' [1]: ' + (array at: 1) asString + ' [l]: ' + (array at: array length) asString ].
        3 to: (array length) do: [ :i |
            (array at: i - 1) > (array at: i)
                ifTrue: [ self error: 'Array is not sorted. [' + i asString + ' - 1]: ' + (array at: i - 1) asString + ' [' + i asString + ']: ' + (array at: i) asString]. ].
        ^ true
    )

    randomArray: size = (
        | array |
```

```
        Random initialize.
        array := Array new: size withAll: [ Random next ].
        smallest := largest := array at: 1.
        array do: [ :elm |
            (elm > largest)  ifTrue: [ largest  := elm ].
            (elm < smallest) ifTrue: [ smallest := elm ]. ].
        ^array
    )

)

```

Examples/Benchmarks/Storage.som

```
"

$Id: Storage.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Storage = Benchmark (

    | count |

    benchmark = (
        Random initialize.
        count := 0.
        self buildTreeDepth: 7.
        ^ count
    )

    verifyResult: result = (
      ^ self assert: 5461 equals: result
    )

    buildTreeDepth: depth = (
        count := count + 1.
        ^(depth = 1)
            ifTrue:  [ Array new: Random next % 10 + 1 ]
            ifFalse: [
                Array new: 4 withAll: [ self buildTreeDepth: depth - 1 ] ]
    )

)
```

Examples/Benchmarks/TestSuite/Array2Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Array2Test = TestCase (
  | a |

  setUp = (
    a := Array new: 3.
    a at: 1 put: 'hello'.
    a at: 2 put: #world.
    a at: 3 put: 23.
  )

  testAt = (
    self assert: 'hello' equals: (a at: 1).
    self assert: #world equals: (a at: 2).
    self assert: 23 equals:     (a at: 3).
  )

  testLength = (
    self assert: 3 equals: a length
  )

  testPutAllBlock = (
    | arr i |
    arr := Array new: 10.

    i := 0.
    arr putAll: [ i := i + 1. i ].
```

```

    1 to: 10 do: [:j |
      self assert: j equals: (arr at: j) ].
  )

  testPutAllInt = (
    | arr i |
    arr := Array new: 10.
    arr putAll: 0.

    1 to: 10 do: [:j |
      self assert: 0 equals: (arr at: j) ].
  )

  testPutAllSym = (
    | arr i |
    arr := Array new: 10.
    arr putAll: #sym.

    1 to: 10 do: [:j |
      self assert: #sym is: (arr at: j) ].
  )

  testFirst = (
    self assert: 'hello' equals: a first.
  )

  testLast = (
    self assert: 23 equals: a last.
  )

  testContains = (
    self assert: (a contains: 23).
    self deny:   (a contains: #notInThere).
  )

  testDo = (
    | j |
    j := 1.

    a do: [:i |
      self assert: (a at: j) is: i.
      j := j + 1.
    ]
  )

  testDoIndexes = (
    | i arr |
    i := 1.
    a doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 4 equals: i.

    arr := Array new: 10.
    i := 1.
    arr doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
```

```
  self assert: 11 equals: i.
)

testFromToDo = (
  | arr i |
  a from: 2 to: 2 do: [:e | self assert: #world is: e ].

  i := 0.
  arr := Array new: 10 withAll: [ i := i + 1. i ].

  i := 3.
  arr from: 3 to: 7 do: [:e |
    self assert: i equals: e.
    i := i + 1 ].

  self assert: 8 equals: i.
)

testSumAndAverage = (
  | arr |
  arr := Array new: 3.
  1 to: 3 do: [ :i | arr at: i put: i ].

  self assert: 6 equals: arr sum.
  self assert: 2 equals: arr average.
)

testCopyFrom = (
  | arr b |
  arr := Array new: 5.
  1 to: 5 do: [ :i | arr at: i put: i ].

  b := arr copyFrom: 2 to: 4.
  self assert: 2 equals: (b at: 1).
  self assert: 3 equals: (b at: 2).
  self assert: 4 equals: (b at: 3).

  b := arr copyFrom: 3.
  self assert: 3 equals: (b at: 1).
  self assert: 4 equals: (b at: 2).
  self assert: 5 equals: (b at: 3).
)

testCopy = (
  | arr |
  arr := a copy.
  self assert: 3 equals: arr length.
  self assert: 'hello' equals: (arr at: 1).
  self assert: #world equals:  (arr at: 2).
  self assert: 23 equals:      (arr at: 3).
)

testReplaceFrom = (
  | arr1 arr2 i |
  arr1 := Array new: 10 withAll: 0.

  i := 0.
  arr2 := Array new: 10 withAll: [ i := i + 1. i ].

  arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.

```

```
    i := 1.
    3 to: 7 do: [:j |
      self assert: i equals: (arr1 at: j).
      i := i + 1.
    ]
  )

  testExtendedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr extendedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a extendedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 4).
  )

  testPrependedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr prependedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a prependedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 1).
  )

  testIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

```

```
    self assert: 2 equals: (arr indexOf: #two).
    self assert: 4 equals: (arr indexOf: #four).
    self assert: 5 equals: (arr indexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 1 equals: (arr indexOf: #one).
  )

  testLastIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

    self assert: 2 equals: (arr lastIndexOf: #two).
    self assert: 4 equals: (arr lastIndexOf: #four).
    self assert: 5 equals: (arr lastIndexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 6 equals: (arr lastIndexOf: #one).
  )

  testCollect = (
    | arr i col |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].
    col := arr collect: [:e | e + 1 ].

    self assert: 10 equals: col length.

    1 to: 10 do: [:i |
      self assert: i + 1 equals: (col at: i) ].
  )

  testInject = (
    | arr result |
    arr := Array new: 10 withAll: 1.

    result := arr inject: 100 into: [:sum :e | sum + e ].

    self assert: 110 equals: result.
  )

  testReject = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr reject: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 1 equals: (result at: 1).
    self assert: 3 equals: (result at: 2).
    self assert: 5 equals: (result at: 3).
    self assert: 7 equals: (result at: 4).
```

```
    self assert: 9 equals: (result at: 5).
  )

  testRejectEmpty = (
    | result |
    result := (Array new: 0) reject: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.
  )

  testDontRejectAny = (
    | result |
    result := (Array new: 10 withAll: 0) reject: [:e | false ].
    self assert: 10 equals: result size.

    self assert: 0 equals: (result at: 5).
  )

  testSelect = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr select: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 2 equals: (result at: 1).
    self assert: 4 equals: (result at: 2).
    self assert: 6 equals: (result at: 3).
    self assert: 8 equals: (result at: 4).
    self assert: 10 equals: (result at: 5).
  )

  testSelectEmpty = (
    | result |
    result := (Array new: 0) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) select: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.
  )

  testSelectAll = (
    | result |
    result := (Array new: 10 withAll: 0) select: [:e | true ].
    self assert: 10 equals: result size.

```

```
    self assert: 0 equals: (result at: 5).
  )

  testUnion = (
    | result |
    result := a union: a.
    self assert: 3 equals: result size.

    self assert: (result contains: #world).
    self assert: (result contains: 23).

    result := a union: #(21 22 23 #world).

    self assert: 5 equals: result size.
    self assert: (result contains: 21).
    self assert: (result contains: 22).
    self assert: (result contains: 23).
    self assert: (result contains: #world).
  )

  testNewWithAll = (
    | arr |
    arr := Array new: 5 withAll: [1].
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].

    arr := Array new: 5 withAll: 1.
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
  )

  testNewWithAllIntAndObjects = (
    | arr o |
    arr := Array new: 5 withAll: 5.
    self assert: 5 equals: (arr at: 3).
    arr at: 3 put: nil.
    self assert: nil equals: (arr at: 3).

    o := Object new.
    arr at: 2 put: o.
    self assert: o equals: (arr at: 2).
  )

  testLiteralArrays = (
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢  ’ W V Ç3¢  à
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢ "’ W V Ç3¢ "à

  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢  ’ W V Ç3¢ Ó à
  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢ "’ W V Ç3¢ Ó#  ã à
  )

  testJoin = (
    | arr |
    arr := Array new: 0.
    self assert: nil is: (arr join: ', ').

    arr := Array with: 1 with: 10 with: 100.
    self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).

    arr := Array with: 'a' with: 'b' with: 'c'.
    self assert: 'a, b, c' equals: (arr join: ', ').
  )
)
```

Examples/Benchmarks/TestSuite/Array3Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Array3Test = TestCase (
  | a |

  setUp = (
    a := Array new: 3.
    a at: 1 put: 'hello'.
    a at: 2 put: #world.
    a at: 3 put: 23.
  )

  testAt = (
    self assert: 'hello' equals: (a at: 1).
    self assert: #world equals: (a at: 2).
    self assert: 23 equals:     (a at: 3).
  )

  testLength = (
    self assert: 3 equals: a length
  )

  testPutAllBlock = (
    | arr i |
    arr := Array new: 10.

    i := 0.
    arr putAll: [ i := i + 1. i ].
```

```

    1 to: 10 do: [:j |
      self assert: j equals: (arr at: j) ].
  )

  testPutAllInt = (
    | arr i |
    arr := Array new: 10.
    arr putAll: 0.

    1 to: 10 do: [:j |
      self assert: 0 equals: (arr at: j) ].
  )

  testPutAllSym = (
    | arr i |
    arr := Array new: 10.
    arr putAll: #sym.

    1 to: 10 do: [:j |
      self assert: #sym is: (arr at: j) ].
  )

  testFirst = (
    self assert: 'hello' equals: a first.
  )

  testLast = (
    self assert: 23 equals: a last.
  )

  testContains = (
    self assert: (a contains: 23).
    self deny:   (a contains: #notInThere).
  )

  testDo = (
    | j |
    j := 1.

    a do: [:i |
      self assert: (a at: j) is: i.
      j := j + 1.
    ]
  )

  testDoIndexes = (
    | i arr |
    i := 1.
    a doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 4 equals: i.

    arr := Array new: 10.
    i := 1.
    arr doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
```

```
    self assert: 11 equals: i.
  )

  testFromToDo = (
    | arr i |
    a from: 2 to: 2 do: [:e | self assert: #world is: e ].

    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    i := 3.
    arr from: 3 to: 7 do: [:e |
      self assert: i equals: e.
      i := i + 1 ].

    self assert: 8 equals: i.
  )

  testSumAndAverage = (
    | arr |
    arr := Array new: 3.
    1 to: 3 do: [ :i | arr at: i put: i ].

    self assert: 6 equals: arr sum.
    self assert: 2 equals: arr average.
  )

  testCopyFrom = (
    | arr b |
    arr := Array new: 5.
    1 to: 5 do: [ :i | arr at: i put: i ].

    b := arr copyFrom: 2 to: 4.
    self assert: 2 equals: (b at: 1).
    self assert: 3 equals: (b at: 2).
    self assert: 4 equals: (b at: 3).

    b := arr copyFrom: 3.
    self assert: 3 equals: (b at: 1).
    self assert: 4 equals: (b at: 2).
    self assert: 5 equals: (b at: 3).
  )

  testCopy = (
    | arr |
    arr := a copy.
    self assert: 3 equals: arr length.
    self assert: 'hello' equals: (arr at: 1).
    self assert: #world equals:  (arr at: 2).
    self assert: 23 equals:      (arr at: 3).
  )

  testReplaceFrom = (
    | arr1 arr2 i |
    arr1 := Array new: 10 withAll: 0.

    i := 0.
    arr2 := Array new: 10 withAll: [ i := i + 1. i ].

    arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.

```

```
    i := 1.
    3 to: 7 do: [:j |
      self assert: i equals: (arr1 at: j).
      i := i + 1.
    ]
  )

  testExtendedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr extendedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a extendedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 4).
  )

  testPrependedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr prependedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a prependedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 1).
  )

  testIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.
```

```
    self assert: 2 equals: (arr indexOf: #two).
    self assert: 4 equals: (arr indexOf: #four).
    self assert: 5 equals: (arr indexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 1 equals: (arr indexOf: #one).
  )

  testLastIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

    self assert: 2 equals: (arr lastIndexOf: #two).
    self assert: 4 equals: (arr lastIndexOf: #four).
    self assert: 5 equals: (arr lastIndexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 6 equals: (arr lastIndexOf: #one).
  )

  testCollect = (
    | arr i col |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].
    col := arr collect: [:e | e + 1 ].

    self assert: 10 equals: col length.

    1 to: 10 do: [:i |
      self assert: i + 1 equals: (col at: i) ].
  )

  testInject = (
    | arr result |
    arr := Array new: 10 withAll: 1.

    result := arr inject: 100 into: [:sum :e | sum + e ].

    self assert: 110 equals: result.
  )

  testReject = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr reject: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 1 equals: (result at: 1).
    self assert: 3 equals: (result at: 2).
    self assert: 5 equals: (result at: 3).
    self assert: 7 equals: (result at: 4).
```

```
    self assert: 9 equals: (result at: 5).
  )

  testRejectEmpty = (
    | result |
    result := (Array new: 0) reject: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.
  )

  testDontRejectAny = (
    | result |
    result := (Array new: 10 withAll: 0) reject: [:e | false ].
    self assert: 10 equals: result size.

    self assert: 0 equals: (result at: 5).
  )

  testSelect = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr select: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 2 equals: (result at: 1).
    self assert: 4 equals: (result at: 2).
    self assert: 6 equals: (result at: 3).
    self assert: 8 equals: (result at: 4).
    self assert: 10 equals: (result at: 5).
  )

  testSelectEmpty = (
    | result |
    result := (Array new: 0) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) select: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.
  )

  testSelectAll = (
    | result |
    result := (Array new: 10 withAll: 0) select: [:e | true ].
    self assert: 10 equals: result size.

```

```
    self assert: 0 equals: (result at: 5).
  )

  testUnion = (
    | result |
    result := a union: a.
    self assert: 3 equals: result size.

    self assert: (result contains: #world).
    self assert: (result contains: 23).

    result := a union: #(21 22 23 #world).

    self assert: 5 equals: result size.
    self assert: (result contains: 21).
    self assert: (result contains: 22).
    self assert: (result contains: 23).
    self assert: (result contains: #world).
  )

  testNewWithAll = (
    | arr |
    arr := Array new: 5 withAll: [1].
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].

    arr := Array new: 5 withAll: 1.
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
  )

  testNewWithAllIntAndObjects = (
    | arr o |
    arr := Array new: 5 withAll: 5.
    self assert: 5 equals: (arr at: 3).
    arr at: 3 put: nil.
    self assert: nil equals: (arr at: 3).

    o := Object new.
    arr at: 2 put: o.
    self assert: o equals: (arr at: 2).
  )

  testLiteralArrays = (
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢  ’ W V Ç3¢  à
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢ "’ W V Ç3¢ "à

  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢  ’ W V Ç3¢ Ó à
  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢ "’ W V Ç3¢ Ó#  ã à
  )

  testJoin = (
    | arr |
    arr := Array new: 0.
    self assert: nil is: (arr join: ', ').

    arr := Array with: 1 with: 10 with: 100.
    self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).

    arr := Array with: 'a' with: 'b' with: 'c'.
    self assert: 'a, b, c' equals: (arr join: ', ').
  )
)
```

Examples/Benchmarks/TestSuite/Array4Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Array4Test = TestCase (
  | a |

  setUp = (
    a := Array new: 3.
    a at: 1 put: 'hello'.
    a at: 2 put: #world.
    a at: 3 put: 23.
  )

  testAt = (
    self assert: 'hello' equals: (a at: 1).
    self assert: #world equals: (a at: 2).
    self assert: 23 equals:     (a at: 3).
  )

  testLength = (
    self assert: 3 equals: a length
  )

  testPutAllBlock = (
    | arr i |
    arr := Array new: 10.

    i := 0.
    arr putAll: [ i := i + 1. i ].
```

```

    1 to: 10 do: [:j |
      self assert: j equals: (arr at: j) ].
  )

  testPutAllInt = (
    | arr i |
    arr := Array new: 10.
    arr putAll: 0.

    1 to: 10 do: [:j |
      self assert: 0 equals: (arr at: j) ].
  )

  testPutAllSym = (
    | arr i |
    arr := Array new: 10.
    arr putAll: #sym.

    1 to: 10 do: [:j |
      self assert: #sym is: (arr at: j) ].
  )

  testFirst = (
    self assert: 'hello' equals: a first.
  )

  testLast = (
    self assert: 23 equals: a last.
  )

  testContains = (
    self assert: (a contains: 23).
    self deny:   (a contains: #notInThere).
  )

  testDo = (
    | j |
    j := 1.

    a do: [:i |
      self assert: (a at: j) is: i.
      j := j + 1.
    ]
  )

  testDoIndexes = (
    | i arr |
    i := 1.
    a doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 4 equals: i.

    arr := Array new: 10.
    i := 1.
    arr doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
```

```
    self assert: 11 equals: i.
  )

  testFromToDo = (
    | arr i |
    a from: 2 to: 2 do: [:e | self assert: #world is: e ].

    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    i := 3.
    arr from: 3 to: 7 do: [:e |
      self assert: i equals: e.
      i := i + 1 ].

    self assert: 8 equals: i.
  )

  testSumAndAverage = (
    | arr |
    arr := Array new: 3.
    1 to: 3 do: [ :i | arr at: i put: i ].

    self assert: 6 equals: arr sum.
    self assert: 2 equals: arr average.
  )

  testCopyFrom = (
    | arr b |
    arr := Array new: 5.
    1 to: 5 do: [ :i | arr at: i put: i ].

    b := arr copyFrom: 2 to: 4.
    self assert: 2 equals: (b at: 1).
    self assert: 3 equals: (b at: 2).
    self assert: 4 equals: (b at: 3).

    b := arr copyFrom: 3.
    self assert: 3 equals: (b at: 1).
    self assert: 4 equals: (b at: 2).
    self assert: 5 equals: (b at: 3).
  )

  testCopy = (
    | arr |
    arr := a copy.
    self assert: 3 equals: arr length.
    self assert: 'hello' equals: (arr at: 1).
    self assert: #world equals:  (arr at: 2).
    self assert: 23 equals:      (arr at: 3).
  )

  testReplaceFrom = (
    | arr1 arr2 i |
    arr1 := Array new: 10 withAll: 0.

    i := 0.
    arr2 := Array new: 10 withAll: [ i := i + 1. i ].

    arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.

```

```
    i := 1.
    3 to: 7 do: [:j |
      self assert: i equals: (arr1 at: j).
      i := i + 1.
    ]
  )

  testExtendedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr extendedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a extendedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 4).
  )

  testPrependedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr prependedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a prependedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 1).
  )

  testIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

```

```
    self assert: 2 equals: (arr indexOf: #two).
    self assert: 4 equals: (arr indexOf: #four).
    self assert: 5 equals: (arr indexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 1 equals: (arr indexOf: #one).
  )

  testLastIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

    self assert: 2 equals: (arr lastIndexOf: #two).
    self assert: 4 equals: (arr lastIndexOf: #four).
    self assert: 5 equals: (arr lastIndexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 6 equals: (arr lastIndexOf: #one).
  )

  testCollect = (
    | arr i col |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].
    col := arr collect: [:e | e + 1 ].

    self assert: 10 equals: col length.

    1 to: 10 do: [:i |
      self assert: i + 1 equals: (col at: i) ].
  )

  testInject = (
    | arr result |
    arr := Array new: 10 withAll: 1.

    result := arr inject: 100 into: [:sum :e | sum + e ].

    self assert: 110 equals: result.
  )

  testReject = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr reject: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 1 equals: (result at: 1).
    self assert: 3 equals: (result at: 2).
    self assert: 5 equals: (result at: 3).
    self assert: 7 equals: (result at: 4).
```

```
    self assert: 9 equals: (result at: 5).
  )

  testRejectEmpty = (
    | result |
    result := (Array new: 0) reject: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.
  )

  testDontRejectAny = (
    | result |
    result := (Array new: 10 withAll: 0) reject: [:e | false ].
    self assert: 10 equals: result size.

    self assert: 0 equals: (result at: 5).
  )

  testSelect = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr select: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 2 equals: (result at: 1).
    self assert: 4 equals: (result at: 2).
    self assert: 6 equals: (result at: 3).
    self assert: 8 equals: (result at: 4).
    self assert: 10 equals: (result at: 5).
  )

  testSelectEmpty = (
    | result |
    result := (Array new: 0) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) select: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.
  )

  testSelectAll = (
    | result |
    result := (Array new: 10 withAll: 0) select: [:e | true ].
    self assert: 10 equals: result size.

```

```
    self assert: 0 equals: (result at: 5).
  )

  testUnion = (
    | result |
    result := a union: a.
    self assert: 3 equals: result size.

    self assert: (result contains: #world).
    self assert: (result contains: 23).

    result := a union: #(21 22 23 #world).

    self assert: 5 equals: result size.
    self assert: (result contains: 21).
    self assert: (result contains: 22).
    self assert: (result contains: 23).
    self assert: (result contains: #world).
  )

  testNewWithAll = (
    | arr |
    arr := Array new: 5 withAll: [1].
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].

    arr := Array new: 5 withAll: 1.
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
  )

  testNewWithAllIntAndObjects = (
    | arr o |
    arr := Array new: 5 withAll: 5.
    self assert: 5 equals: (arr at: 3).
    arr at: 3 put: nil.
    self assert: nil equals: (arr at: 3).

    o := Object new.
    arr at: 2 put: o.
    self assert: o equals: (arr at: 2).
  )

  testLiteralArrays = (
  –6VÆb  76W'C¢ ,2ƒ  "’  C¢  ’ W V Ç3¢  à
  –6VÆb  76W'C¢ ,2ƒ  "’  C¢ "’ W V Ç3¢ "à

  –6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢  ’ W V Ç3¢ Ó à
  –6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢ "’ W V Ç3¢ Ó#  ã à
  )

  testJoin = (
    | arr |
    arr := Array new: 0.
    self assert: nil is: (arr join: ', ').

    arr := Array with: 1 with: 10 with: 100.
    self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).

    arr := Array with: 'a' with: 'b' with: 'c'.
    self assert: 'a, b, c' equals: (arr join: ', ').
  )
)
```

Examples/Benchmarks/TestSuite/Array5Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Array5Test = TestCase (
  | a |

  setUp = (
    a := Array new: 3.
    a at: 1 put: 'hello'.
    a at: 2 put: #world.
    a at: 3 put: 23.
  )

  testAt = (
    self assert: 'hello' equals: (a at: 1).
    self assert: #world equals: (a at: 2).
    self assert: 23 equals:     (a at: 3).
  )

  testLength = (
    self assert: 3 equals: a length
  )

  testPutAllBlock = (
    | arr i |
    arr := Array new: 10.

    i := 0.
    arr putAll: [ i := i + 1. i ].
```

```

    1 to: 10 do: [:j |
      self assert: j equals: (arr at: j) ].
  )

  testPutAllInt = (
    | arr i |
    arr := Array new: 10.
    arr putAll: 0.

    1 to: 10 do: [:j |
      self assert: 0 equals: (arr at: j) ].
  )

  testPutAllSym = (
    | arr i |
    arr := Array new: 10.
    arr putAll: #sym.

    1 to: 10 do: [:j |
      self assert: #sym is: (arr at: j) ].
  )

  testFirst = (
    self assert: 'hello' equals: a first.
  )

  testLast = (
    self assert: 23 equals: a last.
  )

  testContains = (
    self assert: (a contains: 23).
    self deny:   (a contains: #notInThere).
  )

  testDo = (
    | j |
    j := 1.

    a do: [:i |
      self assert: (a at: j) is: i.
      j := j + 1.
    ]
  )

  testDoIndexes = (
    | i arr |
    i := 1.
    a doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 4 equals: i.

    arr := Array new: 10.
    i := 1.
    arr doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
```

```
    self assert: 11 equals: i.
  )

  testFromToDo = (
    | arr i |
    a from: 2 to: 2 do: [:e | self assert: #world is: e ].

    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    i := 3.
    arr from: 3 to: 7 do: [:e |
      self assert: i equals: e.
      i := i + 1 ].

    self assert: 8 equals: i.
  )

  testSumAndAverage = (
    | arr |
    arr := Array new: 3.
    1 to: 3 do: [ :i | arr at: i put: i ].

    self assert: 6 equals: arr sum.
    self assert: 2 equals: arr average.
  )

  testCopyFrom = (
    | arr b |
    arr := Array new: 5.
    1 to: 5 do: [ :i | arr at: i put: i ].

    b := arr copyFrom: 2 to: 4.
    self assert: 2 equals: (b at: 1).
    self assert: 3 equals: (b at: 2).
    self assert: 4 equals: (b at: 3).

    b := arr copyFrom: 3.
    self assert: 3 equals: (b at: 1).
    self assert: 4 equals: (b at: 2).
    self assert: 5 equals: (b at: 3).
  )

  testCopy = (
    | arr |
    arr := a copy.
    self assert: 3 equals: arr length.
    self assert: 'hello' equals: (arr at: 1).
    self assert: #world equals:  (arr at: 2).
    self assert: 23 equals:      (arr at: 3).
  )

  testReplaceFrom = (
    | arr1 arr2 i |
    arr1 := Array new: 10 withAll: 0.

    i := 0.
    arr2 := Array new: 10 withAll: [ i := i + 1. i ].

    arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.

```

```
    i := 1.
    3 to: 7 do: [:j |
      self assert: i equals: (arr1 at: j).
      i := i + 1.
    ]
  )

  testExtendedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr extendedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a extendedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 4).
  )

  testPrependedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr prependedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a prependedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 1).
  )

  testIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

```

```
    self assert: 2 equals: (arr indexOf: #two).
    self assert: 4 equals: (arr indexOf: #four).
    self assert: 5 equals: (arr indexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 1 equals: (arr indexOf: #one).
  )

  testLastIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

    self assert: 2 equals: (arr lastIndexOf: #two).
    self assert: 4 equals: (arr lastIndexOf: #four).
    self assert: 5 equals: (arr lastIndexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 6 equals: (arr lastIndexOf: #one).
  )

  testCollect = (
    | arr i col |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].
    col := arr collect: [:e | e + 1 ].

    self assert: 10 equals: col length.

    1 to: 10 do: [:i |
      self assert: i + 1 equals: (col at: i) ].
  )

  testInject = (
    | arr result |
    arr := Array new: 10 withAll: 1.

    result := arr inject: 100 into: [:sum :e | sum + e ].

    self assert: 110 equals: result.
  )

  testReject = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr reject: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 1 equals: (result at: 1).
    self assert: 3 equals: (result at: 2).
    self assert: 5 equals: (result at: 3).
    self assert: 7 equals: (result at: 4).
```

```
    self assert: 9 equals: (result at: 5).
  )

  testRejectEmpty = (
    | result |
    result := (Array new: 0) reject: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.
  )

  testDontRejectAny = (
    | result |
    result := (Array new: 10 withAll: 0) reject: [:e | false ].
    self assert: 10 equals: result size.

    self assert: 0 equals: (result at: 5).
  )

  testSelect = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr select: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 2 equals: (result at: 1).
    self assert: 4 equals: (result at: 2).
    self assert: 6 equals: (result at: 3).
    self assert: 8 equals: (result at: 4).
    self assert: 10 equals: (result at: 5).
  )

  testSelectEmpty = (
    | result |
    result := (Array new: 0) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) select: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.
  )

  testSelectAll = (
    | result |
    result := (Array new: 10 withAll: 0) select: [:e | true ].
    self assert: 10 equals: result size.

```

```
    self assert: 0 equals: (result at: 5).
  )

  testUnion = (
    | result |
    result := a union: a.
    self assert: 3 equals: result size.

    self assert: (result contains: #world).
    self assert: (result contains: 23).

    result := a union: #(21 22 23 #world).

    self assert: 5 equals: result size.
    self assert: (result contains: 21).
    self assert: (result contains: 22).
    self assert: (result contains: 23).
    self assert: (result contains: #world).
  )

  testNewWithAll = (
    | arr |
    arr := Array new: 5 withAll: [1].
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].

    arr := Array new: 5 withAll: 1.
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
  )

  testNewWithAllIntAndObjects = (
    | arr o |
    arr := Array new: 5 withAll: 5.
    self assert: 5 equals: (arr at: 3).
    arr at: 3 put: nil.
    self assert: nil equals: (arr at: 3).

    o := Object new.
    arr at: 2 put: o.
    self assert: o equals: (arr at: 2).
  )

  testLiteralArrays = (
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢  ’ W V Ç3¢  à
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢ "’ W V Ç3¢ "à

  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢  ’ W V Ç3¢ Ó à
  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢ "’ W V Ç3¢ Ó#  ã à
  )

  testJoin = (
    | arr |
    arr := Array new: 0.
    self assert: nil is: (arr join: ', ').

    arr := Array with: 1 with: 10 with: 100.
    self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).

    arr := Array with: 'a' with: 'b' with: 'c'.
    self assert: 'a, b, c' equals: (arr join: ', ').
  )
)
```

Examples/Benchmarks/TestSuite/ArrayTest.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

ArrayTest = TestCase (
  | a |

  setUp = (
    a := Array new: 3.
    a at: 1 put: 'hello'.
    a at: 2 put: #world.
    a at: 3 put: 23.
  )

  testAt = (
    self assert: 'hello' equals: (a at: 1).
    self assert: #world equals: (a at: 2).
    self assert: 23 equals:     (a at: 3).
  )

  testLength = (
    self assert: 3 equals: a length
  )

  testPutAllBlock = (
    | arr i |
    arr := Array new: 10.

    i := 0.
    arr putAll: [ i := i + 1. i ].
```

```

    1 to: 10 do: [:j |
      self assert: j equals: (arr at: j) ].
  )

  testPutAllInt = (
    | arr i |
    arr := Array new: 10.
    arr putAll: 0.

    1 to: 10 do: [:j |
      self assert: 0 equals: (arr at: j) ].
  )

  testPutAllSym = (
    | arr i |
    arr := Array new: 10.
    arr putAll: #sym.

    1 to: 10 do: [:j |
      self assert: #sym is: (arr at: j) ].
  )

  testFirst = (
    self assert: 'hello' equals: a first.
  )

  testLast = (
    self assert: 23 equals: a last.
  )

  testContains = (
    self assert: (a contains: 23).
    self deny:   (a contains: #notInThere).
  )

  testDo = (
    | j |
    j := 1.

    a do: [:i |
      self assert: (a at: j) is: i.
      j := j + 1.
    ]
  )

  testDoIndexes = (
    | i arr |
    i := 1.
    a doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 4 equals: i.

    arr := Array new: 10.
    i := 1.
    arr doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
```

```
    self assert: 11 equals: i.
  )

  testFromToDo = (
    | arr i |
    a from: 2 to: 2 do: [:e | self assert: #world is: e ].

    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    i := 3.
    arr from: 3 to: 7 do: [:e |
      self assert: i equals: e.
      i := i + 1 ].

    self assert: 8 equals: i.
  )

  testSumAndAverage = (
    | arr |
    arr := Array new: 3.
    1 to: 3 do: [ :i | arr at: i put: i ].

    self assert: 6 equals: arr sum.
    self assert: 2 equals: arr average.
  )

  testCopyFrom = (
    | arr b |
    arr := Array new: 5.
    1 to: 5 do: [ :i | arr at: i put: i ].

    b := arr copyFrom: 2 to: 4.
    self assert: 2 equals: (b at: 1).
    self assert: 3 equals: (b at: 2).
    self assert: 4 equals: (b at: 3).

    b := arr copyFrom: 3.
    self assert: 3 equals: (b at: 1).
    self assert: 4 equals: (b at: 2).
    self assert: 5 equals: (b at: 3).
  )

  testCopy = (
    | arr |
    arr := a copy.
    self assert: 3 equals: arr length.
    self assert: 'hello' equals: (arr at: 1).
    self assert: #world equals:  (arr at: 2).
    self assert: 23 equals:      (arr at: 3).
  )

  testReplaceFrom = (
    | arr1 arr2 i |
    arr1 := Array new: 10 withAll: 0.

    i := 0.
    arr2 := Array new: 10 withAll: [ i := i + 1. i ].

    arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.

```

```
    i := 1.
    3 to: 7 do: [:j |
      self assert: i equals: (arr1 at: j).
      i := i + 1.
    ]
  )

  testExtendedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr extendedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a extendedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 4).
  )

  testPrependedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr prependedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a prependedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 1).
  )

  testIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

```

```
    self assert: 2 equals: (arr indexOf: #two).
    self assert: 4 equals: (arr indexOf: #four).
    self assert: 5 equals: (arr indexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 1 equals: (arr indexOf: #one).
  )

  testLastIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

    self assert: 2 equals: (arr lastIndexOf: #two).
    self assert: 4 equals: (arr lastIndexOf: #four).
    self assert: 5 equals: (arr lastIndexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 6 equals: (arr lastIndexOf: #one).
  )

  testCollect = (
    | arr i col |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].
    col := arr collect: [:e | e + 1 ].

    self assert: 10 equals: col length.

    1 to: 10 do: [:i |
      self assert: i + 1 equals: (col at: i) ].
  )

  testInject = (
    | arr result |
    arr := Array new: 10 withAll: 1.

    result := arr inject: 100 into: [:sum :e | sum + e ].

    self assert: 110 equals: result.
  )

  testReject = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr reject: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 1 equals: (result at: 1).
    self assert: 3 equals: (result at: 2).
    self assert: 5 equals: (result at: 3).
    self assert: 7 equals: (result at: 4).
```

```
    self assert: 9 equals: (result at: 5).
  )

  testRejectEmpty = (
    | result |
    result := (Array new: 0) reject: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.
  )

  testDontRejectAny = (
    | result |
    result := (Array new: 10 withAll: 0) reject: [:e | false ].
    self assert: 10 equals: result size.

    self assert: 0 equals: (result at: 5).
  )

  testSelect = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr select: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 2 equals: (result at: 1).
    self assert: 4 equals: (result at: 2).
    self assert: 6 equals: (result at: 3).
    self assert: 8 equals: (result at: 4).
    self assert: 10 equals: (result at: 5).
  )

  testSelectEmpty = (
    | result |
    result := (Array new: 0) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) select: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.
  )

  testSelectAll = (
    | result |
    result := (Array new: 10 withAll: 0) select: [:e | true ].
    self assert: 10 equals: result size.

```

```
    self assert: 0 equals: (result at: 5).
  )

  testUnion = (
    | result |
    result := a union: a.
    self assert: 3 equals: result size.

    self assert: (result contains: #world).
    self assert: (result contains: 23).

    result := a union: #(21 22 23 #world).

    self assert: 5 equals: result size.
    self assert: (result contains: 21).
    self assert: (result contains: 22).
    self assert: (result contains: 23).
    self assert: (result contains: #world).
  )

  testNewWithAll = (
    | arr |
    arr := Array new: 5 withAll: [1].
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].

    arr := Array new: 5 withAll: 1.
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
  )

  testNewWithAllIntAndObjects = (
    | arr o |
    arr := Array new: 5 withAll: 5.
    self assert: 5 equals: (arr at: 3).
    arr at: 3 put: nil.
    self assert: nil equals: (arr at: 3).

    o := Object new.
    arr at: 2 put: o.
    self assert: o equals: (arr at: 2).
  )

  testLiteralArrays = (
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢  ’ W V Ç3¢  à
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢ "’ W V Ç3¢ "à

  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢  ’ W V Ç3¢ Ó à
  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢ "’ W V Ç3¢ Ó#  ã à
  )

  testJoin = (
    | arr |
    arr := Array new: 0.
    self assert: nil is: (arr join: ', ').

    arr := Array with: 1 with: 10 with: 100.
    self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).

    arr := Array with: 'a' with: 'b' with: 'c'.
    self assert: 'a, b, c' equals: (arr join: ', ').
  )
)
```

Examples/Benchmarks/TestSuite/Block2Test.som

```
Block2Test = TestCase (
    |escape_count escaped_block|

    simpleBlock = (
      ^ [ 42 ]
    )

    incBlock = (
      ^ [:val | val + 1 ]
    )

    "This requires a closure"
    adderBlock: amount = (
      ^ [:val | amount + val ]
    )

    "Closure with mutable state in block"
    counterBlock = (
      |count|
      count := 0.
      ^ [ count := count + 1. count ]
    )

    selfKeeper = (
      ^ [ self ]
    )

    escapingBlock = (
      ^ [ ^ 42 ]
    )

    escapingNestedBlock = (
      [ [ ^ [ ^ 43 ] ] value ] value
    )

    testSimpleBlocks = (
      self assert: 42 equals: self simpleBlock value.
      self assert: 4  equals: (self incBlock value: 3).
      self assert: 43 equals: ((self adderBlock: 13) value: 30).
    )

    testClosure = (
      | counter |
      counter := self counterBlock.
      self assert: 1 equals: counter value.
      self assert: 2 equals: counter value.
      self assert: 1 equals: self counterBlock value. "make sure each copy is independent"
      self assert: 3 equals: counter value.
    )

    testSelfInBlock = (
      | test_inst |
      test_inst := BlockTest new.
      self assert: test_inst equals: test_inst selfKeeper value.
      self assert: self      equals: self selfKeeper value.
    )

```

```
    testEscapedBlock = (
      | escaping_block |

      escape_count := 0.

      escaping_block := self escapingBlock.

      self assert: 0 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 1 equals: escape_count.

      self assert: escaping_block is: escaped_block.


      escaping_block := self escapingNestedBlock.
      self assert: 1 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 2 equals: escape_count.
      self assert: escaping_block is: escaped_block.
    )

    escapedBlock: block = (
      escape_count := escape_count + 1.
      escaped_block := block.

      "return some dummy value to the object that sent 'value' to block"
      ^666
    )

    testWhileTrue = (
      | i |
      i := 1.
      [ i < 10 ] whileTrue: [ i := i + 1 ].
      self assert: 10 equals: i.
    )

    testWhileFalse = (
      | i |
      i := 1.
      [ i >= 10 ] whileFalse: [ i := i + 1 ].
      self assert: 10 equals: i.
    )
)
```

Examples/Benchmarks/TestSuite/Block3Test.som

```
Block3Test = TestCase (
    |escape_count escaped_block|

    simpleBlock = (
      ^ [ 42 ]
    )

    incBlock = (
      ^ [:val | val + 1 ]
    )

    "This requires a closure"
    adderBlock: amount = (
      ^ [:val | amount + val ]
    )

    "Closure with mutable state in block"
    counterBlock = (
      |count|
      count := 0.
      ^ [ count := count + 1. count ]
    )

    selfKeeper = (
      ^ [ self ]
    )

    escapingBlock = (
      ^ [ ^ 42 ]
    )

    escapingNestedBlock = (
      [ [ ^ [ ^ 43 ] ] value ] value
    )

    testSimpleBlocks = (
      self assert: 42 equals: self simpleBlock value.
      self assert: 4  equals: (self incBlock value: 3).
      self assert: 43 equals: ((self adderBlock: 13) value: 30).
    )

    testClosure = (
      | counter |
      counter := self counterBlock.
      self assert: 1 equals: counter value.
      self assert: 2 equals: counter value.
      self assert: 1 equals: self counterBlock value. "make sure each copy is independent"
      self assert: 3 equals: counter value.
    )

    testSelfInBlock = (
      | test_inst |
      test_inst := BlockTest new.
      self assert: test_inst equals: test_inst selfKeeper value.
      self assert: self      equals: self selfKeeper value.
    )

```

```
    testEscapedBlock = (
      | escaping_block |

      escape_count := 0.

      escaping_block := self escapingBlock.

      self assert: 0 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 1 equals: escape_count.

      self assert: escaping_block is: escaped_block.


      escaping_block := self escapingNestedBlock.
      self assert: 1 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 2 equals: escape_count.
      self assert: escaping_block is: escaped_block.
    )

    escapedBlock: block = (
      escape_count := escape_count + 1.
      escaped_block := block.

      "return some dummy value to the object that sent 'value' to block"
      ^666
    )

    testWhileTrue = (
      | i |
      i := 1.
      [ i < 10 ] whileTrue: [ i := i + 1 ].
      self assert: 10 equals: i.
    )

    testWhileFalse = (
      | i |
      i := 1.
      [ i >= 10 ] whileFalse: [ i := i + 1 ].
      self assert: 10 equals: i.
    )
)

```

```
Block4Test = TestCase (
    |escape_count escaped_block|

    simpleBlock = (
      ^ [ 42 ]
    )

    incBlock = (
      ^ [:val | val + 1 ]
    )

    "This requires a closure"
    adderBlock: amount = (
      ^ [:val | amount + val ]
    )

    "Closure with mutable state in block"
    counterBlock = (
      |count|
      count := 0.
      ^ [ count := count + 1. count ]
    )

    selfKeeper = (
      ^ [ self ]
    )

    escapingBlock = (
      ^ [ ^ 42 ]
    )

    escapingNestedBlock = (
      [ [ ^ [ ^ 43 ] ] value ] value
    )

    testSimpleBlocks = (
      self assert: 42 equals: self simpleBlock value.
      self assert: 4  equals: (self incBlock value: 3).
      self assert: 43 equals: ((self adderBlock: 13) value: 30).
    )

    testClosure = (
      | counter |
      counter := self counterBlock.
      self assert: 1 equals: counter value.
      self assert: 2 equals: counter value.
      self assert: 1 equals: self counterBlock value. "make sure each copy is independent"
      self assert: 3 equals: counter value.
    )

    testSelfInBlock = (
      | test_inst |
      test_inst := BlockTest new.
      self assert: test_inst equals: test_inst selfKeeper value.
      self assert: self      equals: self selfKeeper value.
    )

```

```
    testEscapedBlock = (
      | escaping_block |

      escape_count := 0.

      escaping_block := self escapingBlock.

      self assert: 0 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 1 equals: escape_count.

      self assert: escaping_block is: escaped_block.


      escaping_block := self escapingNestedBlock.
      self assert: 1 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 2 equals: escape_count.
      self assert: escaping_block is: escaped_block.
    )

    escapedBlock: block = (
      escape_count := escape_count + 1.
      escaped_block := block.

      "return some dummy value to the object that sent 'value' to block"
      ^666
    )

    testWhileTrue = (
      | i |
      i := 1.
      [ i < 10 ] whileTrue: [ i := i + 1 ].
      self assert: 10 equals: i.
    )

    testWhileFalse = (
      | i |
      i := 1.
      [ i >= 10 ] whileFalse: [ i := i + 1 ].
      self assert: 10 equals: i.
    )
)
```

Examples/Benchmarks/TestSuite/Block5Test.som

```
Block5Test = TestCase (
    |escape_count escaped_block|

    simpleBlock = (
      ^ [ 42 ]
    )

    incBlock = (
      ^ [:val | val + 1 ]
    )

    "This requires a closure"
    adderBlock: amount = (
      ^ [:val | amount + val ]
    )

    "Closure with mutable state in block"
    counterBlock = (
      |count|
      count := 0.
      ^ [ count := count + 1. count ]
    )

    selfKeeper = (
      ^ [ self ]
    )

    escapingBlock = (
      ^ [ ^ 42 ]
    )

    escapingNestedBlock = (
      [ [ ^ [ ^ 43 ] ] value ] value
    )

    testSimpleBlocks = (
      self assert: 42 equals: self simpleBlock value.
      self assert: 4  equals: (self incBlock value: 3).
      self assert: 43 equals: ((self adderBlock: 13) value: 30).
    )

    testClosure = (
      | counter |
      counter := self counterBlock.
      self assert: 1 equals: counter value.
      self assert: 2 equals: counter value.
      self assert: 1 equals: self counterBlock value. "make sure each copy is independent"
      self assert: 3 equals: counter value.
    )

    testSelfInBlock = (
      | test_inst |
      test_inst := BlockTest new.
      self assert: test_inst equals: test_inst selfKeeper value.
      self assert: self      equals: self selfKeeper value.
    )

```

```
    testEscapedBlock = (
      | escaping_block |

      escape_count := 0.

      escaping_block := self escapingBlock.

      self assert: 0 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 1 equals: escape_count.

      self assert: escaping_block is: escaped_block.


      escaping_block := self escapingNestedBlock.
      self assert: 1 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 2 equals: escape_count.
      self assert: escaping_block is: escaped_block.
    )

    escapedBlock: block = (
      escape_count := escape_count + 1.
      escaped_block := block.

      "return some dummy value to the object that sent 'value' to block"
      ^666
    )

    testWhileTrue = (
      | i |
      i := 1.
      [ i < 10 ] whileTrue: [ i := i + 1 ].
      self assert: 10 equals: i.
    )

    testWhileFalse = (
      | i |
      i := 1.
      [ i >= 10 ] whileFalse: [ i := i + 1 ].
      self assert: 10 equals: i.
    )
)
```

Examples/Benchmarks/TestSuite/BlockTest.som

```
BlockTest = TestCase (
    |escape_count escaped_block|

    simpleBlock = (
      ^ [ 42 ]
    )

    incBlock = (
      ^ [:val | val + 1 ]
    )

    "This requires a closure"
    adderBlock: amount = (
      ^ [:val | amount + val ]
    )

    "Closure with mutable state in block"
    counterBlock = (
      |count|
      count := 0.
      ^ [ count := count + 1. count ]
    )

    selfKeeper = (
      ^ [ self ]
    )

    escapingBlock = (
      ^ [ ^ 42 ]
    )

    escapingNestedBlock = (
      [ [ ^ [ ^ 43 ] ] value ] value
    )

    testSimpleBlocks = (
      self assert: 42 equals: self simpleBlock value.
      self assert: 4  equals: (self incBlock value: 3).
      self assert: 43 equals: ((self adderBlock: 13) value: 30).
    )

    testClosure = (
      | counter |
      counter := self counterBlock.
      self assert: 1 equals: counter value.
      self assert: 2 equals: counter value.
      self assert: 1 equals: self counterBlock value. "make sure each copy is independent"
      self assert: 3 equals: counter value.
    )

    testSelfInBlock = (
      | test_inst |
      test_inst := BlockTest new.
      self assert: test_inst equals: test_inst selfKeeper value.
      self assert: self      equals: self selfKeeper value.
    )

```

```
    testEscapedBlock = (
      | escaping_block |

      escape_count := 0.

      escaping_block := self escapingBlock.

      self assert: 0 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 1 equals: escape_count.

      self assert: escaping_block is: escaped_block.


      escaping_block := self escapingNestedBlock.
      self assert: 1 equals: escape_count.
      self assert: 666 equals: escaping_block value.
      self assert: 2 equals: escape_count.
      self assert: escaping_block is: escaped_block.
    )

    escapedBlock: block = (
      escape_count := escape_count + 1.
      escaped_block := block.

      "return some dummy value to the object that sent 'value' to block"
      ^666
    )

    testWhileTrue = (
      | i |
      i := 1.
      [ i < 10 ] whileTrue: [ i := i + 1 ].
      self assert: 10 equals: i.
    )

    testWhileFalse = (
      | i |
      i := 1.
      [ i >= 10 ] whileFalse: [ i := i + 1 ].
      self assert: 10 equals: i.
    )
)
```

```
Boolean2Test = TestCase (

    testIfTrueIfFalse = (
      | b1 b2 |
      b1 := false.
      b2 := false.

      true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b1.
      self deny: b2.

      b1 := false.
      b2 := false.
      false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b2.
      self deny: b1.
    )

    testIfTrue = (
      | b |
      b := false.

      true ifTrue: [ b := true ].
      self assert: b.

      b := false.
      false ifTrue: [ b := true ].
      self deny: b.
    )

    testIfFalse = (
      | b |
      b := false.

      true ifFalse: [ b := true ].
      self deny: b.

      b := false.
      false ifFalse: [ b := true ].
      self assert: b.
    )

    testNot = (
      self deny: true not.
      self assert: false not.
    )

    andBoolTrueTrue   = ( ^ true  and: [ true  ] )
    andBoolTrueFalse  = ( ^ true  and: [ false ] )
    andBoolFalseTrue  = ( ^ false and: [ true  ] )
    andBoolFalseFalse = ( ^ false and: [ false ] )

    testAnd = (
      self assert: self andBoolTrueTrue.
      self deny:   self andBoolTrueFalse.
      self deny:   self andBoolFalseTrue.
      self deny:   self andBoolFalseFalse.
    )
```

```
    ampBoolTrueTrue   = ( ^ true  && [ true  ] )
    ampBoolTrueFalse  = ( ^ true  && [ false ] )
    ampBoolFalseTrue  = ( ^ false && [ true  ] )
    ampBoolFalseFalse = ( ^ false && [ false ] )

    testAmp = (
      self assert: self ampBoolTrueTrue.
      self deny:   self ampBoolTrueFalse.
      self deny:   self ampBoolFalseTrue.
      self deny:   self ampBoolFalseFalse.
    )

    orBoolTrueTrue   = ( ^ true  or: [ true  ] )
    orBoolTrueFalse  = ( ^ true  or: [ false ] )
    orBoolFalseTrue  = ( ^ false or: [ true  ] )
    orBoolFalseFalse = ( ^ false or: [ false ] )

    testOr = (
      self assert: self orBoolTrueTrue.
      self assert: self orBoolTrueFalse.
      self assert: self orBoolFalseTrue.
      self deny:   self orBoolFalseFalse.
    )

    pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
    pipeBoolTrueFalse  = ( ^ true  || [ false ] )
    pipeBoolFalseTrue  = ( ^ false || [ true  ] )
    pipeBoolFalseFalse = ( ^ false || [ false ] )

    testPipe = (
      self assert: self pipeBoolTrueTrue.
      self assert: self pipeBoolTrueFalse.
      self assert: self pipeBoolFalseTrue.
      self deny:   self pipeBoolFalseFalse.
    )

    testAsString = (
      self assert: 'true'  equals: true asString.
      self assert: 'false' equals: false asString.
    )
)
```

```
Boolean3Test = TestCase (

    testIfTrueIfFalse = (
      | b1 b2 |
      b1 := false.
      b2 := false.

      true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b1.
      self deny: b2.

      b1 := false.
      b2 := false.
      false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b2.
      self deny: b1.
    )

    testIfTrue = (
      | b |
      b := false.

      true ifTrue: [ b := true ].
      self assert: b.

      b := false.
      false ifTrue: [ b := true ].
      self deny: b.
    )

    testIfFalse = (
      | b |
      b := false.

      true ifFalse: [ b := true ].
      self deny: b.

      b := false.
      false ifFalse: [ b := true ].
      self assert: b.
    )

    testNot = (
      self deny: true not.
      self assert: false not.
    )

    andBoolTrueTrue   = ( ^ true  and: [ true  ] )
    andBoolTrueFalse  = ( ^ true  and: [ false ] )
    andBoolFalseTrue  = ( ^ false and: [ true  ] )
    andBoolFalseFalse = ( ^ false and: [ false ] )

    testAnd = (
      self assert: self andBoolTrueTrue.
      self deny:   self andBoolTrueFalse.
      self deny:   self andBoolFalseTrue.
      self deny:   self andBoolFalseFalse.
    )
```

```
    ampBoolTrueTrue   = ( ^ true  && [ true  ] )
    ampBoolTrueFalse  = ( ^ true  && [ false ] )
    ampBoolFalseTrue  = ( ^ false && [ true  ] )
    ampBoolFalseFalse = ( ^ false && [ false ] )

    testAmp = (
      self assert: self ampBoolTrueTrue.
      self deny:   self ampBoolTrueFalse.
      self deny:   self ampBoolFalseTrue.
      self deny:   self ampBoolFalseFalse.
    )

    orBoolTrueTrue   = ( ^ true  or: [ true  ] )
    orBoolTrueFalse  = ( ^ true  or: [ false ] )
    orBoolFalseTrue  = ( ^ false or: [ true  ] )
    orBoolFalseFalse = ( ^ false or: [ false ] )

    testOr = (
      self assert: self orBoolTrueTrue.
      self assert: self orBoolTrueFalse.
      self assert: self orBoolFalseTrue.
      self deny:   self orBoolFalseFalse.
    )

    pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
    pipeBoolTrueFalse  = ( ^ true  || [ false ] )
    pipeBoolFalseTrue  = ( ^ false || [ true  ] )
    pipeBoolFalseFalse = ( ^ false || [ false ] )

    testPipe = (
      self assert: self pipeBoolTrueTrue.
      self assert: self pipeBoolTrueFalse.
      self assert: self pipeBoolFalseTrue.
      self deny:   self pipeBoolFalseFalse.
    )

    testAsString = (
      self assert: 'true'  equals: true asString.
      self assert: 'false' equals: false asString.
    )
)
```

```
Boolean4Test = TestCase (

    testIfTrueIfFalse = (
      | b1 b2 |
      b1 := false.
      b2 := false.

      true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b1.
      self deny: b2.

      b1 := false.
      b2 := false.
      false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b2.
      self deny: b1.
    )

    testIfTrue = (
      | b |
      b := false.

      true ifTrue: [ b := true ].
      self assert: b.

      b := false.
      false ifTrue: [ b := true ].
      self deny: b.
    )

    testIfFalse = (
      | b |
      b := false.

      true ifFalse: [ b := true ].
      self deny: b.

      b := false.
      false ifFalse: [ b := true ].
      self assert: b.
    )

    testNot = (
      self deny: true not.
      self assert: false not.
    )

    andBoolTrueTrue   = ( ^ true  and: [ true  ] )
    andBoolTrueFalse  = ( ^ true  and: [ false ] )
    andBoolFalseTrue  = ( ^ false and: [ true  ] )
    andBoolFalseFalse = ( ^ false and: [ false ] )

    testAnd = (
      self assert: self andBoolTrueTrue.
      self deny:   self andBoolTrueFalse.
      self deny:   self andBoolFalseTrue.
      self deny:   self andBoolFalseFalse.
    )
```

```
    ampBoolTrueTrue   = ( ^ true  && [ true  ] )
    ampBoolTrueFalse  = ( ^ true  && [ false ] )
    ampBoolFalseTrue  = ( ^ false && [ true  ] )
    ampBoolFalseFalse = ( ^ false && [ false ] )

    testAmp = (
      self assert: self ampBoolTrueTrue.
      self deny:   self ampBoolTrueFalse.
      self deny:   self ampBoolFalseTrue.
      self deny:   self ampBoolFalseFalse.
    )

    orBoolTrueTrue   = ( ^ true  or: [ true  ] )
    orBoolTrueFalse  = ( ^ true  or: [ false ] )
    orBoolFalseTrue  = ( ^ false or: [ true  ] )
    orBoolFalseFalse = ( ^ false or: [ false ] )

    testOr = (
      self assert: self orBoolTrueTrue.
      self assert: self orBoolTrueFalse.
      self assert: self orBoolFalseTrue.
      self deny:   self orBoolFalseFalse.
    )

    pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
    pipeBoolTrueFalse  = ( ^ true  || [ false ] )
    pipeBoolFalseTrue  = ( ^ false || [ true  ] )
    pipeBoolFalseFalse = ( ^ false || [ false ] )

    testPipe = (
      self assert: self pipeBoolTrueTrue.
      self assert: self pipeBoolTrueFalse.
      self assert: self pipeBoolFalseTrue.
      self deny:   self pipeBoolFalseFalse.
    )

    testAsString = (
      self assert: 'true'  equals: true asString.
      self assert: 'false' equals: false asString.
    )
)
```

```
Boolean5Test = TestCase (

    testIfTrueIfFalse = (
      | b1 b2 |
      b1 := false.
      b2 := false.

      true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b1.
      self deny: b2.

      b1 := false.
      b2 := false.
      false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b2.
      self deny: b1.
    )

    testIfTrue = (
      | b |
      b := false.

      true ifTrue: [ b := true ].
      self assert: b.

      b := false.
      false ifTrue: [ b := true ].
      self deny: b.
    )

    testIfFalse = (
      | b |
      b := false.

      true ifFalse: [ b := true ].
      self deny: b.

      b := false.
      false ifFalse: [ b := true ].
      self assert: b.
    )

    testNot = (
      self deny: true not.
      self assert: false not.
    )

    andBoolTrueTrue   = ( ^ true  and: [ true  ] )
    andBoolTrueFalse  = ( ^ true  and: [ false ] )
    andBoolFalseTrue  = ( ^ false and: [ true  ] )
    andBoolFalseFalse = ( ^ false and: [ false ] )

    testAnd = (
      self assert: self andBoolTrueTrue.
      self deny:   self andBoolTrueFalse.
      self deny:   self andBoolFalseTrue.
      self deny:   self andBoolFalseFalse.
    )
```

```

    ampBoolTrueTrue   = ( ^ true  && [ true  ] )
    ampBoolTrueFalse  = ( ^ true  && [ false ] )
    ampBoolFalseTrue  = ( ^ false && [ true  ] )
    ampBoolFalseFalse = ( ^ false && [ false ] )

    testAmp = (
      self assert: self ampBoolTrueTrue.
      self deny:   self ampBoolTrueFalse.
      self deny:   self ampBoolFalseTrue.
      self deny:   self ampBoolFalseFalse.
    )

    orBoolTrueTrue   = ( ^ true  or: [ true  ] )
    orBoolTrueFalse  = ( ^ true  or: [ false ] )
    orBoolFalseTrue  = ( ^ false or: [ true  ] )
    orBoolFalseFalse = ( ^ false or: [ false ] )

    testOr = (
      self assert: self orBoolTrueTrue.
      self assert: self orBoolTrueFalse.
      self assert: self orBoolFalseTrue.
      self deny:   self orBoolFalseFalse.
    )

    pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
    pipeBoolTrueFalse  = ( ^ true  || [ false ] )
    pipeBoolFalseTrue  = ( ^ false || [ true  ] )
    pipeBoolFalseFalse = ( ^ false || [ false ] )

    testPipe = (
      self assert: self pipeBoolTrueTrue.
      self assert: self pipeBoolTrueFalse.
      self assert: self pipeBoolFalseTrue.
      self deny:   self pipeBoolFalseFalse.
    )

    testAsString = (
      self assert: 'true'  equals: true asString.
      self assert: 'false' equals: false asString.
    )
)

```

```
BooleanTest = TestCase (

    testIfTrueIfFalse = (
      | b1 b2 |
      b1 := false.
      b2 := false.

      true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b1.
      self deny: b2.

      b1 := false.
      b2 := false.
      false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b2.
      self deny: b1.
    )

    testIfTrue = (
      | b |
      b := false.

      true ifTrue: [ b := true ].
      self assert: b.

      b := false.
      false ifTrue: [ b := true ].
      self deny: b.
    )

    testIfFalse = (
      | b |
      b := false.

      true ifFalse: [ b := true ].
      self deny: b.

      b := false.
      false ifFalse: [ b := true ].
      self assert: b.
    )

    testNot = (
      self deny: true not.
      self assert: false not.
    )

    andBoolTrueTrue   = ( ^ true  and: [ true  ] )
    andBoolTrueFalse  = ( ^ true  and: [ false ] )
    andBoolFalseTrue  = ( ^ false and: [ true  ] )
    andBoolFalseFalse = ( ^ false and: [ false ] )

    testAnd = (
      self assert: self andBoolTrueTrue.
      self deny:   self andBoolTrueFalse.
      self deny:   self andBoolFalseTrue.
      self deny:   self andBoolFalseFalse.
    )
```

```
    ampBoolTrueTrue   = ( ^ true  && [ true  ] )
    ampBoolTrueFalse  = ( ^ true  && [ false ] )
    ampBoolFalseTrue  = ( ^ false && [ true  ] )
    ampBoolFalseFalse = ( ^ false && [ false ] )

    testAmp = (
      self assert: self ampBoolTrueTrue.
      self deny:   self ampBoolTrueFalse.
      self deny:   self ampBoolFalseTrue.
      self deny:   self ampBoolFalseFalse.
    )

    orBoolTrueTrue   = ( ^ true  or: [ true  ] )
    orBoolTrueFalse  = ( ^ true  or: [ false ] )
    orBoolFalseTrue  = ( ^ false or: [ true  ] )
    orBoolFalseFalse = ( ^ false or: [ false ] )

    testOr = (
      self assert: self orBoolTrueTrue.
      self assert: self orBoolTrueFalse.
      self assert: self orBoolFalseTrue.
      self deny:   self orBoolFalseFalse.
    )

    pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
    pipeBoolTrueFalse  = ( ^ true  || [ false ] )
    pipeBoolFalseTrue  = ( ^ false || [ true  ] )
    pipeBoolFalseFalse = ( ^ false || [ false ] )

    testPipe = (
      self assert: self pipeBoolTrueTrue.
      self assert: self pipeBoolTrueFalse.
      self assert: self pipeBoolFalseTrue.
      self deny:   self pipeBoolFalseFalse.
    )

    testAsString = (
      self assert: 'true'  equals: true asString.
      self assert: 'false' equals: false asString.
    )
)
```

Examples/Benchmarks/TestSuite/ClassA.som

```
ClassA = (
   | a b |
   result = (
      ^42
   )
   ----
   | c1 c2 c3 |
)

```

Examples/Benchmarks/TestSuite/ClassB.som

```
ClassB = ClassA (
  | c d |
  ----
  | c4 c5 c6 |
)

```

```
ClassC = ClassB (
  | e f |
  a      = ( ^ a )
  a: val = ( a := val )

  f      = ( ^ f )
  f: val = ( f := val )

  ----

  | c7 c8 c9 |

  setAllAndInc: anInt = (
      c1 := anInt.
      c2 := c1 + 1.
      c3 := c2 + 1.
      c4 := c3 + 1.
      c5 := c4 + 1.
      c6 := c5 + 1.
      c7 := c6 + 1.
      c8 := c7 + 1.
      c9 := c8 + 1.
  )

  getAll = (
      | arr |
      arr := Array new: 9.
      arr at: 1 put: c1.
      arr at: 2 put: c2.
      arr at: 3 put: c3.
      arr at: 4 put: c4.
      arr at: 5 put: c5.
      arr at: 6 put: c6.
      arr at: 7 put: c7.
      arr at: 8 put: c8.
      arr at: 9 put: c9.
      ^ arr
  )
)
```

```
ClassLoading2Test = TestCase (
  testEqualityOfClasses = (
    | a b c |
    b := ClassB new.
    a := ClassA new.
    c := ClassC new.

    self assert: 42 equals: b result.
    self assert: 42 equals: c result.

    self assert: a class equals: b class superclass.
    self assert: b class equals: c class superclass.
  )
)
```

```
ClassLoading3Test = TestCase (
  testEqualityOfClasses = (
    | a b c |
    b := ClassB new.
    a := ClassA new.
    c := ClassC new.

    self assert: 42 equals: b result.
    self assert: 42 equals: c result.

    self assert: a class equals: b class superclass.
    self assert: b class equals: c class superclass.
  )
)

```

```
ClassLoading4Test = TestCase (
  testEqualityOfClasses = (
    | a b c |
    b := ClassB new.
    a := ClassA new.
    c := ClassC new.

    self assert: 42 equals: b result.
    self assert: 42 equals: c result.

    self assert: a class equals: b class superclass.
    self assert: b class equals: c class superclass.
  )
)
```

```
ClassLoading5Test = TestCase (
  testEqualityOfClasses = (
    | a b c |
    b := ClassB new.
    a := ClassA new.
    c := ClassC new.

    self assert: 42 equals: b result.
    self assert: 42 equals: c result.

    self assert: a class equals: b class superclass.
    self assert: b class equals: c class superclass.
  )
)

```

```
ClassLoadingTest = TestCase (
  testEqualityOfClasses = (
    | a b c |
    b := ClassB new.
    a := ClassA new.
    c := ClassC new.

    self assert: 42 equals: b result.
    self assert: 42 equals: c result.

    self assert: a class equals: b class superclass.
    self assert: b class equals: c class superclass.
  )
)

```

Examples/Benchmarks/TestSuite/ClassStructure2Test.som

```
ClassStructure2Test = TestCase (

  testClassIdentity = (
    self assert: Array   equals: Array new class.
    self assert: Integer equals: 1 class.
    self assert: Integer equals: 10000000000 class.
    self assert: Double  equals: (1 // 2) class.
    self assert: Double  equals: 0.5 class.
    self assert: Block1  equals: [42] class.
    self assert: Object  equals: Object new class.
    self assert: Set     equals: Set new class.
    self assert: String  equals: 'foo' class.
    self assert: Symbol  equals: #foo class.
    self assert: True    equals: true class.
    self assert: False   equals: false class.
    self assert: Nil     equals: nil class.

    self assert: True superclass equals: False superclass.
    self assert: True superclass equals: Boolean.
    self assert: True superclass equals: Boolean.
  )

  testThatCertainMethodsArePrimitives = (
    | m |
    "This is a little fragile.
     Index needs to be adapted with changing Class definition."
    m := Object methods at: 1.
    "self expect: #class equals: m signature."

    self optional: #invokableTypes assert: Primitive equals: m class. "Class>>#name should be a primitive."

    m := Object methods at: 7.
    "self expect: #asString equals: m signature."

    self optional: #invokableTypes assert: Method equals: m class. "Class>>#asString should be a normal method."
  )

  testAccessToInstanceFields = (
    | o |
    o := ClassC new.
    o a: 333.
    self assert: 333 equals: o a.

    o f: 333.
    self assert: 333 equals: o f.
  )

  testAccessToClassFields = (
    | arr |
    ClassC setAllAndInc: 4.
    arr := ClassC getAll.
    1 to: 9 do: [:i |
      self assert: i + (4 - 1) equals: (arr at: i).
    ].

    "We do that here to make sure that class fields do not interfere with
```

```
     other class properties."
    self assert: ClassB    is: ClassC superclass.
    self assert: Metaclass is: ClassC class class.
    self assert: #ClassC   equals: ClassC name.
  )

  testMetaclasses = (
    self assert: nil              is: Object superclass.
    self assert: Integer          is: 1 class.
    self assert: #'Integer class' is: 1 class class name.
    self assert: Metaclass        is: 1 class class class.

    self assert: #'Metaclass class' is: Metaclass class name.
    self assert: Metaclass          is: Metaclass class class.

    self assert: Object          is: 1 class superclass.
    self assert: #'Object class' is: 1 class class superclass name.
    self assert: Class           is: Object class superclass.
    self assert: Metaclass       is: Class class class.
  )

  testInstanceFields = (
    self assert: 2 equals: ClassA fields length.
    self assert: 4 equals: ClassB fields length.
    self assert: 6 equals: ClassC fields length.
  )
)
```

```
ClassStructure3Test = TestCase (

  testClassIdentity = (
    self assert: Array   equals: Array new class.
    self assert: Integer equals: 1 class.
    self assert: Integer equals: 10000000000 class.
    self assert: Double  equals: (1 // 2) class.
    self assert: Double  equals: 0.5 class.
    self assert: Block1  equals: [42] class.
    self assert: Object  equals: Object new class.
    self assert: Set     equals: Set new class.
    self assert: String  equals: 'foo' class.
    self assert: Symbol  equals: #foo class.
    self assert: True    equals: true class.
    self assert: False   equals: false class.
    self assert: Nil     equals: nil class.

    self assert: True superclass equals: False superclass.
    self assert: True superclass equals: Boolean.
    self assert: True superclass equals: Boolean.
  )

  testThatCertainMethodsArePrimitives = (
    | m |
    "This is a little fragile.
     Index needs to be adapted with changing Class definition."
    m := Object methods at: 1.
    "self expect: #class equals: m signature."

    self optional: #invokableTypes assert: Primitive equals: m class. "Class>>#name should be a primitive."

    m := Object methods at: 7.
    "self expect: #asString equals: m signature."

    self optional: #invokableTypes assert: Method equals: m class. "Class>>#asString should be a normal method."
  )

  testAccessToInstanceFields = (
    | o |
    o := ClassC new.
    o a: 333.
    self assert: 333 equals: o a.

    o f: 333.
    self assert: 333 equals: o f.
  )

  testAccessToClassFields = (
    | arr |
    ClassC setAllAndInc: 4.
    arr := ClassC getAll.
    1 to: 9 do: [:i |
      self assert: i + (4 - 1) equals: (arr at: i).
    ].

    "We do that here to make sure that class fields do not interfere with
```

```
     other class properties."
    self assert: ClassB    is: ClassC superclass.
    self assert: Metaclass is: ClassC class class.
    self assert: #ClassC   equals: ClassC name.
  )

  testMetaclasses = (
    self assert: nil              is: Object superclass.
    self assert: Integer          is: 1 class.
    self assert: #'Integer class' is: 1 class class name.
    self assert: Metaclass        is: 1 class class class.

    self assert: #'Metaclass class' is: Metaclass class name.
    self assert: Metaclass          is: Metaclass class class.

    self assert: Object          is: 1 class superclass.
    self assert: #'Object class' is: 1 class class superclass name.
    self assert: Class           is: Object class superclass.
    self assert: Metaclass       is: Class class class.
  )

  testInstanceFields = (
    self assert: 2 equals: ClassA fields length.
    self assert: 4 equals: ClassB fields length.
    self assert: 6 equals: ClassC fields length.
  )
)
```

Examples/Benchmarks/TestSuite/ClassStructure4Test.som

```
ClassStructure4Test = TestCase (

  testClassIdentity = (
    self assert: Array   equals: Array new class.
    self assert: Integer equals: 1 class.
    self assert: Integer equals: 10000000000 class.
    self assert: Double  equals: (1 // 2) class.
    self assert: Double  equals: 0.5 class.
    self assert: Block1  equals: [42] class.
    self assert: Object  equals: Object new class.
    self assert: Set     equals: Set new class.
    self assert: String  equals: 'foo' class.
    self assert: Symbol  equals: #foo class.
    self assert: True    equals: true class.
    self assert: False   equals: false class.
    self assert: Nil     equals: nil class.

    self assert: True superclass equals: False superclass.
    self assert: True superclass equals: Boolean.
    self assert: True superclass equals: Boolean.
  )

  testThatCertainMethodsArePrimitives = (
    | m |
    "This is a little fragile.
     Index needs to be adapted with changing Class definition."
    m := Object methods at: 1.
    "self expect: #class equals: m signature."

    self optional: #invokableTypes assert: Primitive equals: m class. "Class>>#name should be a primitive."

    m := Object methods at: 7.
    "self expect: #asString equals: m signature."

    self optional: #invokableTypes assert: Method equals: m class. "Class>>#asString should be a normal method."
  )

  testAccessToInstanceFields = (
    | o |
    o := ClassC new.
    o a: 333.
    self assert: 333 equals: o a.

    o f: 333.
    self assert: 333 equals: o f.
  )

  testAccessToClassFields = (
    | arr |
    ClassC setAllAndInc: 4.
    arr := ClassC getAll.
    1 to: 9 do: [:i |
      self assert: i + (4 - 1) equals: (arr at: i).
    ].

    "We do that here to make sure that class fields do not interfere with
```

```
     other class properties."
    self assert: ClassB    is: ClassC superclass.
    self assert: Metaclass is: ClassC class class.
    self assert: #ClassC   equals: ClassC name.
  )

  testMetaclasses = (
    self assert: nil              is: Object superclass.
    self assert: Integer          is: 1 class.
    self assert: #'Integer class' is: 1 class class name.
    self assert: Metaclass        is: 1 class class class.

    self assert: #'Metaclass class' is: Metaclass class name.
    self assert: Metaclass          is: Metaclass class class.

    self assert: Object          is: 1 class superclass.
    self assert: #'Object class' is: 1 class class superclass name.
    self assert: Class           is: Object class superclass.
    self assert: Metaclass       is: Class class class.
  )

  testInstanceFields = (
    self assert: 2 equals: ClassA fields length.
    self assert: 4 equals: ClassB fields length.
    self assert: 6 equals: ClassC fields length.
  )
)
```

```
ClassStructure5Test = TestCase (

  testClassIdentity = (
    self assert: Array   equals: Array new class.
    self assert: Integer equals: 1 class.
    self assert: Integer equals: 10000000000 class.
    self assert: Double  equals: (1 // 2) class.
    self assert: Double  equals: 0.5 class.
    self assert: Block1  equals: [42] class.
    self assert: Object  equals: Object new class.
    self assert: Set     equals: Set new class.
    self assert: String  equals: 'foo' class.
    self assert: Symbol  equals: #foo class.
    self assert: True    equals: true class.
    self assert: False   equals: false class.
    self assert: Nil     equals: nil class.

    self assert: True superclass equals: False superclass.
    self assert: True superclass equals: Boolean.
    self assert: True superclass equals: Boolean.
  )

  testThatCertainMethodsArePrimitives = (
    | m |
    "This is a little fragile.
     Index needs to be adapted with changing Class definition."
    m := Object methods at: 1.
    "self expect: #class equals: m signature."

    self optional: #invokableTypes assert: Primitive equals: m class.
"Class>>#name should be a primitive."

    m := Object methods at: 7.
    "self expect: #asString equals: m signature."

    self optional: #invokableTypes assert: Method equals: m class.
"Class>>#asString should be a normal method."
  )

  testAccessToInstanceFields = (
    | o |
    o := ClassC new.
    o a: 333.
    self assert: 333 equals: o a.

    o f: 333.
    self assert: 333 equals: o f.
  )

  testAccessToClassFields = (
    | arr |
    ClassC setAllAndInc: 4.
    arr := ClassC getAll.
    1 to: 9 do: [:i |
      self assert: i + (4 - 1) equals: (arr at: i).
    ].

    "We do that here to make sure that class fields do not interfere with
```

```
     other class properties."
    self assert: ClassB    is: ClassC superclass.
    self assert: Metaclass is: ClassC class class.
    self assert: #ClassC   equals: ClassC name.
  )

  testMetaclasses = (
    self assert: nil              is: Object superclass.
    self assert: Integer          is: 1 class.
    self assert: #'Integer class' is: 1 class class name.
    self assert: Metaclass        is: 1 class class class.

    self assert: #'Metaclass class' is: Metaclass class name.
    self assert: Metaclass          is: Metaclass class class.

    self assert: Object          is: 1 class superclass.
    self assert: #'Object class' is: 1 class class superclass name.
    self assert: Class           is: Object class superclass.
    self assert: Metaclass       is: Class class class.
  )

  testInstanceFields = (
    self assert: 2 equals: ClassA fields length.
    self assert: 4 equals: ClassB fields length.
    self assert: 6 equals: ClassC fields length.
  )
)

```

Examples/Benchmarks/TestSuite/ClassStructureTest.som

```
ClassStructureTest = TestCase (

  testClassIdentity = (
    self assert: Array   equals: Array new class.
    self assert: Integer equals: 1 class.
    self assert: Integer equals: 10000000000 class.
    self assert: Double  equals: (1 // 2) class.
    self assert: Double  equals: 0.5 class.
    self assert: Block1  equals: [42] class.
    self assert: Object  equals: Object new class.
    self assert: Set     equals: Set new class.
    self assert: String  equals: 'foo' class.
    self assert: Symbol  equals: #foo class.
    self assert: True    equals: true class.
    self assert: False   equals: false class.
    self assert: Nil     equals: nil class.

    self assert: True superclass equals: False superclass.
    self assert: True superclass equals: Boolean.
    self assert: True superclass equals: Boolean.
  )

  testThatCertainMethodsArePrimitives = (
    | m |
    "This is a little fragile.
     Index needs to be adapted with changing Class definition."
    m := Object methods at: 1.
    "self expect: #class equals: m signature."

    self optional: #invokableTypes assert: Primitive equals: m class. "Class>>#name should be a primitive."

    m := Object methods at: 7.
    "self expect: #asString equals: m signature."

    self optional: #invokableTypes assert: Method equals: m class. "Class>>#asString should be a normal method."
  )

  testAccessToInstanceFields = (
    | o |
    o := ClassC new.
    o a: 333.
    self assert: 333 equals: o a.

    o f: 333.
    self assert: 333 equals: o f.
  )

  testAccessToClassFields = (
    | arr |
    ClassC setAllAndInc: 4.
    arr := ClassC getAll.
    1 to: 9 do: [:i |
      self assert: i + (4 - 1) equals: (arr at: i).
    ].

    "We do that here to make sure that class fields do not interfere with
```

```
    other class properties."
   self assert: ClassB    is: ClassC superclass.
   self assert: Metaclass is: ClassC class class.
   self assert: #ClassC   equals: ClassC name.
 )

 testMetaclasses = (
   self assert: nil              is: Object superclass.
   self assert: Integer          is: 1 class.
   self assert: #'Integer class' is: 1 class class name.
   self assert: Metaclass        is: 1 class class class.

   self assert: #'Metaclass class' is: Metaclass class name.
   self assert: Metaclass          is: Metaclass class class.

   self assert: Object          is: 1 class superclass.
   self assert: #'Object class' is: 1 class class superclass name.
   self assert: Class           is: Object class superclass.
   self assert: Metaclass       is: Class class class.
 )

 testInstanceFields = (
   self assert: 2 equals: ClassA fields length.
   self assert: 4 equals: ClassB fields length.
   self assert: 6 equals: ClassC fields length.
 )
)
```

Examples/Benchmarks/TestSuite/Closure2Test.som

```
"

$Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"This test verifies that SOM blocks are indeed closures. The test was found on
Eliot Miranda's Cog Blog."

Closure2Test = TestCase (
  testClosureProperty = (
    | factorial result facs |

    facs := Array new: 10.
    facs at: 1 put: 1.
    facs at: 2 put: 2.
    facs at: 3 put: 6.
    facs at: 4 put: 24.
    facs at: 5 put: 120.
    facs at: 6 put: 720.
    facs at: 7 put: 5040.
    facs at: 8 put: 40320.
    facs at: 9 put: 362880.
    facs at: 10 put: 3628800.

    factorial := [ :n |
        n = 1
            ifTrue: [ 1 ]
            ifFalse: [ (factorial value: n - 1) * n ] ].

    result := (1 to: 10) collect: factorial.
    result doIndexes: [ :i |
      self assert: (facs at: i) equals: (result at: i) ]
  )
```

```
)
```

Examples/Benchmarks/TestSuite/Closure3Test.som

```
"

$Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"This test verifies that SOM blocks are indeed closures. The test was found on
Eliot Miranda's Cog Blog."

Closure3Test = TestCase (
  testClosureProperty = (
    | factorial result facs |

    facs := Array new: 10.
    facs at: 1 put: 1.
    facs at: 2 put: 2.
    facs at: 3 put: 6.
    facs at: 4 put: 24.
    facs at: 5 put: 120.
    facs at: 6 put: 720.
    facs at: 7 put: 5040.
    facs at: 8 put: 40320.
    facs at: 9 put: 362880.
    facs at: 10 put: 3628800.

    factorial := [ :n |
        n = 1
            ifTrue: [ 1 ]
            ifFalse: [ (factorial value: n - 1) * n ] ].

    result := (1 to: 10) collect: factorial.
    result doIndexes: [ :i |
      self assert: (facs at: i) equals: (result at: i) ]
  )
```

)

Examples/Benchmarks/TestSuite/Closure4Test.som

```
"

$Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"This test verifies that SOM blocks are indeed closures. The test was found on
Eliot Miranda's Cog Blog."

Closure4Test = TestCase (
  testClosureProperty = (
    | factorial result facs |

    facs := Array new: 10.
    facs at: 1 put: 1.
    facs at: 2 put: 2.
    facs at: 3 put: 6.
    facs at: 4 put: 24.
    facs at: 5 put: 120.
    facs at: 6 put: 720.
    facs at: 7 put: 5040.
    facs at: 8 put: 40320.
    facs at: 9 put: 362880.
    facs at: 10 put: 3628800.

    factorial := [ :n |
        n = 1
            ifTrue: [ 1 ]
            ifFalse: [ (factorial value: n - 1) * n ] ].

    result := (1 to: 10) collect: factorial.
    result doIndexes: [ :i |
      self assert: (facs at: i) equals: (result at: i) ]
  )
```

```
)
```

```
"

$Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"This test verifies that SOM blocks are indeed closures. The test was found on
Eliot Miranda's Cog Blog."

Closure5Test = TestCase (
  testClosureProperty = (
    | factorial result facs |

    facs := Array new: 10.
    facs at: 1 put: 1.
    facs at: 2 put: 2.
    facs at: 3 put: 6.
    facs at: 4 put: 24.
    facs at: 5 put: 120.
    facs at: 6 put: 720.
    facs at: 7 put: 5040.
    facs at: 8 put: 40320.
    facs at: 9 put: 362880.
    facs at: 10 put: 3628800.

    factorial := [ :n |
        n = 1
            ifTrue: [ 1 ]
            ifFalse: [ (factorial value: n - 1) * n ] ].

    result := (1 to: 10) collect: factorial.
    result doIndexes: [ :i |
      self assert: (facs at: i) equals: (result at: i) ]
  )
```

```
)

```

Examples/Benchmarks/TestSuite/ClosureTest.som

```
"

$Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"This test verifies that SOM blocks are indeed closures. The test was found on
Eliot Miranda's Cog Blog."

ClosureTest = TestCase (
  testClosureProperty = (
    | factorial result facs |

    facs := Array new: 10.
    facs at: 1 put: 1.
    facs at: 2 put: 2.
    facs at: 3 put: 6.
    facs at: 4 put: 24.
    facs at: 5 put: 120.
    facs at: 6 put: 720.
    facs at: 7 put: 5040.
    facs at: 8 put: 40320.
    facs at: 9 put: 362880.
    facs at: 10 put: 3628800.

    factorial := [ :n |
        n = 1
            ifTrue: [ 1 ]
            ifFalse: [ (factorial value: n - 1) * n ] ].

    result := (1 to: 10) collect: factorial.
    result doIndexes: [ :i |
      self assert: (facs at: i) equals: (result at: i) ]
  )
```

54)
55

Examples/Benchmarks/TestSuite/Coercion2Test.som

```
"

$Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Coercion2Test = TestCase (

  testBasicNumberCoercion = (
    self assert: 5 equals: 25 sqrt.
    self assert: 1 equals: (2 // 4) * 2.
    self assert: 1 equals: 2 * (2 // 4).
  )
)

```

```
"

$Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Coercion3Test = TestCase (

  testBasicNumberCoercion = (
    self assert: 5 equals: 25 sqrt.
    self assert: 1 equals: (2 // 4) * 2.
    self assert: 1 equals: 2 * (2 // 4).
  )
)
```

Examples/Benchmarks/TestSuite/Coercion4Test.som

```
"

$Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Coercion4Test = TestCase (

  testBasicNumberCoercion = (
    self assert: 5 equals: 25 sqrt.
    self assert: 1 equals: (2 // 4) * 2.
    self assert: 1 equals: 2 * (2 // 4).
  )
)
```

Examples/Benchmarks/TestSuite/Coercion5Test.som

```
"

$Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Coercion5Test = TestCase (

  testBasicNumberCoercion = (
    self assert: 5 equals: 25 sqrt.
    self assert: 1 equals: (2 // 4) * 2.
    self assert: 1 equals: 2 * (2 // 4).
  )
)
```

Examples/Benchmarks/TestSuite/CoercionTest.som

```
"

$Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CoercionTest = TestCase (

  testBasicNumberCoercion = (
    self assert: 5 equals: 25 sqrt.
    self assert: 1 equals: (2 // 4) * 2.
    self assert: 1 equals: 2 * (2 // 4).
  )
)

```

```
"

$Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2009-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CompilerReturn2Test = TestCase (

    return1 = ( ^self )
    return2 = (       )

    return3: arg = ( ^self )
    return4: arg = (       )

    testExplicitAndImplicitReturns = (
      self assert: self is: self return1.
      self assert: self is: self return2.
      self assert: self is: (self return3: 23).
      self assert: self is: (self return4: 23).
    )


    "In SOM++, code after the #ifTrue: does not seem to be executed, if the
     block expression ends with a dot."
    testIfTrueWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithDot: arr.
      self assertArrayCorrectness: arr.
    )

```

```
    assertArrayCorrectness: arr = (
      self assert: 1 equals: (arr at: 1). "method was not executed"
      self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
      self assert: 3 equals: (arr at: 3). "remainder was not executed"
    )

    testIfTrueWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    usesIfTrueWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfTrueWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    testWriteArgument = (
        self assert: 42 equals: (self dec: 43).
    )

    dec: anInt = (
        anInt := anInt - 1.
        ^ anInt
    )
)
```

```
"

$Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2009-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CompilerReturn3Test = TestCase (

    return1 = ( ^self )
    return2 = (       )

    return3: arg = ( ^self )
    return4: arg = (       )

    testExplicitAndImplicitReturns = (
      self assert: self is: self return1.
      self assert: self is: self return2.
      self assert: self is: (self return3: 23).
      self assert: self is: (self return4: 23).
    )


    "In SOM++, code after the #ifTrue: does not seem to be executed, if the
     block expression ends with a dot."
    testIfTrueWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithDot: arr.
      self assertArrayCorrectness: arr.
    )

```

```
    assertArrayCorrectness: arr = (
      self assert: 1 equals: (arr at: 1). "method was not executed"
      self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
      self assert: 3 equals: (arr at: 3). "remainder was not executed"
    )

    testIfTrueWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    usesIfTrueWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfTrueWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    testWriteArgument = (
        self assert: 42 equals: (self dec: 43).
    )

    dec: anInt = (
        anInt := anInt - 1.
        ^ anInt
    )
)
```

Examples/Benchmarks/TestSuite/CompilerReturn4Test.som

```
"

$Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2009-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CompilerReturn4Test = TestCase (

    return1 = ( ^self )
    return2 = (       )

    return3: arg = ( ^self )
    return4: arg = (       )

    testExplicitAndImplicitReturns = (
      self assert: self is: self return1.
      self assert: self is: self return2.
      self assert: self is: (self return3: 23).
      self assert: self is: (self return4: 23).
    )


    "In SOM++, code after the #ifTrue: does not seem to be executed, if the
     block expression ends with a dot."
    testIfTrueWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithDot: arr.
      self assertArrayCorrectness: arr.
    )

```

```
    assertArrayCorrectness: arr = (
      self assert: 1 equals: (arr at: 1). "method was not executed"
      self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
      self assert: 3 equals: (arr at: 3). "remainder was not executed"
    )

    testIfTrueWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    usesIfTrueWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfTrueWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    testWriteArgument = (
        self assert: 42 equals: (self dec: 43).
    )

    dec: anInt = (
        anInt := anInt - 1.
        ^ anInt
    )
)
```

113

Examples/Benchmarks/TestSuite/CompilerReturn5Test.som

```
"

$Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2009-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CompilerReturn5Test = TestCase (

    return1 = ( ^self )
    return2 = (       )

    return3: arg = ( ^self )
    return4: arg = (       )

    testExplicitAndImplicitReturns = (
      self assert: self is: self return1.
      self assert: self is: self return2.
      self assert: self is: (self return3: 23).
      self assert: self is: (self return4: 23).
    )


    "In SOM++, code after the #ifTrue: does not seem to be executed, if the
     block expression ends with a dot."
    testIfTrueWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithDot: arr.
      self assertArrayCorrectness: arr.
    )

```

```
    assertArrayCorrectness: arr = (
      self assert: 1 equals: (arr at: 1). "method was not executed"
      self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
      self assert: 3 equals: (arr at: 3). "remainder was not executed"
    )

    testIfTrueWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    usesIfTrueWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfTrueWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    testWriteArgument = (
        self assert: 42 equals: (self dec: 43).
    )

    dec: anInt = (
        anInt := anInt - 1.
        ^ anInt
    )
)
```

Examples/Benchmarks/TestSuite/CompilerReturnTest.som

```
"

$Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2009-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CompilerReturnTest = TestCase (

    return1 = ( ^self )
    return2 = (       )

    return3: arg = ( ^self )
    return4: arg = (       )

    testExplicitAndImplicitReturns = (
      self assert: self is: self return1.
      self assert: self is: self return2.
      self assert: self is: (self return3: 23).
      self assert: self is: (self return4: 23).
    )


    "In SOM++, code after the #ifTrue: does not seem to be executed, if the
     block expression ends with a dot."
    testIfTrueWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithDot: arr.
      self assertArrayCorrectness: arr.
    )

```

```
    assertArrayCorrectness: arr = (
      self assert: 1 equals: (arr at: 1). "method was not executed"
      self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
      self assert: 3 equals: (arr at: 3). "remainder was not executed"
    )

    testIfTrueWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    usesIfTrueWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfTrueWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    testWriteArgument = (
        self assert: 42 equals: (self dec: 43).
    )

    dec: anInt = (
        anInt := anInt - 1.
        ^ anInt
    )
)
```

```
Dictionary2Test = TestCase (
  testAtAndAtPut = (
    | dict val |
    dict := Dictionary new.
    val := dict at: 1.

    self assert: nil is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: 42.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: 42 equals: val.
  )

  testContainsKey = (
    | dict |
    dict := Dictionary new.
    self deny: (dict containsKey: 4).

    dict at: 4 put: 34234.
    self assert: (dict containsKey: 4).
  )

  testKeys = (
    | dict keys |
    dict := Dictionary new.

    self assert: 0 equals: dict keys size.

    dict at: 4 put: 423.
    self assert: 1 equals: dict keys size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict keys size.

    dict at: 'as' put: Object new.
    self assert: 2 equals: dict keys size.

    keys := dict keys.
    self assert: 4    equals: (keys at: 1).
    self assert: 'as' equals: (keys at: 2).
  )

  testValues = (
    | dict values v2 |
    dict := Dictionary new.

    self assert: 0 equals: dict values size.
```

```
    dict at: 4 put: 423.
    self assert: 1 equals: dict values size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict values size.

    dict at: 'as' put: #(1 2 3).
    self assert: 2 equals: dict values size.

    values := dict values.
    self assert: #gdfgd is: (values at: 1).

    v2 := values at: 2.
    self assert: 1 equals: (v2 at: 1).
    self assert: 2 equals: (v2 at: 2).
    self assert: 3 equals: (v2 at: 3).
  )

  testDo = (
    | dict expectedKs expectedVs i |
    i := 1.
    dict := Dictionary new.
    dict at: #e put: 344.
    dict at: 1  put: 545.
    dict at: 0  put: 123.

    expectedKs := #(#e 1 0).
    expectedVs := #(344 545 123).

    dict do: [:p |
      self assert: (expectedKs at: i) equals: p key.
      self assert: (expectedVs at: i) equals: p value.
      i := i + 1.
    ].

    self assert: 4 equals: i.
  )
)
```

```
Dictionary3Test = TestCase (
  testAtAndAtPut = (
    | dict val |
    dict := Dictionary new.
    val := dict at: 1.

    self assert: nil is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: 42.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: 42 equals: val.
  )

  testContainsKey = (
    | dict |
    dict := Dictionary new.
    self deny: (dict containsKey: 4).

    dict at: 4 put: 34234.
    self assert: (dict containsKey: 4).
  )

  testKeys = (
    | dict keys |
    dict := Dictionary new.

    self assert: 0 equals: dict keys size.

    dict at: 4 put: 423.
    self assert: 1 equals: dict keys size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict keys size.

    dict at: 'as' put: Object new.
    self assert: 2 equals: dict keys size.

    keys := dict keys.
    self assert: 4    equals: (keys at: 1).
    self assert: 'as' equals: (keys at: 2).
  )

  testValues = (
    | dict values v2 |
    dict := Dictionary new.

    self assert: 0 equals: dict values size.
```

```

    dict at: 4 put: 423.
    self assert: 1 equals: dict values size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict values size.

    dict at: 'as' put: #(1 2 3).
    self assert: 2 equals: dict values size.

    values := dict values.
    self assert: #gdfgd is: (values at: 1).

    v2 := values at: 2.
    self assert: 1 equals: (v2 at: 1).
    self assert: 2 equals: (v2 at: 2).
    self assert: 3 equals: (v2 at: 3).
  )

  testDo = (
    | dict expectedKs expectedVs i |
    i := 1.
    dict := Dictionary new.
    dict at: #e put: 344.
    dict at: 1  put: 545.
    dict at: 0  put: 123.

    expectedKs := #(#e 1 0).
    expectedVs := #(344 545 123).

    dict do: [:p |
      self assert: (expectedKs at: i) equals: p key.
      self assert: (expectedVs at: i) equals: p value.
      i := i + 1.
    ].

    self assert: 4 equals: i.
  )
)
```

```
Dictionary4Test = TestCase (
  testAtAndAtPut = (
    | dict val |
    dict := Dictionary new.
    val := dict at: 1.

    self assert: nil is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: 42.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: 42 equals: val.
  )

  testContainsKey = (
    | dict |
    dict := Dictionary new.
    self deny: (dict containsKey: 4).

    dict at: 4 put: 34234.
    self assert: (dict containsKey: 4).
  )

  testKeys = (
    | dict keys |
    dict := Dictionary new.

    self assert: 0 equals: dict keys size.

    dict at: 4 put: 423.
    self assert: 1 equals: dict keys size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict keys size.

    dict at: 'as' put: Object new.
    self assert: 2 equals: dict keys size.

    keys := dict keys.
    self assert: 4    equals: (keys at: 1).
    self assert: 'as' equals: (keys at: 2).
  )

  testValues = (
    | dict values v2 |
    dict := Dictionary new.

    self assert: 0 equals: dict values size.
```

```

    dict at: 4 put: 423.
    self assert: 1 equals: dict values size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict values size.

    dict at: 'as' put: #(1 2 3).
    self assert: 2 equals: dict values size.

    values := dict values.
    self assert: #gdfgd is: (values at: 1).

    v2 := values at: 2.
    self assert: 1 equals: (v2 at: 1).
    self assert: 2 equals: (v2 at: 2).
    self assert: 3 equals: (v2 at: 3).
  )

  testDo = (
    | dict expectedKs expectedVs i |
    i := 1.
    dict := Dictionary new.
    dict at: #e put: 344.
    dict at: 1  put: 545.
    dict at: 0  put: 123.

    expectedKs := #(#e 1 0).
    expectedVs := #(344 545 123).

    dict do: [:p |
      self assert: (expectedKs at: i) equals: p key.
      self assert: (expectedVs at: i) equals: p value.
      i := i + 1.
    ].

    self assert: 4 equals: i.
  )
)
```

```
Dictionary5Test = TestCase (
  testAtAndAtPut = (
    | dict val |
    dict := Dictionary new.
    val := dict at: 1.

    self assert: nil is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: 42.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: 42 equals: val.
  )

  testContainsKey = (
    | dict |
    dict := Dictionary new.
    self deny: (dict containsKey: 4).

    dict at: 4 put: 34234.
    self assert: (dict containsKey: 4).
  )

  testKeys = (
    | dict keys |
    dict := Dictionary new.

    self assert: 0 equals: dict keys size.

    dict at: 4 put: 423.
    self assert: 1 equals: dict keys size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict keys size.

    dict at: 'as' put: Object new.
    self assert: 2 equals: dict keys size.

    keys := dict keys.
    self assert: 4    equals: (keys at: 1).
    self assert: 'as' equals: (keys at: 2).
  )

  testValues = (
    | dict values v2 |
    dict := Dictionary new.

    self assert: 0 equals: dict values size.
```

```

    dict at: 4 put: 423.
    self assert: 1 equals: dict values size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict values size.

    dict at: 'as' put: #(1 2 3).
    self assert: 2 equals: dict values size.

    values := dict values.
    self assert: #gdfgd is: (values at: 1).

    v2 := values at: 2.
    self assert: 1 equals: (v2 at: 1).
    self assert: 2 equals: (v2 at: 2).
    self assert: 3 equals: (v2 at: 3).
  )

  testDo = (
    | dict expectedKs expectedVs i |
    i := 1.
    dict := Dictionary new.
    dict at: #e put: 344.
    dict at: 1  put: 545.
    dict at: 0  put: 123.

    expectedKs := #(#e 1 0).
    expectedVs := #(344 545 123).

    dict do: [:p |
      self assert: (expectedKs at: i) equals: p key.
      self assert: (expectedVs at: i) equals: p value.
      i := i + 1.
    ].

    self assert: 4 equals: i.
  )
)
```

```
DictionaryTest = TestCase (
  testAtAndAtPut = (
    | dict val |
    dict := Dictionary new.
    val := dict at: 1.

    self assert: nil is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: 42.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: 42 equals: val.
  )

  testContainsKey = (
    | dict |
    dict := Dictionary new.
    self deny: (dict containsKey: 4).

    dict at: 4 put: 34234.
    self assert: (dict containsKey: 4).
  )

  testKeys = (
    | dict keys |
    dict := Dictionary new.

    self assert: 0 equals: dict keys size.

    dict at: 4 put: 423.
    self assert: 1 equals: dict keys size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict keys size.

    dict at: 'as' put: Object new.
    self assert: 2 equals: dict keys size.

    keys := dict keys.
    self assert: 4    equals: (keys at: 1).
    self assert: 'as' equals: (keys at: 2).
  )

  testValues = (
    | dict values v2 |
    dict := Dictionary new.

    self assert: 0 equals: dict values size.
```

```

    dict at: 4 put: 423.
    self assert: 1 equals: dict values size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict values size.

    dict at: 'as' put: #(1 2 3).
    self assert: 2 equals: dict values size.

    values := dict values.
    self assert: #gdfgd is: (values at: 1).

    v2 := values at: 2.
    self assert: 1 equals: (v2 at: 1).
    self assert: 2 equals: (v2 at: 2).
    self assert: 3 equals: (v2 at: 3).
  )

  testDo = (
    | dict expectedKs expectedVs i |
    i := 1.
    dict := Dictionary new.
    dict at: #e put: 344.
    dict at: 1  put: 545.
    dict at: 0  put: 123.

    expectedKs := #(#e 1 0).
    expectedVs := #(344 545 123).

    dict do: [:p |
      self assert: (expectedKs at: i) equals: p key.
      self assert: (expectedVs at: i) equals: p value.
      i := i + 1.
    ].

    self assert: 4 equals: i.
  )
)
```

Examples/Benchmarks/TestSuite/DoesNotUnderstand2Test.som

```
DoesNotUnderstand2Test = TestCase (

  testSimpleUnknownFoo = (
    | result |
    result := self foo.
    self assert: DoesNotUnderstandMessage is: result class.
    self assert: self is: result target.
    self assert: #foo is: result selector.
  )

  testArguments = (
    | result |
    result := self foo.
    self assert: Array is: result arguments class.
    self assert: 0 equals: result arguments length.

    result := self foo: 1.
    self assert: 1 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).

    result := self foo: 1 bar: 2 baz: 3.
    self assert: 3 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).
    self assert: 2 equals: (result arguments at: 2).
    self assert: 3 equals: (result arguments at: 3).
  )

  testRepeat = (
    | result |
    result := Array new: 5.
    1 to: result length do: [:i |
      result at: i put: self foo.

      i > 1 ifTrue: [
        self assert: (result at: i - 1) ~= (result at: i).
      ]
    ].
  )

  doesNotUnderstand: selector arguments: arguments = (
      ^ DoesNotUnderstandMessage to: self selector: selector arguments: arguments.
  )
)
```

Examples/Benchmarks/TestSuite/DoesNotUnderstand3Test.som

```
DoesNotUnderstand3Test = TestCase (

  testSimpleUnknownFoo = (
    | result |
    result := self foo.
    self assert: DoesNotUnderstandMessage is: result class.
    self assert: self is: result target.
    self assert: #foo is: result selector.
  )

  testArguments = (
    | result |
    result := self foo.
    self assert: Array is: result arguments class.
    self assert: 0 equals: result arguments length.

    result := self foo: 1.
    self assert: 1 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).

    result := self foo: 1 bar: 2 baz: 3.
    self assert: 3 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).
    self assert: 2 equals: (result arguments at: 2).
    self assert: 3 equals: (result arguments at: 3).
  )

  testRepeat = (
    | result |
    result := Array new: 5.
    1 to: result length do: [:i |
      result at: i put: self foo.

      i > 1 ifTrue: [
        self assert: (result at: i - 1) ~= (result at: i).
      ]
    ].
  )

  doesNotUnderstand: selector arguments: arguments = (
      ^ DoesNotUnderstandMessage to: self selector: selector arguments: arguments.
  )
)
```

Examples/Benchmarks/TestSuite/DoesNotUnderstand4Test.som

```
DoesNotUnderstand4Test = TestCase (

  testSimpleUnknownFoo = (
    | result |
    result := self foo.
    self assert: DoesNotUnderstandMessage is: result class.
    self assert: self is: result target.
    self assert: #foo is: result selector.
  )

  testArguments = (
    | result |
    result := self foo.
    self assert: Array is: result arguments class.
    self assert: 0 equals: result arguments length.

    result := self foo: 1.
    self assert: 1 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).

    result := self foo: 1 bar: 2 baz: 3.
    self assert: 3 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).
    self assert: 2 equals: (result arguments at: 2).
    self assert: 3 equals: (result arguments at: 3).
  )

  testRepeat = (
    | result |
    result := Array new: 5.
    1 to: result length do: [:i |
      result at: i put: self foo.

      i > 1 ifTrue: [
        self assert: (result at: i - 1) ~= (result at: i).
      ]
    ].
  )

  doesNotUnderstand: selector arguments: arguments = (
      ^ DoesNotUnderstandMessage to: self selector: selector arguments: arguments.
  )
)
```

Examples/Benchmarks/TestSuite/DoesNotUnderstand5Test.som

```
DoesNotUnderstand5Test = TestCase (

  testSimpleUnknownFoo = (
    | result |
    result := self foo.
    self assert: DoesNotUnderstandMessage is: result class.
    self assert: self is: result target.
    self assert: #foo is: result selector.
  )

  testArguments = (
    | result |
    result := self foo.
    self assert: Array is: result arguments class.
    self assert: 0 equals: result arguments length.

    result := self foo: 1.
    self assert: 1 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).

    result := self foo: 1 bar: 2 baz: 3.
    self assert: 3 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).
    self assert: 2 equals: (result arguments at: 2).
    self assert: 3 equals: (result arguments at: 3).
  )

  testRepeat = (
    | result |
    result := Array new: 5.
    1 to: result length do: [:i |
      result at: i put: self foo.

      i > 1 ifTrue: [
        self assert: (result at: i - 1) ~= (result at: i).
      ]
    ].
  )

  doesNotUnderstand: selector arguments: arguments = (
      ^ DoesNotUnderstandMessage to: self selector: selector arguments: arguments.
  )
)
```

```
DoesNotUnderstandMessage = (
    | target selector arguments |

    initializeWith: targetObj selector: aSelector arguments: argArray = (
        target    := targetObj.
        selector  := aSelector.
        arguments := argArray.
    )

    target    = ( ^ target )
    selector  = ( ^ selector )
    arguments = ( ^ arguments )

    ----

    to: target selector: selector arguments: args = (
        | m |
        m := self new.
        m initializeWith: target selector: selector arguments: args.
        ^ m
    )
)
```

```
DoesNotUnderstandTest = TestCase (

  testSimpleUnknownFoo = (
    | result |
    result := self foo.
    self assert: DoesNotUnderstandMessage is: result class.
    self assert: self is: result target.
    self assert: #foo is: result selector.
  )

  testArguments = (
    | result |
    result := self foo.
    self assert: Array is: result arguments class.
    self assert: 0 equals: result arguments length.

    result := self foo: 1.
    self assert: 1 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).

    result := self foo: 1 bar: 2 baz: 3.
    self assert: 3 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).
    self assert: 2 equals: (result arguments at: 2).
    self assert: 3 equals: (result arguments at: 3).
  )

  testRepeat = (
    | result |
    result := Array new: 5.
    1 to: result length do: [:i |
      result at: i put: self foo.

      i > 1 ifTrue: [
        self assert: (result at: i - 1) ~= (result at: i).
      ]
    ].
  )

  doesNotUnderstand: selector arguments: arguments = (
      ^ DoesNotUnderstandMessage to: self selector: selector arguments: arguments.
  )
)
```

Examples/Benchmarks/TestSuite/Double2Test.som

```
"

$Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Double2Test = TestCase (

  testAdd = (
    self assert: 1.0 equals: 0.5 + 0.5.
    self assert: 0.0 equals: 0.5 + -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.
    self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.
  )

  testSubtract = (
    self assert: 0.0 equals: 0.5 - 0.5.
    self assert: 1.0 equals: 0.5 - -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.
    self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.
  )

  testMultiply = (
    self assert:  4.0 equals: 2.0 * 2.0.
    self assert: -4.0 equals: 2.0 * -2.0.

    self assert:  1.0 equals:  4.0 * 0.25.
    self assert: -1.0 equals: -4.0 * 0.25.
  )

  testIntegerDivision = (
    self assert: 1 equals: (4/3) + (4/5)
```

```
  )

  testDoubleDivision = (
    self assert: 32 // 15 equals: (4//3) + (4//5)
  )

  testModulo = (
    self assert: 1.0 equals: 3.0 % 2.0.
    self assert: 0.0 equals: 3.0 % 3.0.

    self assert: -1.0 equals: -3.0 %  2.0.
    self assert: -1.0 equals: -3.0 % -2.0.
    self assert:  1.0 equals:  3.0 % -2.0.

    self assert: 0.0 equals:  3.0 %  3.0.
    self assert: 0.0 equals: -3.0 % -3.0.
    self assert: 0.0 equals:  3.0 % -3.0.
  )

  testAbs = (
    self assert: 1.0 equals:  1.0 abs.
    self assert: 1.0 equals: -1.0 abs.

    self assert: 9007199254740992.0 equals:  9007199254740992.0 abs.
    self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.

    self assert: 19007199254740992.0 equals:  19007199254740992.0 abs.
    self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
  )

  testSqrt = (
    self assert:  3.0 equals:   9.0 sqrt.
    self assert: 16.0 equals: 256.0 sqrt.

    self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
  )

  testNegated = (
    self assert:  0.0 equals:  0.0 negated.
    self assert: -1.0 equals:  1.0 negated.
    self assert:  1.0 equals: -1.0 negated.

    self assert: -9007199254740992.0 equals:  9007199254740992.0 negated.
    self assert:  9007199254740992.0 equals: -9007199254740992.0 negated.

    self assert: -19007199254740992.0 equals:  19007199254740992.0 negated.
    self assert:  19007199254740992.0 equals: -19007199254740992.0 negated.
  )

  testAsString = (
    self assert: '0.5' equals: (1//2) asString.
    self assert: '0.5' equals: 0.5 asString.
  )

  testEquals = (
    self assert: (1.0 = 1).
  )

  testRound = (
```

```
    self assert:   1 equals:         1.0 round.
    self assert:   1 equals:         1.4 round.
    self assert:   1 equals:         1.4999 round.
    self assert:   2 equals:         1.5 round.
    self assert:   2 equals:         1.5000001 round.
    self assert:   1 equals:     (5//10) round.
    self assert:   1 equals:    (14//10) round.
    self assert: 445 equals: (44534//100) round.
  )

  testAsInteger = (
    self assert: 1 equals: 1.0 asInteger.
    self assert: 1 equals: 1.1 asInteger.
    self assert: 1 equals: 1.999 asInteger.

    self assert: -1 equals: -1.0 asInteger.
    self assert: -1 equals: -1.999 asInteger.
  )

  testSin = (
    | pi |
    pi := 3.141592653589.
    self assert: 0.0 equals: 0.0 sin.
    self assert: pi sin abs < 0.00000000001.
    self assert: (pi // 2.0) sin > 0.9999999999.
  )

  testCos = (
    | pi |
    pi := 3.141592653589.
    self assert: 1.0 equals: 0.0 cos.
    self assert: (pi // 2.0) cos abs < 0.00000000001.
    self assert: pi cos < -0.9999999999.
  )

  testInfinity = (
    self assert: Double PositiveInfinity > 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity + 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity - 1.

    self assert: Double PositiveInfinity > (999999 * 999999 * 999999 * 999999).
  )

  testFromString = (
    self assert:  0.0 equals: (Double fromString:  '0.0').
    self assert: -1.1 equals: (Double fromString: '-1.1').

    self assert:  3423.54656 equals: (Double fromString:  '3423.54656').
    self assert: -672.433244 equals: (Double fromString: '-672.433244').
  )

  testEqual = (
    self assert:  0.0 =  0.0.
    self assert:  1.0 =  1.0.
    self assert:  0.0 = -0.0.
    self assert: -0.0 =  0.0.
  )

```

```
  testLessThan = (
    self assert: 0.0 < 1.0.
    self assert: 0.499999999 < 0.5.
    self deny:   1.0 < 0.0.
    self deny:   0.5 < 0.499999999.
  )

  testGreaterThan = (
    self deny:   0.0 > 1.0.
    self deny:   0.499999999 > 0.5.
    self assert: 1.0 > 0.0.
    self assert: 0.5 > 0.499999999.
  )

  testLessThanOrEqual = (
    self assert: 0.0 <= 1.0.
    self assert: 0.499999999 <= 0.5.
    self assert: 0.5 <= 0.5.
    self deny:   1.0 < 0.0.
    self deny:   0.5 < 0.499999999.
  )

  testGreaterThanOrEqual = (
    self deny:   0.0 >= 1.0.
    self deny:   0.499999999 >= 0.5.
    self assert: 0.5 >= 0.5.
    self assert: 1.0 >= 0.0.
    self assert: 0.5 >= 0.499999999.
  )

  testNegative = (
    self assert: -0.000000001 negative.
    self assert: -1.0 negative.
    self assert: -123123.000000001 negative.
    self deny: 0.000000001 negative.
    self deny: 1.0 negative.
    self deny: 123123.000000001 negative.
  )

  testBetween = (
    self assert: (1.0 between: 0.0 and: 2.0).
    self assert: (0.000001 between: 0.0 and: 2.0).
    self assert: (1.999999 between: 0.0 and: 2.0).
    self deny:   (0.0 between: 0.0 and: 2.0).
    self deny:   (2.0 between: 0.0 and: 2.0).
  )

  testToDo = (
    | d |
    d := 0.0.
    0.0 to: 10.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    0.0 to: 10.1 do: [:ii |
      d := d + ii.
    ].

```

```
    self assert: 55.0 equals: d.

    d := 0.0.
    0.1 to: 10.1 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 + 1.1 equals: d.
  )

  testDownToDo = (
    | d |
    d := 0.0.
    10.0 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    10.1 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
  )
)
```

```
"

$Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Double3Test = TestCase (

  testAdd = (
    self assert: 1.0 equals: 0.5 + 0.5.
    self assert: 0.0 equals: 0.5 + -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.
    self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.
  )

  testSubtract = (
    self assert: 0.0 equals: 0.5 - 0.5.
    self assert: 1.0 equals: 0.5 - -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.
    self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.
  )

  testMultiply = (
    self assert:  4.0 equals: 2.0 * 2.0.
    self assert: -4.0 equals: 2.0 * -2.0.

    self assert:  1.0 equals:  4.0 * 0.25.
    self assert: -1.0 equals: -4.0 * 0.25.
  )

  testIntegerDivision = (
    self assert: 1 equals: (4/3) + (4/5)
```

```
  )

  testDoubleDivision = (
    self assert: 32 // 15 equals: (4//3) + (4//5)
  )

  testModulo = (
    self assert: 1.0 equals: 3.0 % 2.0.
    self assert: 0.0 equals: 3.0 % 3.0.

    self assert: -1.0 equals: -3.0 %  2.0.
    self assert: -1.0 equals: -3.0 % -2.0.
    self assert:  1.0 equals:  3.0 % -2.0.

    self assert: 0.0 equals:  3.0 %  3.0.
    self assert: 0.0 equals: -3.0 % -3.0.
    self assert: 0.0 equals:  3.0 % -3.0.
  )

  testAbs = (
    self assert: 1.0 equals:  1.0 abs.
    self assert: 1.0 equals: -1.0 abs.

    self assert: 9007199254740992.0 equals:  9007199254740992.0 abs.
    self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.

    self assert: 19007199254740992.0 equals:  19007199254740992.0 abs.
    self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
  )

  testSqrt = (
    self assert:  3.0 equals:   9.0 sqrt.
    self assert: 16.0 equals: 256.0 sqrt.

    self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
  )

  testNegated = (
    self assert:  0.0 equals:  0.0 negated.
    self assert: -1.0 equals:  1.0 negated.
    self assert:  1.0 equals: -1.0 negated.

    self assert: -9007199254740992.0 equals:  9007199254740992.0 negated.
    self assert:  9007199254740992.0 equals: -9007199254740992.0 negated.

    self assert: -19007199254740992.0 equals:  19007199254740992.0 negated.
    self assert:  19007199254740992.0 equals: -19007199254740992.0 negated.
  )

  testAsString = (
    self assert: '0.5' equals: (1//2) asString.
    self assert: '0.5' equals: 0.5 asString.
  )

  testEquals = (
    self assert: (1.0 = 1).
  )

  testRound = (
```

```
    self assert:   1 equals:         1.0 round.
    self assert:   1 equals:         1.4 round.
    self assert:   1 equals:         1.4999 round.
    self assert:   2 equals:         1.5 round.
    self assert:   2 equals:         1.5000001 round.
    self assert:   1 equals:     (5//10) round.
    self assert:   1 equals:    (14//10) round.
    self assert: 445 equals: (44534//100) round.
  )

  testAsInteger = (
    self assert: 1 equals: 1.0 asInteger.
    self assert: 1 equals: 1.1 asInteger.
    self assert: 1 equals: 1.999 asInteger.

    self assert: -1 equals: -1.0 asInteger.
    self assert: -1 equals: -1.999 asInteger.
  )

  testSin = (
    | pi |
    pi := 3.141592653589.
    self assert: 0.0 equals: 0.0 sin.
    self assert: pi sin abs < 0.00000000001.
    self assert: (pi // 2.0) sin > 0.9999999999.
  )

  testCos = (
    | pi |
    pi := 3.141592653589.
    self assert: 1.0 equals: 0.0 cos.
    self assert: (pi // 2.0) cos abs < 0.00000000001.
    self assert: pi cos < -0.9999999999.
  )

  testInfinity = (
    self assert: Double PositiveInfinity > 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity + 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity - 1.

    self assert: Double PositiveInfinity > (999999 * 999999 * 999999 * 999999).
  )

  testFromString = (
    self assert:  0.0 equals: (Double fromString:  '0.0').
    self assert: -1.1 equals: (Double fromString: '-1.1').

    self assert:  3423.54656 equals: (Double fromString:  '3423.54656').
    self assert: -672.433244 equals: (Double fromString: '-672.433244').
  )

  testEqual = (
    self assert:  0.0 =  0.0.
    self assert:  1.0 =  1.0.
    self assert:  0.0 = -0.0.
    self assert: -0.0 =  0.0.
  )
```

```
testLessThan = (
  self assert: 0.0 < 1.0.
  self assert: 0.499999999 < 0.5.
  self deny:   1.0 < 0.0.
  self deny:   0.5 < 0.499999999.
)

testGreaterThan = (
  self deny:   0.0 > 1.0.
  self deny:   0.499999999 > 0.5.
  self assert: 1.0 > 0.0.
  self assert: 0.5 > 0.499999999.
)

testLessThanOrEqual = (
  self assert: 0.0 <= 1.0.
  self assert: 0.499999999 <= 0.5.
  self assert: 0.5 <= 0.5.
  self deny:   1.0 < 0.0.
  self deny:   0.5 < 0.499999999.
)

testGreaterThanOrEqual = (
  self deny:   0.0 >= 1.0.
  self deny:   0.499999999 >= 0.5.
  self assert: 0.5 >= 0.5.
  self assert: 1.0 >= 0.0.
  self assert: 0.5 >= 0.499999999.
)

testNegative = (
  self assert: -0.000000001 negative.
  self assert: -1.0 negative.
  self assert: -123123.000000001 negative.
  self deny: 0.000000001 negative.
  self deny: 1.0 negative.
  self deny: 123123.000000001 negative.
)

testBetween = (
  self assert: (1.0 between: 0.0 and: 2.0).
  self assert: (0.000001 between: 0.0 and: 2.0).
  self assert: (1.999999 between: 0.0 and: 2.0).
  self deny:   (0.0 between: 0.0 and: 2.0).
  self deny:   (2.0 between: 0.0 and: 2.0).
)

testToDo = (
  | d |
  d := 0.0.
  0.0 to: 10.0 do: [:ii |
    d := d + ii.
  ].

  self assert: 55.0 equals: d.

  d := 0.0.
  0.0 to: 10.1 do: [:ii |
    d := d + ii.
  ].

```

```
    self assert: 55.0 equals: d.

    d := 0.0.
    0.1 to: 10.1 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 + 1.1 equals: d.
  )

  testDownToDo = (
    | d |
    d := 0.0.
    10.0 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    10.1 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
  )
)
```

```
"

$Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Double4Test = TestCase (

  testAdd = (
    self assert: 1.0 equals: 0.5 + 0.5.
    self assert: 0.0 equals: 0.5 + -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.
    self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.
  )

  testSubtract = (
    self assert: 0.0 equals: 0.5 - 0.5.
    self assert: 1.0 equals: 0.5 - -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.
    self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.
  )

  testMultiply = (
    self assert:  4.0 equals: 2.0 * 2.0.
    self assert: -4.0 equals: 2.0 * -2.0.

    self assert:  1.0 equals:  4.0 * 0.25.
    self assert: -1.0 equals: -4.0 * 0.25.
  )

  testIntegerDivision = (
    self assert: 1 equals: (4/3) + (4/5)
```

```
  )

  testDoubleDivision = (
    self assert: 32 // 15 equals: (4//3) + (4//5)
  )

  testModulo = (
    self assert: 1.0 equals: 3.0 % 2.0.
    self assert: 0.0 equals: 3.0 % 3.0.

    self assert: -1.0 equals: -3.0 %  2.0.
    self assert: -1.0 equals: -3.0 % -2.0.
    self assert:  1.0 equals:  3.0 % -2.0.

    self assert: 0.0 equals:  3.0 %  3.0.
    self assert: 0.0 equals: -3.0 % -3.0.
    self assert: 0.0 equals:  3.0 % -3.0.
  )

  testAbs = (
    self assert: 1.0 equals:  1.0 abs.
    self assert: 1.0 equals: -1.0 abs.

    self assert: 9007199254740992.0 equals:  9007199254740992.0 abs.
    self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.

    self assert: 19007199254740992.0 equals:  19007199254740992.0 abs.
    self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
  )

  testSqrt = (
    self assert:  3.0 equals:   9.0 sqrt.
    self assert: 16.0 equals: 256.0 sqrt.

    self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
  )

  testNegated = (
    self assert:  0.0 equals:  0.0 negated.
    self assert: -1.0 equals:  1.0 negated.
    self assert:  1.0 equals: -1.0 negated.

    self assert: -9007199254740992.0 equals:  9007199254740992.0 negated.
    self assert:  9007199254740992.0 equals: -9007199254740992.0 negated.

    self assert: -19007199254740992.0 equals:  19007199254740992.0 negated.
    self assert:  19007199254740992.0 equals: -19007199254740992.0 negated.
  )

  testAsString = (
    self assert: '0.5' equals: (1//2) asString.
    self assert: '0.5' equals: 0.5 asString.
  )

  testEquals = (
    self assert: (1.0 = 1).
  )

  testRound = (
```

```
    self assert:   1 equals:         1.0 round.
    self assert:   1 equals:         1.4 round.
    self assert:   1 equals:         1.4999 round.
    self assert:   2 equals:         1.5 round.
    self assert:   2 equals:         1.5000001 round.
    self assert:   1 equals:     (5//10) round.
    self assert:   1 equals:    (14//10) round.
    self assert: 445 equals: (44534//100) round.
  )

  testAsInteger = (
    self assert: 1 equals: 1.0 asInteger.
    self assert: 1 equals: 1.1 asInteger.
    self assert: 1 equals: 1.999 asInteger.

    self assert: -1 equals: -1.0 asInteger.
    self assert: -1 equals: -1.999 asInteger.
  )

  testSin = (
    | pi |
    pi := 3.141592653589.
    self assert: 0.0 equals: 0.0 sin.
    self assert: pi sin abs < 0.00000000001.
    self assert: (pi // 2.0) sin > 0.9999999999.
  )

  testCos = (
    | pi |
    pi := 3.141592653589.
    self assert: 1.0 equals: 0.0 cos.
    self assert: (pi // 2.0) cos abs < 0.00000000001.
    self assert: pi cos < -0.9999999999.
  )

  testInfinity = (
    self assert: Double PositiveInfinity > 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity + 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity - 1.

    self assert: Double PositiveInfinity > (999999 * 999999 * 999999 * 999999).
  )

  testFromString = (
    self assert:  0.0 equals: (Double fromString:  '0.0').
    self assert: -1.1 equals: (Double fromString: '-1.1').

    self assert:  3423.54656 equals: (Double fromString:  '3423.54656').
    self assert: -672.433244 equals: (Double fromString: '-672.433244').
  )

  testEqual = (
    self assert:  0.0 =  0.0.
    self assert:  1.0 =  1.0.
    self assert:  0.0 = -0.0.
    self assert: -0.0 =  0.0.
  )
```

```
testLessThan = (
  self assert: 0.0 < 1.0.
  self assert: 0.499999999 < 0.5.
  self deny:   1.0 < 0.0.
  self deny:   0.5 < 0.499999999.
)

testGreaterThan = (
  self deny:   0.0 > 1.0.
  self deny:   0.499999999 > 0.5.
  self assert: 1.0 > 0.0.
  self assert: 0.5 > 0.499999999.
)

testLessThanOrEqual = (
  self assert: 0.0 <= 1.0.
  self assert: 0.499999999 <= 0.5.
  self assert: 0.5 <= 0.5.
  self deny:   1.0 < 0.0.
  self deny:   0.5 < 0.499999999.
)

testGreaterThanOrEqual = (
  self deny:   0.0 >= 1.0.
  self deny:   0.499999999 >= 0.5.
  self assert: 0.5 >= 0.5.
  self assert: 1.0 >= 0.0.
  self assert: 0.5 >= 0.499999999.
)

testNegative = (
  self assert: -0.000000001 negative.
  self assert: -1.0 negative.
  self assert: -123123.000000001 negative.
  self deny: 0.000000001 negative.
  self deny: 1.0 negative.
  self deny: 123123.000000001 negative.
)

testBetween = (
  self assert: (1.0 between: 0.0 and: 2.0).
  self assert: (0.000001 between: 0.0 and: 2.0).
  self assert: (1.999999 between: 0.0 and: 2.0).
  self deny:   (0.0 between: 0.0 and: 2.0).
  self deny:   (2.0 between: 0.0 and: 2.0).
)

testToDo = (
  | d |
  d := 0.0.
  0.0 to: 10.0 do: [:ii |
    d := d + ii.
  ].

  self assert: 55.0 equals: d.

  d := 0.0.
  0.0 to: 10.1 do: [:ii |
    d := d + ii.
  ].

```

```
    self assert: 55.0 equals: d.

    d := 0.0.
    0.1 to: 10.1 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 + 1.1 equals: d.
  )

  testDownToDo = (
    | d |
    d := 0.0.
    10.0 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    10.1 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
  )
)
```

```
"

$Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Double5Test = TestCase (

  testAdd = (
    self assert: 1.0 equals: 0.5 + 0.5.
    self assert: 0.0 equals: 0.5 + -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.
    self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.
  )

  testSubtract = (
    self assert: 0.0 equals: 0.5 - 0.5.
    self assert: 1.0 equals: 0.5 - -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.
    self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.
  )

  testMultiply = (
    self assert:  4.0 equals: 2.0 * 2.0.
    self assert: -4.0 equals: 2.0 * -2.0.

    self assert:  1.0 equals:  4.0 * 0.25.
    self assert: -1.0 equals: -4.0 * 0.25.
  )

  testIntegerDivision = (
    self assert: 1 equals: (4/3) + (4/5)
```

```
  )

  testDoubleDivision = (
    self assert: 32 // 15 equals: (4//3) + (4//5)
  )

  testModulo = (
    self assert: 1.0 equals: 3.0 % 2.0.
    self assert: 0.0 equals: 3.0 % 3.0.

    self assert: -1.0 equals: -3.0 %  2.0.
    self assert: -1.0 equals: -3.0 % -2.0.
    self assert:  1.0 equals:  3.0 % -2.0.

    self assert: 0.0 equals:  3.0 %  3.0.
    self assert: 0.0 equals: -3.0 % -3.0.
    self assert: 0.0 equals:  3.0 % -3.0.
  )

  testAbs = (
    self assert: 1.0 equals:  1.0 abs.
    self assert: 1.0 equals: -1.0 abs.

    self assert: 9007199254740992.0 equals:  9007199254740992.0 abs.
    self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.

    self assert: 19007199254740992.0 equals:  19007199254740992.0 abs.
    self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
  )

  testSqrt = (
    self assert:  3.0 equals:   9.0 sqrt.
    self assert: 16.0 equals: 256.0 sqrt.

    self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
  )

  testNegated = (
    self assert:  0.0 equals:  0.0 negated.
    self assert: -1.0 equals:  1.0 negated.
    self assert:  1.0 equals: -1.0 negated.

    self assert: -9007199254740992.0 equals:  9007199254740992.0 negated.
    self assert:  9007199254740992.0 equals: -9007199254740992.0 negated.

    self assert: -19007199254740992.0 equals:  19007199254740992.0 negated.
    self assert:  19007199254740992.0 equals: -19007199254740992.0 negated.
  )

  testAsString = (
    self assert: '0.5' equals: (1//2) asString.
    self assert: '0.5' equals: 0.5 asString.
  )

  testEquals = (
    self assert: (1.0 = 1).
  )

  testRound = (
```

```
    self assert:   1 equals:         1.0 round.
    self assert:   1 equals:         1.4 round.
    self assert:   1 equals:         1.4999 round.
    self assert:   2 equals:         1.5 round.
    self assert:   2 equals:         1.5000001 round.
    self assert:   1 equals:     (5//10) round.
    self assert:   1 equals:    (14//10) round.
    self assert: 445 equals: (44534//100) round.
  )

  testAsInteger = (
    self assert: 1 equals: 1.0 asInteger.
    self assert: 1 equals: 1.1 asInteger.
    self assert: 1 equals: 1.999 asInteger.

    self assert: -1 equals: -1.0 asInteger.
    self assert: -1 equals: -1.999 asInteger.
  )

  testSin = (
    | pi |
    pi := 3.141592653589.
    self assert: 0.0 equals: 0.0 sin.
    self assert: pi sin abs < 0.00000000001.
    self assert: (pi // 2.0) sin > 0.9999999999.
  )

  testCos = (
    | pi |
    pi := 3.141592653589.
    self assert: 1.0 equals: 0.0 cos.
    self assert: (pi // 2.0) cos abs < 0.00000000001.
    self assert: pi cos < -0.9999999999.
  )

  testInfinity = (
    self assert: Double PositiveInfinity > 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity + 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity - 1.

    self assert: Double PositiveInfinity > (999999 * 999999 * 999999 * 999999).
  )

  testFromString = (
    self assert:  0.0 equals: (Double fromString:  '0.0').
    self assert: -1.1 equals: (Double fromString: '-1.1').

    self assert:  3423.54656 equals: (Double fromString:  '3423.54656').
    self assert: -672.433244 equals: (Double fromString: '-672.433244').
  )

  testEqual = (
    self assert:  0.0 =  0.0.
    self assert:  1.0 =  1.0.
    self assert:  0.0 = -0.0.
    self assert: -0.0 =  0.0.
  )

```

```
  testLessThan = (
    self assert: 0.0 < 1.0.
    self assert: 0.499999999 < 0.5.
    self deny:   1.0 < 0.0.
    self deny:   0.5 < 0.499999999.
  )

  testGreaterThan = (
    self deny:   0.0 > 1.0.
    self deny:   0.499999999 > 0.5.
    self assert: 1.0 > 0.0.
    self assert: 0.5 > 0.499999999.
  )

  testLessThanOrEqual = (
    self assert: 0.0 <= 1.0.
    self assert: 0.499999999 <= 0.5.
    self assert: 0.5 <= 0.5.
    self deny:   1.0 < 0.0.
    self deny:   0.5 < 0.499999999.
  )

  testGreaterThanOrEqual = (
    self deny:   0.0 >= 1.0.
    self deny:   0.499999999 >= 0.5.
    self assert: 0.5 >= 0.5.
    self assert: 1.0 >= 0.0.
    self assert: 0.5 >= 0.499999999.
  )

  testNegative = (
    self assert: -0.000000001 negative.
    self assert: -1.0 negative.
    self assert: -123123.000000001 negative.
    self deny: 0.000000001 negative.
    self deny: 1.0 negative.
    self deny: 123123.000000001 negative.
  )

  testBetween = (
    self assert: (1.0 between: 0.0 and: 2.0).
    self assert: (0.000001 between: 0.0 and: 2.0).
    self assert: (1.999999 between: 0.0 and: 2.0).
    self deny:   (0.0 between: 0.0 and: 2.0).
    self deny:   (2.0 between: 0.0 and: 2.0).
  )

  testToDo = (
    | d |
    d := 0.0.
    0.0 to: 10.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    0.0 to: 10.1 do: [:ii |
      d := d + ii.
    ].

```

```
    self assert: 55.0 equals: d.

    d := 0.0.
    0.1 to: 10.1 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 + 1.1 equals: d.
  )

  testDownToDo = (
    | d |
    d := 0.0.
    10.0 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    10.1 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
  )
)
```

```
"

$Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

DoubleTest = TestCase (

  testAdd = (
    self assert: 1.0 equals: 0.5 + 0.5.
    self assert: 0.0 equals: 0.5 + -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.
    self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.
  )

  testSubtract = (
    self assert: 0.0 equals: 0.5 - 0.5.
    self assert: 1.0 equals: 0.5 - -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.
    self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.
  )

  testMultiply = (
    self assert:  4.0 equals: 2.0 * 2.0.
    self assert: -4.0 equals: 2.0 * -2.0.

    self assert:  1.0 equals:  4.0 * 0.25.
    self assert: -1.0 equals: -4.0 * 0.25.
  )

  testIntegerDivision = (
    self assert: 1 equals: (4/3) + (4/5)
```

```
  )

  testDoubleDivision = (
    self assert: 32 // 15 equals: (4//3) + (4//5)
  )

  testModulo = (
    self assert: 1.0 equals: 3.0 % 2.0.
    self assert: 0.0 equals: 3.0 % 3.0.

    self assert: -1.0 equals: -3.0 %  2.0.
    self assert: -1.0 equals: -3.0 % -2.0.
    self assert:  1.0 equals:  3.0 % -2.0.

    self assert: 0.0 equals:  3.0 %  3.0.
    self assert: 0.0 equals: -3.0 % -3.0.
    self assert: 0.0 equals:  3.0 % -3.0.
  )

  testAbs = (
    self assert: 1.0 equals:  1.0 abs.
    self assert: 1.0 equals: -1.0 abs.

    self assert: 9007199254740992.0 equals:  9007199254740992.0 abs.
    self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.

    self assert: 19007199254740992.0 equals:  19007199254740992.0 abs.
    self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
  )

  testSqrt = (
    self assert:  3.0 equals:   9.0 sqrt.
    self assert: 16.0 equals: 256.0 sqrt.

    self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
  )

  testNegated = (
    self assert:  0.0 equals:  0.0 negated.
    self assert: -1.0 equals:  1.0 negated.
    self assert:  1.0 equals: -1.0 negated.

    self assert: -9007199254740992.0 equals:  9007199254740992.0 negated.
    self assert:  9007199254740992.0 equals: -9007199254740992.0 negated.

    self assert: -19007199254740992.0 equals:  19007199254740992.0 negated.
    self assert:  19007199254740992.0 equals: -19007199254740992.0 negated.
  )

  testAsString = (
    self assert: '0.5' equals: (1//2) asString.
    self assert: '0.5' equals: 0.5 asString.
  )

  testEquals = (
    self assert: (1.0 = 1).
  )

  testRound = (
```

```
    self assert:   1 equals:         1.0 round.
    self assert:   1 equals:         1.4 round.
    self assert:   1 equals:         1.4999 round.
    self assert:   2 equals:         1.5 round.
    self assert:   2 equals:         1.5000001 round.
    self assert:   1 equals:     (5//10) round.
    self assert:   1 equals:    (14//10) round.
    self assert: 445 equals: (44534//100) round.
  )

  testAsInteger = (
    self assert: 1 equals: 1.0 asInteger.
    self assert: 1 equals: 1.1 asInteger.
    self assert: 1 equals: 1.999 asInteger.

    self assert: -1 equals: -1.0 asInteger.
    self assert: -1 equals: -1.999 asInteger.
  )

  testSin = (
    | pi |
    pi := 3.141592653589.
    self assert: 0.0 equals: 0.0 sin.
    self assert: pi sin abs < 0.00000000001.
    self assert: (pi // 2.0) sin > 0.9999999999.
  )

  testCos = (
    | pi |
    pi := 3.141592653589.
    self assert: 1.0 equals: 0.0 cos.
    self assert: (pi // 2.0) cos abs < 0.00000000001.
    self assert: pi cos < -0.9999999999.
  )

  testInfinity = (
    self assert: Double PositiveInfinity > 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity + 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity - 1.

    self assert: Double PositiveInfinity > (999999 * 999999 * 999999 * 999999).
  )

  testFromString = (
    self assert:  0.0 equals: (Double fromString:  '0.0').
    self assert: -1.1 equals: (Double fromString: '-1.1').

    self assert:  3423.54656 equals: (Double fromString:  '3423.54656').
    self assert: -672.433244 equals: (Double fromString: '-672.433244').
  )

  testEqual = (
    self assert:  0.0 =  0.0.
    self assert:  1.0 =  1.0.
    self assert:  0.0 = -0.0.
    self assert: -0.0 =  0.0.
  )

```

```
testLessThan = (
  self assert: 0.0 < 1.0.
  self assert: 0.499999999 < 0.5.
  self deny:   1.0 < 0.0.
  self deny:   0.5 < 0.499999999.
)

testGreaterThan = (
  self deny:   0.0 > 1.0.
  self deny:   0.499999999 > 0.5.
  self assert: 1.0 > 0.0.
  self assert: 0.5 > 0.499999999.
)

testLessThanOrEqual = (
  self assert: 0.0 <= 1.0.
  self assert: 0.499999999 <= 0.5.
  self assert: 0.5 <= 0.5.
  self deny:   1.0 < 0.0.
  self deny:   0.5 < 0.499999999.
)

testGreaterThanOrEqual = (
  self deny:   0.0 >= 1.0.
  self deny:   0.499999999 >= 0.5.
  self assert: 0.5 >= 0.5.
  self assert: 1.0 >= 0.0.
  self assert: 0.5 >= 0.499999999.
)

testNegative = (
  self assert: -0.000000001 negative.
  self assert: -1.0 negative.
  self assert: -123123.000000001 negative.
  self deny: 0.000000001 negative.
  self deny: 1.0 negative.
  self deny: 123123.000000001 negative.
)

testBetween = (
  self assert: (1.0 between: 0.0 and: 2.0).
  self assert: (0.000001 between: 0.0 and: 2.0).
  self assert: (1.999999 between: 0.0 and: 2.0).
  self deny:   (0.0 between: 0.0 and: 2.0).
  self deny:   (2.0 between: 0.0 and: 2.0).
)

testToDo = (
  | d |
  d := 0.0.
  0.0 to: 10.0 do: [:ii |
    d := d + ii.
  ].

  self assert: 55.0 equals: d.

  d := 0.0.
  0.0 to: 10.1 do: [:ii |
    d := d + ii.
  ].

```

```
    self assert: 55.0 equals: d.

    d := 0.0.
    0.1 to: 10.1 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 + 1.1 equals: d.
  )

  testDownToDo = (
    | d |
    d := 0.0.
    10.0 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    10.1 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
  )
)

```

Examples/Benchmarks/TestSuite/Empty2Test.som

```
"

$Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Empty2Test = TestCase (

    "This is just an empty TestCase.
     It only tests the basic infrastructure"

)

```

Examples/Benchmarks/TestSuite/Empty3Test.som

```
"

$Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Empty3Test = TestCase (

    "This is just an empty TestCase.
     It only tests the basic infrastructure"

)

```

```
"

$Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Empty4Test = TestCase (

    "This is just an empty TestCase.
     It only tests the basic infrastructure"

)
```

Examples/Benchmarks/TestSuite/Empty5Test.som

```
"

$Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Empty5Test = TestCase (

    "This is just an empty TestCase.
     It only tests the basic infrastructure"

)

```

```
"

$Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

EmptyTest = TestCase (

    "This is just an empty TestCase.
     It only tests the basic infrastructure"

)
```

Examples/Benchmarks/TestSuite/Global2Test.som

```
Global2Test = TestCase (
    | doesntKnow |
    unknownGlobal: name = ( doesntKnow := name. ^ name )

    testUnknownGlobalHandler = (
      self assert: #foobar equals: foobar.     "should return the unknown globals name"
      self assert: #foobar equals: doesntKnow. "and should have set it in the field"
    )

    testKnownGlobals = (
      self assert: True   equals: true  class.
      self assert: False  equals: false class.
      self assert: Nil    equals: nil   class.
      self assert: System equals: system class.
    )

    escapingBlock = (
       ^ [ EscapingBlockGlobal ]
    )

    testUnknownGlobalSemanticsInBlocks = (
      self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
      self assert: #NormalBlockGlobal is: doesntKnow.

      self assert: #EscapingBlockGlobal is: self escapingBlock value.
      self assert: #EscapingBlockGlobal is: doesntKnow.

      self assert: #NestedBlockGlobal is: [
         [ [ NestedBlockGlobal ] value ] value ] value.
      self assert: #NestedBlockGlobal is: doesntKnow.
    )
)
```

Examples/Benchmarks/TestSuite/Global3Test.som

```
Global3Test = TestCase (
    | doesntKnow |
    unknownGlobal: name = ( doesntKnow := name. ^ name )

    testUnknownGlobalHandler = (
      self assert: #foobar equals: foobar.     "should return the unknown globals name"
      self assert: #foobar equals: doesntKnow. "and should have set it in the field"
    )

    testKnownGlobals = (
      self assert: True   equals: true  class.
      self assert: False  equals: false class.
      self assert: Nil    equals: nil   class.
      self assert: System equals: system class.
    )

    escapingBlock = (
       ^ [ EscapingBlockGlobal ]
    )

    testUnknownGlobalSemanticsInBlocks = (
      self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
      self assert: #NormalBlockGlobal is: doesntKnow.

      self assert: #EscapingBlockGlobal is: self escapingBlock value.
      self assert: #EscapingBlockGlobal is: doesntKnow.

      self assert: #NestedBlockGlobal is: [
         [ [ NestedBlockGlobal ] value ] value ] value.
      self assert: #NestedBlockGlobal is: doesntKnow.
    )
)
```

Examples/Benchmarks/TestSuite/Global4Test.som

```
Global4Test = TestCase (
    | doesntKnow |
    unknownGlobal: name = ( doesntKnow := name. ^ name )

    testUnknownGlobalHandler = (
      self assert: #foobar equals: foobar.     "should return the unknown globals name"
      self assert: #foobar equals: doesntKnow. "and should have set it in the field"
    )

    testKnownGlobals = (
      self assert: True   equals: true  class.
      self assert: False  equals: false class.
      self assert: Nil    equals: nil   class.
      self assert: System equals: system class.
    )

    escapingBlock = (
       ^ [ EscapingBlockGlobal ]
    )

    testUnknownGlobalSemanticsInBlocks = (
      self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
      self assert: #NormalBlockGlobal is: doesntKnow.

      self assert: #EscapingBlockGlobal is: self escapingBlock value.
      self assert: #EscapingBlockGlobal is: doesntKnow.

      self assert: #NestedBlockGlobal is: [
         [ [ NestedBlockGlobal ] value ] value ] value.
      self assert: #NestedBlockGlobal is: doesntKnow.
    )
)

```

Examples/Benchmarks/TestSuite/Global5Test.som

```
Global5Test = TestCase (
    | doesntKnow |
    unknownGlobal: name = ( doesntKnow := name. ^ name )

    testUnknownGlobalHandler = (
      self assert: #foobar equals: foobar.     "should return the unknown globals name"
      self assert: #foobar equals: doesntKnow. "and should have set it in the field"
    )

    testKnownGlobals = (
      self assert: True   equals: true  class.
      self assert: False  equals: false class.
      self assert: Nil    equals: nil   class.
      self assert: System equals: system class.
    )

    escapingBlock = (
       ^ [ EscapingBlockGlobal ]
    )

    testUnknownGlobalSemanticsInBlocks = (
      self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
      self assert: #NormalBlockGlobal is: doesntKnow.

      self assert: #EscapingBlockGlobal is: self escapingBlock value.
      self assert: #EscapingBlockGlobal is: doesntKnow.

      self assert: #NestedBlockGlobal is: [
         [ [ NestedBlockGlobal ] value ] value ] value.
      self assert: #NestedBlockGlobal is: doesntKnow.
    )
)
```

Examples/Benchmarks/TestSuite/GlobalTest.som

```
GlobalTest = TestCase (
    | doesntKnow |
    unknownGlobal: name = ( doesntKnow := name. ^ name )

    testUnknownGlobalHandler = (
      self assert: #foobar equals: foobar.     "should return the unknown globals name"
      self assert: #foobar equals: doesntKnow. "and should have set it in the field"
    )

    testKnownGlobals = (
      self assert: True   equals: true  class.
      self assert: False  equals: false class.
      self assert: Nil    equals: nil   class.
      self assert: System equals: system class.
    )

    escapingBlock = (
       ^ [ EscapingBlockGlobal ]
    )

    testUnknownGlobalSemanticsInBlocks = (
      self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
      self assert: #NormalBlockGlobal is: doesntKnow.

      self assert: #EscapingBlockGlobal is: self escapingBlock value.
      self assert: #EscapingBlockGlobal is: doesntKnow.

      self assert: #NestedBlockGlobal is: [
         [ [ NestedBlockGlobal ] value ] value ] value.
      self assert: #NestedBlockGlobal is: doesntKnow.
    )
)

```

```
"

$Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Hash2Test = TestCase (

  testHashtable = (
    | ht string array t |

    ht := Hashtable new.
    self assert: ht isEmpty description: 'new ht needs to be empty'.

    ht at: 'a' put: 'b'.
    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self deny: ht isEmpty.

    self assert: 1 equals: ht size.

    ht at: 'c' put: 'd'.
    self assert: 2 equals: ht size.

    ht at: 1 put: 2.
    t := Hashtable new.
    ht at: Hashtable put: t.

    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self assert: (ht containsValue: 'd') description: 'needs to contain "d"'.
    self assert: (ht containsValue: 2)   description: 'needs to contain "2"'.
    self assert: (ht containsValue: t)   description: 'needs to contain t'.
```

```
    self assert: (ht containsKey: Hashtable) description: 'needs to contain Hashtable'.

    ht clear.
    self assert: ht isEmpty.
    self assert: 0 equals: ht size.

    self assert: nil equals: (ht get: 'a').
  )
)
```

Examples/Benchmarks/TestSuite/Hash3Test.som

```
"

$Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Hash3Test = TestCase (

  testHashtable = (
    | ht string array t |

    ht := Hashtable new.
    self assert: ht isEmpty description: 'new ht needs to be empty'.

    ht at: 'a' put: 'b'.
    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self deny: ht isEmpty.

    self assert: 1 equals: ht size.

    ht at: 'c' put: 'd'.
    self assert: 2 equals: ht size.

    ht at: 1 put: 2.
    t := Hashtable new.
    ht at: Hashtable put: t.

    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self assert: (ht containsValue: 'd') description: 'needs to contain "d"'.
    self assert: (ht containsValue: 2)   description: 'needs to contain "2"'.
    self assert: (ht containsValue: t)   description: 'needs to contain t'.
```

```
    self assert: (ht containsKey: Hashtable) description: 'needs to contain Hashtable'.

    ht clear.
    self assert: ht isEmpty.
    self assert: 0 equals: ht size.

    self assert: nil equals: (ht get: 'a').
  )
)


```

Examples/Benchmarks/TestSuite/Hash4Test.som

```
"

$Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Hash4Test = TestCase (

  testHashtable = (
    | ht string array t |

    ht := Hashtable new.
    self assert: ht isEmpty description: 'new ht needs to be empty'.

    ht at: 'a' put: 'b'.
    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self deny: ht isEmpty.

    self assert: 1 equals: ht size.

    ht at: 'c' put: 'd'.
    self assert: 2 equals: ht size.

    ht at: 1 put: 2.
    t := Hashtable new.
    ht at: Hashtable put: t.

    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self assert: (ht containsValue: 'd') description: 'needs to contain "d"'.
    self assert: (ht containsValue: 2)   description: 'needs to contain "2"'.
    self assert: (ht containsValue: t)   description: 'needs to contain t'.
```

```
    self assert: (ht containsKey: Hashtable) description: 'needs to contain Hashtable'.

    ht clear.
    self assert: ht isEmpty.
    self assert: 0 equals: ht size.

    self assert: nil equals: (ht get: 'a').
  )
)
```

Examples/Benchmarks/TestSuite/Hash5Test.som

```
"

$Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Hash5Test = TestCase (

  testHashtable = (
    | ht string array t |

    ht := Hashtable new.
    self assert: ht isEmpty description: 'new ht needs to be empty'.

    ht at: 'a' put: 'b'.
    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self deny: ht isEmpty.

    self assert: 1 equals: ht size.

    ht at: 'c' put: 'd'.
    self assert: 2 equals: ht size.

    ht at: 1 put: 2.
    t := Hashtable new.
    ht at: Hashtable put: t.

    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self assert: (ht containsValue: 'd') description: 'needs to contain "d"'.
    self assert: (ht containsValue: 2)   description: 'needs to contain "2"'.
    self assert: (ht containsValue: t)   description: 'needs to contain t'.
```

```
    self assert: (ht containsKey: Hashtable) description: 'needs to contain Hashtable'.

    ht clear.
    self assert: ht isEmpty.
    self assert: 0 equals: ht size.

    self assert: nil equals: (ht get: 'a').
  )
)


```

Examples/Benchmarks/TestSuite/HashTest.som

```
"

$Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

HashTest = TestCase (

  testHashtable = (
    | ht string array t |

    ht := Hashtable new.
    self assert: ht isEmpty description: 'new ht needs to be empty'.

    ht at: 'a' put: 'b'.
    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self deny: ht isEmpty.

    self assert: 1 equals: ht size.

    ht at: 'c' put: 'd'.
    self assert: 2 equals: ht size.

    ht at: 1 put: 2.
    t := Hashtable new.
    ht at: Hashtable put: t.

    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self assert: (ht containsValue: 'd') description: 'needs to contain "d"'.
    self assert: (ht containsValue: 2)   description: 'needs to contain "2"'.
    self assert: (ht containsValue: t)   description: 'needs to contain t'.
```

```
    self assert: (ht containsKey: Hashtable) description: 'needs to contain Hashtable'.

    ht clear.
    self assert: ht isEmpty.
    self assert: 0 equals: ht size.

    self assert: nil equals: (ht get: 'a').
  )
)
```

Examples/Benchmarks/TestSuite/Integer2Test.som

```
"

$Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Integer2Test = TestCase (

  testEqualityAndIdentity = (
    | a b |
    a := 42.
    b := 42.

    self assert: a = b   description: 'Integers are equal based on their value'.
    self assert: a == b  description: 'Integers do not have pointer/reference equality. It is also supposed to be value equality'.

    "Sometimes it can be hard to implement efficiently, but it SHOULD really
     be true for all values of integers."
    a := 1 << 30.  b := 1 << 30.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 32.  b := 1 << 32.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 60.  b := 1 << 60.
    self optional: #integerIdentity assert: a is: b.
  )

  testClassAndValueRanges = (
```

```
    | i |
    self assert: Integer equals: -42 class.
    self assert: Integer equals:   0 class.
    self assert: Integer equals:  23 class.
    self assert: Integer equals:  1073741823 class.
    self assert: Integer equals:  1073741824 class.

    "Let's test for size behavior and corresponding class"
    i := 1 << 30.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1073741824' equals: i asString.

    i := 1 << 32.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '4294967296' equals: i asString.

    i := 1 << 60.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1152921504606846976' equals: i asString.

    i := 1 << 70.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self optional: #bigIntShifts assert: '1180591620717411303424' equals: i asString.

    i := -1 << 30.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1073741824' equals: i asString.

    i := -1 << 32.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-4294967296' equals: i asString.

    i := -1 << 60.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1152921504606846976' equals: i asString.

    i := -1 << 70.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self optional: #bigIntShifts assert: '-1180591620717411303424' equals: i asString.
  )

  testStringConversion = (
    self assert:  '0' equals: ( 0 asString).
    self assert:  '1' equals: ( 1 asString).
    self assert:  '2' equals: ( 2 asString).
    self assert: '-1' equals: (-1 asString).
    self assert: '-2' equals: (-2 asString).

    self assert: 42 equals: '42' asInteger.
    self assert: -2 equals: '-2' asInteger.
  )
```

```

  testIntegerLiterals = (
    "Make sure the parser reads literals correctly. So, check some basic properties"
    self assert: 2 / 2 equals:                       1.
    self assert: 50 + 50 equals:                   100.
    self assert: 92233720368 * 100000000 + 54775807        equals: 9223372036854775807.
    self assert: 92233720368 * 100000000 + 54775807 * 100  equals: 922337203685477580700.
    self assert: 50 - 100 equals:                  -50.
    self assert: 21474 * -100000 - 83648 equals:            -2147483648.
    self assert: 92233720368 * 100000000 + 54775807 * -100 equals: -922337203685477580700.
  )

  testFromString = (
    self assert:                     1 equals: (Integer fromString:                     '1').
    self assert:                   100 equals: (Integer fromString:                   '100').
    self assert:    9223372036854775807 equals: (Integer fromString: '9223372036854775807').
    self assert:  922337203685477580700 equals: (Integer fromString: '922337203685477580700').
    self assert:                   -50 equals: (Integer fromString:                   '-50').
    self assert:           -2147483648 equals: (Integer fromString:           '-2147483648').
    self assert: -922337203685477580700 equals: (Integer fromString: '-922337203685477580700').
  )

  testRangeBorders = (
    self assert:   '536870911' equals:   536870911 asString.
    self assert:   '536870912' equals:   536870912 asString.
    self assert:   '536870913' equals:   536870913 asString.
    self assert:  '1073741823' equals:  1073741823 asString.
    self assert:  '1073741824' equals:  1073741824 asString.
    self assert:  '1073741825' equals:  1073741825 asString.
    self assert:  '2147483647' equals:  2147483647 asString.
    self assert:  '-536870911' equals:  -536870911 asString.
    self assert:  '-536870912' equals:  -536870912 asString.
    self assert:  '-536870913' equals:  -536870913 asString.
    self assert: '-1073741823' equals: -1073741823 asString.
    self assert: '-1073741824' equals: -1073741824 asString.
    self assert: '-1073741825' equals: -1073741825 asString.
    self assert: '-2147483647' equals: -2147483647 asString.
    self assert: '-2147483648' equals: -2147483648 asString.
  )

  testComparisons = (
    self assert: ( 9 =  9).
    self deny:   ( 1 =  2).
    self deny:   ( 0 <  0).
    self assert: ( 1 <  2).
    self deny:   ( 2 <  1).
```

```
    self assert: (-3 <  2).
    self deny:   ( 3 < -2).
    self deny:   ( 0 >  0).
    self deny:   ( 1 >  2).
    self assert: ( 2 >  1).
    self deny:   (-3 >  2).
    self assert: ( 3 > -2).
    self assert: ( 4 >= 3).
    self assert: ( 3 >= 3).
    self deny:   ( 2 >= 3).
    self assert: ( 2 <= 4).
    self assert: ( 3 <= 3).
    self deny:   ( 4 <= 3).
  )

  testAddition = (
    self assert: 0 equals: ( 0+0).
    self assert: 1 equals: ( 1+0).
    self assert: 1 equals: ( 0+1).
    self assert: 2 equals: ( 1+1).
    self assert: 0 equals: (-1+1).
    self assert: 1 equals: (-1+2).
  )

  testSubtraction = (
    self assert:  1 equals: (1-0).
    self assert: -1 equals: (0-1).
    self assert:  1 equals: (2-1).
  )

  testMultiplication = (
    self assert:   0 equals: ( 1*  0).
    self assert:  -1 equals: (-1*  1).
    self assert: -25 equals: ( 5* -5).
    self assert:  12 equals: (-3* -4).
  )

  testDivision = (
    self assert:  1 equals: (  1/  1).
    self assert:  1 equals: (  3/  2).
    self assert: -2 equals: (  4/ -2).
    self assert: -2 equals: ( -6/  3).
    self assert:  3 equals: (-12/ -4).
  )

  testDouble = (
    self assert:  6 equals: ( 36//  6).
    self assert: -5 equals: (-10//  2).
    self assert: -4 equals: ( 20// -5).
    self assert:  1 equals: ( -5// -5).
  )

  testModulo = (
    self assert:  1 equals: ( 10 %  3).
    self assert: -2 equals: ( 10 % -3).
    self assert:  2 equals: (-10 %  3).
    self assert: -1 equals: (-10 % -3).
    self assert:  0 equals: ( 10 %  5).

    self assert:  1 equals: ( 10 rem:  3).
    self assert:  1 equals: ( 10 rem: -3).
```

```
    self assert: -1 equals: (-10 rem:  3).
    self assert: -1 equals: (-10 rem: -3).
    self assert:  0 equals: ( 10 rem:  5).
  )

  testAbsoluteValue = (
    self assert: 4 equals: -4 abs.
    self assert: 4 equals:  4 abs.

    self assert: 9223372036854775296 equals: -9223372036854775296 abs.
    self assert: 9223372036854775296 equals:  9223372036854775296 abs.
  )

  testNegated = (
    self assert: -23 equals: ( 23 negated).
    self assert:  23 equals: (-23 negated).
  )

  testSquareRoot = (
    self assert:       5 equals: (25 sqrt).
    self assert: Integer equals: (25 sqrt class).
  )

  testAnd = (
    self assert: 0 equals: (2 & 1).
    self assert: 2 equals: (2 & 2).
  )

  testBitXor = (
    self assert: 0 equals: (1 bitXor: 1).
    self assert: 3 equals: (2 bitXor: 1).
  )

  testAs32BitUnsignedValue = (
    self assert: 1 <<  1 equals: (1 <<  1) as32BitUnsignedValue.
    self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
    self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
    self assert:       0 equals: (1 << 32) as32BitUnsignedValue.
    self assert: 4294967295 equals:     -1 as32BitUnsignedValue.
    self assert: 512        equals: -9223372036854775296 as32BitUnsignedValue.
    self assert: 4294966784 equals:  9223372036854775296 as32BitUnsignedValue.
  )

  testAs32BitSignedValue = (
    self assert:     1 <<  1 equals: (1 <<  1) as32BitSignedValue.
    self assert:     1 << 10 equals: (1 << 10) as32BitSignedValue.
    self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
    self assert:           0 equals: (1 << 32) as32BitSignedValue.

    self assert:  512 equals:  -9223372036854775296 as32BitSignedValue.
    self assert: -512 equals:   9223372036854775296 as32BitSignedValue.
  )

  testAsDouble = (
    self assert: 0.0          equals: 0 asDouble.
    self assert: Double       is:     0 asDouble class.

    self assert: 2147483648.0 equals: 2147483648 asDouble.
    self assert: Double       is:     2147483648 asDouble class.
```

```

    self assert: -2147483648.0 equals: -2147483648 asDouble.
    self assert: Double       is:      -2147483648 asDouble class.
  )

  testUnsignedRightShift = (
    self assert: 0   equals:    1 >>> 1.
    self assert: 512 equals: 1024 >>> 1.
    self assert: 127 equals: 1023 >>> 3.

    "not sure whether we should really insist on this"
    self optional: #toBeSpecified assert: 9223372036854775807 equals: -1 >>> 1.
    self optional: #toBeSpecified assert: 9223372036854775296 equals: -1024 >>> 1.
  )

  testMin = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
                 -50 -2147483648 922337203685477580700 -922337203685477580700
                 922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
                 -51 -2147483650                     1 -922337203685477580701
                 -922337203685477580701).

    big doIndexes: [:i |
      self assert: (small at: i)  equals: ((big   at: i) min: (small at: i)).
      self assert: (small at: i)  equals: ((small at: i) min: (big   at: i)) ]
  )

  testMax = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
                 -50 -2147483648 922337203685477580700 -922337203685477580700
                 922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
                 -51 -2147483650                     1 -922337203685477580701
                 -922337203685477580701).
    big doIndexes: [:i |
      self assert: (big at: i)  equals: ((big   at: i) max: (small at: i)).
      self assert: (big at: i)  equals: ((small at: i) max: (big   at: i)) ]
  )
)

```

Examples/Benchmarks/TestSuite/Integer3Test.som

```
"

$Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Integer3Test = TestCase (

  testEqualityAndIdentity = (
    | a b |
    a := 42.
    b := 42.

    self assert: a = b   description: 'Integers are equal based on their value'.
    self assert: a == b  description: 'Integers do not have pointer/reference equality. It is also supposed to be value equality'.

    "Sometimes it can be hard to implement efficiently, but it SHOULD really
     be true for all values of integers."
    a := 1 << 30.  b := 1 << 30.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 32.  b := 1 << 32.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 60.  b := 1 << 60.
    self optional: #integerIdentity assert: a is: b.
  )

  testClassAndValueRanges = (
```

```
    | i |
    self assert: Integer equals: -42 class.
    self assert: Integer equals:   0 class.
    self assert: Integer equals:  23 class.
    self assert: Integer equals:  1073741823 class.
    self assert: Integer equals:  1073741824 class.

    "Let's test for size behavior and corresponding class"
    i := 1 << 30.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1073741824' equals: i asString.

    i := 1 << 32.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '4294967296' equals: i asString.

    i := 1 << 60.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1152921504606846976' equals: i asString.

    i := 1 << 70.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self optional: #bigIntShifts assert: '1180591620717411303424' equals: i asString.

    i := -1 << 30.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1073741824' equals: i asString.

    i := -1 << 32.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-4294967296' equals: i asString.

    i := -1 << 60.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1152921504606846976' equals: i asString.

    i := -1 << 70.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self optional: #bigIntShifts assert: '-1180591620717411303424' equals: i asString.
  )

  testStringConversion = (
    self assert:  '0' equals: ( 0 asString).
    self assert:  '1' equals: ( 1 asString).
    self assert:  '2' equals: ( 2 asString).
    self assert: '-1' equals: (-1 asString).
    self assert: '-2' equals: (-2 asString).

    self assert: 42 equals: '42' asInteger.
    self assert: -2 equals: '-2' asInteger.
  )
```

```

  testIntegerLiterals = (
    "Make sure the parser reads literals correctly. So, check some basic properties"
    self assert: 2 / 2                                          equals:                    1.
    self assert: 50 + 50                                        equals:                  100.
    self assert: 92233720368 * 100000000 + 54775807        equals: 9223372036854775807.
    self assert: 92233720368 * 100000000 + 54775807 * 100  equals: 922337203685477580700.
    self assert: 50 - 100                                       equals:                  -50.
    self assert: 21474 * -100000 - 83648                        equals:          -2147483648.
    self assert: 92233720368 * 100000000 + 54775807 * -100 equals: -922337203685477580700.
  )

  testFromString = (
    self assert:                      1 equals: (Integer fromString:                      '1').
    self assert:                    100 equals: (Integer fromString:                    '100').
    self assert:    9223372036854775807 equals: (Integer fromString: '9223372036854775807').
    self assert:  922337203685477580700 equals: (Integer fromString: '922337203685477580700').
    self assert:                    -50 equals: (Integer fromString:                    '-50').
    self assert:            -2147483648 equals: (Integer fromString:            '-2147483648').
    self assert: -922337203685477580700 equals: (Integer fromString: '-922337203685477580700').
  )

  testRangeBorders = (
    self assert:   '536870911' equals:   536870911 asString.
    self assert:   '536870912' equals:   536870912 asString.
    self assert:   '536870913' equals:   536870913 asString.
    self assert:  '1073741823' equals:  1073741823 asString.
    self assert:  '1073741824' equals:  1073741824 asString.
    self assert:  '1073741825' equals:  1073741825 asString.
    self assert:  '2147483647' equals:  2147483647 asString.
    self assert:  '-536870911' equals:  -536870911 asString.
    self assert:  '-536870912' equals:  -536870912 asString.
    self assert:  '-536870913' equals:  -536870913 asString.
    self assert: '-1073741823' equals: -1073741823 asString.
    self assert: '-1073741824' equals: -1073741824 asString.
    self assert: '-1073741825' equals: -1073741825 asString.
    self assert: '-2147483647' equals: -2147483647 asString.
    self assert: '-2147483648' equals: -2147483648 asString.
  )

  testComparisons = (
    self assert: ( 9 =  9).
    self deny:   ( 1 =  2).
    self deny:   ( 0 <  0).
    self assert: ( 1 <  2).
    self deny:   ( 2 <  1).
```

```
    self assert: (-3 <  2).
    self deny:   ( 3 < -2).
    self deny:   ( 0 >  0).
    self deny:   ( 1 >  2).
    self assert: ( 2 >  1).
    self deny:   (-3 >  2).
    self assert: ( 3 > -2).
    self assert: ( 4 >= 3).
    self assert: ( 3 >= 3).
    self deny:   ( 2 >= 3).
    self assert: ( 2 <= 4).
    self assert: ( 3 <= 3).
    self deny:   ( 4 <= 3).
  )

  testAddition = (
    self assert: 0 equals: ( 0+0).
    self assert: 1 equals: ( 1+0).
    self assert: 1 equals: ( 0+1).
    self assert: 2 equals: ( 1+1).
    self assert: 0 equals: (-1+1).
    self assert: 1 equals: (-1+2).
  )

  testSubtraction = (
    self assert:  1 equals: (1-0).
    self assert: -1 equals: (0-1).
    self assert:  1 equals: (2-1).
  )

  testMultiplication = (
    self assert:   0 equals: ( 1*  0).
    self assert:  -1 equals: (-1*  1).
    self assert: -25 equals: ( 5* -5).
    self assert:  12 equals: (-3* -4).
  )

  testDivision = (
    self assert:  1 equals: (  1/  1).
    self assert:  1 equals: (  3/  2).
    self assert: -2 equals: (  4/ -2).
    self assert: -2 equals: ( -6/  3).
    self assert:  3 equals: (-12/ -4).
  )

  testDouble = (
    self assert:  6 equals: ( 36//  6).
    self assert: -5 equals: (-10//  2).
    self assert: -4 equals: ( 20// -5).
    self assert:  1 equals: ( -5// -5).
  )

  testModulo = (
    self assert:  1 equals: ( 10 %  3).
    self assert: -2 equals: ( 10 % -3).
    self assert:  2 equals: (-10 %  3).
    self assert: -1 equals: (-10 % -3).
    self assert:  0 equals: ( 10 %  5).

    self assert:  1 equals: ( 10 rem:  3).
    self assert:  1 equals: ( 10 rem: -3).
```

```
    self assert: -1 equals: (-10 rem:  3).
    self assert: -1 equals: (-10 rem: -3).
    self assert:  0 equals: ( 10 rem:  5).
  )

  testAbsoluteValue = (
    self assert: 4 equals: -4 abs.
    self assert: 4 equals:  4 abs.

    self assert: 9223372036854775296 equals: -9223372036854775296 abs.
    self assert: 9223372036854775296 equals:  9223372036854775296 abs.
  )

  testNegated = (
    self assert: -23 equals: ( 23 negated).
    self assert:  23 equals: (-23 negated).
  )

  testSquareRoot = (
    self assert:       5 equals: (25 sqrt).
    self assert: Integer equals: (25 sqrt class).
  )

  testAnd = (
    self assert: 0 equals: (2 & 1).
    self assert: 2 equals: (2 & 2).
  )

  testBitXor = (
    self assert: 0 equals: (1 bitXor: 1).
    self assert: 3 equals: (2 bitXor: 1).
  )

  testAs32BitUnsignedValue = (
    self assert: 1 <<  1 equals: (1 <<  1) as32BitUnsignedValue.
    self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
    self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
    self assert:       0 equals: (1 << 32) as32BitUnsignedValue.
    self assert: 4294967295 equals:     -1 as32BitUnsignedValue.
    self assert: 512        equals: -9223372036854775296 as32BitUnsignedValue.
    self assert: 4294966784 equals:  9223372036854775296 as32BitUnsignedValue.
  )

  testAs32BitSignedValue = (
    self assert:     1 <<  1 equals: (1 <<  1) as32BitSignedValue.
    self assert:     1 << 10 equals: (1 << 10) as32BitSignedValue.
    self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
    self assert:           0 equals: (1 << 32) as32BitSignedValue.

    self assert:  512 equals:  -9223372036854775296 as32BitSignedValue.
    self assert: -512 equals:   9223372036854775296 as32BitSignedValue.
  )

  testAsDouble = (
    self assert: 0.0          equals: 0 asDouble.
    self assert: Double       is:     0 asDouble class.

    self assert: 2147483648.0 equals: 2147483648 asDouble.
    self assert: Double       is:     2147483648 asDouble class.
```

```

    self assert: -2147483648.0 equals: -2147483648 asDouble.
    self assert: Double       is:      -2147483648 asDouble class.
  )

  testUnsignedRightShift = (
    self assert: 0   equals:    1 >>> 1.
    self assert: 512 equals: 1024 >>> 1.
    self assert: 127 equals: 1023 >>> 3.

    "not sure whether we should really insist on this"
    self optional: #toBeSpecified assert: 9223372036854775807 equals: -1 >>> 1.
    self optional: #toBeSpecified assert: 9223372036854775296 equals: -1024 >>> 1.
  )

  testMin = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
                 -50 -2147483648 922337203685477580700 -922337203685477580700
                 922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
                 -51 -2147483650                    1 -922337203685477580701
                 -922337203685477580701).

    big doIndexes: [:i |
      self assert: (small at: i)  equals: ((big   at: i) min: (small at: i)).
      self assert: (small at: i)  equals: ((small at: i) min: (big   at: i)) ]
  )

  testMax = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
                 -50 -2147483648 922337203685477580700 -922337203685477580700
                 922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
                 -51 -2147483650                    1 -922337203685477580701
                 -922337203685477580701).
    big doIndexes: [:i |
      self assert: (big at: i)  equals: ((big   at: i) max: (small at: i)).
      self assert: (big at: i)  equals: ((small at: i) max: (big   at: i)) ]
  )
)

```

Examples/Benchmarks/TestSuite/Integer4Test.som

```
"

$Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Integer4Test = TestCase (

  testEqualityAndIdentity = (
    | a b |
    a := 42.
    b := 42.

    self assert: a = b   description: 'Integers are equal based on their value'.
    self assert: a == b  description: 'Integers do not have pointer/reference equality. It is also supposed to be value equality'.

    "Sometimes it can be hard to implement efficiently, but it SHOULD really
     be true for all values of integers."
    a := 1 << 30.  b := 1 << 30.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 32.  b := 1 << 32.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 60.  b := 1 << 60.
    self optional: #integerIdentity assert: a is: b.
  )

  testClassAndValueRanges = (
```

```
    | i |
    self assert: Integer equals: -42 class.
    self assert: Integer equals:   0 class.
    self assert: Integer equals:  23 class.
    self assert: Integer equals:  1073741823 class.
    self assert: Integer equals:  1073741824 class.

    "Let's test for size behavior and corresponding class"
    i := 1 << 30.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1073741824' equals: i asString.

    i := 1 << 32.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '4294967296' equals: i asString.

    i := 1 << 60.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1152921504606846976' equals: i asString.

    i := 1 << 70.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self optional: #bigIntShifts assert: '1180591620717411303424' equals: i asString.

    i := -1 << 30.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1073741824' equals: i asString.

    i := -1 << 32.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-4294967296' equals: i asString.

    i := -1 << 60.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1152921504606846976' equals: i asString.

    i := -1 << 70.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self optional: #bigIntShifts assert: '-1180591620717411303424' equals: i asString.
  )

  testStringConversion = (
    self assert:  '0' equals: ( 0 asString).
    self assert:  '1' equals: ( 1 asString).
    self assert:  '2' equals: ( 2 asString).
    self assert: '-1' equals: (-1 asString).
    self assert: '-2' equals: (-2 asString).

    self assert: 42 equals: '42' asInteger.
    self assert: -2 equals: '-2' asInteger.
  )
```

```

  testIntegerLiterals = (
    "Make sure the parser reads literals correctly. So, check some basic properties"
    self assert: 2 / 2                                  equals:                    1.
    self assert: 50 + 50                                equals:                  100.
    self assert: 92233720368 * 100000000 + 54775807        equals: 9223372036854775807.
    self assert: 92233720368 * 100000000 + 54775807 * 100  equals: 922337203685477580700.
    self assert: 50 - 100                               equals:                  -50.
    self assert: 21474 * -100000 - 83648                equals:          -2147483648.
    self assert: 92233720368 * 100000000 + 54775807 * -100 equals: -922337203685477580700.
  )

  testFromString = (
    self assert:                      1 equals: (Integer fromString:                      '1').
    self assert:                    100 equals: (Integer fromString:                    '100').
    self assert:    9223372036854775807 equals: (Integer fromString: '9223372036854775807').
    self assert:  922337203685477580700 equals: (Integer fromString: '922337203685477580700').
    self assert:                    -50 equals: (Integer fromString:                    '-50').
    self assert:            -2147483648 equals: (Integer fromString:            '-2147483648').
    self assert: -922337203685477580700 equals: (Integer fromString: '-922337203685477580700').
  )

  testRangeBorders = (
    self assert:   '536870911' equals:   536870911 asString.
    self assert:   '536870912' equals:   536870912 asString.
    self assert:   '536870913' equals:   536870913 asString.
    self assert:  '1073741823' equals:  1073741823 asString.
    self assert:  '1073741824' equals:  1073741824 asString.
    self assert:  '1073741825' equals:  1073741825 asString.
    self assert:  '2147483647' equals:  2147483647 asString.
    self assert:  '-536870911' equals:  -536870911 asString.
    self assert:  '-536870912' equals:  -536870912 asString.
    self assert:  '-536870913' equals:  -536870913 asString.
    self assert: '-1073741823' equals: -1073741823 asString.
    self assert: '-1073741824' equals: -1073741824 asString.
    self assert: '-1073741825' equals: -1073741825 asString.
    self assert: '-2147483647' equals: -2147483647 asString.
    self assert: '-2147483648' equals: -2147483648 asString.
  )

  testComparisons = (
    self assert: ( 9 =  9).
    self deny:   ( 1 =  2).
    self deny:   ( 0 <  0).
    self assert: ( 1 <  2).
    self deny:   ( 2 <  1).
```

```
    self assert: (-3 <  2).
    self deny:   ( 3 < -2).
    self deny:   ( 0 >  0).
    self deny:   ( 1 >  2).
    self assert: ( 2 >  1).
    self deny:   (-3 >  2).
    self assert: ( 3 > -2).
    self assert: ( 4 >= 3).
    self assert: ( 3 >= 3).
    self deny:   ( 2 >= 3).
    self assert: ( 2 <= 4).
    self assert: ( 3 <= 3).
    self deny:   ( 4 <= 3).
  )

  testAddition = (
    self assert: 0 equals: ( 0+0).
    self assert: 1 equals: ( 1+0).
    self assert: 1 equals: ( 0+1).
    self assert: 2 equals: ( 1+1).
    self assert: 0 equals: (-1+1).
    self assert: 1 equals: (-1+2).
  )

  testSubtraction = (
    self assert:  1 equals: (1-0).
    self assert: -1 equals: (0-1).
    self assert:  1 equals: (2-1).
  )

  testMultiplication = (
    self assert:   0 equals: ( 1*  0).
    self assert:  -1 equals: (-1*  1).
    self assert: -25 equals: ( 5* -5).
    self assert:  12 equals: (-3* -4).
  )

  testDivision = (
    self assert:  1 equals: (  1/  1).
    self assert:  1 equals: (  3/  2).
    self assert: -2 equals: (  4/ -2).
    self assert: -2 equals: ( -6/  3).
    self assert:  3 equals: (-12/ -4).
  )

  testDouble = (
    self assert:  6 equals: ( 36//  6).
    self assert: -5 equals: (-10//  2).
    self assert: -4 equals: ( 20// -5).
    self assert:  1 equals: ( -5// -5).
  )

  testModulo = (
    self assert:  1 equals: ( 10 %  3).
    self assert: -2 equals: ( 10 % -3).
    self assert:  2 equals: (-10 %  3).
    self assert: -1 equals: (-10 % -3).
    self assert:  0 equals: ( 10 %  5).

    self assert:  1 equals: ( 10 rem:  3).
    self assert:  1 equals: ( 10 rem: -3).
```

```
    self assert: -1 equals: (-10 rem:  3).
    self assert: -1 equals: (-10 rem: -3).
    self assert:  0 equals: ( 10 rem:  5).
  )

  testAbsoluteValue = (
    self assert: 4 equals: -4 abs.
    self assert: 4 equals:  4 abs.

    self assert: 9223372036854775296 equals: -9223372036854775296 abs.
    self assert: 9223372036854775296 equals:  9223372036854775296 abs.
  )

  testNegated = (
    self assert: -23 equals: ( 23 negated).
    self assert:  23 equals: (-23 negated).
  )

  testSquareRoot = (
    self assert:       5 equals: (25 sqrt).
    self assert: Integer equals: (25 sqrt class).
  )

  testAnd = (
    self assert: 0 equals: (2 & 1).
    self assert: 2 equals: (2 & 2).
  )

  testBitXor = (
    self assert: 0 equals: (1 bitXor: 1).
    self assert: 3 equals: (2 bitXor: 1).
  )

  testAs32BitUnsignedValue = (
    self assert: 1 <<  1 equals: (1 <<  1) as32BitUnsignedValue.
    self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
    self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
    self assert:       0 equals: (1 << 32) as32BitUnsignedValue.
    self assert: 4294967295 equals:     -1 as32BitUnsignedValue.
    self assert: 512        equals: -9223372036854775296 as32BitUnsignedValue.
    self assert: 4294966784 equals:  9223372036854775296 as32BitUnsignedValue.
  )

  testAs32BitSignedValue = (
    self assert:     1 <<  1 equals: (1 <<  1) as32BitSignedValue.
    self assert:     1 << 10 equals: (1 << 10) as32BitSignedValue.
    self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
    self assert:           0 equals: (1 << 32) as32BitSignedValue.

    self assert:  512 equals:  -9223372036854775296 as32BitSignedValue.
    self assert: -512 equals:   9223372036854775296 as32BitSignedValue.
  )

  testAsDouble = (
    self assert: 0.0          equals: 0 asDouble.
    self assert: Double       is:     0 asDouble class.

    self assert: 2147483648.0 equals: 2147483648 asDouble.
    self assert: Double       is:     2147483648 asDouble class.
```

```

    self assert: -2147483648.0 equals: -2147483648 asDouble.
    self assert: Double       is:      -2147483648 asDouble class.
  )

  testUnsignedRightShift = (
    self assert: 0   equals:    1 >>> 1.
    self assert: 512 equals: 1024 >>> 1.
    self assert: 127 equals: 1023 >>> 3.

    "not sure whether we should really insist on this"
    self optional: #toBeSpecified assert: 9223372036854775807 equals: -1 >>> 1.
    self optional: #toBeSpecified assert: 9223372036854775296 equals: -1024 >>> 1.
  )

  testMin = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
                 -50 -2147483648 922337203685477580700 -922337203685477580700
                 922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
                 -51 -2147483650                     1 -922337203685477580701
                 -922337203685477580701).

    big doIndexes: [:i |
      self assert: (small at: i)  equals: ((big   at: i) min: (small at: i)).
      self assert: (small at: i)  equals: ((small at: i) min: (big   at: i)) ]
  )

  testMax = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
                 -50 -2147483648 922337203685477580700 -922337203685477580700
                 922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
                 -51 -2147483650                     1 -922337203685477580701
                 -922337203685477580701).
    big doIndexes: [:i |
      self assert: (big at: i)  equals: ((big   at: i) max: (small at: i)).
      self assert: (big at: i)  equals: ((small at: i) max: (big   at: i)) ]
  )
)

```

Examples/Benchmarks/TestSuite/Integer5Test.som

```
"

$Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Integer5Test = TestCase (

  testEqualityAndIdentity = (
    | a b |
    a := 42.
    b := 42.

    self assert: a = b   description: 'Integers are equal based on their value'.
    self assert: a == b  description: 'Integers do not have pointer/reference equality. It is also supposed to be value equality'.

    "Sometimes it can be hard to implement efficiently, but it SHOULD really
     be true for all values of integers."
    a := 1 << 30.  b := 1 << 30.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 32.  b := 1 << 32.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 60.  b := 1 << 60.
    self optional: #integerIdentity assert: a is: b.
  )

  testClassAndValueRanges = (
```

```
    | i |
    self assert: Integer equals: -42 class.
    self assert: Integer equals:   0 class.
    self assert: Integer equals:  23 class.
    self assert: Integer equals:  1073741823 class.
    self assert: Integer equals:  1073741824 class.

    "Let's test for size behavior and corresponding class"
    i := 1 << 30.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1073741824' equals: i asString.

    i := 1 << 32.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '4294967296' equals: i asString.

    i := 1 << 60.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1152921504606846976' equals: i asString.

    i := 1 << 70.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self optional: #bigIntShifts assert: '1180591620717411303424' equals: i asString.

    i := -1 << 30.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1073741824' equals: i asString.

    i := -1 << 32.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-4294967296' equals: i asString.

    i := -1 << 60.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1152921504606846976' equals: i asString.

    i := -1 << 70.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self optional: #bigIntShifts assert: '-1180591620717411303424' equals: i asString.
  )

  testStringConversion = (
    self assert:  '0' equals: ( 0 asString).
    self assert:  '1' equals: ( 1 asString).
    self assert:  '2' equals: ( 2 asString).
    self assert: '-1' equals: (-1 asString).
    self assert: '-2' equals: (-2 asString).

    self assert: 42 equals: '42' asInteger.
    self assert: -2 equals: '-2' asInteger.
  )
```

```

  testIntegerLiterals = (
    "Make sure the parser reads literals correctly. So, check some basic properties"
    self assert: 2 / 2                 equals:                       1.
    self assert: 50 + 50               equals:                     100.
    self assert: 92233720368 * 100000000 + 54775807        equals: 9223372036854775807.
    self assert: 92233720368 * 100000000 + 54775807 * 100  equals: 922337203685477580700.
    self assert: 50 - 100              equals:                     -50.
    self assert: 21474 * -100000 - 83648   equals:             -2147483648.
    self assert: 92233720368 * 100000000 + 54775807 * -100 equals: -922337203685477580700.
  )

  testFromString = (
    self assert:                      1 equals: (Integer fromString:                      '1').
    self assert:                    100 equals: (Integer fromString:                    '100').
    self assert:    9223372036854775807 equals: (Integer fromString: '9223372036854775807').
    self assert:  922337203685477580700 equals: (Integer fromString: '922337203685477580700').
    self assert:                    -50 equals: (Integer fromString:                    '-50').
    self assert:            -2147483648 equals: (Integer fromString:            '-2147483648').
    self assert: -922337203685477580700 equals: (Integer fromString: '-922337203685477580700').
  )

  testRangeBorders = (
    self assert:   '536870911' equals:   536870911 asString.
    self assert:   '536870912' equals:   536870912 asString.
    self assert:   '536870913' equals:   536870913 asString.
    self assert:  '1073741823' equals:  1073741823 asString.
    self assert:  '1073741824' equals:  1073741824 asString.
    self assert:  '1073741825' equals:  1073741825 asString.
    self assert:  '2147483647' equals:  2147483647 asString.
    self assert:  '-536870911' equals:  -536870911 asString.
    self assert:  '-536870912' equals:  -536870912 asString.
    self assert:  '-536870913' equals:  -536870913 asString.
    self assert: '-1073741823' equals: -1073741823 asString.
    self assert: '-1073741824' equals: -1073741824 asString.
    self assert: '-1073741825' equals: -1073741825 asString.
    self assert: '-2147483647' equals: -2147483647 asString.
    self assert: '-2147483648' equals: -2147483648 asString.
  )

  testComparisons = (
    self assert: ( 9 =  9).
    self deny:   ( 1 =  2).
    self deny:   ( 0 <  0).
    self assert: ( 1 <  2).
    self deny:   ( 2 <  1).
```

```
    self assert: (-3 <  2).
    self deny:   ( 3 < -2).
    self deny:   ( 0 >  0).
    self deny:   ( 1 >  2).
    self assert: ( 2 >  1).
    self deny:   (-3 >  2).
    self assert: ( 3 > -2).
    self assert: ( 4 >= 3).
    self assert: ( 3 >= 3).
    self deny:   ( 2 >= 3).
    self assert: ( 2 <= 4).
    self assert: ( 3 <= 3).
    self deny:   ( 4 <= 3).
  )

  testAddition = (
    self assert: 0 equals: ( 0+0).
    self assert: 1 equals: ( 1+0).
    self assert: 1 equals: ( 0+1).
    self assert: 2 equals: ( 1+1).
    self assert: 0 equals: (-1+1).
    self assert: 1 equals: (-1+2).
  )

  testSubtraction = (
    self assert:  1 equals: (1-0).
    self assert: -1 equals: (0-1).
    self assert:  1 equals: (2-1).
  )

  testMultiplication = (
    self assert:   0 equals: ( 1*  0).
    self assert:  -1 equals: (-1*  1).
    self assert: -25 equals: ( 5* -5).
    self assert:  12 equals: (-3* -4).
  )

  testDivision = (
    self assert:  1 equals: (  1/  1).
    self assert:  1 equals: (  3/  2).
    self assert: -2 equals: (  4/ -2).
    self assert: -2 equals: ( -6/  3).
    self assert:  3 equals: (-12/ -4).
  )

  testDouble = (
    self assert:  6 equals: ( 36//  6).
    self assert: -5 equals: (-10//  2).
    self assert: -4 equals: ( 20// -5).
    self assert:  1 equals: ( -5// -5).
  )

  testModulo = (
    self assert:  1 equals: ( 10 %  3).
    self assert: -2 equals: ( 10 % -3).
    self assert:  2 equals: (-10 %  3).
    self assert: -1 equals: (-10 % -3).
    self assert:  0 equals: ( 10 %  5).

    self assert:  1 equals: ( 10 rem:  3).
    self assert:  1 equals: ( 10 rem: -3).
```

```
    self assert: -1 equals: (-10 rem:  3).
    self assert: -1 equals: (-10 rem: -3).
    self assert:  0 equals: ( 10 rem:  5).
  )

  testAbsoluteValue = (
    self assert: 4 equals: -4 abs.
    self assert: 4 equals:  4 abs.

    self assert: 9223372036854775296 equals: -9223372036854775296 abs.
    self assert: 9223372036854775296 equals:  9223372036854775296 abs.
  )

  testNegated = (
    self assert: -23 equals: ( 23 negated).
    self assert:  23 equals: (-23 negated).
  )

  testSquareRoot = (
    self assert:       5 equals: (25 sqrt).
    self assert: Integer equals: (25 sqrt class).
  )

  testAnd = (
    self assert: 0 equals: (2 & 1).
    self assert: 2 equals: (2 & 2).
  )

  testBitXor = (
    self assert: 0 equals: (1 bitXor: 1).
    self assert: 3 equals: (2 bitXor: 1).
  )

  testAs32BitUnsignedValue = (
    self assert: 1 <<  1 equals: (1 <<  1) as32BitUnsignedValue.
    self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
    self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
    self assert:       0 equals: (1 << 32) as32BitUnsignedValue.
    self assert: 4294967295 equals:     -1 as32BitUnsignedValue.
    self assert: 512        equals: -9223372036854775296 as32BitUnsignedValue.
    self assert: 4294966784 equals:  9223372036854775296 as32BitUnsignedValue.
  )

  testAs32BitSignedValue = (
    self assert:     1 <<  1 equals: (1 <<  1) as32BitSignedValue.
    self assert:     1 << 10 equals: (1 << 10) as32BitSignedValue.
    self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
    self assert:           0 equals: (1 << 32) as32BitSignedValue.

    self assert:  512 equals:  -9223372036854775296 as32BitSignedValue.
    self assert: -512 equals:   9223372036854775296 as32BitSignedValue.
  )

  testAsDouble = (
    self assert: 0.0          equals: 0 asDouble.
    self assert: Double       is:     0 asDouble class.

    self assert: 2147483648.0 equals: 2147483648 asDouble.
    self assert: Double       is:     2147483648 asDouble class.
```

```
    self assert: -2147483648.0 equals: -2147483648 asDouble.
    self assert: Double       is:      -2147483648 asDouble class.
  )

  testUnsignedRightShift = (
    self assert: 0   equals:    1 >>> 1.
    self assert: 512 equals: 1024 >>> 1.
    self assert: 127 equals: 1023 >>> 3.

    "not sure whether we should really insist on this"
    self optional: #toBeSpecified assert: 9223372036854775807 equals: -1 >>> 1.
    self optional: #toBeSpecified assert: 9223372036854775296 equals: -1024 >>> 1.
  )

  testMin = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
               -50 -2147483648 922337203685477580700 -922337203685477580700
               922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
               -51 -2147483650                     1 -922337203685477580701
               -922337203685477580701).

    big doIndexes: [:i |
      self assert: (small at: i)  equals: ((big   at: i) min: (small at: i)).
      self assert: (small at: i)  equals: ((small at: i) min: (big   at: i)) ]
  )

  testMax = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
               -50 -2147483648 922337203685477580700 -922337203685477580700
               922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
               -51 -2147483650                     1 -922337203685477580701
               -922337203685477580701).
    big doIndexes: [:i |
      self assert: (big at: i)  equals: ((big   at: i) max: (small at: i)).
      self assert: (big at: i)  equals: ((small at: i) max: (big   at: i)) ]
  )
)
```

Examples/Benchmarks/TestSuite/IntegerTest.som

```
"

$Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

IntegerTest = TestCase (

  testEqualityAndIdentity = (
    | a b |
    a := 42.
    b := 42.

    self assert: a = b   description: 'Integers are equal based on their value'.
    self assert: a == b  description: 'Integers do not have pointer/reference equality. It is also supposed to be value equality'.

    "Sometimes it can be hard to implement efficiently, but it SHOULD really
     be true for all values of integers."
    a := 1 << 30.  b := 1 << 30.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 32.  b := 1 << 32.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 60.  b := 1 << 60.
    self optional: #integerIdentity assert: a is: b.
  )

  testClassAndValueRanges = (
```

```
    | i |
    self assert: Integer equals: -42 class.
    self assert: Integer equals:   0 class.
    self assert: Integer equals:  23 class.
    self assert: Integer equals:  1073741823 class.
    self assert: Integer equals:  1073741824 class.

    "Let's test for size behavior and corresponding class"
    i := 1 << 30.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1073741824' equals: i asString.

    i := 1 << 32.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '4294967296' equals: i asString.

    i := 1 << 60.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1152921504606846976' equals: i asString.

    i := 1 << 70.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self optional: #bigIntShifts assert: '1180591620717411303424' equals: i asString.

    i := -1 << 30.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1073741824' equals: i asString.

    i := -1 << 32.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-4294967296' equals: i asString.

    i := -1 << 60.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1152921504606846976' equals: i asString.

    i := -1 << 70.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self optional: #bigIntShifts assert: '-1180591620717411303424' equals: i asString.
  )

  testStringConversion = (
    self assert:  '0' equals: ( 0 asString).
    self assert:  '1' equals: ( 1 asString).
    self assert:  '2' equals: ( 2 asString).
    self assert: '-1' equals: (-1 asString).
    self assert: '-2' equals: (-2 asString).

    self assert: 42 equals: '42' asInteger.
    self assert: -2 equals: '-2' asInteger.
  )
```

```

  testIntegerLiterals = (
    "Make sure the parser reads literals correctly. So, check some basic properties"
    self assert: 2 / 2                                   equals:                    1.
    self assert: 50 + 50                                 equals:                  100.
    self assert: 92233720368 * 100000000 + 54775807        equals: 9223372036854775807.
    self assert: 92233720368 * 100000000 + 54775807 * 100  equals: 9223372036854775807 00.
    self assert: 50 - 100                                equals:                  -50.
    self assert: 21474 * -100000 - 83648                 equals:          -2147483648.
    self assert: 92233720368 * 100000000 + 54775807 * -100 equals: -922337203685477580700.
  )

  testFromString = (
    self assert:                      1 equals: (Integer fromString:                      '1').
    self assert:                    100 equals: (Integer fromString:                    '100').
    self assert:    9223372036854775807 equals: (Integer fromString: '9223372036854775807').
    self assert:  922337203685477580700 equals: (Integer fromString: '922337203685477580700').
    self assert:                    -50 equals: (Integer fromString:                    '-50').
    self assert:            -2147483648 equals: (Integer fromString:            '-2147483648').
    self assert: -922337203685477580700 equals: (Integer fromString: '-922337203685477580700').
  )

  testRangeBorders = (
    self assert:   '536870911' equals:   536870911 asString.
    self assert:   '536870912' equals:   536870912 asString.
    self assert:   '536870913' equals:   536870913 asString.
    self assert:  '1073741823' equals:  1073741823 asString.
    self assert:  '1073741824' equals:  1073741824 asString.
    self assert:  '1073741825' equals:  1073741825 asString.
    self assert:  '2147483647' equals:  2147483647 asString.
    self assert:  '-536870911' equals:  -536870911 asString.
    self assert:  '-536870912' equals:  -536870912 asString.
    self assert:  '-536870913' equals:  -536870913 asString.
    self assert: '-1073741823' equals: -1073741823 asString.
    self assert: '-1073741824' equals: -1073741824 asString.
    self assert: '-1073741825' equals: -1073741825 asString.
    self assert: '-2147483647' equals: -2147483647 asString.
    self assert: '-2147483648' equals: -2147483648 asString.
  )

  testComparisons = (
    self assert: ( 9 =  9).
    self deny:   ( 1 =  2).
    self deny:   ( 0 <  0).
    self assert: ( 1 <  2).
    self deny:   ( 2 <  1).
```

```
    self assert: (-3 <  2).
    self deny:   ( 3 < -2).
    self deny:   ( 0 >  0).
    self deny:   ( 1 >  2).
    self assert: ( 2 >  1).
    self deny:   (-3 >  2).
    self assert: ( 3 > -2).
    self assert: ( 4 >= 3).
    self assert: ( 3 >= 3).
    self deny:   ( 2 >= 3).
    self assert: ( 2 <= 4).
    self assert: ( 3 <= 3).
    self deny:   ( 4 <= 3).
  )

  testAddition = (
    self assert: 0 equals: ( 0+0).
    self assert: 1 equals: ( 1+0).
    self assert: 1 equals: ( 0+1).
    self assert: 2 equals: ( 1+1).
    self assert: 0 equals: (-1+1).
    self assert: 1 equals: (-1+2).
  )

  testSubtraction = (
    self assert:  1 equals: (1-0).
    self assert: -1 equals: (0-1).
    self assert:  1 equals: (2-1).
  )

  testMultiplication = (
    self assert:   0 equals: ( 1*  0).
    self assert:  -1 equals: (-1*  1).
    self assert: -25 equals: ( 5* -5).
    self assert:  12 equals: (-3* -4).
  )

  testDivision = (
    self assert:  1 equals: (  1/  1).
    self assert:  1 equals: (  3/  2).
    self assert: -2 equals: (  4/ -2).
    self assert: -2 equals: ( -6/  3).
    self assert:  3 equals: (-12/ -4).
  )

  testDouble = (
    self assert:  6 equals: ( 36//  6).
    self assert: -5 equals: (-10//  2).
    self assert: -4 equals: ( 20// -5).
    self assert:  1 equals: ( -5// -5).
  )

  testModulo = (
    self assert:  1 equals: ( 10 %  3).
    self assert: -2 equals: ( 10 % -3).
    self assert:  2 equals: (-10 %  3).
    self assert: -1 equals: (-10 % -3).
    self assert:  0 equals: ( 10 %  5).

    self assert:  1 equals: ( 10 rem:  3).
    self assert:  1 equals: ( 10 rem: -3).
```

```
    self assert: -1 equals: (-10 rem:  3).
    self assert: -1 equals: (-10 rem: -3).
    self assert:  0 equals: ( 10 rem:  5).
  )

  testAbsoluteValue = (
    self assert: 4 equals: -4 abs.
    self assert: 4 equals:  4 abs.

    self assert: 9223372036854775296 equals: -9223372036854775296 abs.
    self assert: 9223372036854775296 equals:  9223372036854775296 abs.
  )

  testNegated = (
    self assert: -23 equals: ( 23 negated).
    self assert:  23 equals: (-23 negated).
  )

  testSquareRoot = (
    self assert:       5 equals: (25 sqrt).
    self assert: Integer equals: (25 sqrt class).
  )

  testAnd = (
    self assert: 0 equals: (2 & 1).
    self assert: 2 equals: (2 & 2).
  )

  testBitXor = (
    self assert: 0 equals: (1 bitXor: 1).
    self assert: 3 equals: (2 bitXor: 1).
  )

  testAs32BitUnsignedValue = (
    self assert: 1 <<  1 equals: (1 <<  1) as32BitUnsignedValue.
    self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
    self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
    self assert:       0 equals: (1 << 32) as32BitUnsignedValue.
    self assert: 4294967295 equals:     -1 as32BitUnsignedValue.
    self assert: 512        equals: -9223372036854775296 as32BitUnsignedValue.
    self assert: 4294966784 equals:  9223372036854775296 as32BitUnsignedValue.
  )

  testAs32BitSignedValue = (
    self assert:     1 <<  1 equals: (1 <<  1) as32BitSignedValue.
    self assert:     1 << 10 equals: (1 << 10) as32BitSignedValue.
    self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
    self assert:           0 equals: (1 << 32) as32BitSignedValue.

    self assert:  512 equals:  -9223372036854775296 as32BitSignedValue.
    self assert: -512 equals:   9223372036854775296 as32BitSignedValue.
  )

  testAsDouble = (
    self assert: 0.0          equals: 0 asDouble.
    self assert: Double       is:     0 asDouble class.

    self assert: 2147483648.0 equals: 2147483648 asDouble.
    self assert: Double       is:     2147483648 asDouble class.
```

```

    self assert: -2147483648.0 equals: -2147483648 asDouble.
    self assert: Double       is:      -2147483648 asDouble class.
  )

  testUnsignedRightShift = (
    self assert: 0   equals:    1 >>> 1.
    self assert: 512 equals: 1024 >>> 1.
    self assert: 127 equals: 1023 >>> 3.

    "not sure whether we should really insist on this"
    self optional: #toBeSpecified assert: 9223372036854775807 equals: -1 >>> 1.
    self optional: #toBeSpecified assert: 9223372036854775296 equals: -1024 >>> 1.
  )

  testMin = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
               -50 -2147483648 922337203685477580700 -922337203685477580700
               922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
               -51 -2147483650                    1 -922337203685477580701
               -922337203685477580701).

    big doIndexes: [:i |
      self assert: (small at: i)  equals: ((big   at: i) min: (small at: i)).
      self assert: (small at: i)  equals: ((small at: i) min: (big   at: i)) ]
  )

  testMax = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 922337203685477580700
               -50 -2147483648 922337203685477580700 -922337203685477580700
               922337203685477580700).
    small := #(0  52 9223372036854775296 922337203685477529600
               -51 -2147483650                    1 -922337203685477580701
               -922337203685477580701).
    big doIndexes: [:i |
      self assert: (big at: i)  equals: ((big   at: i) max: (small at: i)).
      self assert: (big at: i)  equals: ((small at: i) max: (big   at: i)) ]
  )
)

```

Examples/Benchmarks/TestSuite/Preliminary2Test.som

```
"

$Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"... something just a bit complicated that tests iteration with
blocks, so that we might fail here rather than when the other tests
start, in case things are broken."

Preliminary2Test = TestCase (

    testBasicSanity = (
        | sum |
        sum := 0.
        1, 2, 3 do: [ :i |
            sum := sum + i.
            i<2 ifTrue: [ sum := sum*2 ].
            i>2 ifFalse: [ sum := sum*2 ] ].
        self assert: 15 equals: sum
    )

)

```

```
"

$Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"... something just a bit complicated that tests iteration with
blocks, so that we might fail here rather than when the other tests
start, in case things are broken."

Preliminary3Test = TestCase (

    testBasicSanity = (
        | sum |
        sum := 0.
        1, 2, 3 do: [ :i |
            sum := sum + i.
            i<2 ifTrue: [ sum := sum*2 ].
            i>2 ifFalse: [ sum := sum*2 ] ].
        self assert: 15 equals: sum
    )

)
```

Examples/Benchmarks/TestSuite/Preliminary4Test.som

```
"

$Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"... something just a bit complicated that tests iteration with
blocks, so that we might fail here rather than when the other tests
start, in case things are broken."

Preliminary4Test = TestCase (

    testBasicSanity = (
        | sum |
        sum := 0.
        1, 2, 3 do: [ :i |
            sum := sum + i.
            i<2 ifTrue: [ sum := sum*2 ].
            i>2 ifFalse: [ sum := sum*2 ] ].
        self assert: 15 equals: sum
    )

)
```

```
"

$Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"... something just a bit complicated that tests iteration with
blocks, so that we might fail here rather than when the other tests
start, in case things are broken."

Preliminary5Test = TestCase (

    testBasicSanity = (
        | sum |
        sum := 0.
        1, 2, 3 do: [ :i |
            sum := sum + i.
            i<2 ifTrue: [ sum := sum*2 ].
            i>2 ifFalse: [ sum := sum*2 ] ].
        self assert: 15 equals: sum
    )

)
```

Examples/Benchmarks/TestSuite/PreliminaryTest.som

```
"

$Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"... something just a bit complicated that tests iteration with
blocks, so that we might fail here rather than when the other tests
start, in case things are broken."

PreliminaryTest = TestCase (

    testBasicSanity = (
        | sum |
        sum := 0.
        1, 2, 3 do: [ :i |
            sum := sum + i.
            i<2 ifTrue: [ sum := sum*2 ].
            i>2 ifFalse: [ sum := sum*2 ] ].
        self assert: 15 equals: sum
    )

)
```

Examples/Benchmarks/TestSuite/Reflection2Test.som

```
"

$Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Reflection2Test = TestCase (
  testResondsTo = (
    self assert: (Object new respondsTo: #isNil).
    self assert: (23 respondsTo: #isNil).
    self assert: (23 respondsTo: #+).
  )

  testMethods = (
    "First method in Object should be #class."
    self assert: #class equals: (Object methods at: 1) signature.
    self assert: (Object hasMethod: #==).
  )

  testPerform = (
    | o |
    self assert: Integer equals: (23 perform: #class).
    self assert: (23 perform: #between:and: withArguments: (Array with: 22 with: 24)).

    o := SuperTest new.
    self assert: #super equals: (o perform: #something inSuperclass: SuperTestSuperClass).

    "Trying to see whether the stack in bytecode-based SOMs works properly"
    self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a] ifFalse: [#b]).

```

```
    self assert: 28 equals: 5 + (23 perform: #value).
  )

  testInstVarAtAndPut = (
    | tmp |
    "Testing #at: and #at:put:"
    tmp := Pair withKey: 3 andValue: 42.

    self assert: tmp key equals: (tmp instVarAt: 1).

    tmp instVarAt: 1 put: #foo.
    self assert: #foo    equals: tmp key.
  )

  testName = (
    self assert: #Object equals: Object name.
    self assert: #'Object class' equals: Object class name.
    self assert: #Integer equals: 1 class name.
  )

  testAsString = (
    self assert: 'Object' equals: Object asString.
    self assert: 'Object class' equals: Object class asString.
    self assert: 'Integer' equals: 1 class asString.
  )

  testSelectors = (
    | sels |
    sels := Object selectors.
    self assert: 32 equals: sels length.

    sels contains: #=.
    self assert: (Object hasMethod: #=).

    sels contains: #value.
    self assert: (Object hasMethod: #value).

    sels contains: #notNil.
    self assert: (Object hasMethod: #notNil).
  )
)
```

```
"

$Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Reflection3Test = TestCase (
  testResondsTo = (
    self assert: (Object new respondsTo: #isNil).
    self assert: (23 respondsTo: #isNil).
    self assert: (23 respondsTo: #+).
  )

  testMethods = (
    "First method in Object should be #class."
    self assert: #class equals: (Object methods at: 1) signature.
    self assert: (Object hasMethod: #==).
  )

  testPerform = (
    | o |
    self assert: Integer equals: (23 perform: #class).
    self assert: (23 perform: #between:and: withArguments: (Array with: 22 with: 24)).

    o := SuperTest new.
    self assert: #super equals: (o perform: #something inSuperclass: SuperTestSuperClass).

    "Trying to see whether the stack in bytecode-based SOMs works properly"
    self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a] ifFalse: [#b]).

```

```
    self assert: 28 equals: 5 + (23 perform: #value).
  )

  testInstVarAtAndPut = (
    | tmp |
    "Testing #at: and #at:put:"
    tmp := Pair withKey: 3 andValue: 42.

    self assert: tmp key equals: (tmp instVarAt: 1).

    tmp instVarAt: 1 put: #foo.
    self assert: #foo    equals: tmp key.
  )

  testName = (
    self assert: #Object equals: Object name.
    self assert: #'Object class' equals: Object class name.
    self assert: #Integer equals: 1 class name.
  )

  testAsString = (
    self assert: 'Object' equals: Object asString.
    self assert: 'Object class' equals: Object class asString.
    self assert: 'Integer' equals: 1 class asString.
  )

  testSelectors = (
    | sels |
    sels := Object selectors.
    self assert: 32 equals: sels length.

    sels contains: #=.
    self assert: (Object hasMethod: #=).

    sels contains: #value.
    self assert: (Object hasMethod: #value).

    sels contains: #notNil.
    self assert: (Object hasMethod: #notNil).
  )
)

```

Examples/Benchmarks/TestSuite/Reflection4Test.som

```
"

$Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Reflection4Test = TestCase (
  testResondsTo = (
    self assert: (Object new respondsTo: #isNil).
    self assert: (23 respondsTo: #isNil).
    self assert: (23 respondsTo: #+).
  )

  testMethods = (
    "First method in Object should be #class."
    self assert: #class equals: (Object methods at: 1) signature.
    self assert: (Object hasMethod: #==).
  )

  testPerform = (
    | o |
    self assert: Integer equals: (23 perform: #class).
    self assert: (23 perform: #between:and: withArguments: (Array with: 22 with: 24)).

    o := SuperTest new.
    self assert: #super equals: (o perform: #something inSuperclass: SuperTestSuperClass).

    "Trying to see whether the stack in bytecode-based SOMs works properly"
    self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a] ifFalse: [#b]).

```

```
    self assert: 28 equals: 5 + (23 perform: #value).
  )

  testInstVarAtAndPut = (
    | tmp |
    "Testing #at: and #at:put:"
    tmp := Pair withKey: 3 andValue: 42.

    self assert: tmp key equals: (tmp instVarAt: 1).

    tmp instVarAt: 1 put: #foo.
    self assert: #foo    equals: tmp key.
  )

  testName = (
    self assert: #Object equals: Object name.
    self assert: #'Object class' equals: Object class name.
    self assert: #Integer equals: 1 class name.
  )

  testAsString = (
    self assert: 'Object' equals: Object asString.
    self assert: 'Object class' equals: Object class asString.
    self assert: 'Integer' equals: 1 class asString.
  )

  testSelectors = (
    | sels |
    sels := Object selectors.
    self assert: 32 equals: sels length.

    sels contains: #=.
    self assert: (Object hasMethod: #=).

    sels contains: #value.
    self assert: (Object hasMethod: #value).

    sels contains: #notNil.
    self assert: (Object hasMethod: #notNil).
  )
)
```

Examples/Benchmarks/TestSuite/Reflection5Test.som

```
"

$Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Reflection5Test = TestCase (
  testResondsTo = (
    self assert: (Object new respondsTo: #isNil).
    self assert: (23 respondsTo: #isNil).
    self assert: (23 respondsTo: #+).
  )

  testMethods = (
    "First method in Object should be #class."
    self assert: #class equals: (Object methods at: 1) signature.
    self assert: (Object hasMethod: #==).
  )

  testPerform = (
    | o |
    self assert: Integer equals: (23 perform: #class).
    self assert: (23 perform: #between:and: withArguments: (Array with: 22 with: 24)).

    o := SuperTest new.
    self assert: #super equals: (o perform: #something inSuperclass: SuperTestSuperClass).

    "Trying to see whether the stack in bytecode-based SOMs works properly"
    self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a] ifFalse: [#b]).

```

```
    self assert: 28 equals: 5 + (23 perform: #value).
  )

  testInstVarAtAndPut = (
    | tmp |
    "Testing #at: and #at:put:"
    tmp := Pair withKey: 3 andValue: 42.

    self assert: tmp key equals: (tmp instVarAt: 1).

    tmp instVarAt: 1 put: #foo.
    self assert: #foo    equals: tmp key.
  )

  testName = (
    self assert: #Object equals: Object name.
    self assert: #'Object class' equals: Object class name.
    self assert: #Integer equals: 1 class name.
  )

  testAsString = (
    self assert: 'Object' equals: Object asString.
    self assert: 'Object class' equals: Object class asString.
    self assert: 'Integer' equals: 1 class asString.
  )

  testSelectors = (
    | sels |
    sels := Object selectors.
    self assert: 32 equals: sels length.

    sels contains: #=.
    self assert: (Object hasMethod: #=).

    sels contains: #value.
    self assert: (Object hasMethod: #value).

    sels contains: #notNil.
    self assert: (Object hasMethod: #notNil).
  )
)
```

Examples/Benchmarks/TestSuite/ReflectionTest.som

```
"

$Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

ReflectionTest = TestCase (
  testResondsTo = (
    self assert: (Object new respondsTo: #isNil).
    self assert: (23 respondsTo: #isNil).
    self assert: (23 respondsTo: #+).
  )

  testMethods = (
    "First method in Object should be #class."
    self assert: #class equals: (Object methods at: 1) signature.
    self assert: (Object hasMethod: #==).
  )

  testPerform = (
    | o |
    self assert: Integer equals: (23 perform: #class).
    self assert: (23 perform: #between:and: withArguments: (Array with: 22 with: 24)).

    o := SuperTest new.
    self assert: #super equals: (o perform: #something inSuperclass: SuperTestSuperClass).

    "Trying to see whether the stack in bytecode-based SOMs works properly"
    self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a] ifFalse: [#b]).

```

```
    self assert: 28 equals: 5 + (23 perform: #value).
  )

  testInstVarAtAndPut = (
    | tmp |
    "Testing #at: and #at:put:"
    tmp := Pair withKey: 3 andValue: 42.

    self assert: tmp key equals: (tmp instVarAt: 1).

    tmp instVarAt: 1 put: #foo.
    self assert: #foo    equals: tmp key.
  )

  testName = (
    self assert: #Object equals: Object name.
    self assert: #'Object class' equals: Object class name.
    self assert: #Integer equals: 1 class name.
  )

  testAsString = (
    self assert: 'Object' equals: Object asString.
    self assert: 'Object class' equals: Object class asString.
    self assert: 'Integer' equals: 1 class asString.
  )

  testSelectors = (
    | sels |
    sels := Object selectors.
    self assert: 32 equals: sels length.

    sels contains: #=.
    self assert: (Object hasMethod: #=).

    sels contains: #value.
    self assert: (Object hasMethod: #value).

    sels contains: #notNil.
    self assert: (Object hasMethod: #notNil).
  )
)
```

Examples/Benchmarks/TestSuite/SelfBlock2Test.som

```
"

$Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SelfBlock2Test = TestCase (

  testEscapedBlock = (
    self assert: 42 equals: self give42 value
  )

  give42 = (
    ^[ self giveBlock value ]
  )

  giveBlock = (
    ^self returnBlock value
  )

  returnBlock = (
    ^[ self returnBlock2 value ]
  )

  returnBlock2 = (
    ^[ 42 ]
  )
)
```

```
"

$Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SelfBlock3Test = TestCase (

  testEscapedBlock = (
    self assert: 42 equals: self give42 value
  )

  give42 = (
    ^[ self giveBlock value ]
  )

  giveBlock = (
    ^self returnBlock value
  )

  returnBlock = (
    ^[ self returnBlock2 value ]
  )

  returnBlock2 = (
    ^[ 42 ]
  )
)
```

```
"

$Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SelfBlock4Test = TestCase (

  testEscapedBlock = (
    self assert: 42 equals: self give42 value
  )

  give42 = (
    ^[ self giveBlock value ]
  )

  giveBlock = (
    ^self returnBlock value
  )

  returnBlock = (
    ^[ self returnBlock2 value ]
  )

  returnBlock2 = (
    ^[ 42 ]
  )
)

```

Examples/Benchmarks/TestSuite/SelfBlock5Test.som

```
"

$Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SelfBlock5Test = TestCase (

  testEscapedBlock = (
    self assert: 42 equals: self give42 value
  )

  give42 = (
    ^[ self giveBlock value ]
  )

  giveBlock = (
    ^self returnBlock value
  )

  returnBlock = (
    ^[ self returnBlock2 value ]
  )

  returnBlock2 = (
    ^[ 42 ]
  )
)
```

Examples/Benchmarks/TestSuite/SelfBlockTest.som

```
"

$Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SelfBlockTest = TestCase (

  testEscapedBlock = (
    self assert: 42 equals: self give42 value
  )

  give42 = (
    ^[ self giveBlock value ]
  )

  giveBlock = (
    ^self returnBlock value
  )

  returnBlock = (
    ^[ self returnBlock2 value ]
  )

  returnBlock2 = (
    ^[ 42 ]
  )
)

```

```
Set2Test = TestCase (
  testBasics = (
    | a b t |
    a := Set new.
    b := Set new.

    a add: #a.
    b add: #b.

    self deny: a = b.

    t := Set new.
    t add: #a.

    self deny: a == t. "different objects"
    self assert: a equals: t. "but with equal value"
  )

  testEquality = (
    | a b |
    a := Set new.
    a addAll: #(1 2 3 4).

    b := Set new.
    b addAll: #(1 2 3 4).

    self assert: a = b.
    self deny: a == b.

    a add: 5.

    self deny: a = b.
    b add: 5.

    self assert: a = b.

    b add: #foo.
    self deny: a = b.
    a add: #foo.
    self assert: a = b.
  )

  testUnion = (
    | a b u |
    a := Set new.
    b := Set new.
    a addAll: #(1 2 3 4).
    b addAll: #(1 2 3 4).

    u := a union: b.
    self assert: a = b.
    self assert: u = a.
    self assert: u = b.

    self deny: a == b.
    self deny: u == a.
    self deny: u == b.

```

```
    self assert: 4 equals: u size.

    a add: #mm.
    u := a union: b.
    self assert: 5 equals: u size.

    b add: #oo.
    u := a union: b.
    self assert: 6 equals: u size.

    b add: #mm.
    u := a union: b.
    self assert: 6 equals: u size.
  )

  testIntersection = (
    | a b i |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    i := a intersection: b.
    self assert: 4 equals: i size.

    a do: [:e |
      self assert: (i contains: e) ].

    b := Set new.
    b add: 64.
    b add: 667.

    i := a intersection: b.
    self assert: 2 equals: i size.

    b do: [:e |
      self assert: (i contains: e) ].
  )

  testSetDifference = (
    | a b d |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    d := a - b.
    self assert: d isEmpty.
    self assert: 0 equals: d size.

    b := Set new.
    b add: 43.
    b add: 667.
    b add: 345345.

    d := a - b.
    self assert: 2 equals: d size.
    self assert: (d contains: 64).
    self assert: (d contains: 730).
  )

```

```
testContains = (
  | s |
  s := Set new.

  self deny: (s contains: #'333').
  s add: #'333'.

  self assert: (s contains: #'333').
  s add: 333.
  self assert: (s contains: #'333').
  self assert: (s contains: 333).
)

testRemove = (
  | s o |
  o := Object new.
  s := Set new.
  s add: #sfsdf.
  s add: 323.
  s add: 545.
  s add: self.
  s add: o.

  self assert: (s contains: o).
  self assert: 5 equals: s size.

  s remove: 323.
  self assert: 4 equals: s size.
  self deny: (s contains: 323).
  s add: 65767.
  self assert: 5 equals: s size.

  s remove: o.
  self assert: 4 equals: s size.
  self deny: (s contains: o).
  s remove: self.
  s remove: #sfsdf.
  s remove: 323.
  s remove: 545.
  s remove: 65767.

  self assert: s isEmpty.
)

testFirst = (
  | s |
  s := Set new.
  s addAll: #(233 545 665).

  self assert: 233 equals: s first.

  s remove: 233.
  self assert: 545 equals: s first.

  s remove: 545.
  self assert: 665 equals: s first.
)

testCollect = (
  | s r |
  s := Set new.
```

```
    s addAll: #(21 54642 6753 344 655).

    r := s collect: [:e | e % 10 ].

    self assert: 5 equals: r size.
    self assert: 1 equals: r first.
    self assert: 2 equals: (r at: 2).
  )
)
```

```
Set3Test = TestCase (
  testBasics = (
    | a b t |
    a := Set new.
    b := Set new.

    a add: #a.
    b add: #b.

    self deny: a = b.

    t := Set new.
    t add: #a.

    self deny: a == t. "different objects"
    self assert: a equals: t. "but with equal value"
  )

  testEquality = (
    | a b |
    a := Set new.
    a addAll: #(1 2 3 4).

    b := Set new.
    b addAll: #(1 2 3 4).

    self assert: a = b.
    self deny: a == b.

    a add: 5.

    self deny: a = b.
    b add: 5.

    self assert: a = b.

    b add: #foo.
    self deny: a = b.
    a add: #foo.
    self assert: a = b.
  )

  testUnion = (
    | a b u |
    a := Set new.
    b := Set new.
    a addAll: #(1 2 3 4).
    b addAll: #(1 2 3 4).

    u := a union: b.
    self assert: a = b.
    self assert: u = a.
    self assert: u = b.

    self deny: a == b.
    self deny: u == a.
    self deny: u == b.

```

```
    self assert: 4 equals: u size.

    a add: #mm.
    u := a union: b.
    self assert: 5 equals: u size.

    b add: #oo.
    u := a union: b.
    self assert: 6 equals: u size.

    b add: #mm.
    u := a union: b.
    self assert: 6 equals: u size.
  )

  testIntersection = (
    | a b i |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    i := a intersection: b.
    self assert: 4 equals: i size.

    a do: [:e |
      self assert: (i contains: e) ].

    b := Set new.
    b add: 64.
    b add: 667.

    i := a intersection: b.
    self assert: 2 equals: i size.

    b do: [:e |
      self assert: (i contains: e) ].
  )

  testSetDifference = (
    | a b d |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    d := a - b.
    self assert: d isEmpty.
    self assert: 0 equals: d size.

    b := Set new.
    b add: 43.
    b add: 667.
    b add: 345345.

    d := a - b.
    self assert: 2 equals: d size.
    self assert: (d contains: 64).
    self assert: (d contains: 730).
  )

```

```
testContains = (
  | s |
  s := Set new.

  self deny: (s contains: #'333').
  s add: #'333'.

  self assert: (s contains: #'333').
  s add: 333.
  self assert: (s contains: #'333').
  self assert: (s contains: 333).
)

testRemove = (
  | s o |
  o := Object new.
  s := Set new.
  s add: #sfsdf.
  s add: 323.
  s add: 545.
  s add: self.
  s add: o.

  self assert: (s contains: o).
  self assert: 5 equals: s size.

  s remove: 323.
  self assert: 4 equals: s size.
  self deny: (s contains: 323).
  s add: 65767.
  self assert: 5 equals: s size.

  s remove: o.
  self assert: 4 equals: s size.
  self deny: (s contains: o).
  s remove: self.
  s remove: #sfsdf.
  s remove: 323.
  s remove: 545.
  s remove: 65767.

  self assert: s isEmpty.
)

testFirst = (
  | s |
  s := Set new.
  s addAll: #(233 545 665).

  self assert: 233 equals: s first.

  s remove: 233.
  self assert: 545 equals: s first.

  s remove: 545.
  self assert: 665 equals: s first.
)

testCollect = (
  | s r |
  s := Set new.
```

```
    s addAll: #(21 54642 6753 344 655).

    r := s collect: [:e | e % 10 ].

    self assert: 5 equals: r size.
    self assert: 1 equals: r first.
    self assert: 2 equals: (r at: 2).
  )
)
```

```
Set4Test = TestCase (
  testBasics = (
    | a b t |
    a := Set new.
    b := Set new.

    a add: #a.
    b add: #b.

    self deny: a = b.

    t := Set new.
    t add: #a.

    self deny: a == t. "different objects"
    self assert: a equals: t. "but with equal value"
  )

  testEquality = (
    | a b |
    a := Set new.
    a addAll: #(1 2 3 4).

    b := Set new.
    b addAll: #(1 2 3 4).

    self assert: a = b.
    self deny: a == b.

    a add: 5.

    self deny: a = b.
    b add: 5.

    self assert: a = b.

    b add: #foo.
    self deny: a = b.
    a add: #foo.
    self assert: a = b.
  )

  testUnion = (
    | a b u |
    a := Set new.
    b := Set new.
    a addAll: #(1 2 3 4).
    b addAll: #(1 2 3 4).

    u := a union: b.
    self assert: a = b.
    self assert: u = a.
    self assert: u = b.

    self deny: a == b.
    self deny: u == a.
    self deny: u == b.

```

```
    self assert: 4 equals: u size.

    a add: #mm.
    u := a union: b.
    self assert: 5 equals: u size.

    b add: #oo.
    u := a union: b.
    self assert: 6 equals: u size.

    b add: #mm.
    u := a union: b.
    self assert: 6 equals: u size.
  )

  testIntersection = (
    | a b i |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    i := a intersection: b.
    self assert: 4 equals: i size.

    a do: [:e |
      self assert: (i contains: e) ].

    b := Set new.
    b add: 64.
    b add: 667.

    i := a intersection: b.
    self assert: 2 equals: i size.

    b do: [:e |
      self assert: (i contains: e) ].
  )

  testSetDifference = (
    | a b d |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    d := a - b.
    self assert: d isEmpty.
    self assert: 0 equals: d size.

    b := Set new.
    b add: 43.
    b add: 667.
    b add: 345345.

    d := a - b.
    self assert: 2 equals: d size.
    self assert: (d contains: 64).
    self assert: (d contains: 730).
  )
```

```
  testContains = (
    | s |
    s := Set new.

    self deny: (s contains: #'333').
    s add: #'333'.

    self assert: (s contains: #'333').
    s add: 333.
    self assert: (s contains: #'333').
    self assert: (s contains: 333).
  )

  testRemove = (
    | s o |
    o := Object new.
    s := Set new.
    s add: #sfsdf.
    s add: 323.
    s add: 545.
    s add: self.
    s add: o.

    self assert: (s contains: o).
    self assert: 5 equals: s size.

    s remove: 323.
    self assert: 4 equals: s size.
    self deny: (s contains: 323).
    s add: 65767.
    self assert: 5 equals: s size.

    s remove: o.
    self assert: 4 equals: s size.
    self deny: (s contains: o).
    s remove: self.
    s remove: #sfsdf.
    s remove: 323.
    s remove: 545.
    s remove: 65767.

    self assert: s isEmpty.
  )

  testFirst = (
    | s |
    s := Set new.
    s addAll: #(233 545 665).

    self assert: 233 equals: s first.

    s remove: 233.
    self assert: 545 equals: s first.

    s remove: 545.
    self assert: 665 equals: s first.
  )

  testCollect = (
    | s r |
    s := Set new.
```

```
    s addAll: #(21 54642 6753 344 655).

    r := s collect: [:e | e % 10 ].

    self assert: 5 equals: r size.
    self assert: 1 equals: r first.
    self assert: 2 equals: (r at: 2).
  )
)

```

```
Set5Test = TestCase (
  testBasics = (
    | a b t |
    a := Set new.
    b := Set new.

    a add: #a.
    b add: #b.

    self deny: a = b.

    t := Set new.
    t add: #a.

    self deny: a == t. "different objects"
    self assert: a equals: t. "but with equal value"
  )

  testEquality = (
    | a b |
    a := Set new.
    a addAll: #(1 2 3 4).

    b := Set new.
    b addAll: #(1 2 3 4).

    self assert: a = b.
    self deny: a == b.

    a add: 5.

    self deny: a = b.
    b add: 5.

    self assert: a = b.

    b add: #foo.
    self deny: a = b.
    a add: #foo.
    self assert: a = b.
  )

  testUnion = (
    | a b u |
    a := Set new.
    b := Set new.
    a addAll: #(1 2 3 4).
    b addAll: #(1 2 3 4).

    u := a union: b.
    self assert: a = b.
    self assert: u = a.
    self assert: u = b.

    self deny: a == b.
    self deny: u == a.
    self deny: u == b.

```

```
    self assert: 4 equals: u size.

    a add: #mm.
    u := a union: b.
    self assert: 5 equals: u size.

    b add: #oo.
    u := a union: b.
    self assert: 6 equals: u size.

    b add: #mm.
    u := a union: b.
    self assert: 6 equals: u size.
  )

  testIntersection = (
    | a b i |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    i := a intersection: b.
    self assert: 4 equals: i size.

    a do: [:e |
      self assert: (i contains: e) ].

    b := Set new.
    b add: 64.
    b add: 667.

    i := a intersection: b.
    self assert: 2 equals: i size.

    b do: [:e |
      self assert: (i contains: e) ].
  )

  testSetDifference = (
    | a b d |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    d := a - b.
    self assert: d isEmpty.
    self assert: 0 equals: d size.

    b := Set new.
    b add: 43.
    b add: 667.
    b add: 345345.

    d := a - b.
    self assert: 2 equals: d size.
    self assert: (d contains: 64).
    self assert: (d contains: 730).
  )

```

```
testContains = (
  | s |
  s := Set new.

  self deny: (s contains: #'333').
  s add: #'333'.

  self assert: (s contains: #'333').
  s add: 333.
  self assert: (s contains: #'333').
  self assert: (s contains: 333).
)

testRemove = (
  | s o |
  o := Object new.
  s := Set new.
  s add: #sfsdf.
  s add: 323.
  s add: 545.
  s add: self.
  s add: o.

  self assert: (s contains: o).
  self assert: 5 equals: s size.

  s remove: 323.
  self assert: 4 equals: s size.
  self deny: (s contains: 323).
  s add: 65767.
  self assert: 5 equals: s size.

  s remove: o.
  self assert: 4 equals: s size.
  self deny: (s contains: o).
  s remove: self.
  s remove: #sfsdf.
  s remove: 323.
  s remove: 545.
  s remove: 65767.

  self assert: s isEmpty.
)

testFirst = (
  | s |
  s := Set new.
  s addAll: #(233 545 665).

  self assert: 233 equals: s first.

  s remove: 233.
  self assert: 545 equals: s first.

  s remove: 545.
  self assert: 665 equals: s first.
)

testCollect = (
  | s r |
  s := Set new.
```

```
    s addAll: #(21 54642 6753 344 655).

    r := s collect: [:e | e % 10 ].

    self assert: 5 equals: r size.
    self assert: 1 equals: r first.
    self assert: 2 equals: (r at: 2).
  )
)
```

```
SetTest = TestCase (
  testBasics = (
    | a b t |
    a := Set new.
    b := Set new.

    a add: #a.
    b add: #b.

    self deny: a = b.

    t := Set new.
    t add: #a.

    self deny: a == t. "different objects"
    self assert: a equals: t. "but with equal value"
  )

  testEquality = (
    | a b |
    a := Set new.
    a addAll: #(1 2 3 4).

    b := Set new.
    b addAll: #(1 2 3 4).

    self assert: a = b.
    self deny: a == b.

    a add: 5.

    self deny: a = b.
    b add: 5.

    self assert: a = b.

    b add: #foo.
    self deny: a = b.
    a add: #foo.
    self assert: a = b.
  )

  testUnion = (
    | a b u |
    a := Set new.
    b := Set new.
    a addAll: #(1 2 3 4).
    b addAll: #(1 2 3 4).

    u := a union: b.
    self assert: a = b.
    self assert: u = a.
    self assert: u = b.

    self deny: a == b.
    self deny: u == a.
    self deny: u == b.

```

```
    self assert: 4 equals: u size.

    a add: #mm.
    u := a union: b.
    self assert: 5 equals: u size.

    b add: #oo.
    u := a union: b.
    self assert: 6 equals: u size.

    b add: #mm.
    u := a union: b.
    self assert: 6 equals: u size.
  )

  testIntersection = (
    | a b i |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    i := a intersection: b.
    self assert: 4 equals: i size.

    a do: [:e |
      self assert: (i contains: e) ].

    b := Set new.
    b add: 64.
    b add: 667.

    i := a intersection: b.
    self assert: 2 equals: i size.

    b do: [:e |
      self assert: (i contains: e) ].
  )

  testSetDifference = (
    | a b d |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    d := a - b.
    self assert: d isEmpty.
    self assert: 0 equals: d size.

    b := Set new.
    b add: 43.
    b add: 667.
    b add: 345345.

    d := a - b.
    self assert: 2 equals: d size.
    self assert: (d contains: 64).
    self assert: (d contains: 730).
  )
```

```
  testContains = (
    | s |
    s := Set new.

    self deny: (s contains: #'333').
    s add: #'333'.

    self assert: (s contains: #'333').
    s add: 333.
    self assert: (s contains: #'333').
    self assert: (s contains: 333).
  )

  testRemove = (
    | s o |
    o := Object new.
    s := Set new.
    s add: #sfsdf.
    s add: 323.
    s add: 545.
    s add: self.
    s add: o.

    self assert: (s contains: o).
    self assert: 5 equals: s size.

    s remove: 323.
    self assert: 4 equals: s size.
    self deny: (s contains: 323).
    s add: 65767.
    self assert: 5 equals: s size.

    s remove: o.
    self assert: 4 equals: s size.
    self deny: (s contains: o).
    s remove: self.
    s remove: #sfsdf.
    s remove: 323.
    s remove: 545.
    s remove: 65767.

    self assert: s isEmpty.
  )

  testFirst = (
    | s |
    s := Set new.
    s addAll: #(233 545 665).

    self assert: 233 equals: s first.

    s remove: 233.
    self assert: 545 equals: s first.

    s remove: 545.
    self assert: 665 equals: s first.
  )

  testCollect = (
    | s r |
    s := Set new.
```

```
    s addAll: #(21 54642 6753 344 655).

    r := s collect: [:e | e % 10 ].

    self assert: 5 equals: r size.
    self assert: 1 equals: r first.
    self assert: 2 equals: (r at: 2).
  )
)

```

```
SpecialSelectors2Test = TestCase (
  testMinusMinsPrefix = (
    self assert: self --> 1 equals: 1.
    self assert: self -- 1 equals: 1.
  )

  --> aValue = (
    ^1
  )

  -- aValue = (
  •ã
  )
)

```

Examples/Benchmarks/TestSuite/SpecialSelectors3Test.som

```
SpecialSelectors3Test = TestCase (
  testMinusMinsPrefix = (
    self assert: self --> 1 equals: 1.
    self assert: self -- 1 equals: 1.
  )

  --> aValue = (
    ^1
  )

  -- aValue = (
  •ã
  )
)

```

Examples/Benchmarks/TestSuite/SpecialSelectors4Test.som

```
SpecialSelectors4Test = TestCase (
  testMinusMinsPrefix = (
    self assert: self --> 1 equals: 1.
    self assert: self -- 1 equals: 1.
  )

  --> aValue = (
    ^1
  )

  -- aValue = (
  •ã
  )
)

```

```
SpecialSelectors5Test = TestCase (
  testMinusMinsPrefix = (
    self assert: self --> 1 equals: 1.
    self assert: self -- 1 equals: 1.
  )

  --> aValue = (
    ^1
  )

  -- aValue = (
  •ã
  )
)

```

```
SpecialSelectorsTest = TestCase (
  testMinusMinsPrefix = (
    self assert: self --> 1 equals: 1.
    self assert: self -- 1 equals: 1.
  )

  --> aValue = (
    ^1
  )

  -- aValue = (
  •ã
  )
)
```

Examples/Benchmarks/TestSuite/String2Test.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

String2Test = TestCase (

  testEquality = (
    | str1 str2 |
    str1 := 'foo'.
    str2 := 'bar'.

    self assert: str1 = str1.
    self assert: str1 = 'foo'.
    self assert: str1 = ('f' + 'oo').
    self deny:   str1 = str2.
    self assert: str2 = str2.

    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
    self assert: ('f' + 'o' + 'o') = #foo.
  )

  testEqualEqual = (
    | str1 |
    str1 := 'foo'.
    self assert: str1 == str1.
    self deny:   str1 == str1 asSymbol.
    self deny:   str1 == #foo.
  )

  testLength = (
    self assert: 1 equals: 't' length.
    self assert: 6 equals: ('foo' + 'bar') length.
  )
```

```

  testCharAt = (
    | str |
    str := 'foobar'.
    self assert: 'f' equals: (str charAt: 1).
    self assert: 'o' equals: (str charAt: 2).
    self assert: 'o' equals: (str charAt: 3).
    self assert: 'b' equals: (str charAt: 4).
    self assert: 'a' equals: (str charAt: 5).
    self assert: 'r' equals: (str charAt: 6).
  )

  testStringLiteralLineBreak = (
    | str |
    "Some parsers get the literals and line bounderies wrong"
    str := '
'.
    self assert: '\n' equals: (str charAt: 1).
    self assert: 1 equals: str length.
  )

  testPrimSubstringFrom = (
    | str |
    str := 'foobar'.
    self assert: 'foo'    equals: (str primSubstringFrom: 1 to: 3).
    self assert: 'bar'    equals: (str primSubstringFrom: 4 to: 6).
    self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
    self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
  )

  testSplit = (
    | r |
    r := 'aaaa' split: ','.
    self assert: 1 equals: r length.
    self assert: 'aaaa' equals: (r at: 1).

    r := 'foo.bar' split: '.'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo..bar' split: '.'.
    self assert: 3     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: ''    equals: (r at: 2).
    self assert: 'bar' equals: (r at: 3).

    r := 'foo..bar' split: '..'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo' split: 'bar'.
    self assert: 1     equals: r length.
    self assert: 'foo' equals: (r at: 1).

    self assert: Array is: r class
  )

  testIndexOf = (
    self assert: -1 equals: ('foo' indexOf: 'b').
```

```
    self assert:  1 equals: ('foo' indexOf: 'f').
    self assert:  2 equals: ('foo' indexOf: 'o').
    self assert:  3 equals: ('foo' indexOf: 'o' startingAt: 3).

    self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).

    self assert:  2 equals: ('foo' indexOf: 'oo').
  )

  testBeginsWith = (
    self deny:   ('foo' beginsWith: 'oo').
    self assert: ('foo' beginsWith: 'foo').
  )

  testEndsWith = (
    self assert: ('foo' endsWith: 'foo').
    self assert: ('foo' endsWith: 'oo').
    self deny: ('f' endsWith: 'bar').
    self deny: ('f' endsWith: 'foo').
  )

  testMultiLineString = (
    "Test whether the parser will parse multi-line strings correctly."
    self assert: '
1234567890
1234567890
1234567890
1234567890
1234567890' equals: '
1234567890
1234567890
1234567890
1234567890
1234567890'
  )

  testEscapeSequences = (
    "Tests for escape sequences, not all of them are reliable represented as
     proper strings. So, we do a simple equality test, and check substring or
     length.

     \t’    F " 6† & 7FW
     \b’    & 6·7  6R 6† & 7FW
     \n’    æWvÆ–æR 6† & 7FW
     \r’    6 '&– vR &WGW&â 6† & 7FW
     \f’    f÷&ÖfVVB 6† & 7FW
     \'’    6–ævÆR  V÷FR 6† & 7FW
     \\’  & 6·6Æ 6, 6† & 7FW
     \0   zero byte character
    "

    self assert: '\t' equals: '\t'.
    self assert: 1 equals: '\t' length.

    self assert: '\b' equals: '\b'.
    self assert: 1 equals: '\b' length.

    self assert: '\n' equals: '\n'.
    self assert: 1 equals: '\n' length.
```

```
    self deny: ('\n' endsWith: 'n').

    self assert: '\r' equals: '\r'.
    self assert: 1 equals: '\n' length.
    self deny: ('\r' endsWith: 'r').

    self assert: '\f' equals: '\f'.
    self assert: 1 equals: '\f' length.
    self deny: ('\f' endsWith: 'f').

    self assert: '\'' equals: '\''.
    self assert: 1 equals: '\'' length.

    self assert: '\\' equals: '\\'.
    self assert: 1 equals: '\\' length.

    self assert: '\0' equals: '\0'.
    self assert: 1 equals: '\0' length.
    self assert: 5 equals: '\0rest' length.
  )

  testHash = (
    | str |
    "Hash should be identical for strings that are identical,
     whether given literal or composed at runtime"
    self assert: 'foobar' hashcode equals: 'foobar' hashcode.
    self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode
         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode.

    str := 'foo' + 'bar'.
    str := str + str.
    self assert: 'foobarfoobar' hashcode equals: str hashcode.

    str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw erltnwrtln'.
    self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw erltnwrtln' hashcode
         equals: str hashcode.
  )

  testWhiteSpace = (
    self assert: ' ' isWhiteSpace.
    self assert: '\t' isWhiteSpace.
    self assert: '\t\n   \n \n' isWhiteSpace.

    self deny: '' isWhiteSpace.
    self deny: '\t\n N  \n \n' isWhiteSpace.
    self deny: 'N' isWhiteSpace.
    self deny: '3' isWhiteSpace.
  )

  testLetters = (
    self assert: 'a' isLetters.
    self assert: 'all' isLetters.

    self deny: '' isLetters.
    self deny: ' ' isLetters.
    self deny: '3' isLetters.
    self deny: '3333' isLetters.
    self deny: 'aOo öéÉíä' isLetters.
    self deny: 'aOo1öéÉíä' isLetters.
```

```
  )

  testDigits = (
    self assert: '0' isDigits.
    self assert: '0123' isDigits.
    self assert: '0123456789' isDigits.

    self deny: '' isDigits.
    self deny: ' ' isDigits.
    self deny: 'S' isDigits.
    self deny: '333 3' isDigits.
    self deny: '66i77' isDigits.
    self deny: '66e7' isDigits.
    self deny: 'aOo1öéÉíä' isDigits.
  )

  testAsInteger = (
    self assert:   0 equals:   '0' asInteger.
    self assert: 100 equals: '100' asInteger.
    self assert: 923 equals: '923' asInteger.

    self assert:   -0 equals:   '-0' asInteger.
    self assert: -100 equals: '-100' asInteger.
    self assert: -923 equals: '-923' asInteger.

    self assert: 123342353453453456456456  equals: '123342353453453456456456' asInteger.
  )
)
```

Examples/Benchmarks/TestSuite/String3Test.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

String3Test = TestCase (

  testEquality = (
    | str1 str2 |
    str1 := 'foo'.
    str2 := 'bar'.

    self assert: str1 = str1.
    self assert: str1 = 'foo'.
    self assert: str1 = ('f' + 'oo').
    self deny:   str1 = str2.
    self assert: str2 = str2.

    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
    self assert: ('f' + 'o' + 'o') = #foo.
  )

  testEqualEqual = (
    | str1 |
    str1 := 'foo'.
    self assert: str1 == str1.
    self deny:   str1 == str1 asSymbol.
    self deny:   str1 == #foo.
  )

  testLength = (
    self assert: 1 equals: 't' length.
    self assert: 6 equals: ('foo' + 'bar') length.
  )
```

```

  testCharAt = (
    | str |
    str := 'foobar'.
    self assert: 'f' equals: (str charAt: 1).
    self assert: 'o' equals: (str charAt: 2).
    self assert: 'o' equals: (str charAt: 3).
    self assert: 'b' equals: (str charAt: 4).
    self assert: 'a' equals: (str charAt: 5).
    self assert: 'r' equals: (str charAt: 6).
  )

  testStringLiteralLineBreak = (
    | str |
    "Some parsers get the literals and line bounderies wrong"
    str := '
'.
    self assert: '\n' equals: (str charAt: 1).
    self assert: 1 equals: str length.
  )

  testPrimSubstringFrom = (
    | str |
    str := 'foobar'.
    self assert: 'foo'    equals: (str primSubstringFrom: 1 to: 3).
    self assert: 'bar'    equals: (str primSubstringFrom: 4 to: 6).
    self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
    self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
  )

  testSplit = (
    | r |
    r := 'aaaa' split: ','.
    self assert: 1 equals: r length.
    self assert: 'aaaa' equals: (r at: 1).

    r := 'foo.bar' split: '.'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo..bar' split: '.'.
    self assert: 3     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: ''    equals: (r at: 2).
    self assert: 'bar' equals: (r at: 3).

    r := 'foo..bar' split: '..'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo' split: 'bar'.
    self assert: 1     equals: r length.
    self assert: 'foo' equals: (r at: 1).

    self assert: Array is: r class
  )

  testIndexOf = (
    self assert: -1 equals: ('foo' indexOf: 'b').
```

```
    self assert:  1 equals: ('foo' indexOf: 'f').
    self assert:  2 equals: ('foo' indexOf: 'o').
    self assert:  3 equals: ('foo' indexOf: 'o' startingAt: 3).

    self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).

    self assert:  2 equals: ('foo' indexOf: 'oo').
  )

  testBeginsWith = (
    self deny:   ('foo' beginsWith: 'oo').
    self assert: ('foo' beginsWith: 'foo').
  )

  testEndsWith = (
    self assert: ('foo' endsWith: 'foo').
    self assert: ('foo' endsWith: 'oo').
    self deny: ('f' endsWith: 'bar').
    self deny: ('f' endsWith: 'foo').
  )

  testMultiLineString = (
    "Test whether the parser will parse multi-line strings correctly."
    self assert: '
1234567890
1234567890
1234567890
1234567890
1234567890' equals: '
1234567890
1234567890
1234567890
1234567890
1234567890'
  )

  testEscapeSequences = (
    "Tests for escape sequences, not all of them are reliable represented as
     proper strings. So, we do a simple equality test, and check substring or
     length.

     \t’    F " 6† & 7FW
     \b’    & 6·7  6R 6† & 7FW
     \n’    æWvÆ–æR 6† & 7FW
     \r’    6 '&– vR &WGW&â 6† & 7FW
     \f’    f÷&ÖfVVB 6† & 7FW
     \'’    6–ævÆR  V÷FR 6† & 7FW
     \\’  & 6·6Æ 6, 6† & 7FW
     \0   zero byte character
    "

    self assert: '\t' equals: '\t'.
    self assert: 1 equals: '\t' length.

    self assert: '\b' equals: '\b'.
    self assert: 1 equals: '\b' length.

    self assert: '\n' equals: '\n'.
    self assert: 1 equals: '\n' length.
```

```
    self deny: ('\n' endsWith: 'n').

    self assert: '\r' equals: '\r'.
    self assert: 1 equals: '\n' length.
    self deny: ('\r' endsWith: 'r').

    self assert: '\f' equals: '\f'.
    self assert: 1 equals: '\f' length.
    self deny: ('\f' endsWith: 'f').

    self assert: '\'' equals: '\''.
    self assert: 1 equals: '\'' length.

    self assert: '\\' equals: '\\'.
    self assert: 1 equals: '\\' length.

    self assert: '\0' equals: '\0'.
    self assert: 1 equals: '\0' length.
    self assert: 5 equals: '\0rest' length.
  )

  testHash = (
    | str |
    "Hash should be identical for strings that are identical,
     whether given literal or composed at runtime"
    self assert: 'foobar' hashcode equals: 'foobar' hashcode.
    self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode
         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode.

    str := 'foo' + 'bar'.
    str := str + str.
    self assert: 'foobarfoobar' hashcode equals: str hashcode.

    str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw erltnwrtln'.
    self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw erltnwrtln' hashcode
         equals: str hashcode.
  )

  testWhiteSpace = (
    self assert: ' ' isWhiteSpace.
    self assert: '\t' isWhiteSpace.
    self assert: '\t\n   \n \n' isWhiteSpace.

    self deny: '' isWhiteSpace.
    self deny: '\t\n N  \n \n' isWhiteSpace.
    self deny: 'N' isWhiteSpace.
    self deny: '3' isWhiteSpace.
  )

  testLetters = (
    self assert: 'a' isLetters.
    self assert: 'all' isLetters.

    self deny: '' isLetters.
    self deny: ' ' isLetters.
    self deny: '3' isLetters.
    self deny: '3333' isLetters.
    self deny: 'aOo öéÉíä' isLetters.
    self deny: 'aOo1öéÉíä' isLetters.
```

```
  )

  testDigits = (
    self assert: '0' isDigits.
    self assert: '0123' isDigits.
    self assert: '0123456789' isDigits.

    self deny: '' isDigits.
    self deny: ' ' isDigits.
    self deny: 'S' isDigits.
    self deny: '333 3' isDigits.
    self deny: '66i77' isDigits.
    self deny: '66e7' isDigits.
    self deny: 'aOo1öéÉíä' isDigits.
  )

  testAsInteger = (
    self assert:   0 equals:   '0' asInteger.
    self assert: 100 equals: '100' asInteger.
    self assert: 923 equals: '923' asInteger.

    self assert:   -0 equals:   '-0' asInteger.
    self assert: -100 equals: '-100' asInteger.
    self assert: -923 equals: '-923' asInteger.

    self assert: 123342353453453456456456  equals: '123342353453453456456456' asInteger.
  )
)

```

Examples/Benchmarks/TestSuite/String4Test.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

String4Test = TestCase (

  testEquality = (
    | str1 str2 |
    str1 := 'foo'.
    str2 := 'bar'.

    self assert: str1 = str1.
    self assert: str1 = 'foo'.
    self assert: str1 = ('f' + 'oo').
    self deny:   str1 = str2.
    self assert: str2 = str2.

    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
    self assert: ('f' + 'o' + 'o') = #foo.
  )

  testEqualEqual = (
    | str1 |
    str1 := 'foo'.
    self assert: str1 == str1.
    self deny:   str1 == str1 asSymbol.
    self deny:   str1 == #foo.
  )

  testLength = (
    self assert: 1 equals: 't' length.
    self assert: 6 equals: ('foo' + 'bar') length.
  )
```

```
  testCharAt = (
    | str |
    str := 'foobar'.
    self assert: 'f' equals: (str charAt: 1).
    self assert: 'o' equals: (str charAt: 2).
    self assert: 'o' equals: (str charAt: 3).
    self assert: 'b' equals: (str charAt: 4).
    self assert: 'a' equals: (str charAt: 5).
    self assert: 'r' equals: (str charAt: 6).
  )

  testStringLiteralLineBreak = (
    | str |
    "Some parsers get the literals and line bounderies wrong"
    str := '
'.
    self assert: '\n' equals: (str charAt: 1).
    self assert: 1 equals: str length.
  )

  testPrimSubstringFrom = (
    | str |
    str := 'foobar'.
    self assert: 'foo'    equals: (str primSubstringFrom: 1 to: 3).
    self assert: 'bar'    equals: (str primSubstringFrom: 4 to: 6).
    self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
    self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
  )

  testSplit = (
    | r |
    r := 'aaaa' split: ','.
    self assert: 1 equals: r length.
    self assert: 'aaaa' equals: (r at: 1).

    r := 'foo.bar' split: '.'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo..bar' split: '.'.
    self assert: 3     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: ''    equals: (r at: 2).
    self assert: 'bar' equals: (r at: 3).

    r := 'foo..bar' split: '..'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo' split: 'bar'.
    self assert: 1     equals: r length.
    self assert: 'foo' equals: (r at: 1).

    self assert: Array is: r class
  )

  testIndexOf = (
    self assert: -1 equals: ('foo' indexOf: 'b').
```

```
    self assert:  1 equals: ('foo' indexOf: 'f').
    self assert:  2 equals: ('foo' indexOf: 'o').
    self assert:  3 equals: ('foo' indexOf: 'o' startingAt: 3).

    self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).

    self assert:  2 equals: ('foo' indexOf: 'oo').
  )

  testBeginsWith = (
    self deny:   ('foo' beginsWith: 'oo').
    self assert: ('foo' beginsWith: 'foo').
  )

  testEndsWith = (
    self assert: ('foo' endsWith: 'foo').
    self assert: ('foo' endsWith: 'oo').
    self deny: ('f' endsWith: 'bar').
    self deny: ('f' endsWith: 'foo').
  )

  testMultiLineString = (
    "Test whether the parser will parse multi-line strings correctly."
    self assert: '
1234567890
1234567890
1234567890
1234567890
1234567890' equals: '
1234567890
1234567890
1234567890
1234567890
1234567890'
  )

  testEscapeSequences = (
    "Tests for escape sequences, not all of them are reliable represented as
     proper strings. So, we do a simple equality test, and check substring or
     length.

     \t’    F " 6† & 7FW
     \b’    & 6·7  6R 6† & 7FW
     \n’    æWvÆ–æR 6† & 7FW
     \r’    6 '&– vR &WGW&â 6† & 7FW
     \f’    f÷&ÖfVVB 6† & 7FW
     \'’    6–ævÆR  V÷FR 6† & 7FW
     \\’  & 6·6Æ 6, 6† & 7FW
     \0   zero byte character
    "

    self assert: '\t' equals: '\t'.
    self assert: 1 equals: '\t' length.

    self assert: '\b' equals: '\b'.
    self assert: 1 equals: '\b' length.

    self assert: '\n' equals: '\n'.
    self assert: 1 equals: '\n' length.
```

```
    self deny: ('\n' endsWith: 'n').

    self assert: '\r' equals: '\r'.
    self assert: 1 equals: '\n' length.
    self deny: ('\r' endsWith: 'r').

    self assert: '\f' equals: '\f'.
    self assert: 1 equals: '\f' length.
    self deny: ('\f' endsWith: 'f').

    self assert: '\'' equals: '\''.
    self assert: 1 equals: '\'' length.

    self assert: '\\' equals: '\\'.
    self assert: 1 equals: '\\' length.

    self assert: '\0' equals: '\0'.
    self assert: 1 equals: '\0' length.
    self assert: 5 equals: '\0rest' length.
  )

  testHash = (
    | str |
    "Hash should be identical for strings that are identical,
     whether given literal or composed at runtime"
    self assert: 'foobar' hashcode equals: 'foobar' hashcode.
    self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode
         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode.

    str := 'foo' + 'bar'.
    str := str + str.
    self assert: 'foobarfoobar' hashcode equals: str hashcode.

    str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw erltnwrtln'.
    self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw erltnwrtln' hashcode
         equals: str hashcode.
  )

  testWhiteSpace = (
    self assert: ' ' isWhiteSpace.
    self assert: '\t' isWhiteSpace.
    self assert: '\t\n   \n \n' isWhiteSpace.

    self deny: '' isWhiteSpace.
    self deny: '\t\n N  \n \n' isWhiteSpace.
    self deny: 'N' isWhiteSpace.
    self deny: '3' isWhiteSpace.
  )

  testLetters = (
    self assert: 'a' isLetters.
    self assert: 'all' isLetters.

    self deny: '' isLetters.
    self deny: ' ' isLetters.
    self deny: '3' isLetters.
    self deny: '3333' isLetters.
    self deny: 'aOo öéÉíä' isLetters.
    self deny: 'aOo1öéÉíä' isLetters.
```

```
  )

  testDigits = (
    self assert: '0' isDigits.
    self assert: '0123' isDigits.
    self assert: '0123456789' isDigits.

    self deny: '' isDigits.
    self deny: ' ' isDigits.
    self deny: 'S' isDigits.
    self deny: '333 3' isDigits.
    self deny: '66i77' isDigits.
    self deny: '66e7' isDigits.
    self deny: 'aOo1öéÉíä' isDigits.
  )

  testAsInteger = (
    self assert:   0 equals:   '0' asInteger.
    self assert: 100 equals: '100' asInteger.
    self assert: 923 equals: '923' asInteger.

    self assert:   -0 equals:   '-0' asInteger.
    self assert: -100 equals: '-100' asInteger.
    self assert: -923 equals: '-923' asInteger.

    self assert: 1233423534534534564564 56  equals: '123342353453453456456456' asInteger.
  )
)
```

Examples/Benchmarks/TestSuite/String5Test.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

String5Test = TestCase (

  testEquality = (
    | str1 str2 |
    str1 := 'foo'.
    str2 := 'bar'.

    self assert: str1 = str1.
    self assert: str1 = 'foo'.
    self assert: str1 = ('f' + 'oo').
    self deny:   str1 = str2.
    self assert: str2 = str2.

    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
    self assert: ('f' + 'o' + 'o') = #foo.
  )

  testEqualEqual = (
    | str1 |
    str1 := 'foo'.
    self assert: str1 == str1.
    self deny:   str1 == str1 asSymbol.
    self deny:   str1 == #foo.
  )

  testLength = (
    self assert: 1 equals: 't' length.
    self assert: 6 equals: ('foo' + 'bar') length.
  )
```

```

  testCharAt = (
    | str |
    str := 'foobar'.
    self assert: 'f' equals: (str charAt: 1).
    self assert: 'o' equals: (str charAt: 2).
    self assert: 'o' equals: (str charAt: 3).
    self assert: 'b' equals: (str charAt: 4).
    self assert: 'a' equals: (str charAt: 5).
    self assert: 'r' equals: (str charAt: 6).
  )

  testStringLiteralLineBreak = (
    | str |
    "Some parsers get the literals and line bounderies wrong"
    str := '
'.
    self assert: '\n' equals: (str charAt: 1).
    self assert: 1 equals: str length.
  )

  testPrimSubstringFrom = (
    | str |
    str := 'foobar'.
    self assert: 'foo'    equals: (str primSubstringFrom: 1 to: 3).
    self assert: 'bar'    equals: (str primSubstringFrom: 4 to: 6).
    self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
    self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
  )

  testSplit = (
    | r |
    r := 'aaaa' split: ','.
    self assert: 1 equals: r length.
    self assert: 'aaaa' equals: (r at: 1).

    r := 'foo.bar' split: '.'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo..bar' split: '.'.
    self assert: 3     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: ''    equals: (r at: 2).
    self assert: 'bar' equals: (r at: 3).

    r := 'foo..bar' split: '..'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo' split: 'bar'.
    self assert: 1     equals: r length.
    self assert: 'foo' equals: (r at: 1).

    self assert: Array is: r class
  )

  testIndexOf = (
    self assert: -1 equals: ('foo' indexOf: 'b').
```

```
    self assert:  1 equals: ('foo' indexOf: 'f').
    self assert:  2 equals: ('foo' indexOf: 'o').
    self assert:  3 equals: ('foo' indexOf: 'o' startingAt: 3).

    self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).

    self assert:  2 equals: ('foo' indexOf: 'oo').
  )

  testBeginsWith = (
    self deny:   ('foo' beginsWith: 'oo').
    self assert: ('foo' beginsWith: 'foo').
  )

  testEndsWith = (
    self assert: ('foo' endsWith: 'foo').
    self assert: ('foo' endsWith: 'oo').
    self deny: ('f' endsWith: 'bar').
    self deny: ('f' endsWith: 'foo').
  )

  testMultiLineString = (
    "Test whether the parser will parse multi-line strings correctly."
    self assert: '
1234567890
1234567890
1234567890
1234567890
1234567890' equals: '
1234567890
1234567890
1234567890
1234567890
1234567890'
  )

  testEscapeSequences = (
    "Tests for escape sequences, not all of them are reliable represented as
     proper strings. So, we do a simple equality test, and check substring or
     length.

     \t’    F " 6† & 7FW
     \b’    & 6·7  6R 6† & 7FW
     \n’    æWvÆ–æR 6† & 7FW
     \r’    6 '&– vR &WGW&â 6† & 7FW
     \f’    f÷&ÖfVVB 6† & 7FW
     \'’    6–ævÆR  V÷FR 6† & 7FW
     \\’  & 6·6Æ 6, 6† & 7FW
     \0   zero byte character
    "

    self assert: '\t' equals: '\t'.
    self assert: 1 equals: '\t' length.

    self assert: '\b' equals: '\b'.
    self assert: 1 equals: '\b' length.

    self assert: '\n' equals: '\n'.
    self assert: 1 equals: '\n' length.
```

```
    self deny: ('\n' endsWith: 'n').

    self assert: '\r' equals: '\r'.
    self assert: 1 equals: '\n' length.
    self deny: ('\r' endsWith: 'r').

    self assert: '\f' equals: '\f'.
    self assert: 1 equals: '\f' length.
    self deny: ('\f' endsWith: 'f').

    self assert: '\'' equals: '\''.
    self assert: 1 equals: '\'' length.

    self assert: '\\' equals: '\\'.
    self assert: 1 equals: '\\' length.

    self assert: '\0' equals: '\0'.
    self assert: 1 equals: '\0' length.
    self assert: 5 equals: '\0rest' length.
  )

  testHash = (
    | str |
    "Hash should be identical for strings that are identical,
     whether given literal or composed at runtime"
    self assert: 'foobar' hashcode equals: 'foobar' hashcode.
    self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode
         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode.

    str := 'foo' + 'bar'.
    str := str + str.
    self assert: 'foobarfoobar' hashcode equals: str hashcode.

    str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw erltnwrtln'.
    self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw erltnwrtln' hashcode
         equals: str hashcode.
  )

  testWhiteSpace = (
    self assert: ' ' isWhiteSpace.
    self assert: '\t' isWhiteSpace.
    self assert: '\t\n   \n \n' isWhiteSpace.

    self deny: '' isWhiteSpace.
    self deny: '\t\n N  \n \n' isWhiteSpace.
    self deny: 'N' isWhiteSpace.
    self deny: '3' isWhiteSpace.
  )

  testLetters = (
    self assert: 'a' isLetters.
    self assert: 'all' isLetters.

    self deny: '' isLetters.
    self deny: ' ' isLetters.
    self deny: '3' isLetters.
    self deny: '3333' isLetters.
    self deny: 'aOo öéÉíä' isLetters.
    self deny: 'aOo1öéÉíä' isLetters.
```

```
  )

  testDigits = (
    self assert: '0' isDigits.
    self assert: '0123' isDigits.
    self assert: '0123456789' isDigits.

    self deny: '' isDigits.
    self deny: ' ' isDigits.
    self deny: 'S' isDigits.
    self deny: '333 3' isDigits.
    self deny: '66i77' isDigits.
    self deny: '66e7' isDigits.
    self deny: 'aOo1öéÉíä' isDigits.
  )

  testAsInteger = (
    self assert:   0 equals:   '0' asInteger.
    self assert: 100 equals: '100' asInteger.
    self assert: 923 equals: '923' asInteger.

    self assert:   -0 equals:   '-0' asInteger.
    self assert: -100 equals: '-100' asInteger.
    self assert: -923 equals: '-923' asInteger.

    self assert: 123342353453453456456456  equals: '123342353453453456456456' asInteger.
  )
)

```

Examples/Benchmarks/TestSuite/StringTest.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

StringTest = TestCase (

  testEquality = (
    | str1 str2 |
    str1 := 'foo'.
    str2 := 'bar'.

    self assert: str1 = str1.
    self assert: str1 = 'foo'.
    self assert: str1 = ('f' + 'oo').
    self deny:   str1 = str2.
    self assert: str2 = str2.

    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
    self assert: ('f' + 'o' + 'o') = #foo.
  )

  testEqualEqual = (
    | str1 |
    str1 := 'foo'.
    self assert: str1 == str1.
    self deny:   str1 == str1 asSymbol.
    self deny:   str1 == #foo.
  )

  testLength = (
    self assert: 1 equals: 't' length.
    self assert: 6 equals: ('foo' + 'bar') length.
  )
```

```

  testCharAt = (
    | str |
    str := 'foobar'.
    self assert: 'f' equals: (str charAt: 1).
    self assert: 'o' equals: (str charAt: 2).
    self assert: 'o' equals: (str charAt: 3).
    self assert: 'b' equals: (str charAt: 4).
    self assert: 'a' equals: (str charAt: 5).
    self assert: 'r' equals: (str charAt: 6).
  )

  testStringLiteralLineBreak = (
    | str |
    "Some parsers get the literals and line bounderies wrong"
    str := '
'.
    self assert: '\n' equals: (str charAt: 1).
    self assert: 1 equals: str length.
  )

  testPrimSubstringFrom = (
    | str |
    str := 'foobar'.
    self assert: 'foo'    equals: (str primSubstringFrom: 1 to: 3).
    self assert: 'bar'    equals: (str primSubstringFrom: 4 to: 6).
    self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
    self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
  )

  testSplit = (
    | r |
    r := 'aaaa' split: ','.
    self assert: 1 equals: r length.
    self assert: 'aaaa' equals: (r at: 1).

    r := 'foo.bar' split: '.'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo..bar' split: '.'.
    self assert: 3     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: ''    equals: (r at: 2).
    self assert: 'bar' equals: (r at: 3).

    r := 'foo..bar' split: '..'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo' split: 'bar'.
    self assert: 1     equals: r length.
    self assert: 'foo' equals: (r at: 1).

    self assert: Array is: r class
  )

  testIndexOf = (
    self assert: -1 equals: ('foo' indexOf: 'b').
```

```
    self assert:  1 equals: ('foo' indexOf: 'f').
    self assert:  2 equals: ('foo' indexOf: 'o').
    self assert:  3 equals: ('foo' indexOf: 'o' startingAt: 3).

    self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).

    self assert:  2 equals: ('foo' indexOf: 'oo').
  )

  testBeginsWith = (
    self deny:   ('foo' beginsWith: 'oo').
    self assert: ('foo' beginsWith: 'foo').
  )

  testEndsWith = (
    self assert: ('foo' endsWith: 'foo').
    self assert: ('foo' endsWith: 'oo').
    self deny: ('f' endsWith: 'bar').
    self deny: ('f' endsWith: 'foo').
  )

  testMultiLineString = (
    "Test whether the parser will parse multi-line strings correctly."
    self assert: '
1234567890
1234567890
1234567890
1234567890
1234567890' equals: '
1234567890
1234567890
1234567890
1234567890
1234567890'
  )

  testEscapeSequences = (
    "Tests for escape sequences, not all of them are reliable represented as
     proper strings. So, we do a simple equality test, and check substring or
     length.

     \t’    F " 6† & 7FW
     \b’    & 6·7  6R 6† & 7FW
     \n’    æWvÆ–æR 6† & 7FW
     \r’    6 '&– vR &WGW&â 6† & 7FW
     \f’    f÷&ÖfVVB 6† & 7FW
     \'’    6–ævÆR  V÷FR 6† & 7FW
     \\’  & 6·6Æ 6, 6† & 7FW
     \0   zero byte character
    "

    self assert: '\t' equals: '\t'.
    self assert: 1 equals: '\t' length.

    self assert: '\b' equals: '\b'.
    self assert: 1 equals: '\b' length.

    self assert: '\n' equals: '\n'.
    self assert: 1 equals: '\n' length.
```

```
    self deny: ('\n' endsWith: 'n').

    self assert: '\r' equals: '\r'.
    self assert: 1 equals: '\n' length.
    self deny: ('\r' endsWith: 'r').

    self assert: '\f' equals: '\f'.
    self assert: 1 equals: '\f' length.
    self deny: ('\f' endsWith: 'f').

    self assert: '\'' equals: '\''.
    self assert: 1 equals: '\'' length.

    self assert: '\\' equals: '\\'.
    self assert: 1 equals: '\\' length.

    self assert: '\0' equals: '\0'.
    self assert: 1 equals: '\0' length.
    self assert: 5 equals: '\0rest' length.
  )

  testHash = (
    | str |
    "Hash should be identical for strings that are identical,
     whether given literal or composed at runtime"
    self assert: 'foobar' hashcode equals: 'foobar' hashcode.
    self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode
         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode.

    str := 'foo' + 'bar'.
    str := str + str.
    self assert: 'foobarfoobar' hashcode equals: str hashcode.

    str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw erltnwrtln'.
    self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw erltnwrtln' hashcode
         equals: str hashcode.
  )

  testWhiteSpace = (
    self assert: ' ' isWhiteSpace.
    self assert: '\t' isWhiteSpace.
    self assert: '\t\n   \n \n' isWhiteSpace.

    self deny: '' isWhiteSpace.
    self deny: '\t\n N  \n \n' isWhiteSpace.
    self deny: 'N' isWhiteSpace.
    self deny: '3' isWhiteSpace.
  )

  testLetters = (
    self assert: 'a' isLetters.
    self assert: 'all' isLetters.

    self deny: '' isLetters.
    self deny: ' ' isLetters.
    self deny: '3' isLetters.
    self deny: '3333' isLetters.
    self deny: 'aOo öéÉíä' isLetters.
    self deny: 'aOo1öéÉíä' isLetters.
```

```
  )

  testDigits = (
    self assert: '0' isDigits.
    self assert: '0123' isDigits.
    self assert: '0123456789' isDigits.

    self deny: '' isDigits.
    self deny: ' ' isDigits.
    self deny: 'S' isDigits.
    self deny: '333 3' isDigits.
    self deny: '66i77' isDigits.
    self deny: '66e7' isDigits.
    self deny: 'aOo1öéÉíä' isDigits.
  )

  testAsInteger = (
    self assert:   0 equals:   '0' asInteger.
    self assert: 100 equals: '100' asInteger.
    self assert: 923 equals: '923' asInteger.

    self assert:   -0 equals:   '-0' asInteger.
    self assert: -100 equals: '-100' asInteger.
    self assert: -923 equals: '-923' asInteger.

    self assert: 123342353453453456456456  equals:
'123342353453453456456456' asInteger.
  )
)
```

Examples/Benchmarks/TestSuite/Super2Test.som

```
"

Copyright (c) 2007-2018 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Super2Test = SuperTestSuperClass (

  testSuper = (
    self assert: 42 equals: self give42.
    self assert: 42 equals: self blockGive42.
  )

  yourself = (
    record := record + 1000.
    ^ self
  )

  give42 = (
    ^super give42
  )

  blockGive42 = (
    ^[ super give42 ] value
  )

  something = (
    ^ #sub
  )

  number = (
    ^ 10
  )

```

```
+ other = (
  ^ 11
)

++++ other = (
  ^ 111
)

keyword: other = (
  ^ 1111
)

testBasicUnary = (
  self assert: 10 equals: self number.
  self assert:  1 equals: super number.
)

testBasicBinary = (
  self assert: 11 equals: self + 3.
  self assert: 22 equals: super + 5.
)


testBasicBinaryNonStandardOperator = (
  self assert: 111 equals: self ++++ 3.
  self assert: 222 equals: super ++++ 5.
)

testBasicKeyword = (
  self assert: 1111 equals: (self keyword: 3).
  self assert: 2222 equals: (super keyword: 5).
)

testWithBinaryUnaryMessage = (
  | val |
  record := 0.
  val := super number * super number.
  self assert: 1 equals: val.
)

testWithBinaryUnaryUnaryMessage = (
  | val |
  record := 0.
  super yourself yourself @ super yourself yourself.
  self assert: 2002 equals: record.
)

testWithKeywordUnaryUnaryMessage = (
  | val |
  record := 0.
  super key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.

  record := 0.
  self key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.
)

"Note: testing assigning self was moved to basic interpreter tests"

testGlobalSelfDoesNotShadowKeyword = (
```

```
    | that |
    that := self.
    system global: #self put: 42.
    that optional: #selfSuperBug assert: that is: self.

    self assert: 42 equals: (system global: #self)
  )

  testGlobalSuperDoesNotShadowKeyword = (
    | that |
    that := super.
    system global: #super put: 42.
    that optional: #selfSuperBug assert: that is: super.

    self assert: 42 equals: (system global: #super)
  )
)
```

Examples/Benchmarks/TestSuite/Super3Test.som

```
"

Copyright (c) 2007-2018 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Super3Test = SuperTestSuperClass (

  testSuper = (
    self assert: 42 equals: self give42.
    self assert: 42 equals: self blockGive42.
  )

  yourself = (
    record := record + 1000.
    ^ self
  )

  give42 = (
    ^super give42
  )

  blockGive42 = (
    ^[ super give42 ] value
  )

  something = (
    ^ #sub
  )

  number = (
    ^ 10
  )

```

```
+ other = (
  ^ 11
)

++++ other = (
  ^ 111
)

keyword: other = (
  ^ 1111
)

testBasicUnary = (
  self assert: 10 equals: self number.
  self assert:  1 equals: super number.
)

testBasicBinary = (
  self assert: 11 equals: self + 3.
  self assert: 22 equals: super + 5.
)


testBasicBinaryNonStandardOperator = (
  self assert: 111 equals: self ++++ 3.
  self assert: 222 equals: super ++++ 5.
)

testBasicKeyword = (
  self assert: 1111 equals: (self keyword: 3).
  self assert: 2222 equals: (super keyword: 5).
)

testWithBinaryUnaryMessage = (
  | val |
  record := 0.
  val := super number * super number.
  self assert: 1 equals: val.
)

testWithBinaryUnaryUnaryMessage = (
  | val |
  record := 0.
  super yourself yourself @ super yourself yourself.
  self assert: 2002 equals: record.
)

testWithKeywordUnaryUnaryMessage = (
  | val |
  record := 0.
  super key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.

  record := 0.
  self key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.
)

"Note: testing assigning self was moved to basic interpreter tests"

testGlobalSelfDoesNotShadowKeyword = (
```

```
    | that |
    that := self.
    system global: #self put: 42.
    that optional: #selfSuperBug assert: that is: self.

    self assert: 42 equals: (system global: #self)
  )

  testGlobalSuperDoesNotShadowKeyword = (
    | that |
    that := super.
    system global: #super put: 42.
    that optional: #selfSuperBug assert: that is: super.

    self assert: 42 equals: (system global: #super)
  )
)
```

```
"

Copyright (c) 2007-2018 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Super4Test = SuperTestSuperClass (

  testSuper = (
    self assert: 42 equals: self give42.
    self assert: 42 equals: self blockGive42.
  )

  yourself = (
    record := record + 1000.
    ^ self
  )

  give42 = (
    ^super give42
  )

  blockGive42 = (
    ^[ super give42 ] value
  )

  something = (
    ^ #sub
  )

  number = (
    ^ 10
  )

```

```
  + other = (
    ^ 11
  )

  ++++ other = (
    ^ 111
  )

  keyword: other = (
    ^ 1111
  )

  testBasicUnary = (
    self assert: 10 equals: self number.
    self assert:  1 equals: super number.
  )

  testBasicBinary = (
    self assert: 11 equals: self + 3.
    self assert: 22 equals: super + 5.
  )


  testBasicBinaryNonStandardOperator = (
    self assert: 111 equals: self ++++ 3.
    self assert: 222 equals: super ++++ 5.
  )

  testBasicKeyword = (
    self assert: 1111 equals: (self keyword: 3).
    self assert: 2222 equals: (super keyword: 5).
  )

  testWithBinaryUnaryMessage = (
    | val |
    record := 0.
    val := super number * super number.
    self assert: 1 equals: val.
  )

  testWithBinaryUnaryUnaryMessage = (
    | val |
    record := 0.
    super yourself yourself @ super yourself yourself.
    self assert: 2002 equals: record.
  )

  testWithKeywordUnaryUnaryMessage = (
    | val |
    record := 0.
    super key: super yourself yourself key: super yourself yourself.
    self assert: 2002 equals: record.

    record := 0.
    self key: super yourself yourself key: super yourself yourself.
    self assert: 2002 equals: record.
  )

  "Note: testing assigning self was moved to basic interpreter tests"

  testGlobalSelfDoesNotShadowKeyword = (
```

```
    | that |
    that := self.
    system global: #self put: 42.
    that optional: #selfSuperBug assert: that is: self.

    self assert: 42 equals: (system global: #self)
  )

  testGlobalSuperDoesNotShadowKeyword = (
    | that |
    that := super.
    system global: #super put: 42.
    that optional: #selfSuperBug assert: that is: super.

    self assert: 42 equals: (system global: #super)
  )
)
```

```
"

Copyright (c) 2007-2018 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Super5Test = SuperTestSuperClass (

  testSuper = (
    self assert: 42 equals: self give42.
    self assert: 42 equals: self blockGive42.
  )

  yourself = (
    record := record + 1000.
    ^ self
  )

  give42 = (
    ^super give42
  )

  blockGive42 = (
    ^[ super give42 ] value
  )

  something = (
    ^ #sub
  )

  number = (
    ^ 10
  )

```

```
+ other = (
  ^ 11
)

++++ other = (
  ^ 111
)

keyword: other = (
  ^ 1111
)

testBasicUnary = (
  self assert: 10 equals: self number.
  self assert:  1 equals: super number.
)

testBasicBinary = (
  self assert: 11 equals: self + 3.
  self assert: 22 equals: super + 5.
)


testBasicBinaryNonStandardOperator = (
  self assert: 111 equals: self ++++ 3.
  self assert: 222 equals: super ++++ 5.
)

testBasicKeyword = (
  self assert: 1111 equals: (self keyword: 3).
  self assert: 2222 equals: (super keyword: 5).
)

testWithBinaryUnaryMessage = (
  | val |
  record := 0.
  val := super number * super number.
  self assert: 1 equals: val.
)

testWithBinaryUnaryUnaryMessage = (
  | val |
  record := 0.
  super yourself yourself @ super yourself yourself.
  self assert: 2002 equals: record.
)

testWithKeywordUnaryUnaryMessage = (
  | val |
  record := 0.
  super key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.

  record := 0.
  self key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.
)

"Note: testing assigning self was moved to basic interpreter tests"

testGlobalSelfDoesNotShadowKeyword = (
```

```
    | that |
    that := self.
    system global: #self put: 42.
    that optional: #selfSuperBug assert: that is: self.

    self assert: 42 equals: (system global: #self)
  )

  testGlobalSuperDoesNotShadowKeyword = (
    | that |
    that := super.
    system global: #super put: 42.
    that optional: #selfSuperBug assert: that is: super.

    self assert: 42 equals: (system global: #super)
  )
)

```

```
"

Copyright (c) 2007-2018 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SuperTest = SuperTestSuperClass (

  testSuper = (
    self assert: 42 equals: self give42.
    self assert: 42 equals: self blockGive42.
  )

  yourself = (
    record := record + 1000.
    ^ self
  )

  give42 = (
    ^super give42
  )

  blockGive42 = (
    ^[ super give42 ] value
  )

  something = (
    ^ #sub
  )

  number = (
    ^ 10
  )

```

```
+ other = (
  ^ 11
)

++++ other = (
  ^ 111
)

keyword: other = (
  ^ 1111
)

testBasicUnary = (
  self assert: 10 equals: self number.
  self assert:  1 equals: super number.
)

testBasicBinary = (
  self assert: 11 equals: self + 3.
  self assert: 22 equals: super + 5.
)


testBasicBinaryNonStandardOperator = (
  self assert: 111 equals: self ++++ 3.
  self assert: 222 equals: super ++++ 5.
)

testBasicKeyword = (
  self assert: 1111 equals: (self keyword: 3).
  self assert: 2222 equals: (super keyword: 5).
)

testWithBinaryUnaryMessage = (
  | val |
  record := 0.
  val := super number * super number.
  self assert: 1 equals: val.
)

testWithBinaryUnaryUnaryMessage = (
  | val |
  record := 0.
  super yourself yourself @ super yourself yourself.
  self assert: 2002 equals: record.
)

testWithKeywordUnaryUnaryMessage = (
  | val |
  record := 0.
  super key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.

  record := 0.
  self key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.
)

"Note: testing assigning self was moved to basic interpreter tests"

testGlobalSelfDoesNotShadowKeyword = (
```

```
    | that |
    that := self.
    system global: #self put: 42.
    that optional: #selfSuperBug assert: that is: self.

    self assert: 42 equals: (system global: #self)
  )

  testGlobalSuperDoesNotShadowKeyword = (
    | that |
    that := super.
    system global: #super put: 42.
    that optional: #selfSuperBug assert: that is: super.

    self assert: 42 equals: (system global: #super)
  )
)
```

```
"

$Id: SuperTestSuperClass.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SuperTestSuperClass = TestCase (
    | record |

    yourself = (
        record := record + 1.
        ^ self
    )

    give42 = (
        ^ 42
    )

    something = (
        ^ #super
    )

    number = (
        ^ 1
    )

    + other = (
      ^ 22
    )

    ++++ other = (
      ^ 222
    )
```

```

    keyword: other = (
      ^ 2222
    )

    key: a key: b = (
        ^ self
    )

    @ o = ( ^ self )
)


```

```
"

$Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Symbol2Test = TestCase (

  testConversion = (
    self assert: 'gunk' equals: 'gunk' asSymbol asString.
    self assert: 'oink' equals: #oink asString.
  )

  testEquality = (
    self assert: #oink == #oink.
    self assert: #oink == 'oink' asSymbol.
    self assert: #oink =  #oink.
    self assert: #oink =  'oink' asSymbol.

    self deny: #foo =  #fooo.
    self deny: #foo == #fooo.

    self assert: #foo =  'foo'.
    self deny: #foo == 'fooo'.
    self deny: #foo == #foo asString.
  )

  testSymbolIsString = (
    self assert: (#oink beginsWith: 'oink').
    self assert: 100 equals: #'100' asInteger.
    self assert: String equals: #foo class superclass
  )
)
```

Examples/Benchmarks/TestSuite/Symbol3Test.som

```
"

$Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Symbol3Test = TestCase (

  testConversion = (
    self assert: 'gunk' equals: 'gunk' asSymbol asString.
    self assert: 'oink' equals: #oink asString.
  )

  testEquality = (
    self assert: #oink == #oink.
    self assert: #oink == 'oink' asSymbol.
    self assert: #oink =  #oink.
    self assert: #oink =  'oink' asSymbol.

    self deny: #foo =  #fooo.
    self deny: #foo == #fooo.

    self assert: #foo =  'foo'.
    self deny: #foo == 'fooo'.
    self deny: #foo == #foo asString.
  )

  testSymbolIsString = (
    self assert: (#oink beginsWith: 'oink').
    self assert: 100 equals: #'100' asInteger.
    self assert: String equals: #foo class superclass
  )
)
```

Examples/Benchmarks/TestSuite/Symbol4Test.som

```
"

$Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Symbol4Test = TestCase (

  testConversion = (
    self assert: 'gunk' equals: 'gunk' asSymbol asString.
    self assert: 'oink' equals: #oink asString.
  )

  testEquality = (
    self assert: #oink == #oink.
    self assert: #oink == 'oink' asSymbol.
    self assert: #oink =  #oink.
    self assert: #oink =  'oink' asSymbol.

    self deny: #foo =  #fooo.
    self deny: #foo == #fooo.

    self assert: #foo =  'foo'.
    self deny: #foo == 'fooo'.
    self deny: #foo == #foo asString.
  )

  testSymbolIsString = (
    self assert: (#oink beginsWith: 'oink').
    self assert: 100 equals: #'100' asInteger.
    self assert: String equals: #foo class superclass
  )
)
```

```
"

$Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Symbol5Test = TestCase (

  testConversion = (
    self assert: 'gunk' equals: 'gunk' asSymbol asString.
    self assert: 'oink' equals: #oink asString.
  )

  testEquality = (
    self assert: #oink == #oink.
    self assert: #oink == 'oink' asSymbol.
    self assert: #oink =  #oink.
    self assert: #oink =  'oink' asSymbol.

    self deny: #foo =  #fooo.
    self deny: #foo == #fooo.

    self assert: #foo =  'foo'.
    self deny: #foo == 'fooo'.
    self deny: #foo == #foo asString.
  )

  testSymbolIsString = (
    self assert: (#oink beginsWith: 'oink').
    self assert: 100 equals: #'100' asInteger.
    self assert: String equals: #foo class superclass
  )
)
```

Examples/Benchmarks/TestSuite/SymbolTest.som

```
"

$Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SymbolTest = TestCase (

  testConversion = (
    self assert: 'gunk' equals: 'gunk' asSymbol asString.
    self assert: 'oink' equals: #oink asString.
  )

  testEquality = (
    self assert: #oink == #oink.
    self assert: #oink == 'oink' asSymbol.
    self assert: #oink =  #oink.
    self assert: #oink =  'oink' asSymbol.

    self deny: #foo =  #fooo.
    self deny: #foo == #fooo.

    self assert: #foo =  'foo'.
    self deny: #foo == 'fooo'.
    self deny: #foo == #foo asString.
  )

  testSymbolIsString = (
    self assert: (#oink beginsWith: 'oink').
    self assert: 100 equals: #'100' asInteger.
    self assert: String equals: #foo class superclass
  )
)
```

Examples/Benchmarks/TestSuite/System2Test.som

```
System2Test = TestCase (

  testFullGCSupport = (
    "Test whether #fullGC is support. We expect the VM now to return true,
     to indicate the a GC was done."
    self optional: #fullGCWithEffect assert: system fullGC description: '#fullGC is not supported or has not immediate effect.'
  )

  testTicks = (
    | ticks |
    ticks := system ticks.
    self assert: ticks class equals: Integer.
    self assert: ticks > 0 description: 'Should return the microseconds since the start'
  )
)
```

Examples/Benchmarks/TestSuite/System3Test.som

```
System3Test = TestCase (

  testFullGCSupport = (
    "Test whether #fullGC is support. We expect the VM now to return true,
     to indicate the a GC was done."
    self optional: #fullGCWithEffect assert: system fullGC description: '#fullGC is not supported or has not immediate effect.'
  )

  testTicks = (
    | ticks |
    ticks := system ticks.
    self assert: ticks class equals: Integer.
    self assert: ticks > 0 description: 'Should return the microseconds since the start'
  )
)

```

```
System4Test = TestCase (

  testFullGCSupport = (
    "Test whether #fullGC is support. We expect the VM now to return true,
     to indicate the a GC was done."
    self optional: #fullGCWithEffect assert: system fullGC description: '#fullGC is not supported or has not immediate effect.'
  )

  testTicks = (
    | ticks |
    ticks := system ticks.
    self assert: ticks class equals: Integer.
    self assert: ticks > 0 description: 'Should return the microseconds since the start'
  )
)

```

Examples/Benchmarks/TestSuite/System5Test.som

```
System5Test = TestCase (

  testFullGCSupport = (
    "Test whether #fullGC is support. We expect the VM now to return true,
     to indicate the a GC was done."
    self optional: #fullGCWithEffect assert: system fullGC description: '#fullGC is not supported or has not immediate effect.'
  )

  testTicks = (
    | ticks |
    ticks := system ticks.
    self assert: ticks class equals: Integer.
    self assert: ticks > 0 description: 'Should return the microseconds since the start'
  )
)
```

Examples/Benchmarks/TestSuite/SystemTest.som

```
SystemTest = TestCase (

  testFullGCSupport = (
    "Test whether #fullGC is support. We expect the VM now to return true,
     to indicate the a GC was done."
    self optional: #fullGCWithEffect assert: system fullGC description: '#fullGC is not supported or has not immediate effect.'
  )

  testTicks = (
    | ticks |
    ticks := system ticks.
    self assert: ticks class equals: Integer.
    self assert: ticks > 0 description: 'Should return the microseconds since the start'
  )
)

```

Examples/Benchmarks/TestSuite/Test.som

```
"

$Id: TestHarness.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Test = TestCommon (
    tests = ( "Now ordered by alphabetical order to improve maintainability"
        ^ EmptyTest,
          SpecialSelectorsTest,
          SpecialSelectors2Test,
          SpecialSelectors3Test,
          SpecialSelectors4Test,
          SpecialSelectors5Test,
          ArrayTest,
          Array2Test,
          Array3Test,
          Array4Test,
          Array5Test,
          BlockTest,
          Block2Test,
          Block3Test,
          Block4Test,
          Block5Test,
          BooleanTest,
          Boolean2Test,
          Boolean3Test,
          Boolean4Test,
          Boolean5Test,
          ClassLoadingTest,
          ClassLoading2Test,
          ClassLoading3Test,
          ClassLoading4Test,
```

```
            ClassLoading5Test,
            ClassStructureTest,
            ClassStructure2Test,
            ClassStructure3Test,
            ClassStructure4Test,
            ClassStructure5Test,
            ClosureTest,
            Closure2Test,
            Closure3Test,
            Closure4Test,
            Closure5Test,
            CoercionTest,
            Coercion2Test,
            Coercion3Test,
            Coercion4Test,
            Coercion5Test,
            CompilerReturnTest,
            CompilerReturn2Test,
            CompilerReturn3Test,
            CompilerReturn4Test,
            CompilerReturn5Test,
            DictionaryTest,
            Dictionary2Test,
            Dictionary3Test,
            Dictionary4Test,
            Dictionary5Test,
            DoesNotUnderstandTest,
            DoesNotUnderstand2Test,
            DoesNotUnderstand3Test,
            DoesNotUnderstand4Test,
            DoesNotUnderstand5Test,
            DoubleTest,
            Double2Test,
            Double3Test,
            Double4Test,
            Double5Test,
            GlobalTest,
            Global2Test,
            Global3Test,
            Global4Test,
            Global5Test,
            HashTest,
            Hash2Test,
            Hash3Test,
            Hash4Test,
            Hash5Test,
            IntegerTest,
            Integer2Test,
            Integer3Test,
            Integer4Test,
            Integer5Test,
            PreliminaryTest,
            Preliminary2Test,
            Preliminary3Test,
            Preliminary4Test,
            Preliminary5Test,
            ReflectionTest,
            Reflection2Test,
            Reflection3Test,
            Reflection4Test,
            Reflection5Test,
```

```
            SelfBlockTest,
            SelfBlock2Test,
            SelfBlock3Test,
            SelfBlock4Test,
            SelfBlock5Test,
            SetTest,
            Set2Test,
            Set3Test,
            Set4Test,
            Set5Test,
            StringTest,
            String2Test,
            String3Test,
            String4Test,
            String5Test,
            SuperTest,
            Super2Test,
            Super3Test,
            Super4Test,
            Super5Test,
            SymbolTest,
            Symbol2Test,
            Symbol3Test,
            Symbol4Test,
            Symbol5Test,
            SystemTest,
            System2Test,
            System3Test,
            System4Test,
            System5Test,
            VectorTest,
            Vector2Test,
            Vector3Test,
            Vector4Test,
            Vector5Test
    )

    numExecs = (
    ^ 5
    )
)
```

Examples/Benchmarks/TestSuite/TestCase.som

```
TestCase = (
    | testSelector runner failed |

    selector       = ( ^ testSelector )
    selector: aSym = ( testSelector := aSym )

    "asserting"
    assert: aBoolean = (
        runner countAssert.
        aBoolean ifFalse: [
            self signalFailure: 'Assertion failed' ] )

    assert: aBoolean description: aStringOrBlock = (
        runner countAssert.
        aBoolean ifFalse: [
            self signalFailure: aStringOrBlock value ] )

    assert: expected equals: actual = (
        "test value equality"
        self assert: (expected = actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    assert: expected is: actual = (
        "test reference equality"
        self assert: (expected == actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    optional: aSymbol assert: aBoolean = (
        runner countAssert.
        aBoolean ifFalse: [
            self signalUnsupported: aSymbol description: nil ] )

    optional: aSymbol assert: expected equals: actual = (
        self optional: aSymbol
             assert: (expected = actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    optional: aSymbol assert: expected is: actual = (
        self optional: aSymbol
             assert: (expected == actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    optional: aSymbol assert: aBoolean description: aStringOrBlock = (
        runner countAssert.
        aBoolean ifFalse: [
            self signalUnsupported: aSymbol description: aStringOrBlock value ] )

    deny: aBoolean = (
        self assert: aBoolean not
```

```
    )

    deny: aBooleanOrBlock description: aString = (
        self assert: aBooleanOrBlock value not description: aString
    )

    optional: aSymbol deny: aBoolean = (
        self optional: aSymbol assert: aBoolean not
    )

    optional: aSymbol deny: aBooleanOrBlock description: aString = (
        self optional: aSymbol assert: aBooleanOrBlock value not description: aString
    )

    signalFailure: aString = (
        failed := true.
        runner fail: self class name + '>>#' + testSelector
            because: aString.
    )

    signalUnsupported: aSymbol description: aDescription = (
        runner unsupported: aSymbol
                      test: self class name + '>>#' + testSelector
                   because: aDescription.
    )

    comparingStringBetween: expected and: actual = (
        ^ 'Expected ' + expected asString +
          ' but was ' + actual asString + '.'
    )

    "running"
    run: aRunner = (
        runner := aRunner.
        failed := false.

        self setUp.
        self performTest.
        self tearDown.

        failed ifFalse: [
            runner passed: self class name + '>>#' + testSelector
        ].
    )

    setUp    = ()
    tearDown = ()

    performTest = ( self perform: testSelector )

    ----

    for: aSelector = (
        | case |
        case := self new.
        case selector: aSelector.
        ^ case
    )

    tests = (
```

```
        | tests |
        tests := Vector new: self methods length.
        self methods do: [:m |
            (m signature beginsWith: #test) ifTrue: [
                tests append: (self for: m signature).
            ].
        ].

        ^ tests
    )
)
```

```
"

$Id: TestHarness.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

TestCommon = Benchmark (
    | failOnUnsupportedOptionals |

    tests = ( "Now ordered by alphabetical order to improve maintainability"
        self error: 'Implement in subclass'
    )

    numExecs = (
      self error: 'Implement in subclass and return the number of times a test is executed'
    )

    oneTimeSetup = (
      "Load all Tests. We don't really want to benchmark the parser."
      self tests
    )

    runAllSuites = (
      | totalTestNum successfulTestNum unsupportedTestNum totalAssertionNum arr |
      totalTestNum := 0.
      unsupportedTestNum := 0.
      successfulTestNum := 0.
      totalAssertionNum := 0.

      self tests do: [ :test |
        | runner |
        runner := TestRunner new.
```

```
        runner initializeOn: test.
        runner runAllTests.

        totalTestNum       := totalTestNum + runner expectedPasses.
        unsupportedTestNum := unsupportedTestNum + runner actualUnsupported.
        successfulTestNum  := successfulTestNum + runner actualPasses.
        totalAssertionNum  := totalAssertionNum + runner numAsserts.
      ].

      arr := Array new: 4.
      arr at: 1 put: totalTestNum.
      arr at: 2 put: unsupportedTestNum.
      arr at: 3 put: successfulTestNum.
      arr at: 4 put: totalAssertionNum.
      ^ arr
    )

    runOneSuite: name = (
      | testName runner |
      testName := name.
      (testName endsWith: 'Test') ifFalse: [
        testName := testName + 'Test'].

      runner := TestRunner new.
      runner initializeOn: (system resolve: testName asSymbol).
      runner run.
      runner hasFailures ifTrue: [system exit: 1]
    )

    benchmark = (
        failOnUnsupportedOptionals := false.
        ^ self runAllSuites
    )

    verifyResult: result = (
      "result do: [:e | e println ]."
      ^ (result at: 1) = (190 * self numExecs) and: [
          (result at: 2) = (0 * self numExecs) and: [
            (result at: 3) = (189 * self numExecs) and: [
              (result at: 4) = (992 * self numExecs)
        ] ] ]
    )
)
```

Examples/Benchmarks/TestSuite/TestGC.som

```
"
Copyright (c) 2021 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

TestGC = TestGCCommon (
    tests = ( "Now ordered by alphabetical order to improve maintainability"
        ^ EmptyTest,
          SpecialSelectorsTest,
          SpecialSelectors2Test,
          SpecialSelectors3Test,
          SpecialSelectors4Test,
          SpecialSelectors5Test,
          ArrayTest,
          Array2Test,
          Array3Test,
          Array4Test,
          Array5Test,
          BlockTest,
          Block2Test,
          Block3Test,
          Block4Test,
          Block5Test,
          BooleanTest,
          Boolean2Test,
          Boolean3Test,
          Boolean4Test,
          Boolean5Test,
          ClassLoadingTest,
          ClassLoading2Test,
          ClassLoading3Test,
          ClassLoading4Test,
          ClassLoading5Test,
          ClassStructureTest,
          ClassStructure2Test,
```

```
            ClassStructure3Test,
            ClassStructure4Test,
            ClassStructure5Test,
            ClosureTest,
            Closure2Test,
            Closure3Test,
            Closure4Test,
            Closure5Test,
            CoercionTest,
            Coercion2Test,
            Coercion3Test,
            Coercion4Test,
            Coercion5Test,
            CompilerReturnTest,
            CompilerReturn2Test,
            CompilerReturn3Test,
            CompilerReturn4Test,
            CompilerReturn5Test,
            DictionaryTest,
            Dictionary2Test,
            Dictionary3Test,
            Dictionary4Test,
            Dictionary5Test,
            DoesNotUnderstandTest,
            DoesNotUnderstand2Test,
            DoesNotUnderstand3Test,
            DoesNotUnderstand4Test,
            DoesNotUnderstand5Test,
            DoubleTest,
            Double2Test,
            Double3Test,
            Double4Test,
            Double5Test,
            GlobalTest,
            Global2Test,
            Global3Test,
            Global4Test,
            Global5Test,
            HashTest,
            Hash2Test,
            Hash3Test,
            Hash4Test,
            Hash5Test,
            IntegerTest,
            Integer2Test,
            Integer3Test,
            Integer4Test,
            Integer5Test,
            PreliminaryTest,
            Preliminary2Test,
            Preliminary3Test,
            Preliminary4Test,
            Preliminary5Test,
            ReflectionTest,
            Reflection2Test,
            Reflection3Test,
            Reflection4Test,
            Reflection5Test,
            SelfBlockTest,
            SelfBlock2Test,
            SelfBlock3Test,
```

```
          SelfBlock4Test,
          SelfBlock5Test,
          SetTest,
          Set2Test,
          Set3Test,
          Set4Test,
          Set5Test,
          StringTest,
          String2Test,
          String3Test,
          String4Test,
          String5Test,
          SuperTest,
          Super2Test,
          Super3Test,
          Super4Test,
          Super5Test,
          SymbolTest,
          Symbol2Test,
          Symbol3Test,
          Symbol4Test,
          Symbol5Test,
          SystemTest,
          System2Test,
          System3Test,
          System4Test,
          System5Test,
          VectorTest,
          Vector2Test,
          Vector3Test,
          Vector4Test,
          Vector5Test
    )

    benchmark = (
      ^ system fullGC
    )
)
```

Examples/Benchmarks/TestSuite/TestGCCommon.som

```
"
Copyright (c) 2021 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

TestGCCommon = Benchmark (
    | failOnUnsupportedOptionals |

    tests = ( "Now ordered by alphabetical order to improve maintainability"
        self error: 'Implement in subclass'
    )

    oneTimeSetup = (
      "Load all Tests. We don't really want to benchmark the parser."
      self tests
    )

    runAllSuites = (
      | totalTestNum successfulTestNum unsupportedTestNum totalAssertionNum arr |
      totalTestNum := 0.
      unsupportedTestNum := 0.
      successfulTestNum := 0.
      totalAssertionNum := 0.

      self tests do: [ :test |
        | runner |
        runner := TestRunner new.
        runner initializeOn: test.
        runner runAllTests.

        totalTestNum       := totalTestNum + runner expectedPasses.
        unsupportedTestNum := unsupportedTestNum + runner actualUnsupported.
        successfulTestNum  := successfulTestNum + runner actualPasses.
        totalAssertionNum  := totalAssertionNum + runner numAsserts.
```

```
      ].

      arr := Array new: 4.
      arr at: 1 put: totalTestNum.
      arr at: 2 put: unsupportedTestNum.
      arr at: 3 put: successfulTestNum.
      arr at: 4 put: totalAssertionNum.
      ^ arr
    )

    runOneSuite: name = (
      | testName runner |
      testName := name.
      (testName endsWith: 'Test') ifFalse: [
        testName := testName + 'Test'].

      runner := TestRunner new.
      runner initializeOn: (system resolve: testName asSymbol).
      runner run.
      runner hasFailures ifTrue: [system exit: 1]
    )

    benchmark = (
      ^ system fullGC
    )

    verifyResult: result = (
      "We expect the GC to actually promise us that it did work"
      ^ result
    )
)
```

```
TestRunner = (
  | suite passes unsupported failures numAsserts |

  initializeOn: aSuite = (
    suite := aSuite.

    passes       := Vector new.
    unsupported  := Vector new.
    failures     := Vector new.

    numAsserts := 0.
  )

  hasUnsupported  = ( ^ unsupported size > 0 )
  hasFailures     = ( ^ failures size > 0 )

  actualUnsupported = ( ^ unsupported size )
  expectedPasses = ( ^ suite tests size )
  actualPasses   = ( ^ passes size )

  run = (
    self reportPreRun.
    self runAllTests.
    self reportPostRun.
    self overviewReport.
  )

  countAssert = (
    numAsserts := numAsserts + 1.
  )

  numAsserts = (
    ^ numAsserts
  )

  reportPreRun = (
    ('TestSuite ' + suite name + ':') println.
    ('Tests: ' + suite tests size asString) println.
  )

  reportPostRun = (
    self hasUnsupported ifTrue: [
      ('Unsupported optional: ' + unsupported size asString) println
    ].
    self hasFailures ifTrue: [
      ('Failures: ' + failures size asString) println
    ].
  )

  runAllTests = (
    suite tests do: [ :each |
        each run: self ].
  )

  overviewReport = (
    ('Tests passed: ' + passes size asString) println.

    (self hasFailures or: [self hasUnsupported]) ifTrue: [
```

```
        '------------------------------' println ].

    self hasUnsupported ifTrue: [
      | lastCategory |
      ('Unsupported optional features: ' + unsupported size asString) println.
      unsupported do: [:each |
        | cat |
        cat := each at: 1.
        cat == lastCategory ifFalse: [
          lastCategory := cat.
          ('\t' + cat) println ].
        ('\t\t' + (each at: 2) asString) println.
        ('\t\t\t' + (each at: 3) value asString) println ].
    ].

    self hasFailures ifTrue: [
      ('Failures: ' + failures size asString) println.
      failures do: [:each |
        ('    ' + each key asString) println.
        ('        ' + each value asString) println ].
    ].
  )

  fail: aSignature because: aReason = (
    | pair |
    pair := Pair withKey: aSignature andValue: aReason.
    failures append: pair.
  )

  unsupported: aSymbol test: aSignature because: aReason = (
    | array |
    array := Array with: aSymbol with: aSignature with: aReason.
    unsupported append: array.
  )

  passed: aSignature = (
    passes append: aSignature
  )
)
```

Examples/Benchmarks/TestSuite/Vector2Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Vector2Test = TestCase (
  | a |

  setUp = (
    a := Vector new.
    a append: 'hello'.
    a append: #world.
    a append: 23.
  )

  testSize = (
    self assert: 3 equals: a size.
  )

  testAt = (
    self assert: #world equals: (a at: 2).
    self assert:     23 equals: (a at: 3).
  )

  testFirst = (
    | v |
    self assert: 'hello' equals: a first.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
```

```
      self assert: 1 equals: v first ].

    1 to: 10 do: [:i |
      self assert: 1 equals: v first.
      v removeFirst ]
  )

  testLast = (
    | v |
    self assert: 23 equals: a last.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
      self assert: i equals: v last ].

    10 downTo: 1 do: [:i |
      self assert: i equals: v last.
      v remove ]
  )

  testContains = (
    self assert: (a contains: 23).
    self deny: (a contains: #nono).
  )

  testAppendAndRemoveFirst = (
    | v |
    v := Vector new: 10.
    1 to: 100 do: [:i |
      v append: i ].

    "160 is implementation dependent, just here for orientation"
    self assert: 160 equals: v capacity.
    self assert: 100 equals: v size.

    1 to: 100 do: [:i |
      v removeFirst ].
    1 to: 100 do: [:i |
      v append: i ].

    self assert: 320 equals: v capacity.
    self assert: 100 equals: v size.
  )

  testIndexOf = (
    | v |
    v := Vector new: 3.
    1 to: 17 do: [:i |
      v append: i ].

    self assert: -1 equals: (v indexOf: nil).
    self assert: 1 equals: (v indexOf: 1).

    self assert: 13 equals: (v indexOf: 13).
    v at: 13 put: #test.

    self assert: 13 equals: (v indexOf: #test).

    v removeFirst.
    self assert: -1 equals: (v indexOf: 1).
```

```
  1 to: 12 do: [:i |
    v removeFirst ].
  self assert: -1 equals: (v indexOf: #test).
)

testAppendAll = (
  | v c i |
  v := Vector new: 2.
  v append: 1.
  v append: 2.
  v append: 3.

  c := Array with: 4 with: 5 with: 6.

  v appendAll: c.
  i := 1.

  v do: [:e |
    self assert: i equals: e.
    i := i + 1 ]
)

testAsArray = (
  | v arr |
  v := Vector new.
  self assert: 0 equals: v asArray length.

  v append: 1.
  v append: 2.

  arr := v asArray.
  self assert: 2 equals: arr length.
  self assert: 1 equals: (arr at: 1).
  self assert: 2 equals: (arr at: 2).
)

testAsSet = (
  | v set |
  v := Vector new.
  v append: 1.
  v append: 2.
  v append: 3.
  v append: 4.
  self assert: 4 equals: v size.

  set := v asSet.
  self assert: 4 equals: set size.

  v append: 1.
  v append: 1.
  v append: 1.

  self assert: 4 + 3 equals: v size.

  set := v asSet.
  self assert: 4 equals: set size.
)

testIsEmpty = (
  | v |
```

```
    v := Vector new.
    self assert: v isEmpty.

    v append: 1.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.

    v append: #ee.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.
  )

  testRemoveObj = (
    | v |
    v := Vector new.
    v append: #a.
    v append: #b.
    v append: #c.
    v append: #d.
    v append: #e.
    v append: #f.
    v append: #g.
    v append: #h.

    self assert: 8 equals: v size.

    self deny: (v remove: #aa).
    self assert: (v remove: #e).
    self assert: 7 equals: v size.
  )

  testAppendComma = (
    | v |
    v := Vector new.
    v, #a.
    v, #b.

    self assert: 2 equals: v size.
    self assert: (v contains: #a).
    self assert: (v contains: #b).
  )

  testDoIndexes = (
    | i v |
    v := Vector new.
    v doIndexes: [:j |
      self assert: false ].

    v appendAll: #(1 2 3 4 5).
    i := 1.
    v doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )

```

```
  testDo = (
    | i v |
    v := Vector new.
    v appendAll: #(1 2 3 4 5).
    i := 1.
    v do: [:v |
      self assert: i equals: v.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )
)

```

Examples/Benchmarks/TestSuite/Vector3Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Vector3Test = TestCase (
  | a |

  setUp = (
    a := Vector new.
    a append: 'hello'.
    a append: #world.
    a append: 23.
  )

  testSize = (
    self assert: 3 equals: a size.
  )

  testAt = (
    self assert: #world equals: (a at: 2).
    self assert:     23 equals: (a at: 3).
  )

  testFirst = (
    | v |
    self assert: 'hello' equals: a first.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
```

```
      self assert: 1 equals: v first ].

    1 to: 10 do: [:i |
      self assert: 1 equals: v first.
      v removeFirst ]
  )

  testLast = (
    | v |
    self assert: 23 equals: a last.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
      self assert: i equals: v last ].

    10 downTo: 1 do: [:i |
      self assert: i equals: v last.
      v remove ]
  )

  testContains = (
    self assert: (a contains: 23).
    self deny: (a contains: #nono).
  )

  testAppendAndRemoveFirst = (
    | v |
    v := Vector new: 10.
    1 to: 100 do: [:i |
      v append: i ].

    "160 is implementation dependent, just here for orientation"
    self assert: 160 equals: v capacity.
    self assert: 100 equals: v size.

    1 to: 100 do: [:i |
      v removeFirst ].
    1 to: 100 do: [:i |
      v append: i ].

    self assert: 320 equals: v capacity.
    self assert: 100 equals: v size.
  )

  testIndexOf = (
    | v |
    v := Vector new: 3.
    1 to: 17 do: [:i |
      v append: i ].

    self assert: -1 equals: (v indexOf: nil).
    self assert: 1 equals: (v indexOf: 1).

    self assert: 13 equals: (v indexOf: 13).
    v at: 13 put: #test.

    self assert: 13 equals: (v indexOf: #test).

    v removeFirst.
    self assert: -1 equals: (v indexOf: 1).
```

```
  1 to: 12 do: [:i |
    v removeFirst ].
  self assert: -1 equals: (v indexOf: #test).
)

testAppendAll = (
  | v c i |
  v := Vector new: 2.
  v append: 1.
  v append: 2.
  v append: 3.

  c := Array with: 4 with: 5 with: 6.

  v appendAll: c.
  i := 1.

  v do: [:e |
    self assert: i equals: e.
    i := i + 1 ]
)

testAsArray = (
  | v arr |
  v := Vector new.
  self assert: 0 equals: v asArray length.

  v append: 1.
  v append: 2.

  arr := v asArray.
  self assert: 2 equals: arr length.
  self assert: 1 equals: (arr at: 1).
  self assert: 2 equals: (arr at: 2).
)

testAsSet = (
  | v set |
  v := Vector new.
  v append: 1.
  v append: 2.
  v append: 3.
  v append: 4.
  self assert: 4 equals: v size.

  set := v asSet.
  self assert: 4 equals: set size.

  v append: 1.
  v append: 1.
  v append: 1.

  self assert: 4 + 3 equals: v size.

  set := v asSet.
  self assert: 4 equals: set size.
)

testIsEmpty = (
  | v |
```

```
    v := Vector new.
    self assert: v isEmpty.

    v append: 1.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.

    v append: #ee.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.
  )

  testRemoveObj = (
    | v |
    v := Vector new.
    v append: #a.
    v append: #b.
    v append: #c.
    v append: #d.
    v append: #e.
    v append: #f.
    v append: #g.
    v append: #h.

    self assert: 8 equals: v size.

    self deny: (v remove: #aa).
    self assert: (v remove: #e).
    self assert: 7 equals: v size.
  )

  testAppendComma = (
    | v |
    v := Vector new.
    v, #a.
    v, #b.

    self assert: 2 equals: v size.
    self assert: (v contains: #a).
    self assert: (v contains: #b).
  )

  testDoIndexes = (
    | i v |
    v := Vector new.
    v doIndexes: [:j |
      self assert: false ].

    v appendAll: #(1 2 3 4 5).
    i := 1.
    v doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )

```

```
  testDo = (
    | i v |
    v := Vector new.
    v appendAll: #(1 2 3 4 5).
    i := 1.
    v do: [:v |
      self assert: i equals: v.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )
)
```

Examples/Benchmarks/TestSuite/Vector4Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Vector4Test = TestCase (
  | a |

  setUp = (
    a := Vector new.
    a append: 'hello'.
    a append: #world.
    a append: 23.
  )

  testSize = (
    self assert: 3 equals: a size.
  )

  testAt = (
    self assert: #world equals: (a at: 2).
    self assert:     23 equals: (a at: 3).
  )

  testFirst = (
    | v |
    self assert: 'hello' equals: a first.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
```

```
      self assert: 1 equals: v first ].

    1 to: 10 do: [:i |
      self assert: 1 equals: v first.
      v removeFirst ]
  )

  testLast = (
    | v |
    self assert: 23 equals: a last.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
      self assert: i equals: v last ].

    10 downTo: 1 do: [:i |
      self assert: i equals: v last.
      v remove ]
  )

  testContains = (
    self assert: (a contains: 23).
    self deny: (a contains: #nono).
  )

  testAppendAndRemoveFirst = (
    | v |
    v := Vector new: 10.
    1 to: 100 do: [:i |
      v append: i ].

    "160 is implementation dependent, just here for orientation"
    self assert: 160 equals: v capacity.
    self assert: 100 equals: v size.

    1 to: 100 do: [:i |
      v removeFirst ].
    1 to: 100 do: [:i |
      v append: i ].

    self assert: 320 equals: v capacity.
    self assert: 100 equals: v size.
  )

  testIndexOf = (
    | v |
    v := Vector new: 3.
    1 to: 17 do: [:i |
      v append: i ].

    self assert: -1 equals: (v indexOf: nil).
    self assert: 1 equals: (v indexOf: 1).

    self assert: 13 equals: (v indexOf: 13).
    v at: 13 put: #test.

    self assert: 13 equals: (v indexOf: #test).

    v removeFirst.
    self assert: -1 equals: (v indexOf: 1).
```

```

  1 to: 12 do: [:i |
    v removeFirst ].
  self assert: -1 equals: (v indexOf: #test).
)

testAppendAll = (
  | v c i |
  v := Vector new: 2.
  v append: 1.
  v append: 2.
  v append: 3.

  c := Array with: 4 with: 5 with: 6.

  v appendAll: c.
  i := 1.

  v do: [:e |
    self assert: i equals: e.
    i := i + 1 ]
)

testAsArray = (
  | v arr |
  v := Vector new.
  self assert: 0 equals: v asArray length.

  v append: 1.
  v append: 2.

  arr := v asArray.
  self assert: 2 equals: arr length.
  self assert: 1 equals: (arr at: 1).
  self assert: 2 equals: (arr at: 2).
)

testAsSet = (
  | v set |
  v := Vector new.
  v append: 1.
  v append: 2.
  v append: 3.
  v append: 4.
  self assert: 4 equals: v size.

  set := v asSet.
  self assert: 4 equals: set size.

  v append: 1.
  v append: 1.
  v append: 1.

  self assert: 4 + 3 equals: v size.

  set := v asSet.
  self assert: 4 equals: set size.
)

testIsEmpty = (
  | v |
```

```
    v := Vector new.
    self assert: v isEmpty.

    v append: 1.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.

    v append: #ee.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.
  )

  testRemoveObj = (
    | v |
    v := Vector new.
    v append: #a.
    v append: #b.
    v append: #c.
    v append: #d.
    v append: #e.
    v append: #f.
    v append: #g.
    v append: #h.

    self assert: 8 equals: v size.

    self deny: (v remove: #aa).
    self assert: (v remove: #e).
    self assert: 7 equals: v size.
  )

  testAppendComma = (
    | v |
    v := Vector new.
    v, #a.
    v, #b.

    self assert: 2 equals: v size.
    self assert: (v contains: #a).
    self assert: (v contains: #b).
  )

  testDoIndexes = (
    | i v |
    v := Vector new.
    v doIndexes: [:j |
      self assert: false ].

    v appendAll: #(1 2 3 4 5).
    i := 1.
    v doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )

```

```
  testDo = (
    | i v |
    v := Vector new.
    v appendAll: #(1 2 3 4 5).
    i := 1.
    v do: [:v |
      self assert: i equals: v.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )
)
```

Examples/Benchmarks/TestSuite/Vector5Test.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Vector5Test = TestCase (
  | a |

  setUp = (
    a := Vector new.
    a append: 'hello'.
    a append: #world.
    a append: 23.
  )

  testSize = (
    self assert: 3 equals: a size.
  )

  testAt = (
    self assert: #world equals: (a at: 2).
    self assert:     23 equals: (a at: 3).
  )

  testFirst = (
    | v |
    self assert: 'hello' equals: a first.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
```

```
      self assert: 1 equals: v first ].

    1 to: 10 do: [:i |
      self assert: 1 equals: v first.
      v removeFirst ]
  )

  testLast = (
    | v |
    self assert: 23 equals: a last.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
      self assert: i equals: v last ].

    10 downTo: 1 do: [:i |
      self assert: i equals: v last.
      v remove ]
  )

  testContains = (
    self assert: (a contains: 23).
    self deny: (a contains: #nono).
  )

  testAppendAndRemoveFirst = (
    | v |
    v := Vector new: 10.
    1 to: 100 do: [:i |
      v append: i ].

    "160 is implementation dependent, just here for orientation"
    self assert: 160 equals: v capacity.
    self assert: 100 equals: v size.

    1 to: 100 do: [:i |
      v removeFirst ].
    1 to: 100 do: [:i |
      v append: i ].

    self assert: 320 equals: v capacity.
    self assert: 100 equals: v size.
  )

  testIndexOf = (
    | v |
    v := Vector new: 3.
    1 to: 17 do: [:i |
      v append: i ].

    self assert: -1 equals: (v indexOf: nil).
    self assert: 1 equals: (v indexOf: 1).

    self assert: 13 equals: (v indexOf: 13).
    v at: 13 put: #test.

    self assert: 13 equals: (v indexOf: #test).

    v removeFirst.
    self assert: -1 equals: (v indexOf: 1).
```

```

    1 to: 12 do: [:i |
      v removeFirst ].
    self assert: -1 equals: (v indexOf: #test).
  )

  testAppendAll = (
    | v c i |
    v := Vector new: 2.
    v append: 1.
    v append: 2.
    v append: 3.

    c := Array with: 4 with: 5 with: 6.

    v appendAll: c.
    i := 1.

    v do: [:e |
      self assert: i equals: e.
      i := i + 1 ]
  )

  testAsArray = (
    | v arr |
    v := Vector new.
    self assert: 0 equals: v asArray length.

    v append: 1.
    v append: 2.

    arr := v asArray.
    self assert: 2 equals: arr length.
    self assert: 1 equals: (arr at: 1).
    self assert: 2 equals: (arr at: 2).
  )

  testAsSet = (
    | v set |
    v := Vector new.
    v append: 1.
    v append: 2.
    v append: 3.
    v append: 4.
    self assert: 4 equals: v size.

    set := v asSet.
    self assert: 4 equals: set size.

    v append: 1.
    v append: 1.
    v append: 1.

    self assert: 4 + 3 equals: v size.

    set := v asSet.
    self assert: 4 equals: set size.
  )

  testIsEmpty = (
    | v |
```

```
    v := Vector new.
    self assert: v isEmpty.

    v append: 1.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.

    v append: #ee.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.
  )

  testRemoveObj = (
    | v |
    v := Vector new.
    v append: #a.
    v append: #b.
    v append: #c.
    v append: #d.
    v append: #e.
    v append: #f.
    v append: #g.
    v append: #h.

    self assert: 8 equals: v size.

    self deny: (v remove: #aa).
    self assert: (v remove: #e).
    self assert: 7 equals: v size.
  )

  testAppendComma = (
    | v |
    v := Vector new.
    v, #a.
    v, #b.

    self assert: 2 equals: v size.
    self assert: (v contains: #a).
    self assert: (v contains: #b).
  )

  testDoIndexes = (
    | i v |
    v := Vector new.
    v doIndexes: [:j |
      self assert: false ].

    v appendAll: #(1 2 3 4 5).
    i := 1.
    v doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )

```

```
  testDo = (
    | i v |
    v := Vector new.
    v appendAll: #(1 2 3 4 5).
    i := 1.
    v do: [:v |
      self assert: i equals: v.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )
)
```

Examples/Benchmarks/TestSuite/VectorTest.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

VectorTest = TestCase (
  | a |

  setUp = (
    a := Vector new.
    a append: 'hello'.
    a append: #world.
    a append: 23.
  )

  testSize = (
    self assert: 3 equals: a size.
  )

  testAt = (
    self assert: #world equals: (a at: 2).
    self assert:     23 equals: (a at: 3).
  )

  testFirst = (
    | v |
    self assert: 'hello' equals: a first.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
```

```
      self assert: 1 equals: v first ].

    1 to: 10 do: [:i |
      self assert: 1 equals: v first.
      v removeFirst ]
  )

  testLast = (
    | v |
    self assert: 23 equals: a last.

    v := Vector new.
    1 to: 10 do: [:i |
      v append: i.
      self assert: i equals: v last ].

    10 downTo: 1 do: [:i |
      self assert: i equals: v last.
      v remove ]
  )

  testContains = (
    self assert: (a contains: 23).
    self deny: (a contains: #nono).
  )

  testAppendAndRemoveFirst = (
    | v |
    v := Vector new: 10.
    1 to: 100 do: [:i |
      v append: i ].

    "160 is implementation dependent, just here for orientation"
    self assert: 160 equals: v capacity.
    self assert: 100 equals: v size.

    1 to: 100 do: [:i |
      v removeFirst ].
    1 to: 100 do: [:i |
      v append: i ].

    self assert: 320 equals: v capacity.
    self assert: 100 equals: v size.
  )

  testIndexOf = (
    | v |
    v := Vector new: 3.
    1 to: 17 do: [:i |
      v append: i ].

    self assert: -1 equals: (v indexOf: nil).
    self assert: 1 equals: (v indexOf: 1).

    self assert: 13 equals: (v indexOf: 13).
    v at: 13 put: #test.

    self assert: 13 equals: (v indexOf: #test).

    v removeFirst.
    self assert: -1 equals: (v indexOf: 1).
```

```

    1 to: 12 do: [:i |
      v removeFirst ].
    self assert: -1 equals: (v indexOf: #test).
  )

  testAppendAll = (
    | v c i |
    v := Vector new: 2.
    v append: 1.
    v append: 2.
    v append: 3.

    c := Array with: 4 with: 5 with: 6.

    v appendAll: c.
    i := 1.

    v do: [:e |
      self assert: i equals: e.
      i := i + 1 ]
  )

  testAsArray = (
    | v arr |
    v := Vector new.
    self assert: 0 equals: v asArray length.

    v append: 1.
    v append: 2.

    arr := v asArray.
    self assert: 2 equals: arr length.
    self assert: 1 equals: (arr at: 1).
    self assert: 2 equals: (arr at: 2).
  )

  testAsSet = (
    | v set |
    v := Vector new.
    v append: 1.
    v append: 2.
    v append: 3.
    v append: 4.
    self assert: 4 equals: v size.

    set := v asSet.
    self assert: 4 equals: set size.

    v append: 1.
    v append: 1.
    v append: 1.

    self assert: 4 + 3 equals: v size.

    set := v asSet.
    self assert: 4 equals: set size.
  )

  testIsEmpty = (
    | v |
```

```
    v := Vector new.
    self assert: v isEmpty.

    v append: 1.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.

    v append: #ee.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.
  )

  testRemoveObj = (
    | v |
    v := Vector new.
    v append: #a.
    v append: #b.
    v append: #c.
    v append: #d.
    v append: #e.
    v append: #f.
    v append: #g.
    v append: #h.

    self assert: 8 equals: v size.

    self deny: (v remove: #aa).
    self assert: (v remove: #e).
    self assert: 7 equals: v size.
  )

  testAppendComma = (
    | v |
    v := Vector new.
    v, #a.
    v, #b.

    self assert: 2 equals: v size.
    self assert: (v contains: #a).
    self assert: (v contains: #b).
  )

  testDoIndexes = (
    | i v |
    v := Vector new.
    v doIndexes: [:j |
      self assert: false ].

    v appendAll: #(1 2 3 4 5).
    i := 1.
    v doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )

```

```
  testDo = (
    | i v |
    v := Vector new.
    v appendAll: #(1 2 3 4 5).
    i := 1.
    v do: [:v |
      self assert: i equals: v.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )
)

```

Examples/Benchmarks/TestSuite/duplicate-tests.sh

```
#!/bin/bash
UNAME="$(uname -s)"

echo "Currently, we can only do up to 10 classes, because otherwise, we have too many literals in a single method"
for i in $(seq 1 100)
do
  if [ $(($i % 20)) -eq "0" ]
  then
    echo ". $i"
  else
    echo -n "."
  fi

  TESTS=("Array" "Block" "Boolean" "ClassLoading" "ClassStructure" "Closure" "Coercion"
         "CompilerReturn" "Dictionary" "DoesNotUnderstand" "Double" "Empty" "Global"
         "Hash" "Integer" "Preliminary" "Reflection" "SelfBlock" "Set" "SpecialSelectors"
         "String"
         "Super" "Symbol" "System" "Vector")

  for name in ${TESTS[@]}
  do
    cp "${name}Test.som" "${name}${i}Test.som"
    if [ "${UNAME}" = "Darwin" ]; then
      sed -i '' "s/${name}Test =/${name}${i}Test =/g" "${name}${i}Test.som"
    else
      sed -i'' "s/${name}Test =/${name}${i}Test =/g" "${name}${i}Test.som"
    fi
  done

  # Create TestGC${i}.som
  TEST_GC_FILE="TestGC${i}.som"
  TEST_GC_CLASS="TestGC${i}"

  echo "${TEST_GC_CLASS} = TestGCCommon (" >  "${TEST_GC_FILE}"
  echo "  tests = ("                       >> "${TEST_GC_FILE}"
  echo "    | v |"                         >> "${TEST_GC_FILE}"
  echo "    v := Vector new."              >> "${TEST_GC_FILE}"
  for j in $(seq 1 "$i")
  do
    echo "    self tests${j}: v."          >> "${TEST_GC_FILE}"
  done
  echo "    ^ v"                           >> "${TEST_GC_FILE}"
  echo "  )"                               >> "${TEST_GC_FILE}"
  echo ""                                  >> "${TEST_GC_FILE}"

  for j in $(seq 1 "$i")
  do
    echo "  tests${j}: v = ("              >> "${TEST_GC_FILE}"

    for name in ${TESTS[@]}
    do
      echo "    v append: ${name}${j}Test.">> "${TEST_GC_FILE}"
    done
    echo "  )"                             >> "${TEST_GC_FILE}"
```

```
    echo ""                                >> "${TEST_GC_FILE}"
  done
  echo ")"                                 >> "${TEST_GC_FILE}"

  # Create Test${i}.som
  TEST_FILE="Test${i}.som"
  TEST_CLASS="Test${i}"

  echo "${TEST_CLASS} = TestCommon ("      >  "${TEST_FILE}"
  echo "  numExecs = ("                    >> "${TEST_FILE}"
  echo "    ^ $i"                          >> "${TEST_FILE}"
  echo "  )"                               >> "${TEST_FILE}"
  echo ""                                  >> "${TEST_FILE}"
  echo "  tests = ("                       >> "${TEST_FILE}"
  echo "    | v |"                         >> "${TEST_FILE}"
  echo "    v := Vector new."              >> "${TEST_FILE}"
  for j in $(seq 1 "$i")
  do
    echo "    self tests${j}: v."          >> "${TEST_FILE}"
  done
  echo "    ^ v"                           >> "${TEST_FILE}"
  echo "  )"                               >> "${TEST_FILE}"
  echo ""                                  >> "${TEST_FILE}"

  for j in $(seq 1 "$i")
  do
    echo "  tests${j}: v = ("              >> "${TEST_FILE}"

    for name in ${TESTS[@]}
    do
      echo "    v append: ${name}${j}Test.">> "${TEST_FILE}"
    done
    echo "  )"                             >> "${TEST_FILE}"
    echo ""                                >> "${TEST_FILE}"
  done
  echo ")"                                 >> "${TEST_FILE}"

done

```

Examples/Benchmarks/Towers.som

```
"

$Id: Towers.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"Mmm... Hanoi..."

Towers = Benchmark (

    | piles movesdone |

    pushDisk: disk onPile: pile = (
        | top |

        top := piles at: pile.
        (top isNil not) && [ disk size >= top size ]
            ifTrue: [ self error: 'Cannot put a big disk on a smaller one' ].

        disk next: top.
        piles at: pile put: disk.
    )

    popDiskFrom: pile = (
        | top |

        top := piles at: pile.
        top isNil
            ifTrue: [
                self error: 'Attempting to remove a disk from an empty pile' ].

        piles at: pile put: top next.
        top next: nil.
```

```
        ^top
    )

    moveTopDiskFrom: fromPile to: toPile = (
        self pushDisk: (self popDiskFrom: fromPile) onPile: toPile.
        movesdone := movesdone + 1.
    )

    buildTowerAt: pile disks: disks = (
        disks downTo: 0 do: [ :i |
            self pushDisk: (TowersDisk new: i) onPile: pile ]
    )

    move: disks disksFrom: fromPile to: toPile = (
        disks = 1
            ifTrue: [ self moveTopDiskFrom: fromPile to: toPile ]
            ifFalse: [ | otherPile |
                otherPile := (6 - fromPile) - toPile.
                self move: disks - 1 disksFrom: fromPile to: otherPile.
                self moveTopDiskFrom: fromPile to: toPile.
                self move: disks - 1 disksFrom: otherPile to: toPile. ]
    )

    benchmark = (
        piles := Array new: 4.
        self buildTowerAt: 1 disks: 13.
        movesdone := 0.
        self move: 13 disksFrom: 1 to: 2.
        ^ movesdone
    )

    verifyResult: result = (
      ^ self assert: 8191 equals: result
    )

)
```

Examples/Benchmarks/TowersDisk.som

```
"

$Id: TowersDisk.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

TowersDisk = (

    | size next |

    size        = ( ^size         )
    size: value = ( size := value )
    next        = ( ^next         )
    next: value = ( next := value )

    ----------------------------------

    new: value  = ( ^super new size: value )

)
```

Examples/Benchmarks/TreeNode.som

```
"

$Id: TreeNode.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

TreeNode = (

    | left right value |

    value    = ( ^value     )
    value: v = ( value := v )

    check = (
        ^(left  isNil || [ (left  value <  value) && left  check ]) &&
         (right isNil || [ (right value >= value) && right check ])
    )

    insert: n = (
        (n < value)
            ifTrue: [
                left isNil
                    ifTrue: [ left  := TreeNode new: n ]
                    ifFalse: [ left  insert: n ] ]
            ifFalse: [
                right isNil
                    ifTrue: [ right := TreeNode new: n ]
                    ifFalse: [ right insert: n ] ].
    )

    ------------------------

    new: value = ( ^super new value: value )

)
```

55

Examples/Benchmarks/TreeSort.som

```
"

$Id: TreeSort.som 31 2009-07-31 12:25:18Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

TreeSort = Sort (

    sort: array = (
        | tree |
        array doIndexes: [ :i |
            (i = 1)
                ifTrue: [ tree := TreeNode new: (array at: i) ]
                ifFalse: [ tree insert: (array at: i) ] ].
        ^ tree
    )

    verifyResult: tree = (
       tree check ifFalse: [ self error: 'Invalid result, tree not sorted' ].
       ^ true.
    )

    dataSize = ( ^ 1000 )
)

```

```
"

$Id: Echo.som 226 2008-04-21 12:45:01Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Echo = (

    run: args = (
        args from: 2 to: args length do: [ :arg | arg print. ' ' print ].
        '' println.
    )

)
```

Examples/Hello.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Hello = (

    "The 'run' method is called when initializing the system"
    run = ('Hello, World from SOM' println )

)
```

Examples/Snake/Apple.som

```
"

$Id: Apple.som 191 2008-04-10 18:15:47Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Apple = Element ()

```

Examples/Snake/Board.som

```
"

$Id: Board.som 426 2008-05-22 08:22:07Z stefan.marr $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Board = (
  | view width height board |

  width = ( ^width )
  width: val = ( width := val. )

  height = ( ^height )
  height: val = ( height := val )

  board = ( ^board )
  board: val = ( board := val )

  view = ( ^view )

  addApple = (
    | added x y newApple |
    added := false.

    [ added ] whileFalse: [
      x := 1 atRandom % width. "$x = rand(0, $this->width - 1);"
      y := 1 atRandom % height.

      x := x + 1.
      y := y + 1.

      (self board at: x) isNil ifTrue: [
        self board at: x put: (Array new: height).
      ].
```

```
    ((self board at: x) at: y) isNil ifTrue: [
      newApple := Apple newWithX: x Y: y.
      (self board at: x) at: y put: newApple.
      added := true.
      view isNil ifFalse: [
        view addApple: newApple.
      ].
    ]
  ]
)

view: value = (
  view := value.
  value board: board.
  value updateCompletely.
)

isAppleAtX: x Y: y = (
  ((board at: x) isNil) ifFalse: [
    ((board at: x) at: y) isNil ifFalse: [
      ^((board at: x) at: y) class == Apple
    ]
  ].
  ^false
)
•
isSnakeAtX: x Y: y = (
  ((board at: x) isNil) ifFalse: [
    ((board at: x) at: y) isNil ifFalse: [
      ^((board at: x) at: y) class == SnakeElement
    ]
  ].
  ^false
)
•
remove: element = (
  (self board at: (element x)) at: (element y) put: nil.
  self view remove: element
)
•
add: element = (
  (self board at: element x) isNil ifTrue: [
    self board at: (element x) put: (Array new: height).
  ].
  (self board at: (element x)) at: (element y) put: element.
  view add: element
)

----
™
newWithWidth: width height: height numberOfApples: numberOfApples = (
  | newBoard |
  newBoard := Board new.
  newBoard board: (Array new: width).
  newBoard width: width.
  newBoard height: height.
™
  [numberOfApples >= 0] whileTrue: [
    newBoard addApple.
    numberOfApples := numberOfApples - 1.
```

```
    ].
    ^newBoard
  )
)

```

Examples/Snake/BoardView.som

```
"

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

BoardView = (
  | board width height |
•
  board: value = ( board := value )
  width: value = ( width := value )
  height: value = ( height := value )

  updateCompletely = (
    board do: [ :y |
      y isNil ifFalse: [
        y do: [ :apple |
          apple isNil ifFalse: [
            Terminal cursorToX: (apple x + 1) Y: (apple y + 1).
            Terminal put: 'o'
          ]
        ]
      ]
    ]
  )
•
  remove: snakeElement = (
    Terminal cursorToX: snakeElement x + 1 Y: snakeElement y + 1.
    Terminal put: ' '
  )
•
  add: snakeElement = (
    Terminal cursorToX: snakeElement x + 1 Y: snakeElement y + 1.
    Terminal put: '#'
  )
•
  addApple: apple = (
```

```
    Terminal cursorToX: apple x + 1 Y: apple y + 1.
    Terminal put: 'o'
  )

  drawBoarder = (
    Terminal cursorToX: 1 Y: 1.
    Terminal put: '/'.
    width timesRepeat: [ Terminal put: '-' ].
    Terminal put: '\\'.

    1 to: height do: [ :i |
      Terminal cursorToX: 1 Y: i + 1.
      Terminal put: '|'.
      Terminal cursorToX: (width + 2) Y: i + 1.
      Terminal put: '|'
    ].

    Terminal cursorToX: 1 Y: height + 2.
    Terminal put: '\\'.
    width timesRepeat: [ Terminal put: '-' ].
    Terminal put: '/'.
  )

  ----

  new: board = (
    | newBoardView |
    newBoardView := BoardView new.
    board view: newBoardView.
    newBoardView width: board width.
    newBoardView height: board height.
    ^newBoardView
  )
)
```

Examples/Snake/Core.cpp

```
#include <string.h>

//#include "../misc/defs.h"
//#include "../vmobjects/VMObject.h"
#include "../../vmobjects/PrimitiveRoutine.h"

#include "Terminal.h"

#include "../../primitivesCore/Routine.h"
#include "../../primitivesCore/PrimitiveContainer.h"
#include "../../primitivesCore/PrimitiveLoader.h"

static PrimitiveLoader* loader = nullptr;
//map<pString, PrimitiveContainer*> primitiveObjects;
//"Constructor"
static bool initialized = false;
extern "C" void setup() {
    if (!loader) {
        //Initialize loader
        loader = new PrimitiveLoader();
        loader->AddPrimitiveObject("Terminal",
            static_cast<PrimitiveContainer*>(new Terminal()));
}

extern "C" bool supportsClass(const char* name) {
    //if (!loader) setup();
    return loader->SupportsClass(name);
}


extern "C" void tearDown() {
    //primitiveClasses.clear();
    if (loader) delete loader;
    //if (primitiveObjects) delete primitiveObjects;
}

extern "C" PrimitiveRoutine* create(const pString& cname, const pString& fname, bool isPrimitive) {

#ifdef __DEBUG
    cout << "Loading PrimitiveContainer: " << cname << "::" << fname << endl;
#endif
    //if (!loader) setup();

    return loader->GetPrimitiveRoutine(cname, fname, isPrimitive);
}
```

Examples/Snake/Element.som

```
"

$Id: Element.som 191 2008-04-10 18:15:47Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Element = (
  | x y |

  x = ( ^x )
  x: val = ( x := val )

  y = ( ^y )
  y: val = ( y := val )

  ----
  newWithX: x Y: y = (
    | newElement |
    newElement := self new.
    newElement x: x.
    newElement y: y.
    ^newElement.
  )
)
```

Examples/Snake/Main.som

```
"

$Id: Main.som 191 2008-04-10 18:15:47Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Main = (
  run = (
    | board view chr snake dir continue |
    Terminal init.
    Terminal clear.
    dir := Terminal KEY_UP.

    board := Board newWithWidth: 10 height: 10 numberOfApples: 5.
    view := BoardView new: board.
    view drawBoarder.

    snake := Snake newWithX: 5 Y: 5 andBoard: board.
    continue := true.
    [ continue ] whileTrue: [
      chr := Terminal get.
      ((Terminal KEY_UP = chr)
        || ((Terminal KEY_DOWN) = chr)
        || ((Terminal KEY_LEFT) = chr)
        || ((Terminal KEY_RIGHT) = chr)) ifTrue: [ dir := chr ].
      Terminal sleepFor: 250.

      "Terminal cursorToX: 15 Y: 15."
      (Terminal KEY_UP = dir) ifTrue: [ continue := snake moveUp ].
      (Terminal KEY_DOWN = dir) ifTrue: [ continue := snake moveDown ].
      (Terminal KEY_LEFT = dir) ifTrue: [ continue := snake moveLeft ].
      (Terminal KEY_RIGHT = dir) ifTrue: [ continue := snake moveRight ].
      '' println.
    ].

```

```
    'GAME OVER' println.
    Terminal uninit.
  )
)

"

while (true) {
’F’ Ò C°
—v†–ÆR ,F’ â  ’ °
™$key = Terminal::get(0);
™if (in_array($key, array(Terminal::KEY_UP, Terminal::KEY_DOWN,
Terminal::KEY_LEFT, Terminal::KEY_RIGHT))) {
™’FF—" Ò F¶W“°
™}
™usleep(100000);
™$i--;
—Ð
™
•
––b , G&W7VÇB’ °
™Terminal::cursorTo(5, 15);
™Terminal::put('GAME OVER');
™sleep(5);
™return;
—Ð
}

Terminal::clear();
"

```

Examples/Snake/Makefile

```
#!/usr/bin/env make

TARGET™=Terminal


################ ONLY CHANGE STUFF BELOW IF YOU REALLY HAVE
###################


CC™“Ör²°
CFLAGS™=-Wno-endif-labels -O2 $(DBG_FLAGS) $(INCLUDES)
LDFLAGS™=-enable-auto-import $(LIBRARIES)

INSTALL™=install

############ global stuff -- overridden by ../Makefile

ROOT_DIR“óÒ B… tB’òâãòâà
BUILD_DIR   ?= $(ROOT_DIR)/build
SRC_DIR™?= $(ROOT_DIR)/src

ST_DIR™?= $(ROOT_DIR)/Smalltalk
EX_DIR™?= $(ROOT_DIR)/Examples
TEST_DIR“óÒ B…$ôõEôD•"’õFW7E7V—FP

############# include path

INCLUDES™=-I$(SRC_DIR)
LIBRARIES™=-L$(ROOT_DIR)

ifneq ($(OS),)
# only Windows has OS predefined.
”Ä”%9“ÒÖÆÒ ÖÂâãòâãô5  4ôÒ ÖÂâãòâãõ &–Ö—F—fT6÷&P
endif

all: $(TARGET).csp

debug : DBG_FLAGS=-DDEBUG -g
debug: all

profiling : DBG_FLAGS=-g -pg
profiling : LDFLAGS+=-pg
profiling: all

$(TARGET).csp:
’B„42’ B„4dÄ u2’ B„ÄDdÄ u2’ ×6† &VB Ör À
™$(PWD)/$(TARGET).cpp -o $(TARGET).csp $(LIBS)
•
clean:
—&Ò Õ&b B…D $tUB’æ77
```

Examples/Snake/Snake.som

```
"

$Id: Snake.som 426 2008-05-22 08:22:07Z stefan.marr $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Snake = (
  | head tail board |

  head: val = ( head := val )
  tail: val = ( tail := val )
  board: val = ( board := val )

  moveLeft = (
    | newPos |
    newPos := SnakeElement newWithX: head x Y: head y.
    newPos x: (self overflow: (newPos x - 1) max: board width).
    ^self move: newPos
  )

  moveRight = (
    | newPos |
    newPos := SnakeElement newWithX: head x Y: head y.
    newPos x: (self overflow: (newPos x + 1) max: board width).
    ^self move: newPos
  )

  moveUp = (
    | newPos |
    newPos := SnakeElement newWithX: head x Y: head y.
    newPos y: (self overflow: (newPos y - 1) max: board height).
    ^self move: newPos
  )

```

```
moveDown = (
  | newPos |
  newPos := SnakeElement newWithX: head x Y: head y.
  newPos y: (self overflow: (newPos y + 1) max: board height).
  ^self move: newPos
)

move: newPos = (
  newPos next: head.
  head prev: newPos.
  head := newPos.

  (board isAppleAtX: (newPos x) Y: (newPos y)) ifTrue: [
    board addApple
  ] ifFalse: [

    (board isSnakeAtX: newPos x Y: newPos y) ifTrue: [
      ^false
    ].
    board remove: tail.
    tail := tail prev.
    tail next: nil.
  ].
  board add: newPos.

  ^true
)
•
overflow: val max: max = (
  (val < 1) ifTrue: [ val := max + val ].
  (val > max) ifTrue: [ val := val - max ].
  ^val
)

----
newWithX: x Y: y andBoard: board = (
  | newSnake head |
  newSnake := Snake new.
  head := SnakeElement newWithX: x Y: y.

  newSnake head: head.
  newSnake tail: head.
  newSnake board: board.
  board add: head.
  ^newSnake
)
)
```

Examples/Snake/SnakeElement.som

```
"

$Id: SnakeElement.som 191 2008-04-10 18:15:47Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SnakeElement = Element (
  | next prev |
  next = ( ^next )
  next: val = ( next := val )

  prev = ( ^prev )
  prev: val = ( prev := val )
)

```

Examples/Snake/Terminal.c

```c
/*
Copyright (c) 2001-2008 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
  */

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <stddef.h>


/*************************************************/
#pragma mark * Included Headers                  *
/*************************************************/


#include <vmobjects/VMFrame.h>
#include <vmobjects/VMString.h>
#include <vm/Universe.h>


#include <termios.h>
#include <fcntl.h>

/*************************************************/
#pragma mark * Primitive Foreward Declaration    *
/*************************************************/

extern "C" {

void _Terminal_sleepFor_(pVMObject object, pVMFrame frame);
void _Terminal_getChar(pVMObject object, pVMFrame frame);
void _Terminal_uninit(pVMObject object, pVMFrame frame);
```

```
void _Terminal_init(pVMObject object, pVMFrame frame);

}

/************************************************/
#pragma mark * Internal functions and init.      *
/************************************************/

/*** Lib initialization **/
#ifdef __GNUC__
void init(void) __attribute__((constructor));
void fini(void) __attribute__((destructor));
#else
void _init(void);
void _fini(void);
#pragma init _init
#pragma fini _fini
#endif

#ifdef __GNUC__
void init(void)
#else
void _init(void)
#endif
{
’òò 6 ÆÂ –æ—B gVæ6–öç2à
•

}

#ifdef __GNUC__
void fini(void)
#else
void _fini(void)
#endif
{
•
}

// Classes supported by this lib.
static char *supported_classes[] = {
    "Terminal",
    nullptr
};


/************************************************/
#pragma mark * Exported functions starting here  *
/************************************************/

extern "C" {

// returns, whether this lib is responsible for a specific class
bool™supports_class(const char* name) {
•
–6† " ¢¦—FW#×7W  ÷'FVEö6Æ 76W3°
—v†–ÆR,¦—FW"•
™if (strcmp(name,*iter++)==0)
™—&WGW&â G'VS°
—&WGW&â f Ç6S°
•
```

```
}

bool initialized = false;


/**
 * init_csp()
 *
 * the library initializer. It is not equal to the init/fini-pair, as for them,
 * the init is executed upon every loading of the shared library.
 *
 * All work that needs to be done before the actual primitives are assigned
 * should be called from this function.
 */
void init_csp(void) {
––æ–F– Æ–¦VB Ò f Ç6S°
}


ptrdiff_t terminalStream;
struct termios old_tty;

void init_the_terminal() {
•
––b †–æ–F– Æ–¦VB•
™return;
•
–7G'V7B FW&Ö–÷2 GG“°
•
’òò  W'  &R FW&Ö–æ Â 6WGF–æw2  æB 6† ævR Fò æöâÖ6 æöæ–6 Â ÖöFR f÷" 6† "Ð wise input
–F6vWF GG"ƒ Â föÆE÷GG’“°
–GG’ Ò öÆE÷GG“°
–GG’æ5öÆfÆ r Ò GG’æ5öÆfÆ r b â„T4„ò Â T4„ô² Â ”4 äôâ“°
–GG’æ5ö65µeD”ÔUÒ Ò  °
–F76WF GG"ƒ Â D54 äõrÂ gGG’“°
•
–FW&Ö–æ Å7G&V Ò Ò ÷ Vâ,"öFWb÷GG’"Â õõ$DôäÅ’ Â õôäôä$Äô4²“°
––b ‡FW&Ö–æ Å7G&V Ò Â  ’ °
™Universe_error_exit("Could not open /dev/tty for read\n");
–Ð
––æ–F– Æ–¦VB Ò G'VS°
}


/************************************************/
/************************************************/
/************************************************/
#pragma mark * Primitive Implementatition here   *
/************************************************/
/************************************************/
/************************************************/


void Terminal_getChar(pVMObject object, pVMFrame frame) {
  init_the_terminal();
  char chr;
  char result[2];
  pString str = nullptr;
```

```
  VMObject* vmStr = nullptr;

  pVMObject self __attribute__((unused)) = SEND(frame, pop);

  if (read(terminalStream, &chr, sizeof(chr)) > 0) {
’  &W7VÇE³ Ò Ò 6‡#°
’  &W7VÇE³ Ò Ò  °
’
’  7G" Ò 7G&–æuöæWr‡&W7VÇB“°
’  fÕ7G" Ò ‡ dÔö&¦V7B•dÕ7G&–æuöæWu÷v—F,‡7G"“°
’  4TäB†g& ÖRÂ  W6,Â fÕ7G"“°
  } else {
’  4TäB†g& ÖRÂ  W6,Â æ–Åöö&¦V7B“°
  }
}

void Terminal_uninit(pVMObject object, pVMFrame frame) {
—7G'V7B FW&Ö–÷2 GG“°

–6Æ÷6R‡FW&Ö–æ Å7G&V Ò“°
—F76WF GG"ƒ Â D54 äõrÂ gGG’“°
—F76WF GG"ƒ Â D54 äõrÂ föÆE÷GG’“°
}

void Terminal_init(pVMObject object, pVMFrame frame) {
}

void Terminal_sleepFor_(pVMObject object, pVMFrame frame) {
    pVMInteger miliSeconds = (pVMInteger)SEND(frame, pop);
    int64_t sec = (int64_t)SEND(miliSeconds, get_embedded_integer) * 1000;
    sync();
    usleep(sec);
}

}

/**************************************************/
/**************************************************/
/**************************************************/
#pragma mark * EOF                              *
/**************************************************/
/**************************************************/
/**************************************************/
```

Examples/Snake/Terminal.cpp

```
/*
 * $Id: Terminal.c 426 2008-05-22 08:22:07Z stefan.marr $
 *
Copyright (c) 2001-2008 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
  */

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>


/**************************************************/
#pragma mark * Included Headers                  *
/**************************************************/


#include <vmobjects/VMObject.h>
#include <vmobjects/VMFrame.h>
#include <vmobjects/VMString.h>
#include <vm/Universe.h>

#include <primitivesCore/Routine.h>

#include <termios.h>
#include <fcntl.h>

#include "Terminal.h"

/**************************************************/
#pragma mark * Primitive Foreward Declaration    *
/**************************************************/

```

```

/***********************************************/
/***********************************************/
/***********************************************/
#pragma mark * Primitive Implementatition here   *
/***********************************************/
/***********************************************/
/***********************************************/


struct termios old_tty;
Terminal::Terminal() : PrimitiveContainer() {
    terminalStream = 0;
    SetPrimitive("getChar",   new (GetHeap<HEAP_CLS>()) Routine<Terminal>(this, &Terminal::getChar));
    SetPrimitive("uninit",    new (GetHeap<HEAP_CLS>()) Routine<Terminal>(this, &Terminal::uninit));
    SetPrimitive("init",      new (GetHeap<HEAP_CLS>()) Routine<Terminal>(this, &Terminal::init));
    SetPrimitive("sleepFor_", new (GetHeap<HEAP_CLS>()) Routine<Terminal>(this, &Terminal::sleepFor_));
}

void Terminal::getChar(VMObject* object, VMFrame* frame) {
  char chr;
  char result[2];
  pString str = nullptr;
  VMObject* vmStr = nullptr;
  frame->Pop();
  //VMObject self __attribute__((unused)) = SEND(frame, pop);

  if (read(terminalStream, &chr, sizeof(chr)) > 0) {
’  &W7VÇE³ Ò Ò 6‡#°
’  &W7VÇE³ Ò Ò  °
’
’  7G" Ò  7G&–ær‡&W7VÇB“°
      vmStr = (VMObject*)GetUniverse()->NewString(str);
      frame->Push(vmStr);
  } else {
      frame->Push(Globals::NilObject());
  }
}

void Terminal::uninit(VMObject* object, VMFrame* frame) {
–6Æ÷6R‡FW&Ö–æ Å7G&V Ò“°
—F76WF GG"ƒ Â D54 äõrÂ föÆE÷GG’“°
}

void Terminal::init(VMObject* object, VMFrame* frame) {
—7G'V7B FW&Ö–÷2 GG“°
•
’òò  W'  &R FW&Ö–æ Â 6WGF–æw2  æB 6† ævR Fò æöâÖ6 æöæ–6 Â ÖöFR f÷" 6† "Ð wise input
—F6vWF GG"ƒ Â föÆE÷GG’“°
—GG’ Ò öÆE÷GG“°
—GG’æ5öÆfÆ r Ò GG’æ5öÆfÆ r b â„T4„ò Â T4„ô² Â ”4 äôâ“°
—GG’æ5ö65µeD”ÔUÒ Ò  °
—F76WF GG"ƒ Â D54 äõrÂ gGG’“°
•
—FW&Ö–æ Å7G&V Ò Ò ÷ Vâ,"öFWb÷GG’"Â õõ$DôäÅ’ Â õôäôä$Äô4²“°
––b ‡FW&Ö–æ Å7G&V Ò Â  ’ °
```

```
        GetUniverse()->ErrorExit("Could not open /dev/tty for read\n");
—Đ
}

void Terminal::sleepFor_(VMObject* object, VMFrame* frame) {
    VMInteger* miliSeconds = (VMInteger*)frame->Pop();
    int64_t sec = (int64_t)miliSeconds->GetEmbeddedInteger() * 1000;
    sync();
    usleep(sec);
}

/**************************************************/
/**************************************************/
/**************************************************/
#pragma mark * EOF                               *
/**************************************************/
/**************************************************/
/**************************************************/


```

```
#pragma once

#include <primitivesCore/PrimitiveContainer.h>

class Terminal : public PrimitiveContainer {
public:
    Terminal();
    void sleepFor_(VMObject* object, VMFrame* frame);
    void getChar(VMObject* object, VMFrame* frame);
    void uninit(VMObject* object, VMFrame* frame);
    void init(VMObject* object, VMFrame* frame);
private:
    int terminalStream;
};
```

Examples/Snake/Terminal.som

```
"

$Id: Terminal.som 191 2008-04-10 18:15:47Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Terminal = (

  ----
  KEY_UP = (^#key_up)
  KEY_DOWN = (^#key_down)
  KEY_LEFT = (^#key_left)
  KEY_RIGHT = (^#key_right)
•
  init = primitive

  uninit = primitive

  cursorToX: x Y: y = (
    ' [' print.
    y print.
    ';' print.
    x print.
    'H' print.
  )

  clear = (
    ' [2J' print.
  )

  put: str = (
    str print
  )

  getChar = primitive
```

```

  get = (
    | chr result |
    result := ''.
    chr := self getChar.

    [ chr = nil ] whileFalse: [
      result := result + chr.
      chr := self getChar
    ].

    ^self recognizeKeys: result.
  )

  recognizeKeys: chrs = (
    (chrs = ' [A') ifTrue: [
      ^self KEY_UP.
    ].

    (chrs = ' [B') ifTrue: [
      ^self KEY_DOWN.
    ].

    (chrs = ' [C') ifTrue: [
      ^self KEY_RIGHT.
    ].

    (chrs = ' [D') ifTrue: [
      ^self KEY_LEFT.
    ].

    (chrs = '') ifTrue: [
      ^nil.
    ].

    ^chrs
  )

  sleepFor: sec = primitive
)

```

LICENSE

README.md

````markdown
SOM - Simple Object Machine
===========================

SOM is a minimal Smalltalk dialect used to teach VM construction at the [Hasso
Plattner Institute][SOM]. It was originally built at the University of Århus
(Denmark) where it was also used for teaching.

Currently, implementations exist for Java (SOM), C (CSOM), C++ (SOM++), and
Squeak/Pharo Smalltalk (AweSOM).

### Fork implemented in Go

### SOM

A simple SOM Hello World looks like:

```Smalltalk
Hello = (
  run = (
    'Hello World!' println.
  )
)
```

This repository contains the standard library of SOM, without an actual SOM
implementation. Please see the [main project page][SOMst] for links to the
VM implementation.

With CSOM, the given example could be executed for instance like:
   `./CSOM -cp Smalltalk Hello.som`

AweSOM can be asked to directly evaluate a given string, for instance like:
   `SOMUniverse new eval: '''Hello World!'' println'.'.`

A version of AweSOM is available for Pharo via:
```Smalltalk
Gofer it
    url: 'http://ss3.gemstone.com/ss/AweSOM';
    package: 'ConfigurationOfAweSOM';
    load.
(Smalltalk at: #ConfigurationOfAweSOM) loadDevelopment
```

To install it into a recent Squeak, use the following expression:
```Smalltalk
Installer ss3
    project: 'AweSOM';
    install: 'ConfigurationOfAweSOM'.
(Smalltalk at: #ConfigurationOfAweSOM) perform: #loadDevelopment
```

Information on previous authors are included in the AUTHORS file. This code is
distributed under the MIT License. Please see the LICENSE file for details.
````

```
Build Status
------------

Thanks to Travis CI, all commits of this repository are tested.
The current build status is: [![Build Status](https://travis-ci.org/SOM-st/SOM.png?branch=master)](https://travis-ci.org/SOM-st/SOM/)

The build status of the SOM implementations is as follows:

* CSOM: [![CSOM Build Status](https://travis-ci.org/SOM-st/CSOM.png?branch=master)](https://travis-ci.org/SOM-st/CSOM/)
* 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/)
* PySOM: [![PySOM Build Status](https://travis-ci.org/SOM-st/PySOM.png?branch=master)](https://travis-ci.org/SOM-st/PySOM)
* RPySOM: [![RPySOM Build Status](https://travis-ci.org/SOM-st/RPySOM.png?branch=master)](https://travis-ci.org/SOM-st/RPySOM)
* TruffleSOM: [![TruffleSOM Build Status](https://travis-ci.org/SOM-st/TruffleSOM.png?branch=master)](https://travis-ci.org/SOM-st/TruffleSOM)


 [SOM]: http://www.hpi.uni-potsdam.de/hirschfeld/projects/som/
 [SOMst]: https://travis-ci.org/SOM-st/
```

Smalltalk/Array.som

```
"

$Id: Array.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Array = (

    "Accessing"
    at: index            = primitive
    at: index put: value = primitive
    length               = primitive
    putAll: block        = ( self doIndexes: [ :i |
                                self at: i put: block value ] )
    first = ( ^ self at: 1 )
    last  = ( ^ self at: self length )


    "Iterating"
    do: block            = ( self doIndexes: [ :i |
                                block value: (self at: i) ] )
    doIndexes: block     = ( 1 to: self length do: [:i |
                                block value: i. ] )

    from: start to: end do: block = (
        start to: end do: [:i | block value: (self at: i) ] )

    "Copying (inclusively)"
    copyFrom: start to: end = (
        | result i |

        result := Array new: end - start + 1.
        i := 1.
        self from: start to: end do: [ :e |
```

```
          result at: i put: e.
          i := i + 1 ].

        ^result
    )

    copyFrom: start = ( ^self copyFrom: start to: self length )

    replaceFrom: start to: stop with: replacement startingAt: repStart = (
        "This destructively replaces elements from start to stop in the
        receiver starting at index, repStart, in the sequenceable collection,
        replacementCollection. Answer the receiver. No range checks are
        performed."
        | index repOff |
        repOff := repStart - start.
        index := start - 1.
        [(index := index + 1) <= stop]
            whileTrue: [self at: index put: (replacement at: repOff + index)]
    )

    copy = (^self copyFrom: 1)

    prependedWith: val = (
        | result |
        result := Array new: self length + 1.
        self doIndexes: [:i |
            result at: i + 1 put: (self at: i)].
        result at: 1 put: val.
        ^ result
    )

    extendedWith: val = (
        | result |
        result := Array new: self length + 1.
        self doIndexes: [:i |
            result at: i put: (self at: i)].
        result at: result length put: val.
        ^ result
    )

    "Numerical"
    sum     = ( ^self inject: 0 into: [ :sub :elem | sub + elem ] )
    average = ( ^self sum / self length )

    join: joiner = (
        | result first |
        first := true.

        self do: [:e |
            first
                ifTrue: [
                    result := e.
                    first := false]
                ifFalse: [
                    result := result + joiner + e] ].
        ^ result
    )

    "Containment check"
```

```
    contains: element = ( self do: [ :e | e = element ifTrue: [ ^true ] ].
                          ^false )
    indexOf: element = (
      self doIndexes: [ :i | (self at: i) = element ifTrue: [ ^ i ]].
      ^ nil
    )

    lastIndexOf: element = (
      self length downTo: 1 do: [: i | (self at: i) = element ifTrue: [ ^ i ]].
      ^ nil
    )

    "Collection"
    collect: aBlock = (
        | result |
        result := Array new: self length.
        self doIndexes: [ :i | result at: i put: (aBlock value: (self at: i)) ].
        ^result
    )

    inject: sub into: aBlock = ( | next |
        next := sub.
        self do: [ :e | next := aBlock value: next with: e ].
        ^next
    )

    reject: aBlock = (
        ^ self select: [:element | (aBlock value: element) == false ]
    )

    select: aBlock = (
        "TODO: fix the hard reference to Vector..."
        | newCollection |
        newCollection := Vector new: self length.
        self do: [:each | (aBlock value: each)
                            ifTrue: [newCollection append: each]].
        ^ newCollection
    )

    union: aCollection = (
        | new |
        new := Set new.
        new addAll: self.
        new addAll: aCollection.
        ^ new
    )

    ----------------------------

    "Allocation"
    new                        = ( ^self new: 0 )
    new: length                = primitive
    new: length withAll: block = ( ^((self new: length) putAll: block) )

    "Convenience"
    with: a = (
        | arr |
        arr := self new: 1.
        arr at: 1 put: a.
```

```
        ^ arr
    )

    with: a with: b            = (
        | arr |
        arr := self new: 2.
        arr at: 1 put: a.
        arr at: 2 put: b.
        ^ arr
    )

    with: a with: b with: c    = (
        | arr |
        arr := self new: 3.
        arr at: 1 put: a.
        arr at: 2 put: b.
        arr at: 3 put: c.
        ^ arr
    )
)
```

```
"

$Id: Block.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Block = (

    "For the creation of Block instances, see Universe_new_block()."

    "Evaluation"
    value = primitive

    "Looping"
    whileFalse: block = (
        [ self value not ] whileTrue: block
    )

    whileTrue: block = (
        self value ifFalse: [ ^nil ].
        block value.
        self restart
    )

    "Restarting"
    restart = primitive

)

```

Smalltalk/Block1.som

```
"

$Id: Block1.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Block1 = Block (

    "For the creation of Block instances, see Universe_new_block()."

    "Evaluating"
    value = primitive

)

```

```
"

$Id: Block2.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Block2 = Block (

    "For the creation of Block instances, see Universe_new_block()."

    "Evaluating"
    value           = ( self value: nil )
    value: argument = primitive

)

```

Smalltalk/Block3.som

```
"

$Id: Block3.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Block3 = Block (

    "For the creation of Block instances, see Universe_new_block()."

    "Evaluating"
    value                  = ( self value: nil with: nil )
    value: arg             = ( self value: arg with: nil )
    value: arg1 with: arg2 = primitive

)

```

Smalltalk/Boolean.som

```
"

$Id: Boolean.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Boolean = (

    "Conditional evaluation"
    ifTrue: trueBlock ifFalse: falseBlock = (
        self ifTrue:  [ ^trueBlock value  ].
        self ifFalse: [ ^falseBlock value ].
    )

    "Logical operations"
    || boolean = ( ^self or: boolean  )
    && boolean = ( ^self and: boolean )

)


```

Smalltalk/Class.som

```
"

$Id: Class.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Class = (

    "Accessing"
    name     = primitive

    "Converting"
    asString = ( ^self name asString )

    "Allocation"
    new = primitive

    "Meta Information"
    superclass = primitive
    fields     = primitive
    methods    = primitive
    selectors  = ( ^self methods collect: [:inv | inv signature ] )

    hasMethod: aSymbol = (
        self methods do: [ :m |
            m signature = aSymbol ifTrue: [ ^true ] ].
        ^false
    )

)

```

Smalltalk/Dictionary.som

```
"

$Id: Dictionary.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Dictionary = (

    | pairs |

    at: aKey put: aValue = (
        (self containsKey: aKey)
            ifTrue:  [ (self pairAt: aKey) value: aValue ]
            ifFalse: [ pairs add: (Pair withKey: aKey andValue: aValue) ]
    )

    at: aKey = (
        pairs do: [ :p | p key = aKey ifTrue: [ ^p value ] ].
        ^nil
    )

    containsKey: aKey = (
        pairs do: [ :p | p key = aKey ifTrue: [ ^true ] ].
        ^false
    )

    keys   = ( ^pairs collect: [ :p | p key ] )
    values = ( ^pairs collect: [ :p | p value ] )

    "Iteration"
    do: block = ( pairs do: block )

    "Private"
    pairs: aSet = ( pairs := aSet )
    pairAt: aKey = (
```

```
        pairs do: [ :p | p key = aKey ifTrue: [ ^p ] ].
        ^nil
    )

    "Printing"
    print = ( '{' print. pairs do: [ :p | p print ]. '}' print )
    println = ( self print. '' println )

    ----

    new = (
        | newDictionary |
        newDictionary := super new.
        newDictionary pairs: Set new.
        ^newDictionary
    )

)
```

```
"

$Id: Double.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Double = (

    "Arithmetic"
    + argument  = primitive
    - argument  = primitive
    * argument  = primitive
    // argument  = primitive
    % argument  = primitive
    abs         = ( ^(self < 0.0) ifTrue: (0.0 - self) ifFalse: self )
    sqrt        = primitive
    negated     = ( ^0.0 - self )
    round       = primitive
    asInteger   = primitive
    cos         = primitive
    sin         = primitive

    raisedTo: exponent = (
        "Raise the receiver to the given exponent.
         Currently only positive integer exponents
         are fully supported."
        | output |
        output := 1.0.
        exponent asInteger
          timesRepeat: [ output := output * self ].
        ^ output
    )

    "Comparing"
    =  argument = primitive
```

```
    <  argument = primitive
    >  argument = ( ^(self >= argument) and: [ self <> argument ] )
    >= argument = ( ^(self < argument) not )
    <= argument = ( ^(self < argument) or: [ self = argument ] )
    negative    = ( ^self < 0.0 )
    between: a and: b = ( ^(self > a) and: [ self < b ] )

    "Converting"
    asString    = primitive

    "Iterating"
    to: limit do: block = (
        | i |
        i := self.
        [ i <= limit ] whileTrue: [ block value: i. i := i + 1.0 ]
    )

    downTo: limit do: block = (
        | i |
        i := self.
        [ i >= limit ] whileTrue: [ block value: i. i := i - 1.0 ]
    )

    ----

    PositiveInfinity = primitive

    "Convert string into Double. In case of any errors, the result is NaN"
    fromString: aString = primitive
)
```

Smalltalk/False.som

```
"

$Id: False.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

False = Boolean (

    "Converting"
    asString       = ( ^'false' )

    "Conditional evaluation"
    ifTrue:  block = ( ^nil )
    ifFalse: block = ( ^block value )

    "Logical operations"
    not        = ( ^true )
    or: block  = ( ^block value )
    and: block = ( ^false )

)

```

Smalltalk/HashEntry.som

```
"

$Id: HashEntry.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"
This class is not meant for direct use - it's an internal datastructure
for Hashtable
"

HashEntry = (

    | key value next hash |

    key       = ( ^key )
    value     = ( ^value )
    next      = ( ^next )
    hash      = ( ^hash )

    key: k    = ( key := k )
    value: v  = ( value := v )
    next: n   = ( next := n )
    hash: h   = ( hash := h )

    setKey: key value: value = (
        key = self key
            ifTrue: [ self value: value. ^false. ]
            ifFalse: [
            next isNil
                ifTrue: [
                    self next: (HashEntry newKey: key value: value next: nil).
                    ^true. ]
                ifFalse: [
```

```
                    ^(self next setKey: key value: value) ] ].
    )

    getValue: key = (
        key = self key ifTrue: [ ^value ].
        next isNil ifTrue: [ ^nil ].
        ^next getValue: key.
    )

    containsKey: key = (
        key = self key ifTrue: [ ^true ].
        next isNil ifTrue: [ ^false ].
        ^next containsKey: key.
    )

    containsValue: value = (
        value = self value ifTrue: [ ^true ].
        next isNil ifTrue: [ ^false ].
        ^next containsValue: value.
    )

    keys = (
        next isNil
            ifTrue: [ ^Vector new append: key ]
            ifFalse: [ ^(next keys), key ]
    )

    values = (
        next isNil
            ifTrue: [ ^Vector new append: value ]
            ifFalse: [ ^(next values), value ]
    )

    ----

    newKey: k value: v next: n = (
        | newEntry |
        newEntry := super new.
        newEntry key: k.
        newEntry value: v.
        newEntry next: n.
        newEntry hash: (k hashcode).
        ^newEntry
    )

)
```

Smalltalk/Hashtable.som

```
"

$Id: Hashtable.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Hashtable = (

    | table count |

    "Testing"
    containsKey: key = ( | idx e |
        idx := self indexForKey: key.
        e := table at: idx.
        e isNil ifFalse: [ e keys do: [ :k | k = key ifTrue: [ ^true ] ] ].
        ^false.
    )

    containsValue: val = (
        table do: [ :ent |
            ent isNil ifFalse: [
                ent values do: [ :v | v = val ifTrue: [ ^true ] ] ] ].
        ^false.
    )

    isEmpty = ( ^count = 0 )
    size = ( ^count )

    "Accessing"
    get: key = ( | idx e |
        idx := self indexForKey: key.
        e := table at: idx.
        e isNil ifTrue: [ ^nil ].
```

```
        ^e getValue: key.
    )

    at: key put: value = ( | idx |
        idx := self indexForKey: key.
        (table at: idx) isNil
            ifTrue: [
                table at: idx put:
                    (HashEntry newKey: key value: value next: nil).
                count := count + 1 ]
            ifFalse: [
                ((table at: idx) setKey: key value: value)
                    ifTrue: [ count := count + 1 ] ].
        "TODO: enlarge table, rehash if too large"
    )

    "TODO: some way to delete keys'd be nice..."

    "Enumerate"
    keys = ( | vec |
        vec := Vector new.
        table do: [ :ent |
            ent isNil ifFalse: [ ent keys do: [ :k | vec append: k ] ] ].
        ^vec.
    )

    values = ( | vec |
        vec := Vector new.
        table do: [ :ent |
            ent isNil ifFalse: [ ent values do: [ :v | vec append: v ] ] ].
        ^vec.
    )

    "Clearing"
    clear = ( table := Array new: 11.
              count := 0 )

    "Private"
    indexForKey: aKey = ( ^(aKey hashcode % table length) abs + 1 )

    ----------------

    "Allocation"
    new = ( | ht |
        ht := super new.
        ht clear.
        ^ht.
    )

)
```

Smalltalk/Integer.som

```
"

$Id: Integer.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Integer = (

    "Arithmetic"
    + argument  = primitive
    - argument  = primitive
    * argument  = primitive
    / argument  = primitive
    // argument = primitive
    % argument  = primitive "modulo with sign of divisor"
    rem: argument=primitive "modulo with sign of dividend"
    & argument  = primitive
    << argument = primitive
    >>> argument= primitive
    bitXor: argument = primitive
    abs         = ( ^(self < 0) ifTrue: (0 - self) ifFalse: self )
    sqrt        = primitive
    negated     = ( ^0 - self )

    raisedTo: exponent = (
        "Raise the receiver to the given exponent.
         Currently only positive integer exponents
         are fully supported."
        | output |
        output := 1.
        exponent asInteger
          timesRepeat: [ output := output * self ].
        ^ output
    )
```

```

    "Random numbers"
    atRandom = primitive

    "Comparing"
    =  argument = primitive
    == argument = ( ^self = argument )
    ~= argument = ( ^(self = argument) not )
    <  argument = primitive
    >  argument = ( ^(self >= argument) and: [ self <> argument ] )
    >= argument = ( ^(self < argument) not )
    <= argument = ( ^(self < argument) or: [ self = argument ] )
    negative    = ( ^self < 0 )
    between: a and: b = ( ^(self > a) and: [ self < b ] )

    "Converting"
    asString    = primitive
    as32BitSignedValue   = primitive " returns an int, with the value that a signed 32-bit integer would have"
    as32BitUnsignedValue = primitive " returns an int, with the value that a unsigned 32-bit integer would have"
    asDouble    = primitive
    asInteger   = ( ^self )

    hashcode    = ( ^self )

    "Iterating"
    to: limit do: block = (
        self to: limit by: 1 do: block
    )

    to: limit by: step do: block = (
        | i |
        i := self.
        [ i <= limit ] whileTrue: [ block value: i. i := i + step ]
    )

    downTo: limit do: block = (
        self downTo: limit by: 1 do: block
    )

    downTo: limit by: step do: block = (
        | i |
        i := self.
        [ i >= limit ] whileTrue: [ block value: i. i := i - step ]
    )

    "More Iterations"
    timesRepeat: block = (
        1 to: self do: [ :i | block value ]
    )

    "Range Creation"
    to: upper = (
        | range |
        range := Array new: upper - self + 1.
        self to: upper do: [ :i | range at: i put: i ].
        ^range
    )

    max: otherInt = (
```

```
        (self < otherInt) ifTrue: [^otherInt] ifFalse: [^self].
    )

    min: otherInt = (
        (self > otherInt) ifTrue: [^otherInt] ifFalse: [^self].
    )


    ----

    fromString: aString = primitive

)

```

Smalltalk/Metaclass.som

```
"

$Id: Metaclass.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Metaclass = Class ( )

```

Smalltalk/Method.som

```
"

$Id: Method.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Method = (

    "Meta Information"
    signature = primitive
    holder    = primitive

    "Printing"
    asString = ( ^self holder asString + '>>' + self signature asString )

    invokeOn: obj with: args = primitive

)
```

Smalltalk/Nil.som

```
"

$Id: Nil.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Nil = (

    "Converting"
    asString = ( ^'nil' )

    "Comparing"
    isNil    = ( ^true )
    notNil   = ( ^false )

    "Convenience"
    ifNil: aBlock = (^aBlock value)
    ifNotNil: aBlock = (^self)
    ifNil: goBlock ifNotNil: noGoBlock = (^goBlock value)

)
```

Smalltalk/Object.som

```
"

$Id: Object.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Object = nil (
    class      = primitive
    objectSize = primitive  "size in bytes"

    "Comparing"

    " If you override =, you MUST override hashcode as well.  The rule
      obj1 = obj2   =>  obj1 hashcode = obj2 hashcode
      must be valid for all objects, or Hashtable will not work"
    =  other    = ( ^self == other )
    <> argument = ( ^(self = argument) not )
    == other    = primitive
    ~= other    = (^ (self == other) not )
    isNil       = ( ^false )
    notNil      = ( ^true )

    "Converting"
    asString  = ( ^'instance of ' + (self class) )
    , element = ( ^(Vector new append: self) append: element )
    hashcode  = primitive

    "Evaluating"
    value     = ( ^self )

    "Convenience"
    ifNil: aBlock = (^self)
    ifNotNil: aBlock = (^aBlock value)
    ifNil: noGoBlock ifNotNil: goBlock = (^goBlock value)
```

```

    "Printing"
    print     = ( self asString print )
    println   = ( self print. system printNewline )

    "Debugging"
    inspect   = primitive
    halt      = primitive

    "Error handling"
    error: string = ( '' println. ('ERROR: ' + string) println. system exit: 1 )

    "Abstract method support"
    subclassResponsibility = (
        self error: 'This method is abstract and should be overridden'
    )

    "Error recovering"
    doesNotUnderstand: selector arguments: arguments = (
        self error:
            ('Method ' + selector + ' not found in class ' + self class name)
    )

    escapedBlock: block = (
        self error: 'Block has escaped and cannot be executed'
    )

    unknownGlobal: name = ( ^system resolve: name )

    "Reflection"
    respondsTo: aSymbol = (
        (self class hasMethod: aSymbol)
            ifTrue:  [ ^true ]
            ifFalse: [ | cls |
                cls := self class superclass.
                [ cls isNil ] whileFalse: [
                    (cls hasMethod: aSymbol)
                        ifTrue:  [ ^true ]
                        ifFalse: [ cls := cls superclass ] ].
                ^ false ]
    )

    perform: aSymbol = primitive
    perform: aSymbol withArguments: args = primitive

    perform: aSymbol inSuperclass: cls = primitive
    perform: aSymbol withArguments: args inSuperclass: cls = primitive

    instVarAt: idx          = primitive
    instVarAt: idx put: obj = primitive
    instVarNamed: sym       = primitive

)

```

Smalltalk/Pair.som

```
"

$Id: Pair.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Pair = (

    | key value |

    key = ( ^key )
    value = ( ^value )

    "Private"
    key: aKey = ( key := aKey )
    value: aValue = ( value := aValue )

    "Printing"
    print = ( '[' print. key print. '=>' print. value print. ']' print )
    println = ( self print. '' println )

    ----

    withKey: aKey andValue: aValue = (
        | newPair |
        newPair := super new.
        newPair key: aKey.
        newPair value: aValue.
        ^newPair
    )

)
```

Smalltalk/Primitive.som

```
"

$Id: Primitive.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Primitive = (

    signature = primitive
    holder    = primitive

    invokeOn: obj with: args = primitive

)

```

Smalltalk/Set.som

```
"

$Id: Set.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Set = (

    | items |

    = otherSet = (
        self size = otherSet size ifFalse: [^ false ].

        self do: [:item | (otherSet contains: item) ifFalse: [^ false]. ].

        ^ true.
    )

    add: anObject = (
        (self contains: anObject)
            ifFalse: [ items append: anObject ]
    )

    addAll: aCollection = (
        aCollection do: [:each |
            self add: each]
    )

    union: aCollection = (
        | new |
        new := Set new.
        new addAll: self.
        new addAll: aCollection.
        ^ new
```

```
    )

    intersection: aCollection = (
        | new |
        new := Set new.
        self do: [:it |
            (aCollection contains: it) ifTrue: [ new add: it ]].
        ^ new
    )

    - aCollection = ( "set difference"
        | new |
        new := Set new.
        self do: [:it |
            (aCollection contains: it) ifFalse: [ new add: it ]].
        ^ new
    )

    contains: anObject = (
        items do: [ :it | it == anObject ifTrue: [ ^true ] ].
        ^false
    )

    remove: anObject = (
        | newItems |
        newItems := Vector new.
        items do: [:it |
            it == anObject ifFalse: [ newItems append: it ] ].
        items := newItems
    )

    "Sets do not have the notion of ordering, but
     for convenience we provide those accessors"
    first = (
        ^items at: 1
    )

    isEmpty = (
        ^items isEmpty
    )

    "Iteration"
    do: block = ( items do: block )

    "Collection"
    collect: block = ( | coll |
        coll := Vector new.
        self do: [ :e | coll append: (block value: e) ].
        ^coll
    )

    "Printing"
    println = (
        '(' print.
        self do: [ :it | '(' print. it print. ')' print ].
        ')' println
    )

    asString = (
        | result |
        result := 'a Set('.
```

```
        items do: [:e | result := result + e asString + ', '].
        result := result + ')'.
        ^ result
    )

    size = (
        ^ items size
    )

    "Private"
    items: it = ( items := it )

    ----

    new = (
        | newSet |
        newSet := super new.
        newSet items: Vector new.
        ^newSet
    )

)
```

Smalltalk/String.som

```
"

$Id: String.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

String = (
    "Strings are immutable"

    "Concatenate: returns a new string object"
    concatenate: argument = primitive
    + argument            = ( ^self concatenate: argument asString )

    "Converting"
    asString = ( ^self )
    asSymbol = primitive
    hashcode = primitive

    "Info"
    length = primitive

    "Returns true if all characters in the string are whitespace.
     False otherwise, including for the empty string."
    isWhiteSpace = primitive

    "Returns true if all characters in the string are letters.
     False otherwise, including for the empty string."
    isLetters = primitive

    "Returns true if all characters in the string are digits.
     False otherwise, including for the empty string."
    isDigits = primitive

    "Comparing"
```

```
    = argument = primitive

    "substring: from 'start' to (and including) 'end'."
    primSubstringFrom: start to: end = primitive

    substringFrom: start to: end = (
        ((end <= self length) && (start > 0) && (start <= end))
            ifTrue: [^self primSubstringFrom: start to: end]
            ifFalse: [
                self error: 'Attempting to index string out of its bounds (start: ' + start asString + ' end: ' + end asString + ' length: ' + self length asString + ')' ]
    )

    beginsWith: prefix = (
        self length < prefix length ifTrue: [ ^ false ].

        1 to: prefix length do: [:i |
            ((self charAt: i) = (prefix charAt: i)) ifFalse: [ ^ false ].
        ].

        ^ true
    )

    endsWith: suffix = (
      | l sufL |
      l := self length.
      sufL := suffix length.
      l < sufL ifTrue: [ ^ false ].

      1 to: sufL do: [:i |
        (self charAt: l - sufL + i) = (suffix charAt: i) ifFalse: [ ^ false ]
      ].
      ^ true
    )

    asInteger = (
        ^ Integer fromString: self
    )

    charAt: argument = (
        ^self substringFrom: argument to: argument
    )

    indexOf: aString = (
      ^ self indexOf: aString startingAt: 1
    )

    indexOf: aString startingAt: start = (
      | l |
      l := aString length.
      start + l > (self length + 1) ifTrue: [ ^ -1 ].

      start to: self length - l + 1 do: [:i |
        (self primSubstringFrom: i to: i + l - 1) = aString ifTrue: [ ^ i ].
      ].

      ^ -1
    )
```

```

    split: split = (
      | start newStart result |
      self length < split length ifTrue: [ ^ Array new: self ].
      start := 1.
      result := Vector new.

      [start > 0] whileTrue: [
        newStart := self indexOf: split startingAt: start.
        newStart = -1
          ifTrue: [
            result append: (self primSubstringFrom: start to: self length).
            ^ result asArray ]
          ifFalse: [
            result append: (self primSubstringFrom: start to: newStart - 1).
            start := newStart + split length ] ].
      ^ result asArray
    )

    "Printing"
    print    = ( system printString: self )

)

```

Smalltalk/Symbol.som

```
"

$Id: Symbol.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Symbol = String (

    concatenate: argument = ( ^ (super concatenate: argument) asSymbol )

    "Converting"
    asString = primitive
    asSymbol = ( ^self )

    "Printing"
    print    = ( '#' print. super print )

)
```

Smalltalk/System.som

```
"

$Id: System.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

System = (

    "Accessing"
    global: name            = primitive
    global: name put: value = primitive
    hasGlobal: name         = primitive

    "Initializing"
    initialize: arguments = (
        | application |

        "Make sure we have got at least one argument"
        (arguments length < 1) ifTrue: [ 'No class specified' println. ^nil ].

        "Load the class with the specified name, create an instance of it, and
         run it. If there is more than only the class given on the command line,
         and the class has a method #run:, the arguments array is passed to it,
         otherwise, #run is sent."
        application := (self resolve: (arguments at: 1) asSymbol) new.

        ^ (application respondsTo: #run:)
            ifTrue:  [ application run: arguments ]
            ifFalse: [ application run ]
    )
```

```

    "File support"

    "Load a file identified by a path. Return content as a string"
    loadFile: fileName = primitive

    "Loading and resolving"
    load:    symbol = primitive
    resolve: symbol = (
        | class |

        "Check if we've already got the global"
        (self global: symbol) == nil ifFalse: [ ^self global: symbol ].

        "Try loading the class"
        class := self load: symbol.
        (class == nil) ifFalse: [
            ^class ].
        self error: 'Tried loading \'' + symbol + '\' as a class, but failed.'
    )

    "Exiting"
    exit: error  = primitive
    exit         = ( self exit: 0 )

    "Printing"
    printString: string = primitive
    printNewline        = primitive

    errorPrintln = ( self errorPrintln: '' )
    errorPrintln: string = primitive
    errorPrint: string  = primitive

    printStackTrace = primitive

    "Time"
    time  = primitive
    ticks = primitive      "returns the microseconds since start"

    "Force Garbage Collection"
    fullGC = primitive

    "To be implemented by SOM implementations that gather such statistics."
    gcStats = ( ^ #(
        0 "Total number of GCs"
        0 "Estimated total GC time in milliseconds"
        0 "Approximate number of allocated bytes of current thread") )
    totalCompilationTime = ( ^ 0 "Estimated total compilation time in milliseconds" )

    ----------------------------------

    "Allocation"
    new = ( self error: 'The system object is singular' )

)

```

Smalltalk/True.som

```
"

$Id: True.som 27 2009-07-31 11:17:53Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

True = Boolean (

    "Converting"
    asString       = ( ^'true' )

    "Conditional evaluation"
    ifTrue:  block = ( ^block value )
    ifFalse: block = ( ^nil )

    "Logical operations"
    not        = ( ^false )
    or: block  = ( ^true )
    and: block = ( ^block value )

)

```

Smalltalk/Vector.som

```
"

$Id: Vector.som 29 2009-07-31 11:28:44Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"FIXME: Implement pushFront and popFront..."

Vector = (

    | first last storage |

    "Accessing"
    at: index = (
        | storeIdx |
        storeIdx := index + first - 1.
        ^ self checkIndex: storeIdx ifValid: [ storage at: storeIdx ]
    )

    at: index put: value = (
        | storeIdx |
        storeIdx := index + first - 1.
        ^ self checkIndex: storeIdx ifValid: [ storage at: storeIdx put: value ]
    )

    first = ( ^ (self size > 0) ifTrue: [storage at: first] ifFalse: [nil] )
    last  = ( ^ (self size > 0) ifTrue: [storage at: last - 1] ifFalse: [nil] )

    "Iterating"
    do: block = (
        first to: last - 1 do: [ :i | block value: (storage at: i) ]
```

```
    )

    doIndexes: block = (
        1 to: last - first do: block
    )

    "Adding"
    , element = ( ^self append: element )

    append: element = (
        (last > storage length) ifTrue: [
            "Need to expand capacity first"
            | newStorage |
            newStorage := Array new: (2 * storage length).
            storage doIndexes: [ :i | newStorage at: i put: (storage at: i) ].
            storage := newStorage. ].

        storage at: last put: element.
        last := last + 1.
        ^self
    )

    appendAll: collection = (
        collection do: [:e |
            self append: e ]
    )

    "Removing"
    remove = (
        | value |
        self isEmpty ifTrue: [ self error: 'Vector: Attempting to remove the last element from an empty Vector' ].
        last := last - 1.
        value := storage at: last.
        storage at: last put: nil.
        ^ value
    )

    remove: object = (
        | newArray newLast found |
        newArray := Array new: self capacity.
        newLast := 1.
        found := false.

        self do: [ :it |
          (it == object)
              ifTrue:  [ found := true ]
              ifFalse: [
                  newArray at: newLast put: it.
                  newLast := newLast + 1.
              ]
        ].

        storage := newArray.
        last := newLast.
        first := 1.
        ^found
    )

    contains: anObject = (
```

```
        self do: [ :element |
            element = anObject ifTrue: [ ^ true ] ].
        ^ false
    )

    "If anObject is in vector, return index of first occurrence.
     If it isn't, return -1."
    indexOf: anObject = (
        first to: last - 1 do: [ :i |
            (storage at: i) = anObject ifTrue: [ ^ i - first + 1 ] ].
        ^ -1
    )

    "Printing"
    println = (
        '(' print.
        self do: [ :it | '(' print. it print. ')' print ].
        ')' println
    )

    "Sizing"
    isEmpty  = ( ^last = first   )
    size     = ( ^last - first   )
    capacity = ( ^storage length )

    "Conversion"
    asArray = ( | arr |
        arr := Array new: self size.
        self doIndexes: [ :i | arr at: i put: (self at: i) ].
        ^arr
    )

    "Private"
    initialize: size = (
        storage := Array new: size.
        first := 1.
        last  := 1.
    )

    checkIndex: index ifValid: block = (
        ^ ((first <= index) && (index < last)
            ifTrue: [ block value ]
            ifFalse: [
                self error:
                    'Vector[' + first asString + '..' + last asString +
                    ']: Index ' + index asString + ' out of bounds' ])
    )

    "DeltaBlue"
    removeFirst = (
        | value |
        self isEmpty ifTrue: [ self error: 'Vector: Attempting to remove the first element from an empty Vector' ].
        value := storage at: first.
        storage at: first put: nil.
        first := first + 1.
        ^ value
    )

    "Conversion"
    asSet = (
```

```
        ^ Set new addAll: self
    )

    ----------------------------

    "Allocation"
    new              = ( ^ self new: 50 )
    new: initialSize = ( ^ super new initialize: initialSize )

    with: elem = (
        | newVector |
        newVector := self new: 1.
        newVector append: elem.
        ^ newVector
    )

)
```

SomSom/src/compiler/BytecodeGenerator.som

```
BytecodeGenerator = (
  ----
  emitPop: mgenc = (
    self emit: mgenc bc: #pop
  )

  emit: mgenc pushArgument: idx in: ctx = (
    self emit: mgenc bc: #pushArgument with: idx and: ctx
  )

  emitReturnLocal: mgenc = (
    self emit: mgenc bc: #returnLocal
  )

  emitReturnNonLocal: mgenc = (
    self emit: mgenc bc: #returnNonLocal
  )

  emitDup: mgenc = (
    self emit: mgenc bc: #dup
  )

  emit: mgenc pushBlock: blockMethod = (
    self emit: mgenc bc: #pushBlock with: (mgenc addLiteralIfAbsent: blockMethod)
  )

  emit: mgenc pushLocal: idx in: ctx = (
    idx negative ifTrue: [ self error: 'pushLocal: ' + idx asString].
    self emit: mgenc bc: #pushLocal with: idx and: ctx
  )

  emit: mgenc pushField: aSymbol = (
    (mgenc hasField: aSymbol) ifFalse: [ self error: 'pushField: field unknown ' + aSymbol ].
    self emit: mgenc bc: #pushField with: (mgenc fieldIndex: aSymbol)
  )

  emit: mgenc pushGlobal: aSymbol = (
    self emit: mgenc bc: #pushGlobal with: (mgenc addLiteralIfAbsent: aSymbol)
  )

  emit: mgenc popArgument: idx in: ctx = (
    idx negative ifTrue: [ self error: 'popArgument: ' + idx asString].
    self emit: mgenc bc: #popArgument with: idx and: ctx
  )

  emit: mgenc popLocal: idx in: ctx = (
    idx negative ifTrue: [ self error: 'popLocal: ' + idx asString].
    self emit: mgenc bc: #popLocal with: idx and: ctx
  )

  emit: mgenc popField: aSymbol = (
    (mgenc hasField: aSymbol) ifFalse: [ self error: 'popField: field unknown ' + aSymbol ].
    self emit: mgenc bc: #popField with: (mgenc fieldIndex: aSymbol)
  )
```

```
  emit: mgenc superSend: aSymbol = (
    self emit: mgenc bc: #superSend with: (mgenc addLiteralIfAbsent: aSymbol)
  )

  emit: mgenc send: aSymbol = (
    self emit: mgenc bc: #send with: (mgenc addLiteralIfAbsent: aSymbol)
  )

  emit: mgenc pushConstant: anAbstractObject = (
    self emit: mgenc bc: #pushConstant with: (mgenc addLiteralIfAbsent: anAbstractObject)
  )

  emit: mgenc pushConstantIdx: anInteger = (
    self emit: mgenc bc: #pushConstant with: anInteger
  )

  emit: mgenc bc: aSymbol = (
    mgenc addBytecode: aSymbol.
  )

  emit: mgenc bc: aSymbol with: anInteger = (
    mgenc addBytecode: aSymbol.
    mgenc addBytecode: anInteger.
  )

  emit: mgenc bc: aSymbol with: anInteger and: otherInteger = (
    mgenc addBytecode: aSymbol.
    mgenc addBytecode: anInteger.
    mgenc addBytecode: otherInteger.
  )
)
```

```
ClassGenerationContext = (
  | universe
    name superName
    classSide
    classFields instanceFields
    classMethods instanceMethods |

  initalize: aUniverse = (
    universe := aUniverse.
    classSide := false.
    classFields := Vector new.
    instanceFields := Vector new.
    classMethods := Vector new.
    instanceMethods := Vector new.
  )

  name = (
    ^ name
  )

  name: aSSymbol = (
    name := aSSymbol
  )

  superName = (
    ^ superName
  )

  superName: aSymbol = (
    superName := aSymbol
  )

  instanceFieldsOfSuper: aSArrayOfFieldNames = (
    | numFields |
    numFields := aSArrayOfFieldNames numberOfIndexableFields.
    1 to: numFields do: [:i |
      instanceFields append: (aSArrayOfFieldNames indexableField: i) ]
  )

  classFieldsOfSuper: aSArrayOfFieldNames = (
    | numFields |
    numFields := aSArrayOfFieldNames numberOfIndexableFields.
    1 to: numFields do: [:i |
      classFields append: (aSArrayOfFieldNames indexableField: i) ]
  )

  addField: aSymbol = (
    classSide
      ifTrue: [classFields append: aSymbol]
      ifFalse: [instanceFields append: aSymbol]
  )

  hasField: aSymbol = (
    ^ classSide
      ifTrue: [classFields contains: aSymbol]
      ifFalse: [instanceFields contains: aSymbol]
  )

```

```
  fieldIndex: aSymbol = (
    ^ classSide
      ifTrue: [classFields indexOf: aSymbol]
      ifFalse: [instanceFields indexOf: aSymbol]
  )

  addMethod: anInvokable = (
    classSide
      ifTrue: [classMethods append: anInvokable]
      ifFalse: [instanceMethods append: anInvokable]
  )

  startClassSide = (
    classSide := true
  )

  assemble = (
    | ccname superClass resultClass superMClass result |
    "build class class name"
    ccname := name string + ' class'.

    "Load the super class"
    superClass := universe loadClass: superName.

    "Allocate the class of the resulting class"
    resultClass := universe newClass: universe metaclassClass.

    "Initialize the class of the resulting class"
    resultClass instanceFields: (universe newArrayFromVector: classFields).
    resultClass instanceInvokables: (universe newArrayFromVector: classMethods).
    resultClass name: (universe symbolFor: ccname).

    superMClass := superClass somClass.
    resultClass superClass: superMClass.

    "Allocate the resulting class"
    result := universe newClass: resultClass.

    "Initialize the resulting class"
    result name: name.
    result superClass: superClass.
    result instanceFields: (universe newArrayFromVector: instanceFields).
    result instanceInvokables: (universe newArrayFromVector: instanceMethods).

    ^ result
  )

  assembleSystemClass: systemClass = (
    | superMClass |
    systemClass instanceInvokables: (universe newArrayFromVector: instanceMethods).
    systemClass instanceFields: (universe newArrayFromVector: instanceFields).

    "class-bound == class-instance-bound"
    superMClass := systemClass somClass.
    superMClass instanceInvokables: (universe newArrayFromVector: classMethods).
```

```
    superMClass instanceFields: (universe newArrayFromVector: classFields).

    ^ systemClass
  )

  ----

  new: aUniverse = (
    ^ self new initalize: aUniverse
  )
)

```

```
Disassembler = (

  ----

  dump: cl in: universe = (
    1 to: cl numberOfInstanceInvokables do: [:i |
      | inv |
      inv := cl instanceInvokable: i.

      "output header and skip if the Invokable is a Primitive"
      Universe errorPrint: (cl name string + '>>' + inv signature string + ' = ').

      inv isPrimitive
        ifTrue: [ Universe errorPrintln: '<primitive>' ]
        ifFalse: [ self dumpMethod: inv indent: '\t' in: universe ] ]
  )

  dumpInvokable: inv in: universe = (
    | holderName |
    holderName := inv holder == nil
      ifTrue: ['nil']
      ifFalse: [inv holder name string].
    Universe errorPrint: (holderName + '>>#' + inv signature string + ' = ').
    inv isPrimitive
        ifTrue: [
          Universe errorPrint: '<primitive>: '.
          Universe errorPrintln: inv debugString ]
        ifFalse: [ self dumpMethod: inv indent: '\t' in: universe ]
  )

  dumpMethod: m indent: indent in: universe = (
    | b |
    Universe errorPrintln: '('.

    "output stack information"
    Universe errorPrintln: indent + '<' + m numberOfLocals + ' locals, '
        + m maximumNumberOfStackElements + ' stack, '
        + m numberOfBytecodes + ' bc_count>'.

    "output bytecodes"
    b := 1.
    [b <= m numberOfBytecodes] whileTrue: [
      | bytecode |
      Universe errorPrint: indent.

      b < 10 ifTrue: [ Universe errorPrint: ' ' ].
      b < 100 ifTrue: [ Universe errorPrint: ' ' ].

      Universe errorPrint: ' ' + b + ':'.

      "mnemonic"
      bytecode := m bytecode: b.
      Universe errorPrint: (Bytecodes paddedBytecodeName: bytecode) + ' '.

      "parameters (if any)"
```

```
      (Bytecodes length: bytecode) = 1
        ifTrue: [ Universe errorPrintln ]
        ifFalse: [ self dumpBytecode: bytecode idx: b method: m indent: indent in: universe ].

      b := b + (Bytecodes length: (m bytecode: b)) ].

    Universe errorPrintln: indent + ')'
  )

  dumpBytecode: bc idx: b method: m indent: indent in: universe = (
    bc == #pushLocal ifTrue: [
      Universe errorPrintln: 'local: ' + (m bytecode: b + 1) + ', context: ' + (m bytecode: b + 2).
      ^ self ].
    bc == #pushArgument ifTrue: [
      Universe errorPrintln: 'argument: ' + (m bytecode: b + 1) + ', context: ' + (m bytecode: b + 2).
      ^ self ].
    bc == #pushField ifTrue: [
      | idx fieldName |
      idx := m bytecode: b + 1.
      fieldName := (m holder instanceFields indexableField: idx) string.
      Universe errorPrintln: '(index: ' + idx + ') field: ' + fieldName.
      ^ self ].
    bc == #pushBlock ifTrue: [
      Universe errorPrint: '(block: (index: ' + (m bytecode: b + 1) + ') '.
      self dumpMethod: (m constant: b) indent: indent + '\t' in: universe.
      ^ self ].
    bc == #pushConstant ifTrue: [
      | constant |
      constant := m constant: b.
      Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ') value: '
        + '(' + (constant somClassIn: universe) name string + ') ' + constant debugString.
      ^ self ].
    bc == #pushGlobal ifTrue: [
      Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ') value: #' + (m constant: b) string.
      ^ self ].
    bc == #popLocal ifTrue: [
      Universe errorPrintln: 'local: ' + (m bytecode: b + 1) + ', context: ' + (m bytecode: b + 2).
      ^ self ].
    bc == #popArgument ifTrue: [
      Universe errorPrintln: 'argument: ' + (m bytecode: b + 1) + ', context: ' + (m bytecode: b + 2).
      ^ self ].
    bc == #pushField ifTrue: [
      | idx fieldName |
      idx := m bytecode: b + 1.
      fieldName := (m holder instanceFields indexableField: idx) string.
      Universe errorPrintln: '(index: ' + idx + ') field: ' + fieldName.
      ^ self ].
    bc == #send ifTrue: [
      Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ') signature: #' + (m constant: b) string.
      ^ self ].
    bc == #superSend ifTrue: [
```

```
      Universe errorPrintln: '(index: ' + (m bytecode: b + 1) + ') signature: #' + (m constant: b) string.
      ^ self ].

    Universe errorPrintln: '<unknown bytecode>'
  )
)

```

```
Lexer = (
  | fileContent state stateAfterPeek peekDone
    index
    sym text
    nextSym nextText
  |

  initialize: aString = (
    fileContent := aString.
    peekDone := false.
    index := 1.
  )

  isPeekDone = ( ^ peekDone )
  text = ( ^ text )

  currentTextContext = (
    | start end |
    start := (index - 50) max: 1.
    end := (index + 5) min: fileContent length.
    ^ fileContent substringFrom: start to: end
  )

  peek = (
    | savedSym savedText |
    peekDone ifTrue: [
      self error: 'SOM lexer cannot peek twice. Likely parser bug' ].

    savedSym := sym.
    savedText := text.
    self sym.

    nextSym := sym.
    nextText := text.
    peekDone := true.

    sym := savedSym.
    text := savedText.

    ^ nextSym
  )

  sym = (
    peekDone ifTrue: [
      peekDone := false.
      sym := nextSym.
      text := nextText.
      ^ sym ].

    self hasMoreInput ifFalse: [
      sym := #NONE.
      text := nil.
      ^ sym ].

    [self currentChar isWhiteSpace or: [self currentChar = '"']] whileTrue: [
      self skipWhiteSpace.
      self skipComment ].
```

```

    self currentChar = '\'' ifTrue: [
      ^ self lexString ].
    self currentChar = '[' ifTrue: [
      ^ self match: #newBlock ].
    self currentChar = ']' ifTrue: [
      ^ self match: #endBlock ].

    self currentChar = ':' ifTrue: [
      self nextChar = '='
        ifTrue: [
          index := index + 2.
          sym := #assign.
          text := ':=' ]
        ifFalse: [
          index := index + 1.
          sym := #colon.
          text := ':'
        ].
      ^ sym ].

    self currentChar = '(' ifTrue: [
      ^ self match: #newTerm ].
    self currentChar = ')' ifTrue: [
      ^ self match: #endTerm ].
    self currentChar = '#' ifTrue: [
      ^ self match: #pound ].
    self currentChar = '^' ifTrue: [
      ^ self match: #exit ].
    self currentChar = '.' ifTrue: [
      ^ self match: #period ].
    self currentChar = '-' ifTrue: [
      (self currentMatches: Lexer sepStr)
        ifTrue: [
          text := ''.
          [self currentChar = '-'] whileTrue: [
            text := text + self currentChar.
            index := index + 1 ].

          ^ sym := #separator ]
        ifFalse: [
          ^ self lexOperator ] ].
    (Lexer isOperator: self currentChar) ifTrue: [
      ^ self lexOperator ].
    (self currentMatches: Lexer primStr) ifTrue: [
      index := index + Lexer primStr length.
      text := Lexer primStr.
      ^ sym := #primitive ].
    self currentChar isLetters ifTrue: [
      text := ''.
      [self currentChar isLetters or: [self currentChar isDigits or: [self currentChar = '_']]] whileTrue: [
        text := text + self currentChar.
        index := index + 1 ].
      sym := #identifier.

      self currentChar = ':' ifTrue: [
        sym := #keyword.
        index := index + 1.
        text := text + ':'.
        self currentChar isLetters ifTrue: [
```

```
        sym := #keywordSequence.
        [self currentChar isLetters or: [self currentChar = ':']] whileTrue: [
          text := text + self currentChar.
          index := index + 1 ] ] ].
    ^ sym ].
  self currentChar isDigits ifTrue: [
    ^ self lexNumber ].

  text := self currentChar.
  ^ sym := #NONE
)

lexNumber = (
  | sawDecimalMark |
  sym := #integer.
  text := ''.

  sawDecimalMark := false.

  [self currentChar isDigits] whileTrue: [
    text := text + self currentChar.
    index := index + 1.
    (sawDecimalMark not and: [
      self currentChar = '.' and: [
        self nextChar isDigits]]) ifTrue: [
      sym := #double.
      text := text + self currentChar.
      index := index + 1 ] ].
  ^ sym
)

lexEscapeChar = (
  self currentChar = 't' ifTrue: [ ^ text := text + '\t' ].
  self currentChar = 'b' ifTrue: [ ^ text := text + '\b' ].
  self currentChar = 'n' ifTrue: [ ^ text := text + '\n' ].
  self currentChar = 'r' ifTrue: [ ^ text := text + '\r' ].
  self currentChar = 'f' ifTrue: [ ^ text := text + '\f' ].
  self currentChar = '\'' ifTrue: [ ^ text := text + '\'' ].
  self currentChar = '\\' ifTrue: [ ^ text := text + '\\' ].
  self currentChar = '0' ifTrue: [ ^ text := text + '\0' ].

  self error: 'Unknown escape sequence \\' + self currentChar
)

lexStringChar = (
  self currentChar = '\\'
    ifTrue: [
      index := index + 1.
      self lexEscapeChar.
      index := index + 1 ]
    ifFalse: [
      text := text + self currentChar.
      index := index + 1 ]
)

lexString = (
  sym := #string.
  text := ''.
  index := index + 1.

```

```
  [self currentChar = '\''] whileFalse: [
    self lexStringChar ].

  index := index + 1.
  ^ sym
)

lexOperator = (
  (Lexer isOperator: self nextChar) ifTrue: [
    text := ''.
    [Lexer isOperator: self currentChar] whileTrue: [
      text := text + self currentChar.
      index := index + 1 ].
    ^ sym := #operatorSequence ].
  self currentChar = '~' ifTrue: [
    ^ self match: #not ].
  self currentChar = '&' ifTrue: [
    ^ self match: #and ].
  self currentChar = '|' ifTrue: [
    ^ self match: #or ].
  self currentChar = '*' ifTrue: [
    ^ self match: #star ].
  self currentChar = '/' ifTrue: [
    ^ self match: #div ].
  self currentChar = '\\' ifTrue: [
    ^ self match: #mod ].
  self currentChar = '+' ifTrue: [
    ^ self match: #plus ].
  self currentChar = '=' ifTrue: [
    ^ self match: #equal ].
  self currentChar = '>' ifTrue: [
    ^ self match: #more ].
  self currentChar = '<' ifTrue: [
    ^ self match: #less ].
  self currentChar = ',' ifTrue: [
    ^ self match: #comma ].
  self currentChar = '@' ifTrue: [
    ^ self match: #at ].
  self currentChar = '%' ifTrue: [
    ^ self match: #per ].
  self currentChar = '-' ifTrue: [
    ^ self match: #minus ].
  self error: 'lexOperator ran out of options. This should not happen'
)

skipWhiteSpace = (
  [self currentChar isWhiteSpace] whileTrue: [
    index := index + 1 ]
)

skipComment = (
  self currentChar = '"'
    ifFalse: [ ^ self ].

  index := index + 1.

  [self currentChar = '"'] whileFalse: [
    index := index + 1 ].

  index := index + 1
)
```

```

  currentChar = (
    index > fileContent length ifTrue: [ ^ '\0' ].
    ^ fileContent charAt: index
  )

  nextChar = (
    (index + 1) > fileContent length ifTrue: [ ^ '\0' ].
    ^ fileContent charAt: index + 1
  )

  hasMoreInput = (
    ^ index <= fileContent length
  )

  currentMatches: str = (
    (index + str length) <= fileContent length ifFalse: [ ^ false ].
    ^ str = (fileContent substringFrom: index to: index - 1 + str length)
  )

  match: s = (
    sym := s.
    text := self currentChar.
    index := index + 1.
    ^ sym
  )

  ----

  new: aString = (
    ^ self new initialize: aString
  )

  isOperator: c = (
    c = '~' ifTrue: [ ^ true ].
    c = '&' ifTrue: [ ^ true ].
    c = '|' ifTrue: [ ^ true ].
    c = '*' ifTrue: [ ^ true ].
    c = '/' ifTrue: [ ^ true ].
    c = '\\' ifTrue: [ ^ true ].
    c = '+' ifTrue: [ ^ true ].
    c = '=' ifTrue: [ ^ true ].
    c = '>' ifTrue: [ ^ true ].
    c = '<' ifTrue: [ ^ true ].
    c = ',' ifTrue: [ ^ true ].
    c = '@' ifTrue: [ ^ true ].
    c = '%' ifTrue: [ ^ true ].
    c = '-' ifTrue: [ ^ true ].
    ^ false
  )

  sepStr  = ( ^ '----' )
  primStr = ( ^ 'primitive' )
)

```

```
MethodGenerationContext = (
  | holderGenc outerGenc
    arguments locals literals
    signature
    finished prim blockMethod
    bytecode |

  initializeWith: aHolderGenc and: aOuterGenc = (
    holderGenc := aHolderGenc.
    outerGenc := aOuterGenc.
    arguments := Vector new.
    locals := Vector new.
    literals := Vector new.
    finished := false.
    prim := false.
    blockMethod := false.
    bytecode := Vector new.
  )

  holder = (
    ^ holderGenc
  )

  signature: aSymbol = (
    signature := aSymbol
  )

  addArgument: aString = (
    arguments append: aString
  )

  addArgumentIfAbsent: aString = (
    (arguments contains: aString)
      ifTrue: [^ false].

    arguments append: aString.
    ^ true
  )

  numberOfArguments = (
    ^ arguments size
  )

  addLocalIfAbsent: aString = (
    (locals contains: aString)
      ifTrue: [^ false].

    locals append: aString.
    ^ true
  )

  addLiteralIfAbsent: anAbstractObject = (
    | idx |
    idx := literals indexOf: anAbstractObject.
    idx <> -1 ifTrue: [
      ^ idx ].

    ^ self addLiteral: anAbstractObject
```

```
  )

  addLiteral: anAbstractObject = (
    literals append: anAbstractObject.
    ^ literals size
  )

  updateLiteral: oldVal at: idx put: newVal = (
    (literals at: idx) == oldVal ifFalse: [
      self error: 'updateLiteral saw wrong oldVal, indicates bug in parser' ].
    literals at: idx put: newVal
  )

  findVar: var with: searchResult = (
    "searchResult: index, context, isArgument"
    searchResult at: 1 put: (locals indexOf: var).
    (searchResult at: 1) = -1 ifTrue: [
      searchResult at: 1 put: (arguments indexOf: var).
      (searchResult at: 1) = -1
        ifTrue: [
          outerGenc == nil
            ifTrue: [^ false]
            ifFalse: [
              searchResult at: 2 put: (searchResult at: 2) + 1.
              ^ outerGenc findVar: var with: searchResult ] ]
        ifFalse: [
          searchResult at: 3 put: true ] ].

    ^ true
  )

  markAsFinished = (
    finished := true
  )

  isFinished = (
    ^ finished
  )

  markAsPrimitive = (
    prim := true
  )

  isBlockMethod = (
    ^ blockMethod
  )

  markAsBlockMethod = (
    blockMethod := true
  )

  addBytecode: code = (
    bytecode append: code
  )

  removeLastBytecode = (
    bytecode remove
  )

  hasBytecodes = (
```

```
  ^ bytecode isEmpty not
)

hasField: aSymbol = (
  ^ holderGenc hasField: aSymbol
)

fieldIndex: aSymbol = (
  ^ holderGenc fieldIndex: aSymbol
)

assemble: universe = (
  prim
    ifTrue: [
      ^ SPrimitive emptyPrimitive: signature string in: universe ]
    ifFalse: [
      ^ self assembleMethod: universe ]
)

assembleMethod: universe = (
  | numLocals meth i |
  "create a method instance with the given number of bytecodes"
  numLocals := locals size.

  meth := universe newMethod: signature
      bc: bytecode asArray literals: literals asArray
      numLocals: numLocals maxStack: self computeStackDepth.

  "return the method - the holder field is to be set later on!"
  ^ meth
)

computeStackDepth = (
  | depth maxDepth i |
  depth := 0.
  maxDepth := 0.
  i := 1.

  [i <= bytecode size] whileTrue: [
    | bc |
    bc := bytecode at: i.

    (bc == #dup           or: [
     bc == #pushLocal     or: [
     bc == #pushArgument  or: [
     bc == #pushField     or: [
     bc == #pushBlock     or: [
     bc == #pushConstant  or: [
     bc == #pushGlobal ] ] ] ] ] ])  ifTrue: [
      depth := depth + 1 ] ifFalse: [

    (bc == #pop         or: [
     bc == #popLocal    or: [
     bc == #popArgument or: [
     bc == #popField ] ] ]) ifTrue: [
      depth := depth - 1 ] ifFalse: [

    (bc == #send or: [bc == #superSend]) ifTrue: [
      | sig |
      "these are special: they need to look at the number of
       arguments (extractable from the signature)"
```

```
        sig := literals at: (bytecode at: i + 1).
        depth := depth - sig numberOfSignatureArguments.
        depth := depth + 1 "return value" ] ] ].

      i := i + (Bytecodes length: bc).

      depth > maxDepth ifTrue: [
        maxDepth := depth ] ].

    ^ maxDepth
  )

  ----

  new: holderGenc = (
    ^ self new initializeWith: holderGenc and: nil
  )

  new: holderGenc with: outerGenc = (
    ^ self new initializeWith: holderGenc and: outerGenc
  )
)
```

```
Parser = (
  | lexer sym text nextSym filename cgenc universe bcGen |

  initializeWith: aString for: aFilename in: aUniverse = (
    filename := aFilename.
    lexer := Lexer new: aString.
    universe := aUniverse.
    cgenc := ClassGenerationContext new: universe.
    self takeSymbolFromLexer.

    "This is just a convenient abbreviation."
    bcGen := BytecodeGenerator.
  )

  takeSymbolFromLexer = (
    sym := lexer sym.
    text := lexer text.
    nextSym := #none.
  )

  classdef = (
    cgenc name: (universe symbolFor: text).
    self expect: #identifier.
    self expect: #equal.

    self superclass.

    self expect: #newTerm.
    self classBody.

    (self accept: #separator) ifTrue: [
      cgenc startClassSide.
      self classBody ].

    self expect: #endTerm.
    ^ cgenc
  )

  classBody = (
    self fields.
    [self symIsMethod] whileTrue: [
       | mgenc |
       mgenc := MethodGenerationContext new: cgenc.
       mgenc addArgument: 'self'.
       self method: mgenc.
       cgenc addMethod: (mgenc assemble: universe) ].
  )

  superclass = (
    | superName |
    sym == #identifier
      ifTrue: [
        superName := universe symbolFor: text.
        self accept: #identifier ]
      ifFalse: [
        superName := universe symbolFor: 'Object' ].

    cgenc superName: superName.
```

```
    superName string = 'nil' ifFalse: [
      self initializeFromSuperClass: superName ].
  )

  initializeFromSuperClass: superName = (
    | superClass |
    superClass := universe loadClass: superName.
    superClass == nil ifTrue: [
      self error: 'Was not able to load super class: ' + superName string + ' in ' + filename ].
    cgenc instanceFieldsOfSuper: superClass instanceFields.
    cgenc classFieldsOfSuper: superClass somClass instanceFields.
  )

  fields = (
    (self accept: #or) ifTrue: [
      [sym == #identifier] whileTrue: [
        | var |
        var := self variable.
        cgenc addField: (universe symbolFor: var) ].
      self expect: #or ]
  )

  method: mgenc = (
    self pattern: mgenc.
    self expect: #equal.

    sym == #primitive
      ifTrue: [
        mgenc markAsPrimitive.
        self primBlock ]
      ifFalse: [
        self methodBlock: mgenc ]
  )

  primBlock = (
    self expect: #primitive
  )

  pattern: mgenc = (
    sym == #identifier ifTrue: [
      ^ self unaryPattern: mgenc ].
    sym == #keyword ifTrue: [
      ^ self keywordPattern: mgenc ].
    self binaryPattern: mgenc
  )

  unaryPattern: mgenc = (
    mgenc signature: self unarySelector
  )

  binaryPattern: mgenc = (
    mgenc signature: self binarySelector.
    mgenc addArgumentIfAbsent: self argument
  )

  keywordPattern: mgenc = (
    | kw |
    kw := ''.
```

```
    [sym == #keyword] whileTrue: [
      kw := kw + self keyword.
      mgenc addArgumentIfAbsent: self argument ].

    mgenc signature: (universe symbolFor: kw)
  )

  methodBlock: mgenc = (
    self expect: #newTerm.

    self blockContents: mgenc.

    " if no return has been generated so far, we can be sure there was no . (dot)
      terminating the last expression, so the last expression's value must be
      popped off the stack and a ^self be generated "
    mgenc isFinished ifFalse: [
      bcGen emitPop: mgenc.
      bcGen emit: mgenc pushArgument: 1 in: 0.
      bcGen emitReturnLocal: mgenc.
      mgenc markAsFinished ].

    self expect: #endTerm.
  )

  blockContents: mgenc = (
    (self accept: #or) ifTrue: [
      self locals: mgenc.
      self expect: #or ].
    self blockBody: mgenc sawPeriod: false
  )

  locals: mgenc = (
    [sym == #identifier] whileTrue: [
      mgenc addLocalIfAbsent: self variable ]
  )

  blockBody: mgenc sawPeriod: seenPeriod = (
    (self accept: #exit) ifTrue: [
      ^ self result: mgenc ].

    sym == #endBlock ifTrue: [
      seenPeriod ifTrue: [
        "a POP has been generated which must be elided (blocks always
         return the value of the last expression, regardless of
         whether it was terminated with a . or not)"
        mgenc removeLastBytecode ].

      (mgenc isBlockMethod and: [ mgenc hasBytecodes not ]) ifTrue: [
        | nilSym |
        "if the block is empty, we need to return nil"
        nilSym := universe symbolFor: 'nil'.
        bcGen emit: mgenc pushGlobal: nilSym. ].

      bcGen emitReturnLocal: mgenc.
      mgenc markAsFinished.
      ^ self ].

    sym == #endTerm ifTrue: [
      "it does not matter whether a period has been seen, as the end of
```

```
      the method has been found (EndTerm) - so it is safe to emit a
      'return self'"
     bcGen emit: mgenc pushArgument: 1 in: 0.
     bcGen emitReturnLocal: mgenc.
     mgenc markAsFinished.
     ^ self ].

   self expression: mgenc.
   (self accept: #period) ifTrue: [
     bcGen emitPop: mgenc.
     self blockBody: mgenc sawPeriod: true ]
 )

 unarySelector = (
   ^ universe symbolFor: self identifier
 )

 binarySelector = (
   | s |
   s := text.

   (self accept: #operatorSequence) or: [
   (self acceptOneOf: Parser singleOpSyms) or: [
       self expect: #none ] ].

   ^ universe symbolFor: s
 )

 variable = (
   ^ self identifier
 )

 argument = (
   ^ self variable
 )

 identifier = (
   | s |
   s := text.
   (self accept: #primitive)
     ifFalse: [self expect: #identifier].
   ^ s
 )

 keyword = (
   | s |
   s := text.
   self expect: #keyword.
   ^ s
 )

 string = (
   | s |
   s := text.
   self expect: #string.
   ^ s
 )

 selector = (
   (sym == #operatorSequence or: [self symIn: Parser singleOpSyms])
     ifTrue: [^ self binarySelector].
```

```
  (sym == #keyword or: [sym == #keywordSequence])
    ifTrue: [^ self keywordSelector].

  ^ self unarySelector
)

keywordSelector = (
  | s |
  s := text.
  self expectOneOf: Parser keywordSelectorSyms.
  ^ universe symbolFor: s
)

result: mgenc = (
  self expression: mgenc.

  mgenc isBlockMethod
    ifTrue: [bcGen emitReturnNonLocal: mgenc ]
    ifFalse: [bcGen emitReturnLocal: mgenc ].
  mgenc markAsFinished.

  self accept: #period
)

expression: mgenc = (
  self peekForNextSymbolFromLexer.

  nextSym == #assign
    ifTrue: [self assignation: mgenc]
    ifFalse: [self evaluation: mgenc]
)

assignation: mgenc = (
  | variables |
  variables := Vector new.

  self assignments: mgenc to: variables.
  self evaluation: mgenc.

  variables do: [:v | bcGen emitDup: mgenc ].
  variables do: [:v | self gen: mgenc popVariable: v ]
)

assignments: mgenc to: variables = (
  sym == #identifier ifTrue: [
    variables append: (self assignment: mgenc).
    self peekForNextSymbolFromLexer.
    nextSym == #assign ifTrue: [
      self assignments: mgenc to: variables ] ]
)

assignment: mgenc = (
  | v |
  v := self variable.
  self expect: #assign.
  ^ v
)

evaluation: mgenc = (
  | superSend |
  superSend := self primary: mgenc.
```

```
    self symIsMethod ifTrue: [
      self messages: mgenc with: superSend ]
  )

  primary: mgenc = (
    | superSend |
    superSend := false.

    sym == #identifier ifTrue: [
      | v |
      v := self variable.
      v = 'super' ifTrue: [
        superSend := true.
        " sends to super, but pushes self as receiver"
        v := 'self' ].

      self gen: mgenc pushVariable: v.
      ^ superSend ].

    sym == #newTerm ifTrue: [
      self nestedTerm: mgenc.
      ^ superSend ].

    sym == #newBlock ifTrue: [
      | bgenc blockMethod |
      bgenc := MethodGenerationContext new: mgenc holder with: mgenc.
      bgenc markAsBlockMethod.

      self nestedBlock: bgenc.

      blockMethod := bgenc assembleMethod: universe.
      bcGen emit: mgenc pushBlock: blockMethod.
      ^ superSend ].

    self literal: mgenc.
    ^ superSend
  )

  messages: mgenc with: superSend = (
    sym == #identifier ifTrue: [
      "only the first message in a sequence can be a super send"
      self unaryMessage: mgenc with: superSend.

      [sym == #identifier] whileTrue: [
        self unaryMessage: mgenc with: false ].

      [sym == #operatorSequence or: [self symIn: Parser binaryOpSyms]] whileTrue: [
        self binaryMessage: mgenc with: false ].

      sym == #keyword ifTrue: [
        self keywordMessage: mgenc with: false ].

      ^ self ].

    (sym == #operatorSequence or: [self symIn: Parser binaryOpSyms]) ifTrue: [
      self binaryMessage: mgenc with: superSend.

      [sym == #operatorSequence or: [self symIn: Parser binaryOpSyms]] whileTrue: [
```

```
        self binaryMessage: mgenc with: false ].

      sym == #keyword ifTrue: [
        self keywordMessage: mgenc with: false ].

      ^ self ].

    self keywordMessage: mgenc with: superSend
  )

  unaryMessage: mgenc with: superSend = (
    | msg |
    msg := self unarySelector.

    superSend ifTrue: [ bcGen emit: mgenc superSend: msg ]
              ifFalse: [ bcGen emit: mgenc send: msg ]
  )

  binaryMessage: mgenc with: superSend = (
    | msg |
    msg := self binarySelector.
    self binaryOperand: mgenc.

    superSend ifTrue: [ bcGen emit: mgenc superSend: msg ]
              ifFalse: [ bcGen emit: mgenc send: msg ]
  )

  binaryOperand: mgenc = (
    | superSend |
    superSend := self primary: mgenc.

    [sym == #identifier] whileTrue: [
      self unaryMessage: mgenc with: superSend.
      superSend := false ].

    ^ superSend
  )

  keywordMessage: mgenc with: superSend = (
    | kw msg |
    kw := self keyword.
    self formula: mgenc.

    [sym == #keyword] whileTrue: [
      kw := kw + self keyword.
      self formula: mgenc ].

    msg := universe symbolFor: kw.
    superSend ifTrue: [ bcGen emit: mgenc superSend: msg ]
              ifFalse: [ bcGen emit: mgenc send: msg ]
  )

  formula: mgenc = (
    | superSend |
    superSend := self binaryOperand: mgenc.

    "only the first message in a sequence can be a super send"
    [sym == #operatorSequence or: [self symIn: Parser binaryOpSyms]] whileTrue: [
        self binaryMessage: mgenc with: superSend.
        superSend := false ].
```

```
  )

  nestedTerm: mgenc = (
    self expect: #newTerm.
    self expression: mgenc.
    self expect: #endTerm.
  )

  nestedBlock: mgenc = (
    | blockSig argSize |
    mgenc addArgumentIfAbsent: '$block self'.

    self expect: #newBlock.

    sym == #colon ifTrue: [
      self blockPattern: mgenc ].

    "generate block signature"
    blockSig := '$block method'.
    argSize := mgenc numberOfArguments.
    (argSize - 1) timesRepeat: [
      blockSig := blockSig + ':' ].

    mgenc signature: (universe symbolFor: blockSig).

    self blockContents: mgenc.

    "if no return has been generated, we can be sure that the last expression
     in the block was not terminated by ., and can generate a return"
    mgenc isFinished ifFalse: [
      bcGen emitReturnLocal: mgenc.
      mgenc markAsFinished ].

    self expect: #endBlock
  )

  blockPattern: mgenc = (
    self blockArguments: mgenc.
    self expect: #or.
  )

  blockArguments: mgenc = (
    self expect: #colon.
    mgenc addArgumentIfAbsent: self argument.

    [sym == #colon] whileTrue: [
      self expect: #colon.
      mgenc addArgumentIfAbsent: self argument ]
  )

  literal: mgenc = (
    sym == #pound ifTrue: [
      self peekForNextSymbolFromLexerIfNecessary.
      nextSym == #newTerm
        ifTrue: [ self literalArray: mgenc ]
        ifFalse: [ self literalSymbol: mgenc ].
      ^ self ].

    sym == #string ifTrue: [
      self literalString: mgenc.
```

```
      ^ self ].

    self literalNumber: mgenc
  )

  literalArray: mgenc = (
    | arrayClassName arraySizePlaceholder
      newMessage atPutMessage arraySizeLiteralIndex i |
    self expect: #pound.
    self expect: #newTerm.

    arrayClassName := universe symbolFor: 'Array'.
    arraySizePlaceholder := universe symbolFor: 'ArraySizeLiteralPlaceholder'.
    newMessage := universe symbolFor: 'new:'.
    atPutMessage := universe symbolFor: 'at:put:'.

    "need the array size at a know idx so that we don't need a second pass
     over the array elements"
    arraySizeLiteralIndex := mgenc addLiteral: arraySizePlaceholder.

    "create empty array"
    bcGen emit: mgenc pushGlobal: arrayClassName.
    bcGen emit: mgenc pushConstantIdx: arraySizeLiteralIndex.
    bcGen emit: mgenc send: newMessage.

    i := 1.

    [sym == #endTerm] whileFalse: [
      | pushIndex |
      pushIndex := universe newInteger: i.
      bcGen emit: mgenc pushConstant: pushIndex.
      self literal: mgenc.
      bcGen emit: mgenc send: atPutMessage.
      i := i + 1 ].

    "replace the placeholder with the actual array size"
    mgenc updateLiteral: arraySizePlaceholder at: arraySizeLiteralIndex put: (universe newInteger: i - 1).
    self expect: #endTerm.
  )

  literalSymbol: mgenc = (
    | symb |
    self expect: #pound.
    sym == #string
      ifTrue: [
        | s |
        s := self string.
        symb := universe symbolFor: s ]
      ifFalse: [
        symb := self selector ].
    bcGen emit: mgenc pushConstant: symb
  )

  literalString: mgenc = (
    | s str |
    s := self string.
    str := universe newString: s.
    bcGen emit: mgenc pushConstant: str
  )
```

```

  literalNumber: mgenc = (
    | lit |
    sym == #minus
      ifTrue: [lit := self negativeDecimal]
      ifFalse: [lit := self literalDecimal: false].

    bcGen emit: mgenc pushConstant: lit
  )

  negativeDecimal = (
    self expect: #minus.
    ^ self literalDecimal: true
  )

  literalDecimal: isNegative = (
    sym == #integer
      ifTrue: [^ self literalInteger: isNegative]
      ifFalse: [^ self literalDouble: isNegative]
  )

  literalInteger: isNegative = (
    | i |
    i := Integer fromString: text.
    isNegative ifTrue: [
      i := i negated].

    self expect: #integer.
    ^ universe newInteger: i
  )

  literalDouble: isNegative = (
    | d |
    d := Double fromString: text.
    isNegative ifTrue: [
      d := d negated ].

    self expect: #double.
    ^ universe newDouble: d
  )

  accept: s = (
    sym == s ifTrue: [
      self takeSymbolFromLexer.
      ^ true ].
    ^ false
  )

  acceptOneOf: ss = (
    (self symIn: ss) ifTrue: [
      self takeSymbolFromLexer.
      ^ true ].
    ^ false
  )

  expect: s = (
    (self accept: s) ifTrue: [ ^ true ].

    self error: 'Parsing of ' + filename + ' failed, expected ' + s + ' but found ' + sym +
      ' (' + text + ').\nCurrent parser context: ' + lexer
```

```
currentTextContext
  )

  expectOneOf: ss = (
    | err |
    (self acceptOneOf: ss) ifTrue: [ ^ true ].

    err := 'Parsing of ' + filename + ' failed, expected one of '.

    ss do: [
      err := err + s + ', ' ].
    err := err + 'but found: ' + sym + ' (' + text + ').\nCurrent parser context: ' + lexer currentTextContext.

    self error: err
  )

  symIn: ss = (
    ^ ss contains: sym
  )

  symIsMethod = (
    sym == #identifier       ifTrue: [^ true].
    sym == #keyword          ifTrue: [^ true].
    sym == #operatorSequence ifTrue: [^ true].
    (self symIn: Parser binaryOpSyms) ifTrue: [^ true].
    ^ false
  )

  peekForNextSymbolFromLexer = (
    nextSym := lexer peek
  )

  peekForNextSymbolFromLexerIfNecessary = (
    lexer isPeekDone ifFalse: [
      self peekForNextSymbolFromLexer ]
  )

  gen: mgenc popVariable: var = (
    | searchResult |
    "Needs to determine whether the variable that is to be popped off the stack
     is a local variable, argument, or object field.
     This is done by examining all available lexical contexts, starting with
     the innermost (i.e., the one represented by mgenc)."

    "index, context, isArgument"
    searchResult := Array with: 0 with: 0 with: false. "TODO support: #(0 0 false)"

    (mgenc findVar: var with: searchResult)
      ifTrue: [
        (searchResult at: 3) "isArgument"
          ifTrue: [bcGen emit: mgenc popArgument: (searchResult at: 1) in: (searchResult at: 2)]
          ifFalse: [bcGen emit: mgenc popLocal: (searchResult at: 1) in: (searchResult at: 2)]
      ]
      ifFalse: [
        | varSym |
```

```
        varSym := universe symbolFor: var.
        (mgenc hasField: varSym) ifFalse: [
          ^ self error: 'Write to variable with the name ' + var + ', but there is no variable or field defined with this name' ].
        bcGen emit: mgenc popField: varSym ].
  )

  gen: mgenc pushVariable: var = (
    "Needs to determine whether the variable to be pushed on the stack
     is a local variable, argument, or object field.
     This is done by examining all available lexical contexts, starting with
     the innermost (i.e., the one represented by mgenc)."

    "index, context, isArgument"
    | searchResult |
    searchResult := Array with: 0 with: 0 with: false. "TODO support: #(0 0 false)"

    (mgenc findVar: var with: searchResult)
      ifTrue: [
        (searchResult at: 3) "isArgument"
          ifTrue: [
            bcGen emit: mgenc pushArgument: (searchResult at: 1) in: (searchResult at: 2) ]
          ifFalse: [
            bcGen emit: mgenc pushLocal: (searchResult at: 1) in: (searchResult at: 2) ] ]
      ifFalse: [
        | varSym |
        varSym := universe symbolFor: var.
        (mgenc hasField: varSym)
          ifTrue: [
            bcGen emit: mgenc pushField: varSym ]
          ifFalse: [
            bcGen emit: mgenc pushGlobal: varSym ] ]
  )


  ----
  | singleOpSyms binaryOpSyms keywordSelectorSyms |

  singleOpSyms = (
    singleOpSyms == nil ifTrue: [
      singleOpSyms := #(#not #and #or #star #div #mod #plus #equal
                        #more #less #comma #at #per #minus #none) ].
    ^ singleOpSyms
  )

  binaryOpSyms = (
    binaryOpSyms == nil ifTrue: [
      binaryOpSyms := #(#or #comma #minus #equal #not #and #or #star
                        #div #mod #plus #equal #more #less #comma #at
                        #per #none) ].
    ^ binaryOpSyms
  )

  keywordSelectorSyms = (
    keywordSelectorSyms == nil ifTrue: [
      keywordSelectorSyms := #(#keyword #keywordSequence) ].
    ^ keywordSelectorSyms
```

```
  )

  newWith: aString for: aFilename in: universe = (
    ^ self new initializeWith: aString for: aFilename in: universe
  )

  load: aFileName in: universe = (
    | fileContent |
    fileContent := system loadFile: aFileName.
    fileContent == nil ifTrue: [ ^ nil ].

    ^ self new initializeWith: fileContent for: aFileName in: universe
  )
)
```

```
SourcecodeCompiler = (
  ----
  compileClass: path name: fileName into: systemClass in: universe = (
    | fname parser result cname |
    fname := path + '/' + fileName + '.som'.

    parser := Parser load: fname in: universe.
    parser ifNil: [ ^ nil ].

    result := self compile: parser into: systemClass.

    cname := result name string.

    fileName ~= cname ifTrue: [
      self error: 'File name ' + fname
          + ' does not match class name (' + cname + ') in it.' ].
    ^ result
  )

  compileClass: stmt into: systemClass in: universe = (
    | parser |
    parser := Parser newWith: stmt for: '$string$' in: universe.
    ^ self compile: parser into: systemClass.
  )

  compile: parser into: systemClass = (
    | cgc |
    cgc := parser classdef.

    systemClass == nil
      ifTrue: [ ^ cgc assemble ]
      ifFalse: [ ^ cgc assembleSystemClass: systemClass ]
  )
)
```

SomSom/src/interpreter/Bytecodes.som

```
Bytecodes = (
  ----

  length: bytecode = (
    bytecode == #halt           ifTrue: [ ^ 1 ].
    bytecode == #dup            ifTrue: [ ^ 1 ].
    bytecode == #pushLocal      ifTrue: [ ^ 3 ].
    bytecode == #pushArgument   ifTrue: [ ^ 3 ].
    bytecode == #pushField      ifTrue: [ ^ 2 ].
    bytecode == #pushBlock      ifTrue: [ ^ 2 ].
    bytecode == #pushConstant   ifTrue: [ ^ 2 ].
    bytecode == #pushGlobal     ifTrue: [ ^ 2 ].
    bytecode == #pop            ifTrue: [ ^ 1 ].
    bytecode == #popLocal       ifTrue: [ ^ 3 ].
    bytecode == #popArgument    ifTrue: [ ^ 3 ].
    bytecode == #popField       ifTrue: [ ^ 2 ].
    bytecode == #send           ifTrue: [ ^ 2 ].
    bytecode == #superSend      ifTrue: [ ^ 2 ].
    bytecode == #returnLocal    ifTrue: [ ^ 1 ].
    bytecode == #returnNonLocal ifTrue: [ ^ 1 ].

    self error: 'Unknown bytecode' + bytecode asString
  )

  paddedBytecodeName: bytecodeSymbol = (
    | max padded |
    max := #returnNonLocal length.
    padded := bytecodeSymbol asString.
    [padded length < max] whileTrue: [
      padded := padded + ' ' ].
    ^ padded
  )
)
```

SomSom/src/interpreter/Frame.som

```
Frame = (
"
Frame layout:
+-----------------+
| Arguments       | 1
+-----------------+
| Local Variables | <-- localOffset
+-----------------+
| Stack           | <-- stackPointer
| ...             |
+-----------------+
"
|
  "Points at the top element"
  stackPointer
  bytecodeIndex

  "the offset at which local variables start"
  localOffset

  method
  context
  previousFrame
  stack
|
  initialize: nilObject previous: prevFrame context: contextFrame method: aSMethod maxStack: stackElements = (
    previousFrame := prevFrame.
    context := contextFrame.
    method := aSMethod.
    stack := Array new: stackElements withAll: nilObject.

    "Reset the stack pointer and the bytecode index"
    self resetStackPointer.
    bytecodeIndex := 1.
  )

  previousFrame = (
    ^ previousFrame
  )

  clearPreviousFrame = (
    previousFrame := nil
  )

  hasPreviousFrame = (
    ^ previousFrame ~= nil
  )

  isBootstrapFrame = (
    ^ self hasPreviousFrame not
  )

  context = (
    ^ context
  )

  hasContext = (
```

```
    ^ context ~= nil
  )

  context: level = (
    | frame |
    "Get the context frame at the given level"
    frame := self.

    "Iterate through the context chain until the given level is reached"
    [level > 0] whileTrue: [
      "Get the context of the current frame"
      frame := frame context.

      "Go to the next level"
      level := level - 1 ].

    ^ frame
  )

  outerContext = (
    | frame |
    "Compute the outer context of this frame"
    frame := self.

    "Iterate through the context chain until null is reached"
    [frame hasContext] whileTrue: [
      frame := frame context ].

    ^ frame
  )

  method = (
    ^ method
  )

  pop = (
    | sp |
    "Pop an object from the expression stack and return it"
    sp := stackPointer.
    stackPointer := stackPointer - 1.
    ^ stack at: sp.
  )

  push: aSAbstractObject = (
    "Push an object onto the expression stack"
    | sp |
    sp := stackPointer + 1.
    stack at: sp put: aSAbstractObject.
    stackPointer := sp
  )

  resetStackPointer = (
    "arguments are stored in front of local variables"
    localOffset := method numberOfArguments + 1.

    "Set the stack pointer to its initial value thereby clearing the stack"
    stackPointer := localOffset + method numberOfLocals - 1
  )

  bytecodeIndex = (
```

```
    "Get the current bytecode index for this frame"
    ^ bytecodeIndex
  )

  bytecodeIndex: value = (
    "Set the current bytecode index for this frame"
    bytecodeIndex := value
  )

  stackElement: index = (
    "Get the stack element with the given index
     (an index of zero yields the top element)"
    ^ stack at: stackPointer - index
  )

  stackElement: index put: value = (
    "Set the stack element with the given index to the given value
     (an index of zero yields the top element)"
    stack at: stackPointer - index put: value
  )

  local: index = (
    ^ stack at: localOffset + index - 1
  )

  local: index put: value = (
    stack at: localOffset + index - 1 put: value
  )

  local: index at: contextLevel = (
    "Get the local with the given index in the given context"
    ^ (self context: contextLevel) local: index
  )

  local: index at: contextLevel put: value = (
    "Set the local with the given index in the given context to the given value"
    (self context: contextLevel) local: index put: value
  )

  argument: index = (
    ^ stack at: index
  )

  argument: index put: value = (
    ^ stack at: index put: value
  )

  argument: index at: contextLevel = (
    | context |
    "Get the context"
    context := self context: contextLevel.

    "Get the argument with the given index"
    ^ context argument: index
  )

  argument: index at: contextLevel put: value = (
    | context |
    "Get the context"
    context := self context: contextLevel.
```

```

    "Set the argument with the given index to the given value"
    context argument: index put: value
  )

  copyArgumentsFrom: frame = (
    | numArgs |
    "copy arguments from frame:
     - arguments are at the top of the stack of frame.
     - copy them into the argument area of the current frame"
    numArgs := method numberOfArguments.
    0 to: numArgs - 1 do: [:i |
      stack at: i + 1 put: (frame stackElement: numArgs - 1 - i) ]
  )

  printStackTrace = (
    | className methodName |
    "Print a stack trace starting in this frame"
    self hasPreviousFrame ifTrue: [
      previousFrame printStackTrace ].

    className := method holder name string.
    methodName := method signature string.
    Universe println: className + '>>#' + methodName + ' @bi: ' + bytecodeIndex
  )

  ----

  new: nilObject previous: prevFrame context: contextFrame method: aSMethod maxStack: stackElements = (
    ^ self new initialize: nilObject previous: prevFrame context: contextFrame method: aSMethod maxStack: stackElements
  )
)

```

```
Interpreter = (
  | universe frame |

  initializeWith: aUniverse = (
    universe := aUniverse
  )

  doDup = (
    frame push: (frame stackElement: 0)
  )

  doPushLocal: bytecodeIndex = (
    frame push: (
        frame local: (frame method bytecode: bytecodeIndex + 1)
                 at: (frame method bytecode: bytecodeIndex + 2))
  )

  doPushArgument: bytecodeIndex = (
    frame push: (
        frame argument: (frame method bytecode: bytecodeIndex + 1)
                    at: (frame method bytecode: bytecodeIndex + 2))
  )

  doPushField: bytecodeIndex = (
    | fieldIndex |
    fieldIndex := frame method bytecode: bytecodeIndex + 1.

    "Push the field with the computed index onto the stack"
    frame push: (self getSelf field: fieldIndex)
  )

  doPushBlock: bytecodeIndex = (
    | blockMethod |
    blockMethod := frame method constant: bytecodeIndex.

    "Push a new block with the current frame as context onto the stack"
    frame push: (
        universe newBlock: blockMethod
                     with: frame
                  numArgs: blockMethod numberOfArguments)
  )

  doPushConstant: bytecodeIndex = (
    frame push: (frame method constant: bytecodeIndex)
  )

  doPushGlobal: bytecodeIndex = (
    | globalName global |
    globalName := frame method constant: bytecodeIndex.

    "Get the global from the universe"
    global := universe global: globalName.

    global ~= nil
      ifTrue: [ frame push: global  ]
      ifFalse: [
        "Send 'unknownGlobal:' to self"
        self getSelf sendUnknownGlobal: globalName in: universe using:
```

```
self ]
  )

  doPop = (
    frame pop
  )

  doPopLocal: bytecodeIndex = (
    frame local: (frame method bytecode: bytecodeIndex + 1)
              at: (frame method bytecode: bytecodeIndex + 2)
             put: frame pop
  )

  doPopArgument: bytecodeIndex = (
    frame argument: (frame method bytecode: bytecodeIndex + 1)
                 at: (frame method bytecode: bytecodeIndex + 2)
                put: frame pop
  )

  doPopField: bytecodeIndex = (
    | fieldIndex |
    fieldIndex := frame method bytecode: bytecodeIndex + 1.

    "Set the field with the computed index to the value popped from the stack"
    self getSelf field: fieldIndex put: frame pop
  )

  doSend: bytecodeIndex = (
    | signature numberOfArguments receiver |
    signature := frame method constant: bytecodeIndex.
    numberOfArguments := signature numberOfSignatureArguments.
    receiver := frame stackElement: numberOfArguments - 1.
    self send: signature rcvrClass: (receiver somClassIn: universe)
  )

  doSuperSend: bytecodeIndex = (
    | signature holderSuper invokable |
    signature := frame method constant: bytecodeIndex.

    "Send the message
     Lookup the invokable with the given signature"
    holderSuper := frame method holder superClass.
    invokable := holderSuper lookupInvokable: signature.

    self activate: invokable orDnu: signature
  )

  doReturnLocal = (
    | result |
    result := frame pop.

    "Pop the top frame and push the result"
    self popFrameAndPushResult: result
  )

  doReturnNonLocal = (
    | result context |
    result := frame pop.

    "Compute the context for the non-local return"
```

```
    context := frame outerContext.

    "Make sure the block context is still on the stack"
    context hasPreviousFrame ifFalse: [
      | block sender method numArgs |
      "Try to recover by sending 'escapedBlock:' to the sending object
       this can get a bit nasty when using nested blocks. In this case
       the 'sender' will be the surrounding block and not the object
       that actually sent the 'value' message."
      block := frame argument: 1 at: 0.
      sender := frame previousFrame outerContext argument: 1 at: 0.

      "pop the frame of the currently executing block..."
      self popFrame.

      "pop old arguments from stack"
      method := frame method.
      numArgs := method numberOfArguments.
      numArgs timesRepeat: [ frame pop ].

      "... and execute the escapedBlock message instead"
      sender sendEscapedBlock: block in: universe using: self.
      ^ self ].

    "Unwind the frames"
    [frame ~= context] whileTrue: [
      self popFrame ].

    self popFrameAndPushResult: result
  )

  start = (
    [true] whileTrue: [
      | bytecodeIndex bytecode bytecodeLength nextBytecodeIndex result |
      bytecodeIndex := frame bytecodeIndex.
      bytecode := frame method bytecode: bytecodeIndex.
      bytecodeLength := Bytecodes length: bytecode.
      nextBytecodeIndex := bytecodeIndex + bytecodeLength.
      frame bytecodeIndex: nextBytecodeIndex.

      result := self dispatch: bytecode idx: bytecodeIndex.
      result ~= nil
        ifTrue: [ ^ result ] ]
  )

  dispatch: bytecode idx: bytecodeIndex = (
    bytecode == #halt ifTrue: [
      ^ frame stackElement: 0 ].

    bytecode == #dup ifTrue: [
      self doDup.
      ^ nil ].

    bytecode == #pushLocal ifTrue: [
      self doPushLocal: bytecodeIndex.
      ^ nil ].

    bytecode == #pushArgument ifTrue: [
      self doPushArgument: bytecodeIndex.
      ^ nil ].
```

```
    bytecode == #pushField ifTrue: [
      self doPushField: bytecodeIndex.
      ^ nil ].

    bytecode == #pushBlock ifTrue: [
      self doPushBlock: bytecodeIndex.
      ^ nil ].

    bytecode == #pushConstant ifTrue: [
      self doPushConstant: bytecodeIndex.
      ^ nil ].

    bytecode == #pushGlobal ifTrue: [
      self doPushGlobal: bytecodeIndex.
      ^ nil ].

    bytecode == #pop ifTrue: [
      self doPop.
      ^ nil ].

    bytecode == #popLocal ifTrue: [
      self doPopLocal: bytecodeIndex.
      ^ nil ].

    bytecode == #popArgument ifTrue: [
      self doPopArgument: bytecodeIndex.
      ^ nil ].

    bytecode == #popField ifTrue: [
      self doPopField: bytecodeIndex.
      ^ nil ].

    bytecode == #send ifTrue: [
      self doSend: bytecodeIndex.
      ^ nil ].

    bytecode == #superSend ifTrue: [
      self doSuperSend: bytecodeIndex.
      ^ nil ].

    bytecode == #returnLocal ifTrue: [
      self doReturnLocal.
      ^ nil ].

    bytecode == #returnNonLocal ifTrue: [
      self doReturnNonLocal.
      ^ nil ].

    self error: 'Unknown bytecode' + bytecode asString
  )

  pushNewFrame: method with: contextFrame = (
    frame := universe newFrame: frame with: method with: contextFrame.
    ^ frame
  )

  pushNewFrame: method = (
    ^ self pushNewFrame: method with: nil
  )

  frame = (
```

```
    ^ frame
  )

  method = (
    ^ frame method
  )

  getSelf = (
    "Get the self object from the interpreter"
    ^ frame outerContext argument: 1 at: 0
  )

  send: selector rcvrClass: receiverClass = (
    | invokable |
    invokable := receiverClass lookupInvokable: selector.
    self activate: invokable orDnu: selector
  )

  activate: invokable orDnu: signature = (
    invokable ~= nil
        ifTrue: [
          "Invoke the invokable in the current frame"
          invokable invoke: frame using: self ]
        ifFalse: [
          | numberOfArguments receiver |
          numberOfArguments := signature numberOfSignatureArguments.
          receiver := frame stackElement: numberOfArguments - 1.
          receiver sendDoesNotUnderstand: signature in: universe using: self ]
  )

  popFrame = (
    | result |
    "Save a reference to the top frame"
    result := frame.

    "Pop the top frame from the frame stack"
    frame := frame previousFrame.

    "Destroy the previous pointer on the old top frame"
    result clearPreviousFrame.

    "Return the popped frame"
    ^ result
  )

  popFrameAndPushResult: result = (
    | numberOfArguments |
    "Pop the top frame from the interpreter frame stack and
     get the number of arguments"
    numberOfArguments := self popFrame method numberOfArguments.

    "Pop the arguments"
    numberOfArguments
      timesRepeat: [ frame pop ].

    frame push: result
  )

  ----

```

```
  new: universe = (
    ^ self new initializeWith: universe
  )
)

```

```
ArrayPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'at:' in: universe with: [:frame :interp |
        | idx rcvr |
        idx := frame pop.
        rcvr := frame pop.
        frame push: (rcvr indexableField: idx integer)]).

    self installInstancePrimitive: (
      SPrimitive new: 'at:put:' in: universe with: [:frame :interp |
        | rcvr idx value |
        value := frame pop.
        idx   := frame pop.
        rcvr  := frame stackElement: 0.
        rcvr indexableField: idx integer put: value ]).

    self installInstancePrimitive: (
      SPrimitive new: 'length' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.

        frame push: (universe newInteger: rcvr numberOfIndexableFields) ]).

    self installClassPrimitive: (
      SPrimitive new: 'new:' in: universe with: [:frame :interp |
        | arg |
        arg := frame pop.
        frame pop.

        frame push: (universe newArray: arg integer) ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

```
BlockPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'restart' in: universe with: [:frame :interp |
        frame bytecodeIndex: 1.
        frame resetStackPointer. ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

```
ClassPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'new' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInstance: rcvr) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'name' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr name ]).

    self installInstancePrimitive: (
      SPrimitive new: 'superclass' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr superClass ]).

    self installInstancePrimitive: (
      SPrimitive new: 'fields' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr instanceFields ]).

    self installInstancePrimitive: (
      SPrimitive new: 'methods' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr instanceInvokables ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

SomSom/src/primitives/DoublePrimitives.som

```
DoublePrimitives = Primitives (

  coerceToDouble: anSAbstractObject = (
    anSAbstractObject class == SDouble ifTrue: [
      ^ anSAbstractObject double ].
    anSAbstractObject class == SInteger ifTrue: [
      ^ anSAbstractObject integer asDouble ].
    self error: 'Cannot coerce ' + anSAbstractObject debugString + ' to double'.
  )

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'asString' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newString: rcvr double asString) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'asInteger' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: rcvr double asInteger) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'sqrt' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double sqrt) ]).

    self installInstancePrimitive: (
      SPrimitive new: '+' in: universe with: [:frame :interp |
        | rcvr arg |
        arg  := self coerceToDouble: frame pop.
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double + arg) ]).

    self installInstancePrimitive: (
      SPrimitive new: '-' in: universe with: [:frame :interp |
        | rcvr arg |
        arg  := self coerceToDouble: frame pop.
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double - arg) ]).

    self installInstancePrimitive: (
      SPrimitive new: '*' in: universe with: [:frame :interp |
        | rcvr arg |
        arg  := self coerceToDouble: frame pop.
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double * arg) ]).

    self installInstancePrimitive: (
      SPrimitive new: '//' in: universe with: [:frame :interp |
        | rcvr arg |
        arg  := self coerceToDouble: frame pop.
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double // arg) ]).

```

```
    self installInstancePrimitive: (
      SPrimitive new: '%' in: universe with: [:frame :interp |
        | rcvr arg |
        arg  := self coerceToDouble: frame pop.
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double % arg) ]).

    self installInstancePrimitive: (
      SPrimitive new: '=' in: universe with: [:frame :interp |
        | argument rcvr left |
        argument := frame pop.
        rcvr := frame pop.
        left := rcvr double.

        frame push: (self somBool: (
          (argument class == SDouble)
            ifTrue: [left = argument double]
            ifFalse: [
              argument class == SInteger
                ifTrue: [left = argument integer]
                ifFalse: [ false ] ])) ]).

    self installInstancePrimitive: (
      SPrimitive new: '<' in: universe with: [:frame :interp |
        | rcvr arg |
        arg  := self coerceToDouble: frame pop.
        rcvr := frame pop.
        frame push: (self somBool: rcvr double < arg) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'round' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: rcvr double round) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'sin' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double sin) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'cos' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr double cos) ]).

    self installClassPrimitive: (
      SPrimitive new: 'PositiveInfinity' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newDouble: Double PositiveInfinity) ]).

    self installClassPrimitive: (
      SPrimitive new: 'fromString:' in: universe with: [:frame :interp |
        | rcvr arg |
        arg  := frame pop.
        rcvr := frame pop.
        frame push: (universe newDouble: (Double fromString: arg string)) ]).
```

```
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)

```

SomSom/src/primitives/IntegerPrimitives.som

```
IntegerPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'asString' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newString: rcvr integer asString) ]).


    self installInstancePrimitive: (
      SPrimitive new: 'sqrt' in: universe with: [:frame :interp |
        | rcvr result |
        rcvr := frame pop.
        result := rcvr integer sqrt.
        result class == Integer
          ifTrue: [frame push: (universe newInteger: result)]
          ifFalse: [frame push: (universe newDouble: result)] ]).

    self installInstancePrimitive: (
      SPrimitive new: 'atRandom' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: rcvr integer atRandom) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'asDouble' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newDouble: rcvr integer asDouble) ]).

    self installInstancePrimitive: (
      SPrimitive new: '+' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.

        frame push: (argument class == SDouble
          ifTrue: [universe newDouble: rcvr integer + argument double]
          ifFalse: [universe newInteger: rcvr integer + argument integer]) ]).

    self installInstancePrimitive: (
      SPrimitive new: '-' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.

        frame push: (argument class == SDouble
          ifTrue: [universe newDouble: rcvr integer - argument double]
          ifFalse: [universe newInteger: rcvr integer - argument integer]) ]).

    self installInstancePrimitive: (
      SPrimitive new: '*' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.
```

```

        frame push: (argument class == SDouble
          ifTrue: [universe newDouble: rcvr integer * argument double]
          ifFalse: [universe newInteger: rcvr integer * argument integer]) ]).

    self installInstancePrimitive: (
      SPrimitive new: '//' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.

        frame push: (universe newDouble:
          (argument class == SDouble
            ifTrue: [rcvr integer // argument double]
            ifFalse: [rcvr integer // argument integer])) ]).

    self installInstancePrimitive: (
      SPrimitive new: '/' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.

        frame push: (universe newInteger:
          (argument class == SDouble
            ifTrue: [rcvr integer / argument double]
            ifFalse: [rcvr integer / argument integer])) ]).

    self installInstancePrimitive: (
      SPrimitive new: '%' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.

        frame push: (argument class == SDouble
            ifTrue: [universe newDouble: rcvr integer % argument double]
            ifFalse: [universe newInteger: rcvr integer % argument integer]) ]).


    self installInstancePrimitive: (
      SPrimitive new: 'rem:' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.
        frame push: (universe newInteger: (rcvr integer rem: argument integer))]).

    self installInstancePrimitive: (
      SPrimitive new: '&' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.
        frame push: (universe newInteger: (rcvr integer & argument integer)) ]).

    self installInstancePrimitive: (
      SPrimitive new: '=' in: universe with: [:frame :interp |
        | argument rcvr left |
        argument := frame pop.
        rcvr := frame pop.
```

```
        left := rcvr integer.

        frame push: (self somBool: (
          (argument class == SDouble)
            ifTrue: [left = argument double]
            ifFalse: [
              argument class == SInteger
                ifTrue: [left = argument integer]
                ifFalse: [ false ] ])) ]).

    self installInstancePrimitive: (
      SPrimitive new: '==' in: universe with: [:frame :interp |
        | argument rcvr left |
        argument := frame pop.
        rcvr := frame pop.
        left := rcvr integer.

        frame push: (self somBool: (
          argument class == SInteger
            ifTrue: [left = argument integer]
            ifFalse: [ false ] )) ]) dontWarn: true.

    self installInstancePrimitive: (
      SPrimitive new: '<<' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.
        frame push: (universe newInteger: (rcvr integer << argument integer)) ]).

    self installInstancePrimitive: (
      SPrimitive new: '<' in: universe with: [:frame :interp |
        | argument rcvr left |
        argument := frame pop.
        rcvr := frame pop.
        left := rcvr integer.

        frame push: (self somBool: (
          (argument class == SDouble)
            ifTrue: [left < argument double]
            ifFalse: [
              argument class == SInteger
                ifTrue: [left < argument integer]
                ifFalse: [ false ] ])) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'bitXor:' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.
        frame push: (universe newInteger: (rcvr integer bitXor: argument integer)) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'as32BitSignedValue' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: (rcvr integer as32BitSignedValue)) ]).

```

```
    self installInstancePrimitive: (
      SPrimitive new: 'as32BitUnsignedValue' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: (rcvr integer as32BitUnsignedValue)) ]).

    self installInstancePrimitive: (
      SPrimitive new: '>>>' in: universe with: [:frame :interp |
        | argument rcvr |
        argument := frame pop.
        rcvr := frame pop.
        frame push: (universe newInteger: (rcvr integer >>> argument integer)) ]).

    self installClassPrimitive: (
      SPrimitive new: 'fromString:' in: universe with: [:frame :interp |
        | argument |
        argument := frame pop.
        frame pop.
        frame push: (universe newInteger: (Integer fromString: argument string)) ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

```
MethodPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'holder' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr holder ]).

    self installInstancePrimitive: (
      SPrimitive new: 'signature' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr signature ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

SomSom/src/primitives/ObjectPrimitives.som

```
ObjectPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: '==' in: universe with: [:frame :interp |
        | op1 op2 |
        op1 := frame pop.
        op2 := frame pop.

        frame push: (self somBool: op1 == op2) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'hashcode' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: rcvr hashcode) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'objectSize' in: universe with: [:frame :interp |
        | rcvr size clazz |
        rcvr := frame pop.

        size := 1.
        clazz := (rcvr somClassIn: universe).
        clazz == SArray ifTrue: [
          size := size + rcvr numberOfIndexableFields ].
        clazz == SObject ifTrue: [
          size := size + rcvr numberOfFields ].

        frame push: (universe newInteger: size) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'perform:' in: universe with: [:frame :interp |
        | selector rcvr invokable |
        selector := frame pop.
        rcvr := frame stackElement: 0.

        invokable := (rcvr somClassIn: universe) lookupInvokable: selector.
        invokable invoke: frame using: interp ]).

    self installInstancePrimitive: (
      SPrimitive new: 'perform:inSuperclass:' in: universe with: [:frame :interp |
        | selector clazz invokable |
        clazz := frame pop.
        selector := frame pop.

        invokable := clazz lookupInvokable: selector.
        invokable invoke: frame using: interp ]).

    self installInstancePrimitive: (
      SPrimitive new: 'perform:withArguments:' in: universe with: [:frame :interp |
        | args selector rcvr invokable |
        args := frame pop.
        selector := frame pop.
        rcvr := frame stackElement: 0.

```

```
        1 to: args numberOfIndexableFields do: [:i |
          frame push: (args indexableField: i) ].

        invokable := (rcvr somClassIn: universe) lookupInvokable: selector.
        invokable invoke: frame using: interp ]).

    self installInstancePrimitive: (
      SPrimitive new: 'instVarAt:' in: universe with: [:frame :interp |
        | idx rcvr invokable |
        idx := frame pop.
        rcvr := frame pop.

        frame push: (rcvr field: idx integer) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'instVarAt:put:' in: universe with: [:frame :interp |
        | idx rcvr invokable val |
        val := frame pop.
        idx := frame pop.
        rcvr := frame stackElement: 0.

        rcvr field: idx integer put: val ]).

    self installInstancePrimitive: (
      SPrimitive new: 'class' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (rcvr somClassIn: universe) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'halt' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame stackElement: 0.
        rcvr halt ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

```
PrimitivePrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'holder' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr holder ]).

    self installInstancePrimitive: (
      SPrimitive new: 'signature' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: rcvr signature ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

```
Primitives = (
  | universe holder |
  initialize: aUniverse = (
    universe := aUniverse
  )

  installPrimitivesIn: aSClass = (
    holder := aSClass.

    self installPrimitives
  )

  installInstancePrimitive: prim = (
    self installInstancePrimitive: prim dontWarn: false
  )

  installInstancePrimitive: prim dontWarn: suppressWarning = (
    holder addInstancePrimitive: prim dontWarn: suppressWarning
  )

  installClassPrimitive: prim = (
    holder somClass addInstancePrimitive: prim
  )

  somBool: aBool = (
    ^ aBool
        ifTrue: [ universe trueObject ]
        ifFalse: [ universe falseObject ]
  )
)
```

SomSom/src/primitives/StringPrimitives.som

```
StringPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'concatenate:' in: universe with: [:frame :interp |
        | rcvr argument |
        argument := frame pop.
        rcvr := frame pop.

        frame push: (universe newString: rcvr string + argument string) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'asSymbol' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe symbolFor: rcvr string) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'length' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: rcvr string length) ]).

    self installInstancePrimitive: (
      SPrimitive new: '=' in: universe with: [:frame :interp |
        | rcvr argument argCls |
        argument := frame pop.
        rcvr := frame pop.

        argCls := argument somClassIn: universe.

        frame push: (self somBool:
          ((argCls == universe stringClass or: [argCls == universe symbolClass])
            and: [rcvr string = argument string])) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'primSubstringFrom:to:' in: universe with: [:frame :interp |
        | rcvr from to |
        to := frame pop.
        from := frame pop.
        rcvr := frame pop.

        frame push: (universe newString: (rcvr string primSubstringFrom: from integer to: to integer)) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'hashcode' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (universe newInteger: rcvr string hashcode) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'isWhiteSpace' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (self somBool: rcvr string isWhiteSpace) ]).
```

```

    self installInstancePrimitive: (
      SPrimitive new: 'isLetters' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (self somBool: rcvr string isLetters) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'isDigits' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.
        frame push: (self somBool: rcvr string isDigits) ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)

```

```
SymbolPrimitives = Primitives (

  installPrimitives = (
    self installInstancePrimitive: (
      SPrimitive new: 'asString' in: universe with: [:frame :interp |
        | rcvr |
        rcvr := frame pop.

        frame push: (universe newString: rcvr string) ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)

```

```
SystemPrimitives = Primitives (
  installPrimitives = (

    self installInstancePrimitive: (
      SPrimitive new: 'load:' in: universe with: [:frame :interp |
        | arg result |
        arg := frame pop.
        frame pop.

        result := universe loadClass: arg.

        frame push: (result == nil
          ifTrue: [ universe nilObject ]
          ifFalse: [ result ]) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'exit:' in: universe with: [:frame :interp |
        | error |
        frame printStackTrace.
        error := frame pop.
        universe exit: error integer ]).

    self installInstancePrimitive: (
      SPrimitive new: 'global:' in: universe with: [:frame :interp |
        | argument result |
        argument := frame pop.
        frame pop.

        result := universe global: argument.
        frame push: (result == nil
          ifTrue: [ universe nilObject ]
          ifFalse: [ result ]) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'global:put:' in: universe with: [:frame :interp |
        | value argument |
        value := frame pop.
        argument := frame pop.

        universe global: argument put: value ]).

    self installInstancePrimitive: (
      SPrimitive new: 'printString:' in: universe with: [:frame :interp |
        | arg |
        arg := frame pop.
        "Universe print: arg somClass asString."
        Universe print: arg string ]).

    self installInstancePrimitive: (
      SPrimitive new: 'printNewline' in: universe with: [:frame :interp |
        Universe println ]).

    self installInstancePrimitive: (
      SPrimitive new: 'errorPrint:' in: universe with: [:frame :interp |
        | arg |
        arg := frame pop.
        Universe errorPrint: arg string ]).

```

```
    self installInstancePrimitive: (
      SPrimitive new: 'errorPrintln:' in: universe with: [:frame :interp |
        | arg |
        arg := frame pop.
        Universe errorPrintln: arg string ]).

    self installInstancePrimitive: (
      SPrimitive new: 'time' in: universe with: [:frame :interp |
        | time |
        frame pop. "ignore"
        time := system time.
        frame push: (universe newInteger: time) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'ticks' in: universe with: [:frame :interp |
        | ticks |
        frame pop. "ignore"
        ticks := system ticks.
        frame push: (universe newInteger: ticks) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'gcStats' in: universe with: [:frame :interp |
        | gcStats arr |
        frame pop. "ignore"
        gcStats := system gcStats.
        arr := universe newArray: 3.
        arr indexableField: 1 put: (universe newInteger: (gcStats at: 1)).
        arr indexableField: 2 put: (universe newInteger: (gcStats at: 2)).
        arr indexableField: 3 put: (universe newInteger: (gcStats at: 3)).

        frame push: arr ]).

    self installInstancePrimitive: (
      SPrimitive new: 'totalCompilationTime' in: universe with: [:frame :interp |
        | cTime |
        frame pop. "ignore"
        cTime := system totalCompilationTime.
        frame push: (universe newInteger: cTime) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'fullGC' in: universe with: [:frame :interp |
        frame pop. "ignore"
        system fullGC.
        frame push: (universe trueObject) ]).

    self installInstancePrimitive: (
      SPrimitive new: 'loadFile:' in: universe with: [:frame :interp |
        | fileName content |
        fileName := frame pop.
        frame pop.

        content := system loadFile: fileName string.
        content == nil
          ifTrue: [frame push: universe nilObject]
          ifFalse: [frame push: (universe newString: content)] ]).

    self installInstancePrimitive: (
      SPrimitive new: 'printStackTrace' in: universe with: [:frame :interp |
        frame pop. "ignore"
```

```
        frame printStackTrace.
        frame push: (universe trueObject) ]).
  )

  ----

  new: universe = (
    ^ self new initialize: universe
  )
)
```

SomSom/src/vm/Main.som

```
Main = (
  run: args = (
    | u args2 |
    u := Universe new.
    args2 := args copyFrom: 2.
    u interpret: args2.
    u exit: 0.
  )
)
```

```
MainLoadAll = (
  loadAllSomSomSources = (
    #(
        #Bytecodes
        #Interpreter
        #Frame
        #SString
        #SObject
        #SAbstractObject
        #SSymbol
        #SBlock
        #SDouble
        #SArray
        #SPrimitive
        #SMethod
        #SClass
        #SInteger
        #SystemPrimitives
        #ClassPrimitives
        #DoublePrimitives
        #Primitives
        #IntegerPrimitives
        #PrimitivePrimitives
        #SymbolPrimitives
        #MethodPrimitives
        #StringPrimitives
        #BlockPrimitives
        #ObjectPrimitives
        #ArrayPrimitives
        #Main
        #Universe
        #MainLoadAll
        #Parser
        #BytecodeGenerator
        #ClassGenerationContext
        #Lexer
        #SourcecodeCompiler
        #Disassembler
        #MethodGenerationContext
    ) do: [:className |
      (system load: className) println. ]
  )
  run: args = (
    | u args2 |
    u := Universe new.
    args2 := args copyFrom: 2.
    u interpret: args2.
    u exit: 0.
  )
)
```

SomSom/src/vm/Universe.som

```
Universe = (
  | symbolTable globals classPath dumpBytecodes interpreter

    avoidExit
    lastExitCode
    exitBlock

    nilObject
    trueObject
    falseObject

    objectClass
    classClass
    metaclassClass

    nilClass
    integerClass
    arrayClass
    methodClass
    symbolClass
    primClass
    stringClass
    systemClass
    blockClass
    doubleClass

    trueClass
    falseClass
  |

  initialize = (
    symbolTable := Dictionary new.
    globals := Dictionary new.
    interpreter := Interpreter new: self.
    dumpBytecodes := false.
    avoidExit := false
  )

  initialize: aBool = (
    self initialize.
    avoidExit := aBool
  )

  exit: errorCode = (
    "Exit from the Java system"
    avoidExit
      ifTrue: [
        lastExitCode := errorCode.
        exitBlock value: errorCode ]
      ifFalse: [system exit: errorCode]
  )

  lastExitCode = (
    ^ lastExitCode
  )

  errorExit: message = (
    Universe errorPrintln: 'Runtime Error: ' + message.
```

```
    self exit: 1
  )

  nilObject   = ( ^ nilObject )
  trueObject  = ( ^ trueObject )
  falseObject = ( ^ falseObject )
  metaclassClass = ( ^ metaclassClass )

  arrayClass  = ( ^ arrayClass )
  blockClass  = ( ^ blockClass )
  doubleClass = ( ^ doubleClass )
  integerClass = ( ^ integerClass )
  methodClass = ( ^ methodClass )
  primClass = ( ^ primClass )
  stringClass = ( ^ stringClass )
  symbolClass = ( ^ symbolClass )

  defaultClassPath = (
    ^ #('.')
  )

  setupClassPath: cp = (
    | paths cps |
    "Create a new tokenizer to split up the string of directories"
    paths := cp split: ':'.

    cps := Vector new.
    cps appendAll: self defaultClassPath.
    cps appendAll: paths.

    classPath := cps asArray
  )

  handleArguments: args = (
    | gotClasspath remainingArgs cnt i sawOthers |
    gotClasspath := false.
    remainingArgs := Vector new.

    "read dash arguments only while we haven't seen other kind of arguments"
    sawOthers := false.

    i := 1.

    [i <= args length] whileTrue: [
      ((args at: i) = '-cp' and: sawOthers not)
        ifTrue: [
          i + 1 > args length ifTrue: [
            self printUsageAndExit ].
          self setupClassPath: (args at: i + 1).
          i := i + 1.
          gotClasspath := true ]
        ifFalse: [
          ((args at: i) = '-d' and: sawOthers not)
            ifTrue: [ dumpBytecodes := true ]
            ifFalse: [
              sawOthers := true.
              remainingArgs append: (args at: i) ] ].
        i := i + 1 ].

    gotClasspath ifFalse: [
```

```
      classPath := self defaultClassPath ].

    remainingArgs isEmpty ifFalse: [
      | split |
      split := self pathClassExtension: (remainingArgs at: 1).
      (split at: 1) = '' ifFalse: [
        classPath := classPath prependedWith: (split at: 1) ].
      remainingArgs at: 1 put: (split at: 2) ].

    ^ remainingArgs asArray
  )

  pathClassExtension: str = (
    | pathElements fileName parentPath nameParts |
    pathElements := str split: '/'.
    fileName := pathElements last.

    parentPath := ''.
    1 to: pathElements length - 1 do: [:i |
      parentPath = '' ifFalse: [
        parentPath := parentPath + '/' ].
      parentPath := parentPath + (pathElements at: i) ].

    nameParts := fileName split: '.'.
    ^ Array with: parentPath with: (nameParts at: 1)
  )

  interpret: args = (
    | remainingArgs result |
    remainingArgs := self handleArguments: args.
    result := self initializeInterpreter: remainingArgs.
    result class == SInteger
      ifTrue: [ ^ result integer ]
      ifFalse: [ ^ 1 ]
  )

  interpret: className with: selector = (
    | clazz initialize |
    self initializeObjectSystem.

    clazz := self loadClass: (self symbolFor: className).

    "Lookup the initialize invokable on the system class"
    initialize := (clazz somClassIn: self) lookupInvokable: (self symbolFor: selector).

    initialize == nil ifTrue: [
      self error: 'Lookup of ' + className + '>>#' + selector + ' failed' ].

    ^ self interpret: initialize in: clazz with: nil
  )

  initializeInterpreter: arguments = (
    | systemObject initialize argumentsArray |
    systemObject := self initializeObjectSystem.

    "Start the shell if no filename is given"
    arguments length == 0 ifTrue: [
      | shell |
      shell := Shell for: self using: interpreter.
```

```
      shell bootstrapMethod: self createBootstrapMethod.
      ^ shell start ].

    "Lookup the initialize invokable on the system class"
    initialize := systemClass lookupInvokable: (self symbolFor: 'initialize:').

    "Convert the arguments into an array"
    argumentsArray := self newArrayFromStrings: arguments.

    ^ self interpret: initialize in: systemObject with: argumentsArray
  )

  createBootstrapMethod = (
    | bootstrapMethod |
    "Create a fake bootstrap method to simplify later frame traversal"
    bootstrapMethod := self newMethod: (self symbolFor: 'bootstrap')
      bc: #(#halt) literals: #() numLocals: 0 maxStack: 2.

    bootstrapMethod holder: systemClass.
    ^ bootstrapMethod
  )

  interpret: invokable in: receiver with: arguments = (
    | bootstrapMethod bootstrapFrame |
    exitBlock := [:errorCode | ^ errorCode ].

    bootstrapMethod := self createBootstrapMethod.

    "Create a fake bootstrap frame with the system object on the stack"
    bootstrapFrame := interpreter pushNewFrame: bootstrapMethod.
    bootstrapFrame push: receiver.

    arguments ~= nil ifTrue: [
      bootstrapFrame push: arguments ].

    "Invoke the initialize invokable"
    invokable invoke: bootstrapFrame using: interpreter.

    "Start the interpreter"
    ^ interpreter start
  )

  initializeObjectSystem = (
    | trueSymbol falseSymbol systemObject |

    "Allocate the nil object"
    nilObject := SObject new.

    "Allocate the Metaclass classes"
    metaclassClass := self newMetaclassClass.

    "Allocate the rest of the system classes"
    objectClass := self newSystemClass.
    nilClass := self newSystemClass.
    classClass := self newSystemClass.
    arrayClass := self newSystemClass.
    symbolClass := self newSystemClass.
    methodClass := self newSystemClass.
    integerClass := self newSystemClass.
    primClass := self newSystemClass.
```

```
    stringClass := self newSystemClass.
    doubleClass := self newSystemClass.

    "Setup the class reference for the nil object"
    nilObject somClass: nilClass.

    "Initialize the system classes."
    self initializeSystemClass: objectClass superClass: nil name: 'Object'.
    self initializeSystemClass: classClass superClass: objectClass name: 'Class'.
    self initializeSystemClass: metaclassClass superClass: classClass name: 'Metaclass'.
    self initializeSystemClass: nilClass superClass: objectClass name: 'Nil'.
    self initializeSystemClass: arrayClass superClass: objectClass name: 'Array'.
    self initializeSystemClass: methodClass superClass: arrayClass name: 'Method'.
    self initializeSystemClass: stringClass superClass: objectClass name: 'String'.
    self initializeSystemClass: symbolClass superClass: stringClass name: 'Symbol'.
    self initializeSystemClass: integerClass superClass: objectClass name: 'Integer'.
    self initializeSystemClass: primClass superClass: objectClass name: 'Primitive'.
    self initializeSystemClass: doubleClass superClass: objectClass name: 'Double'.

    "Load methods and fields into the system classes"
    self loadSystemClass: objectClass.
    self loadSystemClass: classClass.
    self loadSystemClass: metaclassClass.
    self loadSystemClass: nilClass.
    self loadSystemClass: arrayClass.
    self loadSystemClass: methodClass.
    self loadSystemClass: symbolClass.
    self loadSystemClass: integerClass.
    self loadSystemClass: primClass.
    self loadSystemClass: stringClass.
    self loadSystemClass: doubleClass.

    "Fix up objectClass"
    objectClass superClass: nilObject.

    "Load the generic block class"
    blockClass := self loadClass: (self symbolFor: 'Block').

    "Setup the true and false objects"
    trueSymbol := self symbolFor: 'True'.
    trueClass := self loadClass: trueSymbol.
    trueObject := self newInstance: trueClass.

    falseSymbol := self symbolFor: 'False'.
    falseClass := self loadClass: falseSymbol.
    falseObject := self newInstance: falseClass.

    "Load the system class and create an instance of it"
    systemClass := self loadClass: (self symbolFor: 'System').
    systemObject := self newInstance: systemClass.
```

```
  "Put special objects and classes into the dictionary of globals"
  self global: (self symbolFor: 'nil') put: nilObject.
  self global: (self symbolFor: 'true') put: trueObject.
  self global: (self symbolFor: 'false') put: falseObject.
  self global: (self symbolFor: 'system') put: systemObject.
  self global: (self symbolFor: 'System') put: systemClass.
  self global: (self symbolFor: 'Block') put: blockClass.
  self global: trueSymbol  put: trueClass.
  self global: falseSymbol put: falseClass.
  ^ systemObject
)

symbolFor: aString = (
  | result |
  result := symbolTable at: aString.
  result == nil ifFalse: [
    ^ result ].

  ^ self newSymbol: aString
)

newArray: size = (
  ^ SArray new: size with: nilObject
)

newArrayFromStrings: strArray = (
  | sArr |
  sArr := self newArray: strArray length.
  1 to: strArray length do: [:i |
    sArr indexableField: i put: (self newString: (strArray at: i))].
  ^ sArr
)

newArrayFromVector: vector = (
  | result |
  "Allocate a new array with the same length as the list"
  result := self newArray: vector size.

  "Copy all elements from the list into the array"
  vector doIndexes: [:i |
    result indexableField: i put: (vector at: i) ].

  "Return the allocated and initialized array"
  ^ result
)

newBlock: method with: context numArgs: arguments = (
  ^ SBlock new: method in: context with: (self blockClass: arguments)
)

newClass: classClass = (
  | result |
  "Allocate a new class and set its class to be the given class class"
  result := SClass new: classClass numberOfInstanceFields in: self.
  result somClass: classClass.

  "Return the freshly allocated class"
  ^ result
)
```

```
  newFrame: previousFrame with: method with: contextFrame = (
    | length result |
    "Compute the maximum number of stack locations (including arguments,
     locals and extra buffer to support doesNotUnderstand) and set the number
     of indexable fields accordingly"
    length := method numberOfArguments
        + method numberOfLocals
        + method maximumNumberOfStackElements + 2.

    result := Frame new: nilObject previous: previousFrame context: contextFrame method: method maxStack: length.

    "Return the freshly allocated frame"
    ^ result
  )

  newSymbol: aString = (
    | result |
    result := SSymbol new: aString.
    symbolTable at: aString put: result.
    ^ result
  )

  newInstance: instanceClass = (
    | result |
    result := SObject new: instanceClass numberOfInstanceFields with: nilObject.
    result somClass: instanceClass.

    ^ result
  )

  newInteger: anInteger = (
    ^ SInteger for: anInteger
  )

  newDouble: aDouble = (
    ^ SDouble for: aDouble
  )

  newMetaclassClass = (
    | result |
    "Allocate the metaclass classes"
    result := SClass new: self.
    result somClass: (SClass new: self).

    "Setup the metaclass hierarchy"
    result somClass somClass: result.

    "Return the freshly allocated metaclass class"
    ^ result
  )

  newMethod: aSSymbol bc: bcArray literals: literalsArray numLocals: numLocals maxStack: maxStack = (
    ^ SMethod new: aSSymbol bc: bcArray literals: literalsArray numLocals: numLocals maxStack: maxStack
  )

  newString: aString = (
```

```
    ^ SString new: aString
  )

  newSystemClass = (
    | symbolClass |
    "Allocate the new system class"
    systemClass := SClass new: self.

    "Setup the metaclass hierarchy"
    systemClass somClass: (SClass new: self).
    systemClass somClass somClass: metaclassClass.

    "Return the freshly allocated system class"
    ^ systemClass
  )

  initializeSystemClass: systemClass superClass: superClass name: name = (
    "Initialize the superclass hierarchy"
    superClass ~= nil
      ifTrue: [
        systemClass superClass: superClass.
        systemClass somClass superClass: (superClass somClass) ]
      ifFalse: [
        systemClass somClass superClass: classClass ].

    "Initialize the array of instance fields"
    systemClass instanceFields: (self newArray: 0).
    systemClass somClass instanceFields: (self newArray: 0).

    "Initialize the array of instance invokables"
    systemClass instanceInvokables: (self newArray: 0).
    systemClass somClass instanceInvokables: (self newArray: 0).

    "Initialize the name of the system class"
    systemClass name: (self symbolFor: name).
    systemClass somClass name: (self symbolFor: name + ' class').

    "Insert the system class into the dictionary of globals"
    self global: systemClass name put: systemClass.
  )

  global: aSSymbol = (
    "Return the global with the given name if it's in the dictionary of globals"
    (self hasGlobal: aSSymbol) ifTrue: [
      ^ globals at: aSSymbol ].

    "Global not found"
    ^ nil
  )

  global: aSSymbol put: aSAbstractObject = (
    "Insert the given value into the dictionary of globals"
    globals at: aSSymbol put: aSAbstractObject
  )

  hasGlobal: aSSymbol = (
    "Returns if the universe has a value for the global of the given name"
    ^ globals containsKey: aSSymbol
  )
```

```
  blockClass: numberOfArguments = (
    | name result |
    "Determine the name of the block class with the given number of arguments"
    name := self symbolFor: 'Block' + numberOfArguments.

    "Lookup the block class in the dictionary of globals and return it"
    (self hasGlobal: name) ifTrue: [
      ^ self global: name ].

    result := self loadClass: name into: nil.

    "Add the appropriate value primitive to the block class"
    result addInstancePrimitive:
      (SBlock evaluationPrimitive: numberOfArguments in: self).

    self global: name put: result.
    ^ result
  )

  loadClass: name = (
    | result |
    "Check if the requested class is already in the dictionary of globals"
    (self hasGlobal: name) ifTrue: [
      ^ self global: name ].

    "Load the class"
    result := self loadClass: name into: nil.

    "Load primitives (if necessary) and return the resulting class"
    (result ~= nil and: [result hasPrimitives]) ifTrue: [
      result loadPrimitives ].

    self global: name put: result.
    ^ result
  )

  loadSystemClass: systemClass = (
    | result |
    "Load the system class"
    result := self loadClass: systemClass name into: systemClass.

    "Load primitives if necessary"
    result hasPrimitives ifTrue: [
      result loadPrimitives ].
  )

  loadClass: name into: systemClass = (
    "Try loading the class from all different paths"
    classPath do: [:cpEntry |
      | result |
      "Load the class from a file and return the loaded class"
      result := SourcecodeCompiler compileClass: cpEntry name: name string into: systemClass in: self.

      (result notNil and: dumpBytecodes) ifTrue: [
        Disassembler dump: result somClass in: self.
        Disassembler dump: result in: self ].

      result ifNotNil: [ ^ result ] ].

```

```
    "The class could not be found."
    ^ nil
  )

  loadShellClass: stmt = (
    | result |
    "Load the class from a stream and return the loaded class"
    result := SourcecodeCompiler compileClass: stmt into: nil in: self.
    dumpBytecodes ifTrue: [
      Disassembler dump: result in: self ].
    ^ result
  )

  ----

  new = (
    ^ super new initialize
  )

  new: avoidExit = (
    ^ super new initialize: avoidExit
  )

  errorPrint: msg = (
    system errorPrint: msg
  )

  errorPrintln: msg = (
    system errorPrintln: msg
  )

  errorPrintln = (
    system errorPrintln: ''
  )

  print: msg = (
    system errorPrint: msg
  )

  println: msg = (
    system errorPrintln: msg
  )

  println = (
    system errorPrintln
  )
)

```

SomSom/src/vmobjects/SAbstractObject.som

```
SAbstractObject = (
  send: selectorString with: arguments in: universe using: interpreter = (
    | selector invokable |
    selector := universe symbolFor: selectorString.

    interpreter frame push: self.

    arguments do: [:arg |
      interpreter frame push: arg ].

    invokable := (self somClassIn: universe) lookupInvokable: selector.

    invokable invoke: interpreter frame using: interpreter
  )

  sendDoesNotUnderstand: selector in: universe using: interpreter = (
    | numberOfArguments frame argumentsArray args |
    numberOfArguments := selector numberOfSignatureArguments.

    frame := interpreter frame.
    frame printStackTrace.

    "Allocate an array with enough room to hold all arguments
     except for the receiver, which is passed implicitly, as receiver of #dnu."
    argumentsArray := universe newArray: numberOfArguments - 1.

    "Remove all arguments and put them in the freshly allocated array"
    numberOfArguments - 1 downTo: 1 do: [:i |
      argumentsArray indexableField: i put: frame pop ].

    frame pop. "pop receiver"

    args := Array with: selector with: argumentsArray.
    self send: 'doesNotUnderstand:arguments:' with: args in: universe using: interpreter
  )

  sendUnknownGlobal: globalName in: universe using: interpreter = (
    | arguments |
    arguments := Array with: globalName.
    self send: 'unknownGlobal:' with: arguments in: universe using: interpreter
  )

  sendEscapedBlock: block in: universe using: interpreter = (
    | arguments |
    arguments := Array with: block.
    self send: 'escapedBlock:' with: arguments in: universe using: interpreter
  )
)
```

```
SArray = SAbstractObject (
  | indexableFields |

  initializeWith: length and: nilObject = (
    indexableFields := Array new: length withAll: nilObject.
  )

  somClassIn: universe = (
    ^ universe arrayClass
  )

  indexableField: idx = (
    ^ indexableFields at: idx
  )

  indexableField: idx put: val = (
    ^ indexableFields at: idx put: val
  )

  numberOfIndexableFields = (
    ^ indexableFields length
  )

  copyAndExtendWith: value in: universe = (
    | result newLength |
    newLength := indexableFields length + 1.
    result := universe newArray: newLength.

    self copyIndexableFieldsTo: result.

    result indexableField: newLength put: value.
    ^ result
  )

  copyIndexableFieldsTo: destination = (
    indexableFields doIndexes: [:i |
      destination indexableField: i put: (indexableFields at: i) ]
  )

  "For using in debugging tools such as the Diassembler"
  debugString = (
    | elems |
    elems := ''.
    indexableFields do: [:e |
      elems = '' ifTrue: [elems := e debugString]
                 ifFalse: [ elems := elems + ', ' + e debugString] ].
     ^ 'SArray(' + indexableFields length + '; ' + elems + ')' )

  ----

  new: length with: nilObject = (
    ^ self new initializeWith: length and: nilObject
  )
)
```

```
SBlock = SAbstractObject (
  | method context blockClass |

  initialize: aSMethod in: aContext with: aBlockClass = (
    method := aSMethod.
    context := aContext.
    blockClass := aBlockClass.
  )

  method = (
    ^ method
  )

  context = (
    ^ context
  )

  somClassIn: universe = (
    ^ blockClass
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SBlock(' + method asString + ')' )

  ----

  new: aSMethod in: aContext with: aBlockClass = (
    ^ self new initialize: aSMethod in: aContext with: aBlockClass
  )

  evaluationPrimitive: numberOfArguments in: universe = (
    ^ SPrimitive new: (self computeSignatureString: numberOfArguments)
                  in: universe
                with: [:frame :interp |
        | rcvr context newFrame |
        "Get the block (the receiver) from the stack"
        rcvr := frame stackElement: numberOfArguments - 1.

        "Get the context of the block"
        context := rcvr context.

        "Push a new frame and set its context to be the one specified in
         the block"
        newFrame := interp pushNewFrame: rcvr method with: context.
        newFrame copyArgumentsFrom: frame ]
  )

  computeSignatureString: numberOfArguments = (
    | signatureString |
    signatureString := 'value'.
    numberOfArguments > 1 ifTrue: [
      signatureString := signatureString + ':' ].

    "Add extra with: selector elements if necessary"
    2 to: numberOfArguments - 1 do: [:i |
        signatureString := signatureString + 'with:' ].

    ^ signatureString
```

)
)

SomSom/src/vmobjects/SClass.som

```
SClass = SObject (
  | universe
    superClass
    name
    instanceInvokables instanceFields|

  initialize: aUniverse = (
    universe := aUniverse
  )

  initialize: numberOfFields in: aUniverse = (
    super initialize: numberOfFields with: aUniverse nilObject.
    universe := aUniverse
  )

  superClass = (
    ^ superClass
  )

  superClass: aSClass = (
    superClass := aSClass
  )

  hasSuperClass = (
    ^ superClass ~= universe nilObject
  )

  name = (
    ^ name
  )

  name: aSSymbol = (
    name := aSSymbol
  )

  instanceFields = (
    ^ instanceFields
  )

  instanceFields: aSArray = (
    instanceFields := aSArray
  )

  instanceInvokables = (
    ^ instanceInvokables
  )

  instanceInvokables: aSArray = (
    instanceInvokables := aSArray.

    "Make sure this class is the holder of all invokables in the array"
    1 to: self numberOfInstanceInvokables do: [:i |
      (instanceInvokables indexableField: i) holder: self ]
  )

  numberOfInstanceInvokables = (
    ^ instanceInvokables numberOfIndexableFields
  )
```

```
  instanceInvokable: idx = (
    ^ instanceInvokables indexableField: idx
  )

  instanceInvokable: idx put: aSInvokable = (
    aSInvokable holder: self.
    instanceInvokables indexableField: idx put: aSInvokable
  )

  lookupInvokable: signature = (
    | invokable |

    "Lookup invokable with given signature in array of instance invokables"
    1 to: instanceInvokables numberOfIndexableFields do: [:i |
      "Get the next invokable in the instance invokable array"
      invokable := instanceInvokables indexableField: i.

      "Return the invokable if the signature matches"
      invokable signature == signature ifTrue: [
        ^ invokable ] ].

    "Traverse the super class chain by calling lookup on the super class"
    self hasSuperClass ifTrue: [
      invokable := superClass lookupInvokable: signature.
      invokable ~= nil ifTrue: [
        ^ invokable ] ].

    "Invokable not found"
    ^ nil
  )

  lookupFieldIndex: fieldName = (
    "Lookup field with given name in array of instance fields"

    self numberOfInstanceFields downTo: 1 do: [:i |
      "Return the current index if the name matches"
      fieldName == (self instanceFieldName: i)
        ifTrue: [ ^ i ] ].

    "Field not found"
    ^ -1
  )

  addInstanceInvokable: value = (
    "Add the given invokable to the array of instance invokables"
    1 to: self numberOfInstanceInvokables do: [:i |
      "Get the next invokable in the instance invokable array"
      | invokable |
      invokable := self instanceInvokable: i.

      "Replace the invokable with the given one if the signature matches"
      invokable signature == value signature ifTrue: [
        self instanceInvokable: i put: value.
        ^ false ] ].

    "Append the given method to the array of instance methods"
    instanceInvokables := instanceInvokables copyAndExtendWith: value in: universe.
    ^ true
```

```
  )

  addInstancePrimitive: value = (
    self addInstancePrimitive: value dontWarn: false
  )

  addInstancePrimitive: value dontWarn: suppressWarning = (
    value holder: self.
    ((self addInstanceInvokable: value) and: [suppressWarning not]) ifTrue: [
      Universe print: 'Warning: Primitive ' + value signature string.
      Universe println: ' is not in class definition for class ' + name string ]
  )

  instanceFieldName: index = (
    "Get the name of the instance field with the given index"
    index > self numberOfSuperInstanceFields
      ifTrue: [
        | idx |
        "Adjust the index to account for fields defined in the super class"
        idx := index - self numberOfSuperInstanceFields.

        "Return the symbol representing the instance fields name"
        ^ instanceFields indexableField: idx ]
      ifFalse: [
        "Ask the super class to return the name of the instance field"
        ^ superClass instanceFieldName: index ]
  )

  numberOfInstanceFields = (
    "Get the total number of instance fields in this class"
    ^ instanceFields numberOfIndexableFields + self numberOfSuperInstanceFields
  )

  numberOfSuperInstanceFields = (
    self hasSuperClass
      ifTrue: [ ^ self superClass numberOfInstanceFields ]
      ifFalse: [ ^ 0 ]
  )

  hasPrimitives = (
    "Lookup invokable with given signature in array of instance invokables"
    1 to: self numberOfInstanceInvokables do: [:i |
      "Get the next invokable in the instance invokable array"
      (self instanceInvokable: i) isPrimitive
        ifTrue: [ ^ true ] ].
    ^ false
  )

  loadPrimitives = (
    | className primsClass |
    className := (name string + 'Primitives') asSymbol.

    "Try loading the primitives"

    primsClass := system load: className.
    primsClass ~= nil
```

```
      ifTrue: [
        (primsClass new: universe) installPrimitivesIn: self ]
      ifFalse: [
        Universe println: 'Primitives class ' + className + ' not found' ]
  )


  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SClass(' + name string + ')' )

  ----

  new: universe = (
    ^ self new initialize: universe
  )

  new: numberOfFields in: universe = (
    ^ self new initialize: numberOfFields in: universe
  )
)
```

SomSom/src/vmobjects/SDouble.som

```
SDouble = SAbstractObject (
  | value |

  initialize: aDouble = (
    value := aDouble
  )

  double = ( ^ value )

  somClassIn: universe = (
    ^ universe doubleClass
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SDouble(' + value asString + ')' )

  ----

  for: aDouble = (
    ^ self new initialize: aDouble
  )
)
```

```
SInteger = SAbstractObject (
  | value |

  initialize: anInteger = (
    value := anInteger
  )

  integer = ( ^ value )

  somClassIn: universe = (
    ^ universe integerClass
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SInteger(' + value asString + ')' )

  ----

  "TODO: see whether it makes sense to have a cache"
  for: anInteger = (
    ^ self new initialize: anInteger
  )
)
```

SomSom/src/vmobjects/SMethod.som

```
SMethod = SAbstractObject (
  | signature
    holder
    bytecodes literals
    numberOfLocals maximumNumberOfStackElements |

  initializeWith: aSSymbol bc: bcArray literals: literalsArray numLocals: numLocals maxStack: maxStack = (
    signature := aSSymbol.
    bytecodes := bcArray.
    literals := literalsArray.
    numberOfLocals := numLocals.
    maximumNumberOfStackElements := maxStack.
  )

  isPrimitive = (
    ^ false
  )

  numberOfLocals = (
    ^ numberOfLocals
  )

  maximumNumberOfStackElements = (
    ^ maximumNumberOfStackElements
  )

  signature = (
    ^ signature
  )

  holder = (
    ^ holder
  )

  holder: value = (
    holder := value.

    literals == nil ifTrue: [ ^ self ].

    "Make sure all nested invokables have the same holder"
    literals do: [:l |
      (l class == SMethod or: [l class == SPrimitive]) ifTrue: [
        l holder: value ] ]
  )

  constant: bytecodeIndex = (
    "Get the constant associated to a given bytecode index"
    ^ literals at: (bytecodes at: bytecodeIndex + 1)
  )

  numberOfArguments = (
    "Get the number of arguments of this method"
    ^ signature numberOfSignatureArguments
  )

  numberOfBytecodes = (
    "Get the number of bytecodes in this method"
```

```
    ^ bytecodes length
  )

  bytecode: index = (
    "Get the bytecode at the given index"
    ^ bytecodes at: index
  )

  invoke: frame using: interpreter = (
    | newFrame |
    "Allocate and push a new frame on the interpreter stack"
    newFrame := interpreter pushNewFrame: self.
    newFrame copyArgumentsFrom: frame
  )

  somClassIn: universe = (
    ^ universe methodClass
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SMethod(' + holder name + '>>#' + signature string + ')' )

  ----

  new: aSSymbol bc: bcArray literals: literalsArray numLocals: numLocals maxStack: maxStack = (
    ^ self new
        initializeWith: aSSymbol bc: bcArray literals: literalsArray numLocals: numLocals maxStack: maxStack
  )

)
```

```
SObject = SAbstractObject (
  | fields clazz |

  initialize: numberOfFields with: nilObject = (
    fields := Array new: numberOfFields withAll: nilObject
  )

  somClass = (
    ^ clazz
  )

  somClass: aSClass = (
    clazz := aSClass
  )

  somClassIn: universe = (
    ^ clazz
  )

  fieldName: index = (
    "Get the name of the field with the given index"
    ^ clazz instanceFieldName: index
  )

  fieldIndex: name = (
    "Get the index for the field with the given name"
    ^ clazz lookupFieldIndex: name
  )

  numberOfFields = (
    "Get the number of fields in this object"
    ^ fields length
  )

  field: index = (
    "Get the field with the given index"
    ^ fields at: index
  )

  field: index put: value = (
    "Set the field with the given index to the given value"
    fields at: index put: value
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SObject(' + clazz name string + ')' )

  ----

  new: numberOfFields with: nilObject = (
    ^ self new initialize: numberOfFields with: nilObject
  )
)
```

SomSom/src/vmobjects/SPrimitive.som

```
SPrimitive = SAbstractObject (
  | signature holder isEmpty operation |

  initialize: aSSymbol with: aBlock = (
    signature := aSSymbol.
    isEmpty := false.
    operation := aBlock.
  )

  initializeEmpty: aSSymbol in: universe = (
    signature := aSSymbol.
    isEmpty := true.
    operation := [:frame :interp |
      | receiver msg |
      signature numberOfSignatureArguments timesRepeat: [
        receiver := frame pop ].
      msg := 'Undefined primitive ' + (receiver somClassIn: universe) name string +
        '>>#' + signature string + ' called'.
      self send: 'error:' with: (Array with: receiver with: (universe newString: msg))
           in: universe using: interp ].
  )

  isPrimitive = ( ^ true )

  signature = (
    ^ signature
  )

  holder = (
    ^ holder
  )

  holder: aSClass = (
    holder := aSClass
  )

  isEmpty = (
    ^ isEmpty
  )

  invoke: frame using: interp = (
    ^ operation value: frame with: interp
  )

  somClassIn: universe = (
    ^ universe primClass
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SPrimitive(' + holder name string + '>>#' + signature string + ')' )

  ----

  new: signatureString in: universe with: aBlock = (
    ^ self new initialize: (universe symbolFor: signatureString)
```

```
                with: aBlock
  )

  emptyPrimitive: signatureString in: universe = (
    ^ self new initializeEmpty: (universe symbolFor: signatureString)
                in: universe
  )
)

```

```
SString = SAbstractObject (
  | string |

  initializeWith: aString = (
    string := aString.
  )

  string = ( ^ string )

  somClassIn: universe = (
    ^ universe stringClass
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SString(' + string + ')' )

  ----

  new: aString = (
    ^ self new initializeWith: aString
  )
)
```

```
SSymbol = SString (
  | numSignatureArguments |

  initializeWith: aString = (
    super initializeWith: aString.
    numSignatureArguments := self determineNumberOfArguments
  )

  determineNumberOfArguments = (
    | numColons |
    self isBinarySignature ifTrue: [ ^ 2 ].

    numColons := 0.

    1 to: string length do: [:i |
      ':' = (string charAt: i) ifTrue: [
        numColons := numColons + 1 ] ].
    ^ numColons + 1
  )

  isBinarySignature = (
    1 to: string length do: [:i |
      (Lexer isOperator: (string charAt: i))
        ifFalse: [ ^ false ] ].
    ^ true
  )

  numberOfSignatureArguments = (
    ^ numSignatureArguments
  )

  somClassIn: universe = (
    ^ universe symbolClass
  )

  "For using in debugging tools such as the Diassembler"
  debugString = ( ^ 'SSymbol(' + string + ')' )

  ----

  new: aString = (
    ^ self new initializeWith: aString
  )
)
```

SomSom/tests/BasicInterpreterTests.som

```
BasicInterpreterTests = TestCase (
  | testClass testSel
    expectedResult resultType |

  testBasicInterpreter = (
    | arr |
    arr := BasicInterpreterTests nextTest.

    testClass := arr at: 1.
    testSel := arr at: 2.
    expectedResult := arr at: 3.
    resultType := arr at: 4.

    testSelector := '  ' + testClass + '>>#' + testSel.

    self doBasicInterpreterBehavior.
  )

  doBasicInterpreterBehavior = (
    | u actualResult |
    u := Universe new: true.
    u setupClassPath: 'Smalltalk:TestSuite/BasicInterpreterTests'.

    actualResult := u interpret: testClass with: testSel.

    self assertExpectedEqualsSOMValue: actualResult.
  )

  assertExpectedEqualsSOMValue: actualResult = (
    resultType ~= actualResult class ifTrue: [
      self signalFailure: 'Unexpected result type: ' + actualResult debugString.
      ^ self ].

    resultType == SInteger ifTrue: [
      self assert: expectedResult equals: actualResult integer.
      ^ self ].

    resultType == SDouble ifTrue: [
      "TODO: allow for small errors/inaccuracies"
      self assert: expectedResult equals: actualResult double.
      ^ self ].

    resultType == SClass ifTrue: [
      self assert: expectedResult equals: actualResult name string.
      ^ self ].

    resultType == SSymbol ifTrue: [
      self assert: expectedResult equals: actualResult string.
      ^ self ].

    self signalFailure: 'resultType not currently supported: ' + resultType name string
  )

  ----

  | basicTests next |
```

```

  tests = (
    | tests |
    next := 1.
    basicTests := Vector new.
    tests := Vector new.

    self setupBasicTest.
    self setupBasicTest2.

    basicTests size timesRepeat: [
      tests append: (self for: #testBasicInterpreter) ].
    ^ tests
  )

  nextTest = (
    | test |
    test := basicTests at: next.
    next := next + 1.
    ^ test
  )

  setupBasicTest = (
    self c: 'MethodCall'  t: 'test' e: 42 t: SInteger.
    self c: 'MethodCall'  t: 'test2' e: 42 t: SInteger.

    self c: 'NonLocalReturn'  t: 'test1' e: 42 t: SInteger.
    self c: 'NonLocalReturn'  t: 'test2' e: 43 t: SInteger.
    self c: 'NonLocalReturn'  t: 'test3' e:  3 t: SInteger.
    self c: 'NonLocalReturn'  t: 'test4' e: 42 t: SInteger.
    self c: 'NonLocalReturn'  t: 'test5' e: 22 t: SInteger.

    self c: 'Blocks'  t: 'testArg1' e: 42 t: SInteger.
    self c: 'Blocks'  t: 'testArg2' e: 77 t: SInteger.
    self c: 'Blocks'  t: 'testArgAndLocal'   e: 8 t: SInteger.
    self c: 'Blocks'  t: 'testArgAndContext' e: 8 t: SInteger.
    self c: 'Blocks'  t: 'testEmptyZeroArg'  e: 1 t: SInteger.
    self c: 'Blocks'  t: 'testEmptyOneArg'   e: 1 t: SInteger.
    self c: 'Blocks'  t: 'testEmptyTwoArg'   e: 1 t: SInteger.

    self c: 'Return'  t: 'testReturnSelf' e: 'Return' t: SClass.
    self c: 'Return'  t: 'testReturnSelfImplicitly' e: 'Return' t: SClass.
    self c: 'Return'  t: 'testNoReturnReturnsSelf' e: 'Return' t: SClass.
    self c: 'Return'  t: 'testBlockReturnsImplicitlyLastValue' e: 4 t: SInteger.

    self c: 'IfTrueIfFalse'  t: 'test'  e: 42 t: SInteger.
    self c: 'IfTrueIfFalse'  t: 'test2' e: 33 t: SInteger.
    self c: 'IfTrueIfFalse'  t: 'test3' e:  4 t: SInteger.

    self c: 'IfTrueIfFalse'  t: 'testIfTrueTrueResult'   e: 'Integer' t: SClass.
    self c: 'IfTrueIfFalse'  t: 'testIfTrueFalseResult'  e: 'Nil'     t: SClass.
    self c: 'IfTrueIfFalse'  t: 'testIfFalseTrueResult'  e: 'Nil'     t: SClass.
    self c: 'IfTrueIfFalse'  t: 'testIfFalseFalseResult' e: 'Integer' t: SClass.
  )

  setupBasicTest2 = (
```

```
    self c: 'CompilerSimplification'  t: 'testReturnConstantSymbol' e: 'constant' t: SSymbol.
    self c: 'CompilerSimplification'  t: 'testReturnConstantInt' e: 42 t: SInteger.
    self c: 'CompilerSimplification'  t: 'testReturnSelf' e: 'CompilerSimplification' t: SClass.
    self c: 'CompilerSimplification'  t: 'testReturnSelfImplicitly' e: 'CompilerSimplification' t: SClass.

    self c: 'CompilerSimplification'  t: 'testReturnArgumentN' e: 55 t: SInteger.
    self c: 'CompilerSimplification'  t: 'testReturnArgumentA' e: 44 t: SInteger.
    self c: 'CompilerSimplification'  t: 'testSetField'        e: 'foo' t: SSymbol.
    self c: 'CompilerSimplification'  t: 'testGetField'        e: 40 t: SInteger.

    self c: 'Hash'  t: 'testHash'  e: 444 t: SInteger.

    self c: 'Arrays'  t: 'testEmptyToInts'  e: 3 t: SInteger.
    self c: 'Arrays'  t: 'testPutAllInt'    e: 5 t: SInteger.
    self c: 'Arrays'  t: 'testPutAllNil'    e: 'Nil' t: SClass.
    self c: 'Arrays'  t: 'testPutAllBlock'  e: 3 t: SInteger.
    self c: 'Arrays'  t: 'testNewWithAll'   e: 1 t: SInteger.

    self c: 'BlockInlining'  t: 'testNoInlining'   e: 1 t: SInteger.
    self c: 'BlockInlining'  t: 'testOneLevelInlining'   e: 1 t: SInteger.
    self c: 'BlockInlining'  t: 'testOneLevelInliningWithLocalShadowTrue'   e: 2 t: SInteger.
    self c: 'BlockInlining'  t: 'testOneLevelInliningWithLocalShadowFalse'   e: 1 t: SInteger.

    self c: 'BlockInlining'  t: 'testShadowDoesntStoreWrongLocal' e: 33 t: SInteger.
    self c: 'BlockInlining'  t: 'testShadowDoesntReadUnrelated'   e: 'Nil' t: SClass.

    self c: 'BlockInlining'  t: 'testBlockNestedInIfTrue'   e: 2  t: SInteger.
    self c: 'BlockInlining'  t: 'testBlockNestedInIfFalse'  e: 42 t: SInteger.

    self c: 'BlockInlining'  t: 'testStackDisciplineTrue'   e: 1 t: SInteger.
    self c: 'BlockInlining'  t: 'testStackDisciplineFalse'  e: 2 t: SInteger.

    self c: 'BlockInlining'  t: 'testDeepNestedInlinedIfTrue'   e: 3 t: SInteger.
    self c: 'BlockInlining'  t: 'testDeepNestedInlinedIfFalse'   e: 42 t: SInteger.

    self c: 'BlockInlining'  t: 'testDeepNestedBlocksInInlinedIfTrue' e: 5 t: SInteger.
    self c: 'BlockInlining'  t: 'testDeepNestedBlocksInInlinedIfFalse' e: 43 t: SInteger.

    self c: 'BlockInlining'  t: 'testDeepDeepNestedTrue'   e: 9 t: SInteger.
    self c: 'BlockInlining'  t: 'testDeepDeepNestedFalse'   e: 43 t:
```

```
SInteger.

    self c: 'BlockInlining'  t: 'testToDoNestDoNestIfTrue'   e: 2 t: SInteger.

    self c: 'NonLocalVars'  t: 'testWriteDifferentTypes'   e: 3.75 t: SDouble.

    "self c: 'ObjectCreation'  t: 'test'   e: 1000000 t: SInteger."

    self c: 'Regressions'  t: 'testSymbolEquality'            e: 1 t: SInteger.
    self c: 'Regressions'  t: 'testSymbolReferenceEquality'   e: 1 t: SInteger.
    self c: 'Regressions'  t: 'testUninitializedLocal'        e: 1 t: SInteger.
    self c: 'Regressions'  t: 'testUninitializedLocalInBlock' e: 1 t: SInteger.

    self c: 'BinaryOperation'  t: 'test' e: 3 + 8 t: SInteger.

    self c: 'NumberOfTests'  t: 'numberOfTests' e: 65 t: SInteger.
  )

  c: className t: testName e: value t: resultClass = (
    | arr |
    arr := Array new: 4.
    arr at: 1 put: className.
    arr at: 2 put: testName.
    arr at: 3 put: value.
    arr at: 4 put: resultClass.

    basicTests append: arr.
  )
)
```

SomSom/tests/FrameTests.som

```
FrameTests = TestCase (
  | u a b c d e f g h |

  initialize = (
    u := Universe new: true.
    u setupClassPath: 'Smalltalk:TestSuite/BasicInterpreterTests'.
    u initializeObjectSystem.

    a := SSymbol new: 'a'.
    b := SSymbol new: 'b'.
    c := SSymbol new: 'c'.
    d := SSymbol new: 'd'.
    e := SSymbol new: 'e'.
    g := SSymbol new: 'g'.
    h := SSymbol new: 'h'.
  )

  method: name numLocals: numLocals = (
    | sym clazz method |
    sym := SSymbol new: name.
    clazz := SClass new: u.
    clazz name: (SSymbol new: 'Holder').
    method := SMethod new: sym bc: #() literals: #() numLocals: numLocals maxStack: 4.
    method holder: clazz.
    ^ method
  )

  testPushPop = (
    | f length m |
    m := self method: 'testPushPop' numLocals: 0.
    length := 4 + m numberOfArguments + m numberOfLocals.
    f := Frame new: u nilObject previous: nil context: nil method: m maxStack: length.
    f resetStackPointer.

    f push: a.
    self assert: 'a' equals: (f stackElement: 0) string.
    f push: b.
    self assert: 'b' equals: (f stackElement: 0) string.
    f push: c.
    self assert: 'c' equals: (f stackElement: 0) string.
    f pop.
    self assert: 'b' equals: (f stackElement: 0) string.
    f pop.
    self assert: 'a' equals: (f stackElement: 0) string.
    f pop.
  )

  testArgsAndLocal = (
    | f length m |
    m := self method: 'testArgsAndLocal' numLocals: 2.
    length := 4 + m numberOfArguments + m numberOfLocals.
    f := Frame new: u nilObject previous: nil context: nil method: m maxStack: length.
    f resetStackPointer.

    f argument: 1 put: a. "rcvr"
```

```
    f argument: 2 put: b. "local 1"
    f argument: 3 put: c. "local 2"
    f push: d.
    f push: e.
    f push: g.
    f push: h.

    self assert: 'a' equals: (f argument: 1) string.
    self assert: 'b' equals: (f local: 1) string.
    self assert: 'c' equals: (f local: 2) string.
    self assert: 'h' equals: (f stackElement: 0) string.
    self assert: 'g' equals: (f stackElement: 1) string.
    self assert: 'e' equals: (f stackElement: 2) string.
    self assert: 'd' equals: (f stackElement: 3) string.
    self assert: 'c' equals: (f stackElement: 4) string.
  )

  testCopyArgs = (
    | f length m1 m2 copyF |
    m1 := self method: 'sourceTest:copy:args:' numLocals: 2.
    m2 := self method: 'targetTest:copy:args:' numLocals: 2.
    length := 5 + m1 numberOfArguments + m1 numberOfLocals.
    f := Frame new: u nilObject previous: nil context: nil method: m1 maxStack: length.
    f resetStackPointer.

    f local: 1 put: e. "local 1"
    f local: 2 put: g. "local 2"
    f push: h. "stack 1"
    "stuff to be copied"
    f push: a. "rcvr"
    f push: b. "arg test:"
    f push: c. "arg copy:"
    f push: d. "arg args:"

    copyF := Frame new: u nilObject previous: nil context: nil method: m2 maxStack: length.
    copyF resetStackPointer.
    copyF copyArgumentsFrom: f.

    self assert: 'a' equals: (copyF argument: 1) string.
    self assert: 'b' equals: (copyF argument: 2) string.
    self assert: 'c' equals: (copyF argument: 3) string.
    self assert: 'd' equals: (copyF argument: 4) string.
    self assert: u nilObject is: (copyF local: 1).
    self assert: u nilObject is: (copyF local: 2).
    self assert: u nilObject is: (copyF stackElement: 0).

    copyF push: e. "arg args:"

    self assert: 'a' equals: (copyF argument: 1) string.
    self assert: 'b' equals: (copyF argument: 2) string.
    self assert: 'c' equals: (copyF argument: 3) string.
    self assert: 'd' equals: (copyF argument: 4) string.
    self assert: u nilObject is: (copyF local: 1).
    self assert: u nilObject is: (copyF local: 2).
    self assert: 'e' equals: (copyF stackElement: 0) string.

  )
  ----
```

```
  new = (
    ^ super new initialize
  )
)

```

```
LexerTests = TestCase (

  testEmptyClass = (
    | l |
    l := Lexer new: 'Foo = ()'.

    self assert: #identifier is: l sym.
    self assert: 'Foo' equals: l text.

    self assert: #equal is: l sym.
    self assert: '=' equals: l text.

    self assert: #newTerm is: l sym.
    self assert: '(' equals: l text.

    self assert: #endTerm is: l sym.
    self assert: ')' equals: l text.

    self assert: #NONE is: l sym.
  )

  testKeywordSymbol = (
    | l |
    l := Lexer new: '#key:word:'.

    self assert: #pound is: l sym.
    self assert: '#' equals: l text.

    self assert: #keywordSequence is: l sym.
    self assert: 'key:word:' equals: l text.

    self assert: #NONE is: l sym.
  )

  testIntMessage = (
    | l |
    l := Lexer new: '314 println'.

    self assert: #integer is: l sym.
    self assert: '314' equals: l text.

    self assert: #identifier is: l sym.
    self assert: 'println' equals: l text.

    self assert: #NONE is: l sym.
  )

  testAssignDouble = (
    | l |
    l := Lexer new: 'var := 3.14.'.

    self assert: #identifier is: l sym.
    self assert: 'var' equals: l text.

    self assert: #assign is: l sym.
    self assert: ':=' equals: l text.

    self assert: #double is: l sym.
```

```
    self assert: '3.14' equals: l text.

    self assert: #period is: l sym.
    self assert: '.' equals: l text.

    self assert: #NONE is: l sym.
  )

  testString = (
    | l str |
    str := '\'some string with new\nline\''.
    l := Lexer new: str.

    self assert: #string is: l sym.
    self assert: 'some string with new\nline' equals: l text.

    self assert: #NONE is: l sym.
  )

  testEscapeChars = (
    | l str |
    str := '\'\\n\\0\\t\''.
    l := Lexer new: str.

    self assert: #string is: l sym.
    self assert: '\n\0\t' equals: l text.
    self assert: 3 equals: l text length.

    self assert: #NONE is: l sym.
  )

  testPrimitiveMethod = (
    | l |
    l := Lexer new: ' foo: bar = primitive '.

    self assert: #keyword is: l sym.
    self assert: 'foo:' equals: l text.

    self assert: #identifier is: l sym.
    self assert: 'bar' equals: l text.

    self assert: #equal is: l sym.
    self assert: '=' equals: l text.

    self assert: #primitive is: l sym.
    self assert: 'primitive' equals: l text.
  )

  testMathBlock = (
    | l syms |
    l := Lexer new: '[ 0 ~ 1 & 2 | 3 * 4 / 5 \\ 6 + 7 - 8 > 9 < 10 , 11 @ 12 % 13]'.

    syms := #(#not #and #or #star #div #mod #plus #minus #more #less #comma #at #per).

    self assert: #newBlock is: l sym.
    self assert: '[' equals: l text.

    0 to: 12 do: [:i |
      self assert: #integer is: l sym.
```

```
      self assert: i asString equals: l text.
      self assert: (syms at: i + 1) is: l sym ].

    self assert: #integer is: l sym.
    self assert: '13' equals: l text.

    self assert: #endBlock is: l sym.
    self assert: ']' equals: l text.
  )

  testOperatorSequence = (
    | l |
    l := Lexer new: '1 ------ ---- --> 2'.

    self assert: #integer is: l sym.
    self assert: '1' equals: l text.

    self assert: #separator is: l sym.
    self assert: '------' equals: l text.

    self assert: #separator is: l sym.
    self assert: '----' equals: l text.

    self assert: #operatorSequence is: l sym.
    self assert: '-->' equals: l text.

    self assert: #integer is: l sym.
    self assert: '2' equals: l text.
  )

  testBlock = (
    | l |
    l := Lexer new: '[:x | ^ 1]'.

    self assert: #newBlock is: l sym.
    self assert: '[' equals: l text.

    self assert: #colon is: l sym.
    self assert: ':' equals: l text.

    self assert: #identifier is: l sym.
    self assert: 'x' equals: l text.

    self assert: #or is: l sym.
    self assert: '|' equals: l text.

    self assert: #exit is: l sym.
    self assert: '^' equals: l text.

    self assert: #integer is: l sym.
    self assert: '1' equals: l text.

    self assert: #endBlock is: l sym.
    self assert: ']' equals: l text.
  )


  "  Colon, Exit  "
)
```

SomSom/tests/ParserTests.som

```
ParserTests = TestCase (
  | universe |
  testEmptyClass = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = ()' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testSpaceBeforeEmptyClass = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: '
        Foo = ()' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testCommentBeforeEmptyClass = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: '
      "This is a Foo Class"
      Foo = ()' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testEmptyWithNilSuperClass = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = nil ()' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testEmptyWithObjectSuperClass = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = Object ()' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  parseAndCaptureError: parser = (
    parser errorHandler: [:msg | ^ msg ].
    ^ parser classdef.
  )

  testEmptyClassMissingEqual = (
    | cgenc parser u |
    u := self initUniverse.
    parser := ParserWithError newWith: 'Foo ()' for: 'Foo.som' in: u.
    cgenc := self parseAndCaptureError: parser.
    self assert: (cgenc beginsWith: 'Parsing of Foo.som failed, expected equal but found newTerm')
  )

  testEmptyClassWithComment = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = ( "comment" )' for: 'Foo.som' in: u.
```

```
    cgenc := parser classdef.
  )

  testClassWithFields = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = (|a b c|)' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testClassWithUnaryMethod = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = ( m = () )' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testClassWithBinaryMethod = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = ( * o = () )' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testClassWithKeywordMethod = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = ( m: o = () )' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testClassWithKeywordPrimitive = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: 'Foo = ( m: o = primitive )' for: 'Foo.som' in: u.
    cgenc := parser classdef.
  )

  testClassWithVariousMethods = (
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser newWith: '
    ClassWithVariousMethods = (
      a: o = ( | s n v | )
      b: o = ( ^ 1 )
      bn: o = ( ^ -1 )
      c: o = ( ^ 2.2 )
      cn: o = ( ^ -2.2 )
      d: o = ( ^ \'ss\' )
      e: o = ( ^ #sym )
      f: o = ( | a | a := a := 2 )
      g: o = ( o )
      h: o = ( self foo )
      i: o = ( super foo )
      j: o = ( (1) )
      k: o = ( [1] )
      l: o = ( 1 foo: 4 )
      m: o = ( 1 + 4 )
      n: o = ( 1 ++ 2 )
      o: o = ( #(2 3 4 5) )
```

```
  )' for: 'ClassWithVariousMethods.som' in: u.
  cgenc := parser classdef.
)

testSmalltalkFolder = (
  | files |
  files := #(
    'Array.som'
    'Block.som'
    'Block1.som'
    'Block2.som'
    'Block3.som'
    'Boolean.som'
    'Class.som'
    'Dictionary.som'
    'Double.som'
    'False.som'
    'HashEntry.som'
    'Hashtable.som'
    'Integer.som'
    'Metaclass.som'
    'Method.som'
    'Nil.som'
    'Object.som'
    'Pair.som'
    'Primitive.som'
    'Set.som'
    'String.som'
    'Symbol.som'
    'System.som'
    'True.som'
    'Vector.som' ).

  files do: [:f |
    | cgenc parser u |
    u := self initUniverse.
    parser := Parser load: 'Smalltalk/' + f in: u.
    self deny: parser isNil.
    cgenc := parser classdef ].
)

testTestSuiteFolder = (
  | files |
  files := #(
    'ArrayTest.som'
    'BlockTest.som'
    'ClassA.som'
    'ClassB.som'
    'ClassC.som'
    'ClassLoadingTest.som'
    'ClassStructureTest.som'
    'ClosureTest.som'
    'CoercionTest.som'
    'CompilerReturnTest.som'
    'DoesNotUnderstandMessage.som'
    'DoesNotUnderstandTest.som'
    'DoubleTest.som'
    'EmptyTest.som'
    'GlobalTest.som'
    'HashTest.som'
    'IntegerTest.som'
```

```
      'PreliminaryTest.som'
      'ReflectionTest.som'
      'SelfBlockTest.som'
      'SetTest.som'
      'SpecialSelectorsTest.som'
      'StringTest.som'
      'SuperTest.som'
      'SuperTestSuperClass.som'
      'SymbolTest.som'
      'SystemTest.som'
      'TestCase.som'
      'TestHarness.som'
      'TestRunner.som'
      'VectorTest.som'
    ).

    files do: [:f |
      | cgenc parser u |
      u := self initUniverse.
      parser := Parser load: 'TestSuite/' + f in: u.
      self deny: parser isNil.
      cgenc := parser classdef ].
  )

  initUniverse = (
    | u |
    universe ifNil: [
      u := Universe new.
      u setupClassPath: 'Smalltalk:TestSuite'.
      u initializeObjectSystem.
      universe := u ].
    ^ universe
  )
)
```

```
ParserWithError = Parser (
  | errorHandler |

  errorHandler: aBlock = (
    errorHandler := aBlock
  )

  error: message = (
    errorHandler value: message
  )
)
```

```
SomSomTests = TestHarness (
  tests = (
    ^ EmptyTest,
      FrameTests,
      LexerTests,
      ParserTests,
      BasicInterpreterTests,
      SomTests
  )
)

```

```
SomTests = TestCase (

  testArray          = ( self doTest: 'Array' )
  testBlock          = ( self doTest: 'Block' )
  testBoolean        = ( self doTest: 'Boolean' )
  testClassLoading   = ( self doTest: 'ClassLoading' )
  testClassStructure = ( self doTest: 'ClassStructure' )

  testClosure           = ( self doTest: 'Closure' )
  testCoercion          = ( self doTest: 'Coercion' )
  testCompilerReturn    = ( self doTest: 'CompilerReturn' )
  testDictionary        = ( self doTest: 'Dictionary' )
  testDoesNotUnderstand = ( self doTest: 'DoesNotUnderstand' )
  testDouble            = ( self doTest: 'Double' )

  testEmpty             = ( self doTest: 'Empty' )
  testGlobal            = ( self doTest: 'Global' )
  testHash              = ( self doTest: 'Hash' )
  testInteger           = ( self doTest: 'Integer' )

  testPreliminary       = ( self doTest: 'Preliminary' )
  testReflection        = ( self doTest: 'Reflection' )
  testSelfBlock         = ( self doTest: 'SelfBlock' )
  testSpecialSelectorsTest = ( self doTest: 'SpecialSelectorsTest' )
  testSuper             = ( self doTest: 'Super' )

  testSet                = ( self doTest: 'Set' )
  testString             = ( self doTest: 'String' )
  testSymbol             = ( self doTest: 'Symbol' )
  testSystem             = ( self doTest: 'System' )
  testVector             = ( self doTest: 'Vector' )

  doTest: testName = (
    | args u exitCode |
    args := Array new: 4.
    args at: 1 put: '-cp'.
    args at: 2 put: 'Smalltalk'.
    args at: 3 put: 'TestSuite/TestHarness.som'.
    args at: 4 put: testName.

    u := Universe new: true.

    exitCode := u interpret: args.

    self assert: 0 equals: exitCode.
  )
)
```

TestSuite/ArrayTest.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

ArrayTest = TestCase (
  | a |

  setUp = (
    a := Array new: 3.
    a at: 1 put: 'hello'.
    a at: 2 put: #world.
    a at: 3 put: 23.
  )

  testAt = (
    self assert: 'hello' equals: (a at: 1).
    self assert: #world equals: (a at: 2).
    self assert: 23 equals:     (a at: 3).
  )

  testLength = (
    self assert: 3 equals: a length
  )

  testPutAllBlock = (
    | arr i |
    arr := Array new: 10.

    i := 0.
    arr putAll: [ i := i + 1. i ].
```

```
  1 to: 10 do: [:j |
    self assert: j equals: (arr at: j) ].
)

testPutAllInt = (
  | arr i |
  arr := Array new: 10.
  arr putAll: 0.

  1 to: 10 do: [:j |
    self assert: 0 equals: (arr at: j) ].
)

testPutAllSym = (
  | arr i |
  arr := Array new: 10.
  arr putAll: #sym.

  1 to: 10 do: [:j |
    self assert: #sym is: (arr at: j) ].
)

testFirst = (
  self assert: 'hello' equals: a first.
)

testLast = (
  self assert: 23 equals: a last.
)

testContains = (
  self assert: (a contains: 23).
  self deny:   (a contains: #notInThere).
)

testDo = (
  | j |
  j := 1.

  a do: [:i |
    self assert: (a at: j) is: i.
    j := j + 1.
  ]
)

testDoIndexes = (
  | i arr |
  i := 1.
  a doIndexes: [:j |
    self assert: i equals: j.
    i := i + 1.
  ].
  self assert: 4 equals: i.

  arr := Array new: 10.
  i := 1.
  arr doIndexes: [:j |
    self assert: i equals: j.
    i := i + 1.
  ].
```

```
    self assert: 11 equals: i.
  )

  testFromToDo = (
    | arr i |
    a from: 2 to: 2 do: [:e | self assert: #world is: e ].

    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    i := 3.
    arr from: 3 to: 7 do: [:e |
      self assert: i equals: e.
      i := i + 1 ].

    self assert: 8 equals: i.
  )

  testSumAndAverage = (
    | arr |
    arr := Array new: 3.
    1 to: 3 do: [ :i | arr at: i put: i ].

    self assert: 6 equals: arr sum.
    self assert: 2 equals: arr average.
  )

  testCopyFrom = (
    | arr b |
    arr := Array new: 5.
    1 to: 5 do: [ :i | arr at: i put: i ].

    b := arr copyFrom: 2 to: 4.
    self assert: 2 equals: (b at: 1).
    self assert: 3 equals: (b at: 2).
    self assert: 4 equals: (b at: 3).

    b := arr copyFrom: 3.
    self assert: 3 equals: (b at: 1).
    self assert: 4 equals: (b at: 2).
    self assert: 5 equals: (b at: 3).
  )

  testCopy = (
    | arr |
    arr := a copy.
    self assert: 3 equals: arr length.
    self assert: 'hello' equals: (arr at: 1).
    self assert: #world equals:  (arr at: 2).
    self assert: 23 equals:      (arr at: 3).
  )

  testReplaceFrom = (
    | arr1 arr2 i |
    arr1 := Array new: 10 withAll: 0.

    i := 0.
    arr2 := Array new: 10 withAll: [ i := i + 1. i ].

    arr1 replaceFrom: 3 to: 7 with: arr2 startingAt: 1.

```

```
    i := 1.
    3 to: 7 do: [:j |
      self assert: i equals: (arr1 at: j).
      i := i + 1.
    ]
  )

  testExtendedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr extendedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a extendedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 4).
  )

  testPrependedWith = (
    | arr newArr |
    arr := Array new: 0.
    newArr := arr prependedWith: 33.

    self assert: 1 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 33 equals: (newArr at: 1).

    self testAt.  "confirm a is correct"
    self testLength.

    newArr := a prependedWith: 44.

    self testAt.  "confirm a is correct"
    self testLength.

    self assert: 4 equals: newArr length.
    self assert: 0 equals: arr length.
    self assert: 44 equals: (newArr at: 1).
  )

  testIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

```

```
    self assert: 2 equals: (arr indexOf: #two).
    self assert: 4 equals: (arr indexOf: #four).
    self assert: 5 equals: (arr indexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 1 equals: (arr indexOf: #one).
  )

  testLastIndexOf = (
    | arr |
    arr := Array new: 6.
    arr at: 1 put: #one.
    arr at: 2 put: #two.
    arr at: 3 put: #three.
    arr at: 4 put: #four.
    arr at: 5 put: #five.
    arr at: 6 put: #one.

    self assert: 2 equals: (arr lastIndexOf: #two).
    self assert: 4 equals: (arr lastIndexOf: #four).
    self assert: 5 equals: (arr lastIndexOf: #five).

    self assert: nil equals: (arr indexOf: #notIncluded).

    self assert: 6 equals: (arr lastIndexOf: #one).
  )

  testCollect = (
    | arr i col |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].
    col := arr collect: [:e | e + 1 ].

    self assert: 10 equals: col length.

    1 to: 10 do: [:i |
      self assert: i + 1 equals: (col at: i) ].
  )

  testInject = (
    | arr result |
    arr := Array new: 10 withAll: 1.

    result := arr inject: 100 into: [:sum :e | sum + e ].

    self assert: 110 equals: result.
  )

  testReject = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr reject: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 1 equals: (result at: 1).
    self assert: 3 equals: (result at: 2).
    self assert: 5 equals: (result at: 3).
    self assert: 7 equals: (result at: 4).
```

```
    self assert: 9 equals: (result at: 5).
  )

  testRejectEmpty = (
    | result |
    result := (Array new: 0) reject: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) reject: [:e | true ].
    self assert: 0 equals: result size.
  )

  testDontRejectAny = (
    | result |
    result := (Array new: 10 withAll: 0) reject: [:e | false ].
    self assert: 10 equals: result size.

    self assert: 0 equals: (result at: 5).
  )

  testSelect = (
    | arr i result |
    i := 0.
    arr := Array new: 10 withAll: [ i := i + 1. i ].

    result := arr select: [:e | e % 2 = 0 ].

    self assert: 5 equals: result size.
    self assert: 2 equals: (result at: 1).
    self assert: 4 equals: (result at: 2).
    self assert: 6 equals: (result at: 3).
    self assert: 8 equals: (result at: 4).
    self assert: 10 equals: (result at: 5).
  )

  testSelectEmpty = (
    | result |
    result := (Array new: 0) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 0) select: [:e | true ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.

    result := (Array new: 10 withAll: 4) select: [:e | false ].
    self assert: 0 equals: result size.
  )

  testSelectAll = (
    | result |
    result := (Array new: 10 withAll: 0) select: [:e | true ].
    self assert: 10 equals: result size.

```

```
    self assert: 0 equals: (result at: 5).
  )

  testUnion = (
    | result |
    result := a union: a.
    self assert: 3 equals: result size.

    self assert: (result contains: #world).
    self assert: (result contains: 23).

    result := a union: #(21 22 23 #world).

    self assert: 5 equals: result size.
    self assert: (result contains: 21).
    self assert: (result contains: 22).
    self assert: (result contains: 23).
    self assert: (result contains: #world).
  )

  testNewWithAll = (
    | arr |
    arr := Array new: 5 withAll: [1].
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].

    arr := Array new: 5 withAll: 1.
    1 to: 5 do: [:i | self assert: 1 equals: (arr at: i)].
  )

  testNewWithAllIntAndObjects = (
    | arr o |
    arr := Array new: 5 withAll: 5.
    self assert: 5 equals: (arr at: 3).
    arr at: 3 put: nil.
    self assert: nil equals: (arr at: 3).

    o := Object new.
    arr at: 2 put: o.
    self assert: o equals: (arr at: 2).
  )

  testLiteralArrays = (
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢  ’ W V Ç3¢  à
  —6VÆb  76W'C¢ ,2ƒ  "’  C¢ "’ W V Ç3¢ "à

  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢  ’ W V Ç3¢ Ó à
  —6VÆb  76W'C¢ ,2,Ó  Ó#  ã ’  C¢ "’ W V Ç3¢ Ó#  ã à
  )

  testJoin = (
    | arr |
    arr := Array new: 0.
    self assert: nil is: (arr join: ', ').

    arr := Array with: 1 with: 10 with: 100.
    self assert: 1 + 20000 + 10 + 20000 + 100 equals: (arr join: 20000).

    arr := Array with: 'a' with: 'b' with: 'c'.
    self assert: 'a, b, c' equals: (arr join: ', ').
  )
)
```

TestSuite/BasicInterpreterTests/Arrays.som

```
Arrays = (
  ----

  testEmptyToInts = (
    | arr |
    arr := Array new: 5.
    (arr at: 1) ifNotNil: [self error: 'should be initialized to nil'].

    1 to: 5 do: [:i |
      arr at: i put: i.
      (arr at: i) = i ifFalse: [self error: 'should be i'].
    ].

    (arr at: 1) = 1 ifFalse: [self error: 'should be 1'].
    (arr at: 5) = 5 ifFalse: [self error: 'should be 1'].
    ^ arr at: 3
  )

  testPutAllInt = (
    | arr |
    arr := Array new: 5.
    arr putAll: 5.
    ^ arr at: 3
  )

  testPutAllNil = (
    | arr |
    arr := Array new: 5.
    (arr at: 4) ifNotNil: [self error: 'should be initialized to nil'].

    arr putAll: 5.
    (arr at: 4) = 5 ifFalse: [self error: 'should be set to 5'].

    arr putAll: nil.

    ^ (arr at: 3) class
  )

  testPutAllBlock = (
    | arr b cnt |
    cnt := 0.
    b := [cnt := cnt + 1. cnt].
    arr := Array new: 5.
    arr putAll: b.

    1 to: 5 do: [:i|
      (arr at: i) = i ifFalse: [self error: 'block not properly evaluated?']
    ].

    ^ arr at: 3
  )

  testNewWithAll = (
    | arr |
    arr := Array new: 5 withAll: [1].
    1 to: 5 do: [:i | (arr at: i) = 1 ifFalse: [self error: 'wrong result']].
```

```
    ^ arr at: 3
  )

)
```

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

BinaryOperation = (

    ----

    test = (
        ^ (self foo: 1) + (self foo2: 2)
    )

    foo: aNumber = (
        ^ 3
    )

    foo2: aNumber = (
        ^ 8
    )


)
```

```
"
Copyright (c) 2015 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

BlockInlining = (
    ----

    testNoInlining = (
      | a block |
      a := 1.
      block := [ a ].
      ^ block value
    )

    testOneLevelInlining = (
      | a |
      a := 1.
      ^ true ifTrue: [ a ] ifFalse: [ 42 ]
    )

    oneLevelInliningWithLocalShadow: bool = (
      | a |
      a := 1.
      ^ bool
          ifTrue: [ | a |
            a := 2.
            a
          ] ifFalse: [ a "that's outer a" ]
    )

    testOneLevelInliningWithLocalShadowTrue = (
      ^ self oneLevelInliningWithLocalShadow: true
    )

    testOneLevelInliningWithLocalShadowFalse = (
```

```
    ^ self oneLevelInliningWithLocalShadow: false
  )

  testShadowDoesntStoreWrongLocal = (
    | a |
    a := 33.
    true ifTrue: [
      | a |
      a := 2 ].
    ^ a
  )

  testShadowDoesntReadUnrelated = (
    | a |
    a := 33.
    true ifTrue: [
      | a |
      a := 2 ].

    true ifTrue: [
      | a |
      ^ a class ].
  )

  deepNestedInlinedIf: bool = (
    | block  a block2 |
    a := 1.
    block := [ "not inlined"
      a := a + 1.
      block2 := [ "not inlined"
        bool ifTrue: [ ^ a := a + 1.]
             ifFalse:[ |a| a := 42. a ]
      ].
      block2 value
    ].
    ^ block value
  )

  testDeepNestedInlinedIfTrue  = ( ^ self deepNestedInlinedIf: true  )
  testDeepNestedInlinedIfFalse = ( ^ self deepNestedInlinedIf: false )

  blockNestedInIf: bool = (
    | a |
    a := 1.
    bool ifTrue: [
      | block |
      block := [ a := a + 1 ].
      block value
    ] ifFalse: [
      a := 42.
    ].
    ^ a
  )

  testBlockNestedInIfTrue  = ( ^ self blockNestedInIf: true  )
  testBlockNestedInIfFalse = ( ^ self blockNestedInIf: false )

  testStackDisciplineTrue = (
    | result |
    result := 0 max: (1 > 0 ifTrue: [1] ifFalse: [2]).
    ^ result
```

```
    )

    testStackDisciplineFalse = (
      | result |
      result := 0 max: (1 < 0 ifTrue: [1] ifFalse: [2]).
      ^ result
    )

    deepNestedBlocksInInlinedIf: bool = (
      | block a block2 block3 |
      a := 1.
      block := [ "not inlined"
        a := a + 1.
        block2 := [ "not inlined"
          bool ifTrue: [ a := a + 1. "inlined"
            block3 := [ |block4|
              a := a + 1.
              block4 := [ "not inlined"
                a := a + 1.
                a
              ].
              block4 value
            ].
            block3 value
          ] ifFalse:[ |a block4| "inlined"
            a := 42.
            block4 := [ ^ a := a + 1 ]. "not inlined"
            block4 value
          ]
        ].
        block2 value
      ].
      ^ block value
    )

    testDeepNestedBlocksInInlinedIfTrue  = ( ^ self deepNestedBlocksInInlinedIf: true  )
    testDeepNestedBlocksInInlinedIfFalse = ( ^ self deepNestedBlocksInInlinedIf: false )

    deepDeepNested: bool = (
      | block a block2 block3 |
      a := 1.
      block := [ "not inlined"
        a := a + 1.
        block2 := [ "not inlined"
          bool ifTrue: [ a := a + 1. "inlined"
            block3 := [ |block4|
              a := a + 1.
              block4 := [ "not inlined"
                a := a + 1.


                  block := [ "not inlined"
                    a := a + 1.
                    block2 := [ "not inlined"
                      bool ifTrue: [ a := a + 1. "inlined"
                        block3 := [ |block4|
                          a := a + 1.
                          block4 := [ "not inlined"
                            a := a + 1.
```

```
                            a
                          ].
                          block4 value
                        ].
                        block3 value
                      ] ifFalse:[ |a block4| a := 42. "inlined"
                        block4 := [^ a := a + 1]. "not inlined"
                        block4 value
                      ]
                    ].
                    block2 value
                  ].
                  block value


              ].
              block4 value
            ].
            block3 value
          ] ifFalse:[ |a block4| a := 42. "inlined"
            block4 := [^ a := a + 1]. "not inlined"
            block4 value
          ]
        ].
        block2 value
      ].
      ^ block value
    )

    testDeepDeepNestedTrue  = ( ^ self deepDeepNested: true )
    testDeepDeepNestedFalse = ( ^ self deepDeepNested: false )

    testToDoNestDoNestIfTrue = (
      "from the bounce benchmark"
      | balls bounces |
      balls := Array new: 1 withAll: true.
      bounces := 0.

      1 to: 2 do: [ :i |
          balls do: [ :ball |
              ball ifTrue: [ bounces := bounces + 1 ] ] ].

      ^ bounces
    )
)
```

TestSuite/BasicInterpreterTests/Blocks.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Blocks = (

    ----

    testArg1 = ( ^ [:a | a - 1] value: 43 )

    testArg2 = ( ^ [:a :b | a * b ] value: 11 with: 7 )

    testArgAndLocal = (
      ^ ([:a |
        | blockLocal |
        blockLocal := 3.
        a + blockLocal] value: 5)
    )

    testArgAndContext = (
      | methodLocal |
      ^ [:a |
          methodLocal := 3.
          a + methodLocal] value: 5
    )

    testEmptyZeroArg = (
      [] value == nil ifTrue: [ ^ 1 ].
      ^ 2
    )

    testEmptyOneArg = (
      ([:x | ] value: 4) == nil ifTrue: [ ^ 1 ].
      ^ 2
    )

```

```
    testEmptyTwoArg = (
      ([:x :y | ] value: 4 with: 5) == nil ifTrue: [ ^ 1 ].
      ^ 2
    )
)

```

```
"
Copyright (c) 2014 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CompilerSimplification = (
    ----
    | aField |

    testReturnConstantSymbol = ( ^ #constant )
    testReturnConstantInt    = ( ^ 42 )

    testReturnSelf           = (^ self)
    testReturnSelfImplicitly = ()

    testReturnArgument: n      = ( ^ n )
    testReturnArgument: n a: a = ( ^ a )

    testReturnArgumentN = ( ^ self testReturnArgument: 55 )
    testReturnArgumentA = ( ^ self testReturnArgument: 55 a: 44 )


    setField: val = ( aField := val )
    testSetField  = (
        aField := #bar.
        self setField: #foo.
        ^ aField
    )

    getField = (^ aField)
    testGetField = (
        aField := 40.
        ^ self getField
    )
)

```

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Hash = (

    ----

    testHash = (
        | ht string array t |

        ht := Hashtable new.
        ht isEmpty
            ifFalse: [ 'New Hashtable not empty!'. ^ #notEmpty ].

        ht at: 'a' put: 'b'.
        (ht containsValue: 'b')
            ifFalse: [ '1 not in Hashtable'. ^ 1 ].
        ht isEmpty
            ifTrue: [ 'Nonempty Hashtable empty!'. ^ #notEmpty ].
        ((ht size) = 1)
            ifFalse: [ 'Hashtable has wrong size!'. ^ #wrongSize ].

        ht at: 'c' put: 'd'.
        ((ht size) = 2)
            ifFalse: [ 'Hashtable has wrong size!'. ^ #wrongSize ].

        ht at: 1 put: 2.
        t := Hashtable new.
        ht at: Hashtable put: t.
        (ht containsValue: 'b')
            ifFalse: [ '1 not in Hashtable'. ^ 1 ].
        (ht containsValue: 'd')
            ifFalse: [ '2 not in Hashtable'. ^ 2 ].

        (ht containsValue: 2)
```

```
            ifFalse: [ '3 not in Hashtable'. ^ 3 ].
        (ht containsValue: t)
            ifFalse: [ '4 not in Hashtable'. ^ 4 ].
        (ht containsKey: Hashtable)
            ifFalse: [ 'key not found'. ^ #keyNotFound ].

        ht clear.
        ht isEmpty ifFalse: [ 'cleared hashtable is not empty!'. ^ #notEmpty ].
        ht size = 0 ifFalse: ['cleared hashtable has elements!'. ^ #hasElementsAfterCleaning ].

        string := (ht get: 'a').
        (string = 'b') ifTrue: [ 'get from Hashtable'. ^ 5 ].

        ^ 444
    )
)

```

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

IfTrueIfFalse = (

    ----

    test = (
        ^ self resolve: 42.
    )

    test2 = (
        self resolve: 42.
        self resolve: 42.
        ^ self resolve: 33
    )

    test3 = (
        | i a |
        i := 4.
        [ i > 0 ] whileTrue: [
            a := self resolve: 4.
            i := i - 1.
        ].
        ^ 4
    )

    testIfTrueTrueResult = (
      | result |
      result := true ifTrue: [ 1 ].
      ^ result class
    )

    testIfTrueFalseResult = (
      | result |
```

```
      result := false ifTrue: [ 1 ].
      ^ result class
    )

    testIfFalseTrueResult = (
      | result |
      result := true ifFalse: [ 1 ].
      ^ result class
    )

    testIfFalseFalseResult = (
      | result |
      result := false ifFalse: [ 1 ].
      ^ result class
    )

    resolve: a = (
        (a == nil) ifFalse: [ ^ a ].
    )

    value: aBlock = (
        ^ aBlock value
    )
)

```

TestSuite/BasicInterpreterTests/MethodCall.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

MethodCall = (
    ----

    test = (
        ^ self test2
    )

    test2 = ( ^ 42 )
)

```

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

NonLocalReturn = (

    ----

    test1 = ( | t1Frame |
      [ | nlrFrame |
        ^ 42 ] value
    )
    test2 = ( ^ self test1 + 1 )
    test3 = ( [ self test1. ^ 3 ] value )

    test4 = ( ^ self at: 11 )
    test5 = ( ^ self at: 10000 )

    "Test case borrowed from Vector"
    at: index = ( self checkIndex: index ifValid: [ ^ 42 ].
      "else" ^ 22 )
    checkIndex: index ifValid: block = (
        (10 <= index) && (index <= 100)
            ifTrue:  [ ^ block value ]
            ifFalse: [ #dontcare ]
    )
)

```

```
"
Copyright (c) 2016 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

NonLocalVars = (
     ----

     testWriteDifferentTypes = (
        | value |
        1 to: 10 do: [:index |
            value := 0.
            self collection do: [:index | value := value + index].
            value := value // 4.
        ].
        ^value.
    )

    collection = (^#(7 8))
)
```

TestSuite/BasicInterpreterTests/NumberOfTests.som

```
"
Copyright (c) 2019 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

NumberOfTests = (

    ----

    "Return the known number of tests,
     should be used in basic interpreter test harness to confirm completeness"
    numberOfTests = ( ^ 65 )
)
```

TestSuite/BasicInterpreterTests/ObjectCreation.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

ObjectCreation = (

    ----

    test = (
        | i |
        i := 0.

        [i < 1000000] whileTrue: [
            self new.
            i := i + 1.
        ].
        ^ i
    )
)
```

```
"
Copyright (c) 2019 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Regressions = (

    ----

    testSymbolEquality = (
      'foo:' asSymbol = #foo: ifTrue: [ ^ 1 ].
      ^ 2
    )

    testSymbolReferenceEquality = (
      'foo:' asSymbol == #foo: ifTrue: [ ^ 1 ].
      ^ 2
    )

    testUninitializedLocal = (
      | local |
      local == nil ifTrue: [ ^ 1 ].
      ^ 2
    )

    testUninitializedLocalInBlock = (
      [ | local |
        local == nil ifTrue: [ ^ 1 ] ] value.
      ^ 2
    )
)

```

TestSuite/BasicInterpreterTests/Return.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Return = (

    ----

    testReturnSelf = ( ^ self )

    testReturnSelfImplicitly = ( )

    testNoReturnReturnsSelf = ( 1 )

    testBlockReturnsImplicitlyLastValue = ( ^ ([4] value) )
)

```

TestSuite/BasicInterpreterTests/Self.som

```
"
Copyright (c) 2001-2018 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

Self = (

    ----

    testAssignSuper = (
      super := 42.
      ^ super
    )

    testAssignSelf = (
      self := 42.
      ^ self
    )
)

```

TestSuite/BasicInterpreterTests/number-of-tests.sh

```
#!/bin/sh
SCRIPT_PATH=`dirname $0`
TEST_FILE="${SCRIPT_PATH}/NumberOfTests.som"

# find all tests, count them, trim whitespace from result
NUM_TESTS=`grep -R "test[^[:space:]]*[[:space:]]\+= (" "${SCRIPT_PATH}" | wc -l | tr -d '[:space:]'`

TEST_CODE="    numberOfTests = ( ^ ${NUM_TESTS} )"

sed -i'.old' -e 's/.*numberOfTests.*/'"${TEST_CODE}/" "${TEST_FILE}"

git --no-pager diff --exit-code "${TEST_FILE}"

```

TestSuite/BlockTest.som

```
BlockTest = TestCase (
    |escape_count escaped_block|

    simpleBlock = (
      ^ [ 42 ]
    )

    incBlock = (
      ^ [:val | val + 1 ]
    )

    "This requires a closure"
    adderBlock: amount = (
      ^ [:val | amount + val ]
    )

    "Closure with mutable state in block"
    counterBlock = (
      |count|
      count := 0.
      ^ [ count := count + 1. count ]
    )

    selfKeeper = (
      ^ [ self ]
    )

    escapingBlock = (
      ^ [ ^ 42 ]
    )

    escapingNestedBlock = (
      [ [ ^ [ ^ 43 ] ] value ] value
    )

    testSimpleBlocks = (
      self assert: 42 equals: self simpleBlock value.
      self assert: 4  equals: (self incBlock value: 3).
      self assert: 43 equals: ((self adderBlock: 13) value: 30).
    )

    testClosure = (
      | counter |
      counter := self counterBlock.
      self assert: 1 equals: counter value.
      self assert: 2 equals: counter value.
      self assert: 1 equals: self counterBlock value. "make sure each copy is independent"
      self assert: 3 equals: counter value.
    )

    testSelfInBlock = (
      | test_inst |
      test_inst := BlockTest new.
      self assert: test_inst equals: test_inst selfKeeper value.
      self assert: self      equals: self selfKeeper value.
    )

```

```
  testEscapedBlock = (
    | escaping_block |

    escape_count := 0.

    escaping_block := self escapingBlock.

    self assert: 0 equals: escape_count.
    self assert: 666 equals: escaping_block value.
    self assert: 1 equals: escape_count.

    self assert: escaping_block is: escaped_block.


    escaping_block := self escapingNestedBlock.
    self assert: 1 equals: escape_count.
    self assert: 666 equals: escaping_block value.
    self assert: 2 equals: escape_count.
    self assert: escaping_block is: escaped_block.
  )

  escapedBlock: block = (
    escape_count := escape_count + 1.
    escaped_block := block.

    "return some dummy value to the object that sent 'value' to block"
    ^666
  )

  testWhileTrue = (
    | i |
    i := 1.
    [ i < 10 ] whileTrue: [ i := i + 1 ].
    self assert: 10 equals: i.
  )

  testWhileFalse = (
    | i |
    i := 1.
    [ i >= 10 ] whileFalse: [ i := i + 1 ].
    self assert: 10 equals: i.
  )

  returnVal: val predicate: p = (
    p ifNil: [ ^ val ].
    ^ 0
  )

  testToDoAsResultOfIfTrue = (
    | result |
    result := true ifTrue: [
      1 to: 2 do: [:i | 4 ]
    ].

    self assert: 1 equals: result
  )

  testReturnWithSomething = (
    | result |
    result := self returnVal: 3 predicate: nil.
    self assert: 3 equals: result.
```

```
    result := self returnVal: 4 predicate: self.
    self assert: 0 equals: result.
  )

  testSuperExpressionInBlock = (
    | result |
    result := BlockTest helperForTestSuperExpressionInBlock.
    self assert: BlockTest is: result.
  )

  ----

  doInTest: block = (
    ^ block value
  )

  helperForTestSuperExpressionInBlock = (
    ^ self doInTest: [ (super new) class ]
  )
)
```

```
BooleanTest = TestCase (

    testIfTrueIfFalse = (
      | b1 b2 |
      b1 := false.
      b2 := false.

      true ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b1.
      self deny: b2.

      b1 := false.
      b2 := false.
      false ifTrue: [ b1 := true ] ifFalse: [ b2 := true ].
      self assert: b2.
      self deny: b1.
    )

    testIfTrue = (
      | b |
      b := false.

      true ifTrue: [ b := true ].
      self assert: b.

      b := false.
      false ifTrue: [ b := true ].
      self deny: b.
    )

    testIfTrueWithValueBlock = (
      | b block |
      b := false.
      block := [ b := true ].

      true ifTrue: block.
      self assert: b.

      b := false.
      false ifTrue: block.
      self deny: b.
    )

    testIfFalse = (
      | b |
      b := false.

      true ifFalse: [ b := true ].
      self deny: b.

      b := false.
      false ifFalse: [ b := true ].
      self assert: b.
    )

    testIfFalseWithValueBlock = (
      | b block |
      b := false.
```

```
    block := [ b := true ].

    true ifFalse: block.
    self deny: b.

    b := false.
    false ifFalse: block.
    self assert: b.
  )

  testNot = (
    self deny: true not.
    self assert: false not.
  )

  andBoolTrueTrue   = ( ^ true  and: [ true  ] )
  andBoolTrueFalse  = ( ^ true  and: [ false ] )
  andBoolFalseTrue  = ( ^ false and: [ true  ] )
  andBoolFalseFalse = ( ^ false and: [ false ] )

  testAnd = (
    self assert: self andBoolTrueTrue.
    self deny:   self andBoolTrueFalse.
    self deny:   self andBoolFalseTrue.
    self deny:   self andBoolFalseFalse.
  )

  ampBoolTrueTrue   = ( ^ true  && [ true  ] )
  ampBoolTrueFalse  = ( ^ true  && [ false ] )
  ampBoolFalseTrue  = ( ^ false && [ true  ] )
  ampBoolFalseFalse = ( ^ false && [ false ] )

  testAmp = (
    self assert: self ampBoolTrueTrue.
    self deny:   self ampBoolTrueFalse.
    self deny:   self ampBoolFalseTrue.
    self deny:   self ampBoolFalseFalse.
  )

  orBoolTrueTrue   = ( ^ true  or: [ true  ] )
  orBoolTrueFalse  = ( ^ true  or: [ false ] )
  orBoolFalseTrue  = ( ^ false or: [ true  ] )
  orBoolFalseFalse = ( ^ false or: [ false ] )

  testOr = (
    self assert: self orBoolTrueTrue.
    self assert: self orBoolTrueFalse.
    self assert: self orBoolFalseTrue.
    self deny:   self orBoolFalseFalse.
  )

  pipeBoolTrueTrue   = ( ^ true  || [ true  ] )
  pipeBoolTrueFalse  = ( ^ true  || [ false ] )
  pipeBoolFalseTrue  = ( ^ false || [ true  ] )
  pipeBoolFalseFalse = ( ^ false || [ false ] )

  testPipe = (
    self assert: self pipeBoolTrueTrue.
    self assert: self pipeBoolTrueFalse.
    self assert: self pipeBoolFalseTrue.
    self deny:   self pipeBoolFalseFalse.
```

```
    )

    testBlockSideEffectsOnLogicOps = (
      | changed unused |
      changed := false.

      "#and:"
      unused := true and: [ changed := #case1. true ].
      self assert: changed is: #case1.

      unused := false and: [ changed := #no. true ].
      self assert: changed is: #case1.

      "&&"
      unused := true && [ changed := #case2. true ].
      self assert: changed is: #case2.

      unused := false && [ changed := #no. true ].
      self assert: changed is: #case2.

      "#or:"
      unused := true or: [ changed := #no. true ].
      self assert: changed is: #case2.

      unused := false or: [ changed := #case3. true ].
      self assert: changed is: #case3.

      "||"
      unused := true || [ changed := #no. true ].
      self assert: changed is: #case3.

      unused := false || [ changed := #case4. true ].
      self assert: changed is: #case4.
    )

    testAsString = (
      self assert: 'true'  equals: true asString.
      self assert: 'false' equals: false asString.
    )

    testIfNil = (
      self assert: (nil ifNil: [ true ]).
      self deny:   (nil ifNil: [ false ]).

      self assert: (self ifNil: [ #notExec ]) is: self.
      self assert: (self ifNil: [ #notExec ]) is: self.
    )

    testIfNotNil = (
      self assert: (self ifNotNil: [ true ]).
      self deny:   (self ifNotNil: [ false ]).

      self assert: (nil ifNotNil: [ #notExec ]) is: nil.
      self assert: (nil ifNotNil: [ #notExec ]) is: nil.
    )
)
```

```
ClassA = (
   | a b |
   result = (
      ^42
   )
   ----
   | c1 c2 c3 |
)

```

TestSuite/ClassB.som

```
ClassB = ClassA (
  | c d |
  ----
  | c4 c5 c6 |
)

```

```
ClassC = ClassB (
  | e f |
  a      = ( ^ a )
  a: val = ( a := val )

  f      = ( ^ f )
  f: val = ( f := val )

  ----

  | c7 c8 c9 |

  setAllAndInc: anInt = (
      c1 := anInt.
      c2 := c1 + 1.
      c3 := c2 + 1.
      c4 := c3 + 1.
      c5 := c4 + 1.
      c6 := c5 + 1.
      c7 := c6 + 1.
      c8 := c7 + 1.
      c9 := c8 + 1.
  )

  getAll = (
      | arr |
      arr := Array new: 9.
      arr at: 1 put: c1.
      arr at: 2 put: c2.
      arr at: 3 put: c3.
      arr at: 4 put: c4.
      arr at: 5 put: c5.
      arr at: 6 put: c6.
      arr at: 7 put: c7.
      arr at: 8 put: c8.
      arr at: 9 put: c9.
      ^ arr
  )
)
```

```
ClassLoadingTest = TestCase (
  testEqualityOfClasses = (
    | a b c |
    b := ClassB new.
    a := ClassA new.
    c := ClassC new.

    self assert: 42 equals: b result.
    self assert: 42 equals: c result.

    self assert: a class equals: b class superclass.
    self assert: b class equals: c class superclass.
  )
)

```

TestSuite/ClassStructureTest.som

```
ClassStructureTest = TestCase (

  testClassIdentity = (
    self assert: Array   equals: Array new class.
    self assert: Integer equals: 1 class.
    self assert: Integer equals: 10000000000 class.
    self assert: Double  equals: (1 // 2) class.
    self assert: Double  equals: 0.5 class.
    self assert: Block1  equals: [42] class.
    self assert: Object  equals: Object new class.
    self assert: Set     equals: Set new class.
    self assert: String  equals: 'foo' class.
    self assert: Symbol  equals: #foo class.
    self assert: True    equals: true class.
    self assert: False   equals: false class.
    self assert: Nil     equals: nil class.

    self assert: True superclass equals: False superclass.
    self assert: True superclass equals: Boolean.
    self assert: True superclass equals: Boolean.
  )

  testThatCertainMethodsArePrimitives = (
    | m |
    "This is a little fragile.
     Index needs to be adapted with changing Class definition."
    m := Object methods at: 1.
    "self expect: #class equals: m signature."

    self optional: #invokableTypes assert: Primitive equals: m class. "Class>>#name should be a primitive."

    m := Object methods at: 7.
    "self expect: #asString equals: m signature."

    self optional: #invokableTypes assert: Method equals: m class. "Class>>#asString should be a normal method."
  )

  testAccessToInstanceFields = (
    | o |
    o := ClassC new.
    o a: 333.
    self assert: 333 equals: o a.

    o f: 333.
    self assert: 333 equals: o f.
  )

  testAccessToClassFields = (
    | arr |
    ClassC setAllAndInc: 4.
    arr := ClassC getAll.
    1 to: 9 do: [:i |
      self assert: i + (4 - 1) equals: (arr at: i).
    ].

    "We do that here to make sure that class fields do not interfere with
```

```
     other class properties."
    self assert: ClassB    is: ClassC superclass.
    self assert: Metaclass is: ClassC class class.
    self assert: #ClassC   equals: ClassC name.
  )

  testMetaclasses = (
    self assert: nil              is: Object superclass.
    self assert: Integer          is: 1 class.
    self assert: #'Integer class' is: 1 class class name.
    self assert: Metaclass        is: 1 class class class.

    self assert: #'Metaclass class' is: Metaclass class name.
    self assert: Metaclass          is: Metaclass class class.

    self assert: Object          is: 1 class superclass.
    self assert: #'Object class' is: 1 class class superclass name.
    self assert: Class           is: Object class superclass.
    self assert: Metaclass       is: Class class class.
  )

  testInstanceFields = (
    self assert: 2 equals: ClassA fields length.
    self assert: 4 equals: ClassB fields length.
    self assert: 6 equals: ClassC fields length.
  )
)
```

TestSuite/ClosureTest.som

```
"

$Id: ClosureTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"This test verifies that SOM blocks are indeed closures. The test was found on
Eliot Miranda's Cog Blog."

ClosureTest = TestCase (
  testClosureProperty = (
    | factorial result facs |

    facs := Array new: 10.
    facs at: 1 put: 1.
    facs at: 2 put: 2.
    facs at: 3 put: 6.
    facs at: 4 put: 24.
    facs at: 5 put: 120.
    facs at: 6 put: 720.
    facs at: 7 put: 5040.
    facs at: 8 put: 40320.
    facs at: 9 put: 362880.
    facs at: 10 put: 3628800.

    factorial := [ :n |
        n = 1
            ifTrue: [ 1 ]
            ifFalse: [ (factorial value: n - 1) * n ] ].

    result := (1 to: 10) collect: factorial.
    result doIndexes: [ :i |
      self assert: (facs at: i) equals: (result at: i) ]
  )
```

```
)
```

TestSuite/CoercionTest.som

```
"

$Id: CoercionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CoercionTest = TestCase (

  testBasicNumberCoercion = (
    self assert: 5 equals: 25 sqrt.
    self assert: 1 equals: (2 // 4) * 2.
    self assert: 1 equals: 2 * (2 // 4).
  )
)
```

TestSuite/CompilerReturnTest.som

```
"

$Id: CompilerReturnTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2009-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

CompilerReturnTest = TestCase (

    return1 = ( ^self )
    return2 = (       )

    return3: arg = ( ^self )
    return4: arg = (       )

    testExplicitAndImplicitReturns = (
      self assert: self is: self return1.
      self assert: self is: self return2.
      self assert: self is: (self return3: 23).
      self assert: self is: (self return4: 23).
    )


    "In SOM++, code after the #ifTrue: does not seem to be executed, if the
     block expression ends with a dot."
    testIfTrueWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithDot: arr.
      self assertArrayCorrectness: arr.
    )

```

```
    assertArrayCorrectness: arr = (
      self assert: 1 equals: (arr at: 1). "method was not executed"
      self assert: 2 equals: (arr at: 2). "ifTrue was not executed"
      self assert: 3 equals: (arr at: 3). "remainder was not executed"
    )

    testIfTrueWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfTrueWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithDot: arr.
      self assertArrayCorrectness: arr.
    )

    testIfFalseWithoutDot = (
      | arr |
      arr := Array new: 3.
      self usesIfFalseWithoutDot: arr.
      self assertArrayCorrectness: arr.
    )

    usesIfTrueWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfTrueWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2. "WITH DOT"].
        arr at: 3 put: 3.
    )

    usesIfFalseWithoutDot: arr = (
        arr at: 1 put: 1.
        (3 >= 1) ifTrue: [arr at: 2 put: 2 "WITHOUT DOT"].
        arr at: 3 put: 3.
    )

    testWriteArgument = (
        self assert: 42 equals: (self dec: 43).
    )

    dec: anInt = (
        anInt := anInt - 1.
        ^ anInt
    )
)
```

113

TestSuite/DictionaryTest.som

```
DictionaryTest = TestCase (
  testAtAndAtPut = (
    | dict val |
    dict := Dictionary new.
    val := dict at: 1.

    self assert: nil is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: #foo.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: #foo is: val.

    val := dict at: 1 put: 42.
    self assert: dict is: val.
    val := dict at: 1.
    self assert: 42 equals: val.
  )

  testContainsKey = (
    | dict |
    dict := Dictionary new.
    self deny: (dict containsKey: 4).

    dict at: 4 put: 34234.
    self assert: (dict containsKey: 4).
  )

  testKeys = (
    | dict keys |
    dict := Dictionary new.

    self assert: 0 equals: dict keys size.

    dict at: 4 put: 423.
    self assert: 1 equals: dict keys size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict keys size.

    dict at: 'as' put: Object new.
    self assert: 2 equals: dict keys size.

    keys := dict keys.
    self assert: 4    equals: (keys at: 1).
    self assert: 'as' equals: (keys at: 2).
  )

  testValues = (
    | dict values v2 |
    dict := Dictionary new.

    self assert: 0 equals: dict values size.
```

```

    dict at: 4 put: 423.
    self assert: 1 equals: dict values size.

    dict at: 4 put: #gdfgd.
    self assert: 1 equals: dict values size.

    dict at: 'as' put: #(1 2 3).
    self assert: 2 equals: dict values size.

    values := dict values.
    self assert: #gdfgd is: (values at: 1).

    v2 := values at: 2.
    self assert: 1 equals: (v2 at: 1).
    self assert: 2 equals: (v2 at: 2).
    self assert: 3 equals: (v2 at: 3).
  )

  testDo = (
    | dict expectedKs expectedVs i |
    i := 1.
    dict := Dictionary new.
    dict at: #e put: 344.
    dict at: 1  put: 545.
    dict at: 0  put: 123.

    expectedKs := #(#e 1 0).
    expectedVs := #(344 545 123).

    dict do: [:p |
      self assert: (expectedKs at: i) equals: p key.
      self assert: (expectedVs at: i) equals: p value.
      i := i + 1.
    ].

    self assert: 4 equals: i.
  )
)
```

```
DoesNotUnderstandMessage = (
    | target selector arguments |

    initializeWith: targetObj selector: aSelector arguments: argArray = (
        target    := targetObj.
        selector  := aSelector.
        arguments := argArray.
    )

    target    = ( ^ target )
    selector  = ( ^ selector )
    arguments = ( ^ arguments )

    ----

    to: target selector: selector arguments: args = (
        | m |
        m := self new.
        m initializeWith: target selector: selector arguments: args.
        ^ m
    )
)
```

TestSuite/DoesNotUnderstandTest.som

```
DoesNotUnderstandTest = TestCase (

  testSimpleUnknownFoo = (
    | result |
    result := self foo.
    self assert: DoesNotUnderstandMessage is: result class.
    self assert: self is: result target.
    self assert: #foo is: result selector.
  )

  testArguments = (
    | result |
    result := self foo.
    self assert: Array is: result arguments class.
    self assert: 0 equals: result arguments length.

    result := self foo: 1.
    self assert: 1 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).

    result := self foo: 1 bar: 2 baz: 3.
    self assert: 3 equals: result arguments length.
    self assert: 1 equals: (result arguments at: 1).
    self assert: 2 equals: (result arguments at: 2).
    self assert: 3 equals: (result arguments at: 3).
  )

  testRepeat = (
    | result |
    result := Array new: 5.
    1 to: result length do: [:i |
      result at: i put: self foo.

      i > 1 ifTrue: [
        self assert: (result at: i - 1) ~= (result at: i).
      ]
    ].
  )

  doesNotUnderstand: selector arguments: arguments = (
      ^ DoesNotUnderstandMessage to: self selector: selector arguments: arguments.
  )
)
```

TestSuite/DoubleTest.som

```
"

$Id: DoubleTest.som 48 2009-08-12 12:57:20Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

DoubleTest = TestCase (

  testAdd = (
    self assert: 1.0 equals: 0.5 + 0.5.
    self assert: 0.0 equals: 0.5 + -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 + 0.1.
    self assert: 9007199254741000.1 equals: 9007199254740990.0 + 10.1.
  )

  testSubtract = (
    self assert: 0.0 equals: 0.5 - 0.5.
    self assert: 1.0 equals: 0.5 - -0.5.

    self assert: 9007199254740992.0 equals: 9007199254740992.0 - 0.1.
    self assert: 9007199254740990.0 equals: 9007199254741000.1 - 10.1.
  )

  testMultiply = (
    self assert:  4.0 equals: 2.0 * 2.0.
    self assert: -4.0 equals: 2.0 * -2.0.

    self assert:  1.0 equals:  4.0 * 0.25.
    self assert: -1.0 equals: -4.0 * 0.25.
  )

  testIntegerDivision = (
    self assert: 1 equals: (4/3) + (4/5)
```

```
  )

  testDoubleDivision = (
    self assert: 32 // 15 equals: (4//3) + (4//5)
  )

  testModulo = (
    self assert: 1.0 equals: 3.0 % 2.0.
    self assert: 0.0 equals: 3.0 % 3.0.

    self assert: -1.0 equals: -3.0 %  2.0.
    self assert: -1.0 equals: -3.0 % -2.0.
    self assert:  1.0 equals:  3.0 % -2.0.

    self assert: 0.0 equals:  3.0 %  3.0.
    self assert: 0.0 equals: -3.0 % -3.0.
    self assert: 0.0 equals:  3.0 % -3.0.
  )

  testAbs = (
    self assert: 1.0 equals:  1.0 abs.
    self assert: 1.0 equals: -1.0 abs.

    self assert: 9007199254740992.0 equals:  9007199254740992.0 abs.
    self assert: 9007199254740992.0 equals: -9007199254740992.0 abs.

    self assert: 19007199254740992.0 equals:  19007199254740992.0 abs.
    self assert: 19007199254740992.0 equals: -19007199254740992.0 abs.
  )

  testSqrt = (
    self assert:  3.0 equals:   9.0 sqrt.
    self assert: 16.0 equals: 256.0 sqrt.

    self assert: 23453456.0 equals: (23453456.0 * 23453456.0) sqrt.
  )

  testRaisedTo = (
    self assert:    1.0 equals: ( 2.0 raisedTo: 0).
    self assert:    2.0 equals: ( 2.0 raisedTo: 1).
    self assert:    8.0 equals: ( 2.0 raisedTo: 3).
    self assert:  256.0 equals: ( 2.0 raisedTo: 8).

    self assert:  256.0 equals: (-2.0 raisedTo: 8).
    self assert: -128.0 equals: (-2.0 raisedTo: 7).

    self assert:   6.25 equals: ( 2.5 raisedTo: 2).
    self assert: 5.0625 equals: ( 1.5 raisedTo: 4).

    self assert:    0.0 equals: ( 0.0 raisedTo: 5).
    self assert:    1.0 equals: ( 0.0 raisedTo: 0).

    "Negative exponents are not yet supported"
    self assert: 1.0 equals: ( 0.0 raisedTo:  -1).
    self assert: 1.0 equals: ( 0.0 raisedTo:  -2).
    self assert: 1.0 equals: (10.0 raisedTo:  -1).
    self assert: 1.0 equals: (10.0 raisedTo:  -2).

    "Double exponents are not yet supported"
    self assert: 2.0 equals: ( 2.0 raisedTo:  1.5).
    self assert: 4.0 equals: ( 2.0 raisedTo:  2.4).
```

```
    self assert: 4.0 equals: ( 2.0 raisedTo:  2.9).
    self assert: 1.0 equals: ( 2.0 raisedTo:  -2.2).
  )

  testNegated = (
    self assert:  0.0 equals:  0.0 negated.
    self assert: -1.0 equals:  1.0 negated.
    self assert:  1.0 equals: -1.0 negated.

    self assert: -9007199254740992.0 equals:  9007199254740992.0 negated.
    self assert:  9007199254740992.0 equals: -9007199254740992.0 negated.

    self assert: -19007199254740992.0 equals:  19007199254740992.0 negated.
    self assert:  19007199254740992.0 equals: -19007199254740992.0 negated.
  )

  testAsString = (
    self assert: '0.5' equals: (1//2) asString.
    self assert: '0.5' equals: 0.5 asString.
  )

  testEquals = (
    self assert: (1.0 = 1).
  )

  testRound = (
    self assert:   1 equals:         1.0 round.
    self assert:   1 equals:         1.4 round.
    self assert:   1 equals:         1.4999 round.
    self assert:   2 equals:         1.5 round.
    self assert:   2 equals:         1.5000001 round.
    self assert:   1 equals:     (5//10) round.
    self assert:   1 equals:    (14//10) round.
    self assert: 445 equals: (44534//100) round.
  )

  testAsInteger = (
    self assert: 1 equals: 1.0 asInteger.
    self assert: 1 equals: 1.1 asInteger.
    self assert: 1 equals: 1.999 asInteger.

    self assert: -1 equals: -1.0 asInteger.
    self assert: -1 equals: -1.999 asInteger.
  )

  testSin = (
    | pi |
    pi := 3.141592653589.
    self assert: 0.0 equals: 0.0 sin.
    self assert: pi sin abs < 0.00000000001.
    self assert: (pi // 2.0) sin > 0.9999999999.
  )

  testCos = (
    | pi |
    pi := 3.141592653589.
    self assert: 1.0 equals: 0.0 cos.
    self assert: (pi // 2.0) cos abs < 0.00000000001.
    self assert: pi cos < -0.9999999999.
```

```
  )

  testInfinity = (
    self assert: Double PositiveInfinity > 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity + 1.
    self assert: Double PositiveInfinity equals: Double PositiveInfinity - 1.

    self assert: Double PositiveInfinity > (999999 * 999999 * 999999 * 999999).
  )

  testFromString = (
    self assert:  0.0 equals: (Double fromString:  '0.0').
    self assert: -1.1 equals: (Double fromString: '-1.1').

    self assert:  3423.54656 equals: (Double fromString:  '3423.54656').
    self assert: -672.433244 equals: (Double fromString: '-672.433244').
  )

  testEqual = (
    self assert:  0.0 =  0.0.
    self assert:  1.0 =  1.0.
    self assert:  0.0 = -0.0.
    self assert: -0.0 =  0.0.
  )

  testLessThan = (
    self assert: 0.0 < 1.0.
    self assert: 0.499999999 < 0.5.
    self deny:   1.0 < 0.0.
    self deny:   0.5 < 0.499999999.
  )

  testGreaterThan = (
    self deny:   0.0 > 1.0.
    self deny:   0.499999999 > 0.5.
    self assert: 1.0 > 0.0.
    self assert: 0.5 > 0.499999999.
  )

  testLessThanOrEqual = (
    self assert: 0.0 <= 1.0.
    self assert: 0.499999999 <= 0.5.
    self assert: 0.5 <= 0.5.
    self deny:   1.0 < 0.0.
    self deny:   0.5 < 0.499999999.
  )

  testGreaterThanOrEqual = (
    self deny:   0.0 >= 1.0.
    self deny:   0.499999999 >= 0.5.
    self assert: 0.5 >= 0.5.
    self assert: 1.0 >= 0.0.
    self assert: 0.5 >= 0.499999999.
  )

  testNegative = (
    self assert: -0.000000001 negative.
    self assert: -1.0 negative.
```

```
    self assert: -123123.000000001 negative.
    self deny: 0.000000001 negative.
    self deny: 1.0 negative.
    self deny: 123123.000000001 negative.
  )

  testBetween = (
    self assert: (1.0 between: 0.0 and: 2.0).
    self assert: (0.000001 between: 0.0 and: 2.0).
    self assert: (1.999999 between: 0.0 and: 2.0).
    self deny:   (0.0 between: 0.0 and: 2.0).
    self deny:   (2.0 between: 0.0 and: 2.0).
  )

  testToDo = (
    | d |
    d := 0.0.
    0.0 to: 10.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    0.0 to: 10.1 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    0.1 to: 10.1 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 + 1.1 equals: d.
  )

  testDownToDo = (
    | d |
    d := 0.0.
    10.0 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: 55.0 equals: d.

    d := 0.0.
    10.1 downTo: 0.0 do: [:ii |
      d := d + ii.
    ].

    self assert: ((55.0 + 1.1) * 10.0) round equals: (d * 10.0) round.
  )
)
```

TestSuite/EmptyTest.som

```
"

$Id: EmptyTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

EmptyTest = TestCase (

    "This is just an empty TestCase.
     It only tests the basic infrastructure"

)
```

TestSuite/GlobalTest.som

```
GlobalTest = TestCase (
    | doesntKnow |
    unknownGlobal: name = ( doesntKnow := name. ^ name )

    testUnknownGlobalHandler = (
      self assert: #foobar equals: foobar.     "should return the unknown globals name"
      self assert: #foobar equals: doesntKnow. "and should have set it in the field"
    )

    testKnownGlobals = (
      self assert: True   equals: true  class.
      self assert: False  equals: false class.
      self assert: Nil    equals: nil   class.
      self assert: System equals: system class.
    )

    escapingBlock = (
       ^ [ EscapingBlockGlobal ]
    )

    testUnknownGlobalSemanticsInBlocks = (
      self assert: #NormalBlockGlobal is: [ NormalBlockGlobal ] value.
      self assert: #NormalBlockGlobal is: doesntKnow.

      self assert: #EscapingBlockGlobal is: self escapingBlock value.
      self assert: #EscapingBlockGlobal is: doesntKnow.

      self assert: #NestedBlockGlobal is: [
         [ [ NestedBlockGlobal ] value ] value ] value.
      self assert: #NestedBlockGlobal is: doesntKnow.
    )
)
```

```
"

$Id: HashTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

HashTest = TestCase (

  testHashtable = (
    | ht string array t |

    ht := Hashtable new.
    self assert: ht isEmpty description: 'new ht needs to be empty'.

    ht at: 'a' put: 'b'.
    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self deny: ht isEmpty.

    self assert: 1 equals: ht size.

    ht at: 'c' put: 'd'.
    self assert: 2 equals: ht size.

    ht at: 1 put: 2.
    t := Hashtable new.
    ht at: Hashtable put: t.
    system fullGC.

    self assert: (ht containsValue: 'b') description: 'needs to contain "b"'.
    self assert: (ht containsValue: 'd') description: 'needs to contain "d"'.
    self assert: (ht containsValue: 2)   description: 'needs to contain "2"'.
```

```
    self assert: (ht containsValue: t)   description: 'needs to contain t'.
    self assert: (ht containsKey: Hashtable) description: 'needs to contain Hashtable'.

    ht clear.
    self assert: ht isEmpty.
    self assert: 0 equals: ht size.

    self assert: nil equals: (ht get: 'a').
  )
)
```

TestSuite/IntegerTest.som

```
"

$Id: IntegerTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

IntegerTest = TestCase (

  testEqualityAndIdentity = (
    | a b |
    a := 42.
    b := 42.

    self assert: a = b   description: 'Integers are equal based on their value'.
    self assert: a == b  description: 'Integers do not have pointer/reference equality. It is also supposed to be value equality'.

    "Sometimes it can be hard to implement efficiently, but it SHOULD really
     be true for all values of integers."
    a := 1 << 30.  b := 1 << 30.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 32.  b := 1 << 32.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 60.  b := 1 << 60.
    self optional: #integerIdentity assert: a is: b.

    a := 1 << 70.  b := 1 << 70.
    self optional: #integerIdentity assert: a is: b.
```

```

    a := 1 << 100.  b := 1 << 100.
    self optional: #integerIdentity assert: a is: b.
  )

  testClassAndValueRanges = (
    | i |
    self assert: Integer equals: -42 class.
    self assert: Integer equals:   0 class.
    self assert: Integer equals:  23 class.
    self assert: Integer equals:  1073741823 class.
    self assert: Integer equals:  1073741824 class.

    "Let's test for size behavior and corresponding class"
    i := 1 << 30.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1073741824' equals: i asString.

    i := 1 << 32.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '4294967296' equals: i asString.

    i := 1 << 60.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self assert: '1152921504606846976' equals: i asString.

    i := 1 << 70.
    self assert: Integer equals: i class.
    self assert: i > 0 description: 'should not overflow'.
    self optional: #bigIntShifts assert: '1180591620717411303424' equals: i asString.

    i := -1 << 30.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1073741824' equals: i asString.

    i := -1 << 32.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-4294967296' equals: i asString.

    i := -1 << 60.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self assert: '-1152921504606846976' equals: i asString.

    i := -1 << 70.
    self assert: Integer equals: i class.
    self assert: i < 0 description: 'should not underflow'.
    self optional: #bigIntShifts assert: '-1180591620717411303424' equals: i asString.
  )

  testStringConversion = (
    self assert:  '0' equals: ( 0 asString).
    self assert:  '1' equals: ( 1 asString).
    self assert:  '2' equals: ( 2 asString).
```

```
    self assert: '-1' equals: (-1 asString).
    self assert: '-2' equals: (-2 asString).

    self assert: 42 equals: '42' asInteger.
    self assert: -2 equals: '-2' asInteger.
  )

  testIntegerLiterals = (
    "Make sure the parser reads literals correctly. So, check some basic properties"
    self assert:                      1 equals:  2 / 2.
    self assert:                    100 equals:  50 + 50.
    self assert:    9223372036854775807 equals:  92233720368 * 100000000 + 54775807.
    self assert:                    -50 equals:  50 - 100.
    self assert:            -2147483648 equals:  21474 * -100000 - 83648.

    self optional: #bigInteger
         assert:  922337203685477580700 equals:  92233720368 * 100000000 + 54775807 * 100.
    self optional: #bigInteger
         assert: -922337203685477580700 equals:  92233720368 * 100000000 + 54775807 * -100.
  )

  testFromString = (
    self assert:                      1 equals: (Integer fromString:                      '1').
    self assert:                    100 equals: (Integer fromString:                    '100').
    self assert:    9223372036854775807 equals: (Integer fromString: '9223372036854775807').
    self assert:                    -50 equals: (Integer fromString:                    '-50').
    self assert:            -2147483648 equals: (Integer fromString:            '-2147483648').

    self optional: #bigInteger
         assert:  922337203685477580700 equals: (Integer fromString: '922337203685477580700').
    self optional: #bigInteger
         assert: -922337203685477580700 equals: (Integer fromString: '-922337203685477580700').
  )

  testRangeBorders = (
    self assert:   '536870911' equals:   536870911 asString.
    self assert:   '536870912' equals:   536870912 asString.
    self assert:   '536870913' equals:   536870913 asString.
    self assert:  '1073741823' equals:  1073741823 asString.
    self assert:  '1073741824' equals:  1073741824 asString.
    self assert:  '1073741825' equals:  1073741825 asString.
    self assert:  '2147483647' equals:  2147483647 asString.
    self assert:  '-536870911' equals:  -536870911 asString.
    self assert:  '-536870912' equals:  -536870912 asString.
    self assert:  '-536870913' equals:  -536870913 asString.
    self assert: '-1073741823' equals: -1073741823 asString.
    self assert: '-1073741824' equals: -1073741824 asString.
    self assert: '-1073741825' equals: -1073741825 asString.
    self assert: '-2147483647' equals: -2147483647 asString.
    self assert: '-2147483648' equals: -2147483648 asString.
```

```
  )

  testComparisons = (
    self assert: ( 9 =  9).
    self deny:   ( 1 =  2).
    self deny:   ( 0 <  0).
    self assert: ( 1 <  2).
    self deny:   ( 2 <  1).
    self assert: (-3 <  2).
    self deny:   ( 3 < -2).
    self deny:   ( 0 >  0).
    self deny:   ( 1 >  2).
    self assert: ( 2 >  1).
    self deny:   (-3 >  2).
    self assert: ( 3 > -2).
    self assert: ( 4 >= 3).
    self assert: ( 3 >= 3).
    self deny:   ( 2 >= 3).
    self assert: ( 2 <= 4).
    self assert: ( 3 <= 3).
    self deny:   ( 4 <= 3).
  )

  testAddition = (
    self assert: 0 equals: ( 0+0).
    self assert: 1 equals: ( 1+0).
    self assert: 1 equals: ( 0+1).
    self assert: 2 equals: ( 1+1).
    self assert: 0 equals: (-1+1).
    self assert: 1 equals: (-1+2).
  )

  testSubtraction = (
    self assert:  1 equals: (1-0).
    self assert: -1 equals: (0-1).
    self assert:  1 equals: (2-1).
  )

  testMultiplication = (
    self assert:   0 equals: ( 1*  0).
    self assert:  -1 equals: (-1*  1).
    self assert: -25 equals: ( 5* -5).
    self assert:  12 equals: (-3* -4).
  )

  testDivision = (
    self assert:  1 equals: (  1/  1).
    self assert:  1 equals: (  3/  2).
    self assert: -2 equals: (  4/ -2).
    self assert: -2 equals: ( -6/  3).
    self assert:  3 equals: (-12/ -4).
  )

  testDouble = (
    self assert:  6 equals: ( 36//  6).
    self assert: -5 equals: (-10//  2).
    self assert: -4 equals: ( 20// -5).
    self assert:  1 equals: ( -5// -5).
  )

  testModulo = (
```

```
    self assert:  1 equals: ( 10 %  3).
    self assert: -2 equals: ( 10 % -3).
    self assert:  2 equals: (-10 %  3).
    self assert: -1 equals: (-10 % -3).
    self assert:  0 equals: ( 10 %  5).

    self assert:  1 equals: ( 10 rem:  3).
    self assert:  1 equals: ( 10 rem: -3).
    self assert: -1 equals: (-10 rem:  3).
    self assert: -1 equals: (-10 rem: -3).
    self assert:  0 equals: ( 10 rem:  5).
  )

  testAbsoluteValue = (
    self assert: 4 equals: -4 abs.
    self assert: 4 equals:  4 abs.

    self assert: 9223372036854775296 equals: -9223372036854775296 abs.
    self assert: 9223372036854775296 equals:  9223372036854775296 abs.
  )

  testNegated = (
    self assert: -23 equals: ( 23 negated).
    self assert:  23 equals: (-23 negated).
  )

  testSquareRoot = (
    self assert:       5 equals: (25 sqrt).
    self assert: Integer equals: (25 sqrt class).
  )

  testRaisedTo = (
    self assert:                               1 equals: ( 2 raisedTo: 0).
    self assert:                               2 equals: ( 2 raisedTo: 1).
    self assert:                               8 equals: ( 2 raisedTo: 3).
    self assert:                             256 equals: ( 2 raisedTo: 8).
    self assert: 1267650600228229401496703205376 equals: ( 2 raisedTo: 100).

    self assert:                             256 equals: (-2 raisedTo: 8).
    self assert:                            -128 equals: (-2 raisedTo: 7).

    self assert:                               0 equals: ( 0 raisedTo: 5).
    self assert:                               1 equals: ( 0 raisedTo: 0).

    "Negative exponents are not yet supported"
    self assert:                               1 equals: ( 0 raisedTo: -1).
    self assert:                               1 equals: ( 0 raisedTo: -2).
    self assert:                               1 equals: (10 raisedTo: -1).
    self assert:                               1 equals: (10 raisedTo:
```

```
-2).

    "Double exponents are not yet supported"
    self assert:                               2 equals: ( 2 raisedTo:
1.5).
    self assert:                               4 equals: ( 2 raisedTo:
2.4).
    self assert:                               4 equals: ( 2 raisedTo:
2.9).
    self assert:                               1 equals: ( 2 raisedTo:
-2.2).
  )

  testAnd = (
    self assert: 0 equals: (2 & 1).
    self assert: 2 equals: (2 & 2).
  )

  testBitXor = (
    self assert: 0 equals: (1 bitXor: 1).
    self assert: 3 equals: (2 bitXor: 1).
  )

  testAs32BitUnsignedValue = (
    self assert: 1 <<  1 equals: (1 <<  1) as32BitUnsignedValue.
    self assert: 1 << 10 equals: (1 << 10) as32BitUnsignedValue.
    self assert: 1 << 31 equals: (1 << 31) as32BitUnsignedValue.
    self assert:       0 equals: (1 << 32) as32BitUnsignedValue.
    self assert: 4294967295 equals:     -1 as32BitUnsignedValue.
    self assert: 512        equals: -9223372036854775296
as32BitUnsignedValue.
    self assert: 4294966784 equals:  9223372036854775296
as32BitUnsignedValue.
  )

  testAs32BitSignedValue = (
    self assert:     1 <<  1 equals: (1 <<  1) as32BitSignedValue.
    self assert:     1 << 10 equals: (1 << 10) as32BitSignedValue.
    self assert: -2147483648 equals: (1 << 31) as32BitSignedValue.
    self assert:           0 equals: (1 << 32) as32BitSignedValue.

    self assert:  512 equals:  -9223372036854775296 as32BitSignedValue.
    self assert: -512 equals:   9223372036854775296 as32BitSignedValue.
  )

  testAsDouble = (
    self assert: 0.0          equals: 0 asDouble.
    self assert: Double       is:     0 asDouble class.

    self assert: 2147483648.0 equals: 2147483648 asDouble.
    self assert: Double       is:     2147483648 asDouble class.

    self assert: -2147483648.0 equals: -2147483648 asDouble.
    self assert: Double       is:      -2147483648 asDouble class.
  )

  testUnsignedRightShift = (
    self assert: 0   equals:    1 >>> 1.
    self assert: 512 equals: 1024 >>> 1.
    self assert: 127 equals: 1023 >>> 3.

```

```
    "not sure whether we should really insist on this"
    self optional: #toBeSpecified assert: 9223372036854775807 equals: -1 >>> 1.
    self optional: #toBeSpecified assert: 9223372036854775296 equals: -1024 >>> 1.
  )

  testMin = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 -50 -2147483648).
    small := #(0  52 9223372036854775296 -51 -2147483650).

    big doIndexes: [:i |
      self assert: (small at: i)  equals: ((big   at: i) min: (small at: i)).
      self assert: (small at: i)  equals: ((small at: i) min: (big   at: i)) ].

    "not sure whether we should really insist on this"
    big   := #( 9223372036854775807000  9223372036854775807000
               -9223372036854775807000  9223372036854775807000).
    small := #( 9223372036854775296000                      1
               -9223372036854775807001 -9223372036854775807001).
    big doIndexes: [:i |
      self optional: #toBeSpecified
           assert: (small at: i)  equals: ((big   at: i) min: (small at: i)).
      self optional: #toBeSpecified
           assert: (small at: i)  equals: ((small at: i) min: (big   at: i)) ].
  )

  testMax = (
    "We need to test numbers that are 64bit or less, larger than 64bit,
     positive, and negative"
    | big small |
    big   := #(1 100 9223372036854775807 -50 -2147483648).
    small := #(0  52 9223372036854775296 -51 -2147483650).
    big doIndexes: [:i |
      self assert: (big at: i)  equals: ((big   at: i) max: (small at: i)).
      self assert: (big at: i)  equals: ((small at: i) max: (big   at: i)) ].

    big   := #( 9223372036854775807000
                9223372036854775807000
               -9223372036854775807000
                9223372036854775807000).
    small := #( 9223372036854775296000
                                     1
               -9223372036854775807001
               -9223372036854775807001).

    big doIndexes: [:i |
      self optional: #toBeSpecified
          assert: (big at: i)  equals: ((big   at: i) max: (small at: i)).
      self optional: #toBeSpecified
          assert: (big at: i)  equals: ((small at: i) max: (big   at: i)) ].
```

```
  )
)
```

TestSuite/PreliminaryTest.som

```
"

$Id: PreliminaryTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

"... something just a bit complicated that tests iteration with
blocks, so that we might fail here rather than when the other tests
start, in case things are broken."

PreliminaryTest = TestCase (

    testBasicSanity = (
        | sum |
        sum := 0.
        1, 2, 3 do: [ :i |
            sum := sum + i.
            i<2 ifTrue: [ sum := sum*2 ].
            i>2 ifFalse: [ sum := sum*2 ] ].
        self assert: 15 equals: sum
    )

)
```

TestSuite/ReflectionTest.som

```
"

$Id: ReflectionTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

ReflectionTest = TestCase (
  testResondsTo = (
    self assert: (Object new respondsTo: #isNil).
    self assert: (23 respondsTo: #isNil).
    self assert: (23 respondsTo: #+).
  )

  testMethods = (
    "First method in Object should be #class."
    self assert: #class equals: (Object methods at: 1) signature.
    self assert: (Object hasMethod: #==).
  )

  testPerform = (
    | o |
    self assert: Integer equals: (23 perform: #class).
    self assert: (23 perform: #between:and: withArguments: (Array with: 22 with: 24)).

    o := SuperTest new.
    self assert: #super equals: (o perform: #something inSuperclass: SuperTestSuperClass).

    "Trying to see whether the stack in bytecode-based SOMs works properly"
    self assert: #a equals: ((23 perform: #class) = Integer ifTrue: [#a] ifFalse: [#b]).

```

```
    self assert: 28 equals: 5 + (23 perform: #value).
  )

  testInstVarAtAndPut = (
    | tmp |
    "Testing #at: and #at:put:"
    tmp := Pair withKey: 3 andValue: 42.

    self assert: tmp key equals: (tmp instVarAt: 1).

    tmp instVarAt: 1 put: #foo.
    self assert: #foo    equals: tmp key.
  )

  testName = (
    self assert: #Object equals: Object name.
    self assert: #'Object class' equals: Object class name.
    self assert: #Integer equals: 1 class name.
  )

  testAsString = (
    self assert: 'Object' equals: Object asString.
    self assert: 'Object class' equals: Object class asString.
    self assert: 'Integer' equals: 1 class asString.
  )

  testSelectors = (
    | sels |
    sels := Object selectors.
    self assert: 32 equals: sels length.

    sels contains: #=.
    self assert: (Object hasMethod: #=).

    sels contains: #value.
    self assert: (Object hasMethod: #value).

    sels contains: #notNil.
    self assert: (Object hasMethod: #notNil).
  )
)
```

TestSuite/SelfBlockTest.som

```
"

$Id: SelfBlockTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SelfBlockTest = TestCase (

  testEscapedBlock = (
    self assert: 42 equals: self give42 value
  )

  give42 = (
    ^[ self giveBlock value ]
  )

  giveBlock = (
    ^self returnBlock value
  )

  returnBlock = (
    ^[ self returnBlock2 value ]
  )

  returnBlock2 = (
    ^[ 42 ]
  )
)

```

TestSuite/SetTest.som

```
SetTest = TestCase (
  testBasics = (
    | a b t |
    a := Set new.
    b := Set new.

    a add: #a.
    b add: #b.

    self deny: a = b.

    t := Set new.
    t add: #a.

    self deny: a == t. "different objects"
    self assert: a equals: t. "but with equal value"
  )

  testEquality = (
    | a b |
    a := Set new.
    a addAll: #(1 2 3 4).

    b := Set new.
    b addAll: #(1 2 3 4).

    self assert: a = b.
    self deny: a == b.

    a add: 5.

    self deny: a = b.
    b add: 5.

    self assert: a = b.

    b add: #foo.
    self deny: a = b.
    a add: #foo.
    self assert: a = b.
  )

  testUnion = (
    | a b u |
    a := Set new.
    b := Set new.
    a addAll: #(1 2 3 4).
    b addAll: #(1 2 3 4).

    u := a union: b.
    self assert: a = b.
    self assert: u = a.
    self assert: u = b.

    self deny: a == b.
    self deny: u == a.
    self deny: u == b.

```

```
    self assert: 4 equals: u size.

    a add: #mm.
    u := a union: b.
    self assert: 5 equals: u size.

    b add: #oo.
    u := a union: b.
    self assert: 6 equals: u size.

    b add: #mm.
    u := a union: b.
    self assert: 6 equals: u size.
  )

  testIntersection = (
    | a b i |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    i := a intersection: b.
    self assert: 4 equals: i size.

    a do: [:e |
      self assert: (i contains: e) ].

    b := Set new.
    b add: 64.
    b add: 667.

    i := a intersection: b.
    self assert: 2 equals: i size.

    b do: [:e |
      self assert: (i contains: e) ].
  )

  testSetDifference = (
    | a b d |
    a := Set new.
    b := Set new.
    a addAll: #(43 64 730 667).
    b addAll: #(43 64 730 667).

    d := a - b.
    self assert: d isEmpty.
    self assert: 0 equals: d size.

    b := Set new.
    b add: 43.
    b add: 667.
    b add: 345345.

    d := a - b.
    self assert: 2 equals: d size.
    self assert: (d contains: 64).
    self assert: (d contains: 730).
  )

```

```
testContains = (
  | s |
  s := Set new.

  self deny: (s contains: #'333').
  s add: #'333'.

  self assert: (s contains: #'333').
  s add: 333.
  self assert: (s contains: #'333').
  self assert: (s contains: 333).
)

testRemove = (
  | s o |
  o := Object new.
  s := Set new.
  s add: #sfsdf.
  s add: 323.
  s add: 545.
  s add: self.
  s add: o.

  self assert: (s contains: o).
  self assert: 5 equals: s size.

  s remove: 323.
  self assert: 4 equals: s size.
  self deny: (s contains: 323).
  s add: 65767.
  self assert: 5 equals: s size.

  s remove: o.
  self assert: 4 equals: s size.
  self deny: (s contains: o).
  s remove: self.
  s remove: #sfsdf.
  s remove: 323.
  s remove: 545.
  s remove: 65767.

  self assert: s isEmpty.
)

testFirst = (
  | s |
  s := Set new.
  s addAll: #(233 545 665).

  self assert: 233 equals: s first.

  s remove: 233.
  self assert: 545 equals: s first.

  s remove: 545.
  self assert: 665 equals: s first.
)

testCollect = (
  | s r |
  s := Set new.
```

```
    s addAll: #(21 54642 6753 344 655).

    r := s collect: [:e | e % 10 ].

    self assert: 5 equals: r size.
    self assert: 1 equals: r first.
    self assert: 2 equals: (r at: 2).
  )
)

```

TestSuite/SpecialSelectorsTest.som

```
SpecialSelectorsTest = TestCase (
  testMinusMinsPrefix = (
    self assert: self --> 1 equals: 1.
    self assert: self -- 1 equals: 1.
  )

  --> aValue = (
    ^1
  )

  -- aValue = (
  •ã
  )
)

```

TestSuite/StringTest.som

```
"
Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

StringTest = TestCase (

  testEquality = (
    | str1 str2 |
    str1 := 'foo'.
    str2 := 'bar'.

    self assert: str1 = str1.
    self assert: str1 = 'foo'.
    self assert: str1 = ('f' + 'oo').
    self deny:   str1 = str2.
    self assert: str2 = str2.

    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o').
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asString.
    self assert: ('f' + 'o' + 'o') = ('f' + 'o' + 'o') asSymbol.
    self assert: ('f' + 'o' + 'o') = #foo.
  )

  testEqualEqual = (
    | str1 |
    str1 := 'foo'.
    self assert: str1 == str1.
    self deny:   str1 == str1 asSymbol.
    self deny:   str1 == #foo.
  )

  testLength = (
    self assert: 1 equals: 't' length.
    self assert: 6 equals: ('foo' + 'bar') length.
  )
```

```

  testCharAt = (
    | str |
    str := 'foobar'.
    self assert: 'f' equals: (str charAt: 1).
    self assert: 'o' equals: (str charAt: 2).
    self assert: 'o' equals: (str charAt: 3).
    self assert: 'b' equals: (str charAt: 4).
    self assert: 'a' equals: (str charAt: 5).
    self assert: 'r' equals: (str charAt: 6).
  )

  testStringLiteralLineBreak = (
    | str |
    "Some parsers get the literals and line bounderies wrong"
    str := '
'.
    self assert: '\n' equals: (str charAt: 1).
    self assert: 1 equals: str length.
  )

  testPrimSubstringFrom = (
    | str |
    str := 'foobar'.
    self assert: 'foo'    equals: (str primSubstringFrom: 1 to: 3).
    self assert: 'bar'    equals: (str primSubstringFrom: 4 to: 6).
    self assert: 'foobar' equals: (str primSubstringFrom: 1 to: 6).
    self assert: 'oob' equals: ('foobar' substringFrom: 2 to: 4).
  )

  testSplit = (
    | r |
    r := 'aaaa' split: ','.
    self assert: 1 equals: r length.
    self assert: 'aaaa' equals: (r at: 1).

    r := 'foo.bar' split: '.'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo..bar' split: '.'.
    self assert: 3     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: ''    equals: (r at: 2).
    self assert: 'bar' equals: (r at: 3).

    r := 'foo..bar' split: '..'.
    self assert: 2     equals: r length.
    self assert: 'foo' equals: (r at: 1).
    self assert: 'bar' equals: (r at: 2).

    r := 'foo' split: 'bar'.
    self assert: 1     equals: r length.
    self assert: 'foo' equals: (r at: 1).

    self assert: Array is: r class
  )

  testIndexOf = (
    self assert: -1 equals: ('foo' indexOf: 'b').
```

```
    self assert:  1 equals: ('foo' indexOf: 'f').
    self assert:  2 equals: ('foo' indexOf: 'o').
    self assert:  3 equals: ('foo' indexOf: 'o' startingAt: 3).

    self assert: -1 equals: ('foo' indexOf: 'b' startingAt: 4).

    self assert:  2 equals: ('foo' indexOf: 'oo').
  )

  testBeginsWith = (
    self deny:   ('foo' beginsWith: 'oo').
    self assert: ('foo' beginsWith: 'foo').
  )

  testEndsWith = (
    self assert: ('foo' endsWith: 'foo').
    self assert: ('foo' endsWith: 'oo').
    self deny: ('f' endsWith: 'bar').
    self deny: ('f' endsWith: 'foo').
  )

  testMultiLineString = (
    "Test whether the parser will parse multi-line strings correctly."
    self assert: '
1234567890
1234567890
1234567890
1234567890
1234567890' equals: '
1234567890
1234567890
1234567890
1234567890
1234567890'
  )

  testEscapeSequences = (
    "Tests for escape sequences, not all of them are reliable represented as
     proper strings. So, we do a simple equality test, and check substring or
     length.

     \t'    F " 6† & 7FW
     \b'    & 6·7  6R 6† & 7FW
     \n'    æWvÆ–æR 6† & 7FW
     \r'    6 '&– vR &WGW&â 6† & 7FW
     \f'    f÷&ÖfVVB 6† & 7FW
     \''    6–ævÆR  V÷FR 6† & 7FW
     \\'  & 6·6Æ 6, 6† & 7FW
     \0   zero byte character
    "

    self assert: '\t' equals: '\t'.
    self assert: 1 equals: '\t' length.

    self assert: '\b' equals: '\b'.
    self assert: 1 equals: '\b' length.

    self assert: '\n' equals: '\n'.
    self assert: 1 equals: '\n' length.
```

```
    self deny: ('\n' endsWith: 'n').

    self assert: '\r' equals: '\r'.
    self assert: 1 equals: '\n' length.
    self deny: ('\r' endsWith: 'r').

    self assert: '\f' equals: '\f'.
    self assert: 1 equals: '\f' length.
    self deny: ('\f' endsWith: 'f').

    self assert: '\'' equals: '\''.
    self assert: 1 equals: '\'' length.

    self assert: '\\' equals: '\\'.
    self assert: 1 equals: '\\' length.

    self assert: '\0' equals: '\0'.
    self assert: 1 equals: '\0' length.
    self assert: 5 equals: '\0rest' length.
  )

  testHash = (
    | str |
    "Hash should be identical for strings that are identical,
     whether given literal or composed at runtime"
    self assert: 'foobar' hashcode equals: 'foobar' hashcode.
    self assert: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode
         equals: 'ssdf aksdf; kasd;fk a;dfk a;dfk a;d' hashcode.

    str := 'foo' + 'bar'.
    str := str + str.
    self assert: 'foobarfoobar' hashcode equals: str hashcode.

    str := 'dfadf fgsfg sfg sdfg sfg sfg' + '345243n 24n5 kwertlw erltnwrtln'.
    self assert: 'dfadf fgsfg sfg sdfg sfg sfg345243n 24n5 kwertlw erltnwrtln' hashcode
         equals: str hashcode.
  )

  testWhiteSpace = (
    self assert: ' ' isWhiteSpace.
    self assert: '\t' isWhiteSpace.
    self assert: '\t\n   \n \n' isWhiteSpace.

    self deny: '' isWhiteSpace.
    self deny: '\t\n N  \n \n' isWhiteSpace.
    self deny: 'N' isWhiteSpace.
    self deny: '3' isWhiteSpace.
  )

  testLetters = (
    self assert: 'a' isLetters.
    self assert: 'all' isLetters.
    self optional: #unicode assert: 'aOoöéÉíä' isLetters description: 'Does not support Unicode'.

    self deny: '' isLetters.
    self deny: ' ' isLetters.
    self deny: '3' isLetters.
    self deny: '3333' isLetters.
```

```
    self deny: 'aOo öéÉíä' isLetters.
    self deny: 'aOo1öéÉíä' isLetters.
  )

  testDigits = (
    self assert: '0' isDigits.
    self assert: '0123' isDigits.
    self assert: '0123456789' isDigits.

    self deny: '' isDigits.
    self deny: ' ' isDigits.
    self deny: 'S' isDigits.
    self deny: '333 3' isDigits.
    self deny: '66i77' isDigits.
    self deny: '66e7' isDigits.
    self deny: 'aOo1öéÉíä' isDigits.
  )

  testAsInteger = (
    self assert:   0 equals:   '0' asInteger.
    self assert: 100 equals: '100' asInteger.
    self assert: 923 equals: '923' asInteger.

    self assert:   -0 equals:   '-0' asInteger.
    self assert: -100 equals: '-100' asInteger.
    self assert: -923 equals: '-923' asInteger.

    self assert: 123342353453453456456456  equals: '123342353453453456456456' asInteger.
  )
)
```

TestSuite/SuperTest.som

```
"

Copyright (c) 2007-2018 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SuperTest = SuperTestSuperClass (

  testSuper = (
    self assert: 42 equals: self give42.
    self assert: 42 equals: self blockGive42.
  )

  yourself = (
    record := record + 1000.
    ^ self
  )

  give42 = (
    ^super give42
  )

  blockGive42 = (
    ^[ super give42 ] value
  )

  something = (
    ^ #sub
  )

  number = (
    ^ 10
  )

```

```
+ other = (
  ^ 11
)

++++ other = (
  ^ 111
)

keyword: other = (
  ^ 1111
)

testBasicUnary = (
  self assert: 10 equals: self number.
  self assert:  1 equals: super number.
)

testBasicBinary = (
  self assert: 11 equals: self + 3.
  self assert: 22 equals: super + 5.
)


testBasicBinaryNonStandardOperator = (
  self assert: 111 equals: self ++++ 3.
  self assert: 222 equals: super ++++ 5.
)

testBasicKeyword = (
  self assert: 1111 equals: (self keyword: 3).
  self assert: 2222 equals: (super keyword: 5).
)

testWithBinaryUnaryMessage = (
  | val |
  record := 0.
  val := super number * super number.
  self assert: 1 equals: val.
)

testWithBinaryUnaryUnaryMessage = (
  | val |
  record := 0.
  super yourself yourself @ super yourself yourself.
  self assert: 2002 equals: record.
)

testWithKeywordUnaryUnaryMessage = (
  | val |
  record := 0.
  super key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.

  record := 0.
  self key: super yourself yourself key: super yourself yourself.
  self assert: 2002 equals: record.
)

"Note: testing assigning self was moved to basic interpreter tests"

testGlobalSelfDoesNotShadowKeyword = (
```

```
    | that |
    that := self.
    system global: #self put: 42.
    that optional: #selfSuperBug assert: that is: self.

    self assert: 42 equals: (system global: #self)
  )

  testGlobalSuperDoesNotShadowKeyword = (
    | that |
    that := super.
    system global: #super put: 42.
    that optional: #selfSuperBug assert: that is: super.

    self assert: 42 equals: (system global: #super)
  )
)
```

TestSuite/SuperTestSuperClass.som

```
"

$Id: SuperTestSuperClass.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SuperTestSuperClass = TestCase (
    | record |

    yourself = (
        record := record + 1.
        ^ self
    )

    give42 = (
        ^ 42
    )

    something = (
        ^ #super
    )

    number = (
        ^ 1
    )

    + other = (
      ^ 22
    )

    ++++ other = (
      ^ 222
    )
```

```

    keyword: other = (
      ^ 2222
    )

    key: a key: b = (
        ^ self
    )

    @ o = ( ^ self )
)


```

TestSuite/SymbolTest.som

```
"

$Id: SymbolTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

SymbolTest = TestCase (

  testConcatenation = (
    self assert: #a + #b is: #ab.
    self assert: #a + 'b' is: #ab.
  )

  testConversion = (
    self assert: 'gunk' equals: 'gunk' asSymbol asString.
    self assert: 'oink' equals: #oink asString.
  )

  testEquality = (
    self assert: #oink is: #oink.
    self assert: #oink is: 'oink' asSymbol.
    self assert: #oink equals: #oink.
    self assert: #oink equals: 'oink' asSymbol.

    self deny: #foo equals: #fooo.
    self deny: #foo is:     #fooo.

    self assert: #foo equals: 'foo'.
    self deny: #foo is: 'fooo'.
    self deny: #foo is: #foo asString.
  )

  testSymbolIsString = (
```

```
    self assert: (#oink beginsWith: 'oink').
    self assert: 100 equals: #'100' asInteger.
    self assert: String equals: #foo class superclass
  )

  testOperatorSymbols = (
    self assert: #~ equals: '~' asSymbol.
    self assert: #& equals: '&' asSymbol.
    self assert: #| equals: '|' asSymbol.
    self assert: #* equals: '*' asSymbol.
    self assert: #/ equals: '/' asSymbol.
    self assert: #\ equals: '\\' asSymbol.
    self assert: #+ equals: '+' asSymbol.
    self assert: #= equals: '=' asSymbol.
    self assert: #> equals: '>' asSymbol.
    self assert: #< equals: '<' asSymbol.
    self assert: #, equals: ',' asSymbol.
    self assert: #@ equals: '@' asSymbol.
    self assert: #% equals: '%' asSymbol.
    self assert: #- equals: '-' asSymbol.
  )
)
```

```
SystemTest = TestCase (

  testFullGCSupport = (
    "Test whether #fullGC is support. We expect the VM now to return true,
     to indicate the a GC was done."
    self optional: #fullGCWithEffect assert: system fullGC description: '#fullGC is not supported or has not immediate effect.'
  )

  testTicks = (
    | ticks |
    ticks := system ticks.
    self assert: ticks class equals: Integer.
    self assert: ticks > 0 description: 'Should return the microseconds since the start'
  )
)
```

TestSuite/TestCase.som

```
TestCase = (
    | testSelector runner failed |

    selector       = ( ^ testSelector )
    selector: aSym = ( testSelector := aSym )

    "asserting"
    assert: aBoolean = (
        runner countAssert.
        aBoolean ifFalse: [
            self signalFailure: 'Assertion failed' ] )

    assert: aBoolean description: aStringOrBlock = (
        runner countAssert.
        aBoolean ifFalse: [
            self signalFailure: aStringOrBlock value ] )

    assert: expected equals: actual = (
        "test value equality"
        self assert: (expected = actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    assert: expected equals: actual description: aStringOrBlock = (
        "test value equality"
        self assert: (expected = actual)
             description: aStringOrBlock
    )

    assert: expected is: actual = (
        "test reference equality"
        self assert: (expected == actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    optional: aSymbol assert: aBoolean = (
        runner countAssert.
        aBoolean ifFalse: [
            self signalUnsupported: aSymbol description: nil ] )

    optional: aSymbol assert: expected equals: actual = (
        self optional: aSymbol
             assert: (expected = actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    optional: aSymbol assert: expected is: actual = (
        self optional: aSymbol
             assert: (expected == actual)
             description: [self comparingStringBetween: expected and: actual]
    )

    optional: aSymbol assert: aBoolean description: aStringOrBlock = (
        runner countAssert.
```

```
        aBoolean ifFalse: [
            self signalUnsupported: aSymbol description: aStringOrBlock value ] )

    deny: aBoolean = (
        self assert: aBoolean not
    )

    deny: aBooleanOrBlock description: aString = (
        self assert: aBooleanOrBlock value not description: aString
    )

    deny: expected equals: actual = (
        "test value equality"
        self deny: (expected = actual)
             description: [
                'Expected ' + expected asString +
                ' to differ from ' + actual asString + '.' ]
    )

    deny: expected is: actual = (
        "test value equality"
        self deny: (expected == actual)
             description: [
                'Expected ' + expected asString +
                ' to have different identity from ' + actual asString + '.' ]
    )

    optional: aSymbol deny: aBoolean = (
        self optional: aSymbol assert: aBoolean not
    )

    optional: aSymbol deny: aBooleanOrBlock description: aString = (
        self optional: aSymbol assert: aBooleanOrBlock value not description: aString
    )

    signalFailure: aString = (
        failed := true.
        runner fail: self class name + '>>#' + testSelector
            because: aString.
    )

    signalUnsupported: aSymbol description: aDescription = (
        runner unsupported: aSymbol
                      test: self class name + '>>#' + testSelector
                   because: aDescription.
    )

    comparingStringBetween: expected and: actual = (
        ^ 'Expected ' + expected asString +
          ' but was ' + actual asString + '.'
    )

    "running"
    run: aRunner = (
        runner := aRunner.
        failed := false.

        self setUp.
```

```
        self performTest.
        self tearDown.

        failed ifFalse: [
            runner passed: self class name + '>>#' + testSelector
        ].
    )

    setUp    = ()
    tearDown = ()

    performTest = ( self perform: testSelector )

    ----

    for: aSelector = (
        | case |
        case := self new.
        case selector: aSelector.
        ^ case
    )

    tests = (
        | tests |
        tests := Vector new: self methods length.
        self methods do: [:m |
            (m signature beginsWith: #test) ifTrue: [
                tests append: (self for: m signature).
            ].
        ].

        ^ tests
    )
)
```

TestSuite/TestHarness.som

```
"

$Id: TestHarness.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2001-2013 see AUTHORS file

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

TestHarness = (
    | failOnUnsupportedOptionals |


    tests = ( "Now ordered by alphabetical order to improve maintainability"
        ^ EmptyTest,
          SpecialSelectorsTest,
          ArrayTest,
          BlockTest,
          BooleanTest,
          ClassLoadingTest,
          ClassStructureTest,
          ClosureTest,
          CoercionTest,
          CompilerReturnTest,
          DictionaryTest,
          DoesNotUnderstandTest,
          DoubleTest,
          GlobalTest,
          HashTest,
          IntegerTest,
          PreliminaryTest,
          ReflectionTest,
          SelfBlockTest,
          SetTest,
          StringTest,
          SuperTest,
```

```
        SymbolTest,
        SystemTest,
        VectorTest
  )


  runAllSuites = (
    | totalTestNum successfulTestNum unsupportedTestNum totalAssertionNum |
    totalTestNum := 0.
    unsupportedTestNum := 0.
    successfulTestNum := 0.
    totalAssertionNum := 0.

    self tests do: [ :test |
      | runner |
      runner := TestRunner new.
      runner initializeOn: test.
      runner runAllTests.
      (runner hasUnsupported or: [runner hasFailures])
        ifTrue: [
          'Test Suite: ' print.
          test name println.
          runner overviewReport.
          '' println ].

      totalTestNum       := totalTestNum + runner expectedPasses.
      unsupportedTestNum := unsupportedTestNum + runner actualUnsupported.
      successfulTestNum  := successfulTestNum + runner actualPasses.
      totalAssertionNum  := totalAssertionNum + runner numAsserts.
    ].

    'Total number of tests:           ' print.
    totalTestNum println.
    'Number of unsupported optionals: ' print.
    unsupportedTestNum println.
    'Number of successful tests:      ' print.
    successfulTestNum println.
    'Number of assertions tested:     ' print.
    totalAssertionNum println.

    (failOnUnsupportedOptionals and: [unsupportedTestNum > 0])
      ifTrue: [system exit: 1].
    totalTestNum = successfulTestNum
      ifFalse: [system exit: 1].
  )

  runOneSuite: name = (
    | testName runner |
    testName := name.
    (testName endsWith: 'Test') ifFalse: [
      testName := testName + 'Test'].

    runner := TestRunner new.
    runner initializeOn: (system resolve: testName asSymbol).
    runner run.
    runner hasFailures ifTrue: [system exit: 1]
  )

  run: args = (
```

```
        failOnUnsupportedOptionals := false.

        args length = 1 ifTrue:  [ self runAllSuites. ].
        args length = 2 ifTrue:  [
          ((args at: 2) beginsWith: '--') ifTrue: [
            (args at: 2) = '--help' ifTrue: [
              'TestHarness.som [--help] [--fail-on-optionals] [TestSuiteName]' println.
              system exit: 0 ].

            (args at: 2) = '--fail-on-optionals' ifTrue: [
              failOnUnsupportedOptionals := true ].

            self runAllSuites
          ] ifFalse: [
            self runOneSuite: (args at: 2) ]].
      ^ 0
    )
)
```

```
TestRunner = (
  | suite passes unsupported failures numAsserts |

  initializeOn: aSuite = (
    suite := aSuite.

    passes       := Vector new.
    unsupported  := Vector new.
    failures     := Vector new.

    numAsserts := 0.
  )

  hasUnsupported  = ( ^ unsupported size > 0 )
  hasFailures     = ( ^ failures size > 0 )

  actualUnsupported = ( ^ unsupported size )
  expectedPasses = ( ^ suite tests size )
  actualPasses   = ( ^ passes size )

  run = (
    self reportPreRun.
    self runAllTests.
    self reportPostRun.
    self overviewReport.
  )

  countAssert = (
    numAsserts := numAsserts + 1.
  )

  numAsserts = (
    ^ numAsserts
  )

  reportPreRun = (
    ('TestSuite ' + suite name + ':') println.
    ('Tests: ' + suite tests size asString) println.
  )

  reportPostRun = (
    self hasUnsupported ifTrue: [
      ('Unsupported optional: ' + unsupported size asString) println
    ].
    self hasFailures ifTrue: [
      ('Failures: ' + failures size asString) println
    ].
  )

  runAllTests = (
    suite tests do: [ :each |
        each run: self ].
  )

  overviewReport = (
    ('Tests passed: ' + passes size asString) println.

    (self hasFailures or: [self hasUnsupported]) ifTrue: [
```

```
        '------------------------------' println ].

    self hasUnsupported ifTrue: [
      | lastCategory |
      ('Unsupported optional features: ' + unsupported size asString) println.
      unsupported do: [:each |
        | cat |
        cat := each at: 1.
        cat == lastCategory ifFalse: [
          lastCategory := cat.
          ('\t' + cat) println ].
        ('\t\t' + (each at: 2) asString) println.
        ('\t\t\t' + (each at: 3) value asString) println ].
    ].

    self hasFailures ifTrue: [
      ('Failures: ' + failures size asString) println.
      failures do: [:each |
        ('    ' + each key asString) println.
        ('        ' + each value asString) println ].
    ].
  )

  fail: aSignature because: aReason = (
    | pair |
    pair := Pair withKey: aSignature andValue: aReason.
    failures append: pair.
  )

  unsupported: aSymbol test: aSignature because: aReason = (
    | array |
    array := Array with: aSymbol with: aSignature with: aReason.
    unsupported append: array.
  )

  passed: aSignature = (
    passes append: aSignature
  )
)
```

TestSuite/VectorTest.som

```
"

$Id: ArrayTest.som 30 2009-07-31 12:20:25Z michael.haupt $

Copyright (c) 2007-2013 see AUTHORS file
Software Architecture Group, Hasso Plattner Institute, Potsdam, Germany
http://www.hpi.uni-potsdam.de/swa/

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the 'Software'), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
"

VectorTest = TestCase (
  | a |

  setUp = (
    a := Vector new.
    a append: 'hello'.
    a append: #world.
    a append: 23.
  )

  testSize = (
    self assert: 3 equals: a size.
  )

  testAt = (
    self assert: #world equals: (a at: 2).
    self assert:     23 equals: (a at: 3).
  )

  testAtPut = (
    self assert: 'hello' equals: (a at: 1).
    a at: 1 put: 11.

    self assert: 11 equals: (a at: 1).

    self assert: #world equals: (a at: 2).
```

```
  self assert:     23 equals: (a at: 3).

  a at: 3 put: 33.
  self assert: 33 equals: (a at: 3).
)

testAtAfterRemoveFirst = (
  self assert: 'hello' equals: (a at: 1).
  a removeFirst.
  self assert: #world equals: (a at: 1).
  self assert:     23 equals: (a at: 2).
)

testAtPutAfterRemoveFirst = (
  a removeFirst.
  a at: 1 put: 11.
  a at: 2 put: 22.

  self assert: 11 equals: (a at: 1).
  self assert: 22 equals: (a at: 2).

  self assert: 2 equals: a size
)

testFirst = (
  | v |
  self assert: 'hello' equals: a first.

  v := Vector new.
  1 to: 10 do: [:i |
    v append: i.
    self assert: 1 equals: v first ].

  1 to: 10 do: [:i |
    self assert: i equals: v first.
    v removeFirst ]
)

testLast = (
  | v |
  self assert: 23 equals: a last.

  v := Vector new.
  1 to: 10 do: [:i |
    v append: i.
    self assert: i equals: v last ].

  10 downTo: 1 do: [:i |
    self assert: i equals: v last.
    v remove ]
)

testContains = (
  self assert: (a contains: 'hello').
  self assert: (a contains: #world).
  self assert: (a contains: 23).
  self deny: (a contains: #nono).
)

testContainsAfterRemoveFirst = (
  a removeFirst.
```

```
    self deny: (a contains: 'hello').
    self assert: (a contains: #world).
    self assert: (a contains: 23).
    self deny: (a contains: #nono).

    a removeFirst.

    self deny: (a contains: 'hello').
    self deny: (a contains: #world).
    self assert: (a contains: 23).
    self deny: (a contains: #nono).

    a removeFirst.

    self deny: (a contains: 'hello').
    self deny: (a contains: #world).
    self deny: (a contains: 23).
    self deny: (a contains: #nono).
  )

  testContainsAfterRemovals = (
    self deny: (a contains: nil).

    a removeFirst.
    self deny: (a contains: nil).

    a remove.
    self deny: (a contains: nil).

    self assert: (a contains: #world).
  )

  testAppendAndRemoveFirst = (
    | v |
    v := Vector new: 10.
    1 to: 100 do: [:i |
      v append: i ].

    "160 is implementation dependent, just here for orientation"
    self assert: 160 equals: v capacity.
    self assert: 100 equals: v size.

    1 to: 100 do: [:i |
      v removeFirst ].
    1 to: 100 do: [:i |
      v append: i ].

    self assert: 320 equals: v capacity.
    self assert: 100 equals: v size.
  )

  testIndexOf = (
    | v |
    v := Vector new: 3.
    1 to: 17 do: [:i |
      v append: i ].

    self assert: -1 equals: (v indexOf: nil).
    self assert: 1 equals: (v indexOf: 1).
```

```
    self assert: 13 equals: (v indexOf: 13).
    v at: 13 put: #test.

    self assert: 13 equals: (v indexOf: #test).

    v removeFirst.
    self assert: -1 equals: (v indexOf: 1).

    1 to: 12 do: [:i |
      v removeFirst ].
    self assert: -1 equals: (v indexOf: #test).
  )

  testAppendAll = (
    | v c i |
    v := Vector new: 2.
    v append: 1.
    v append: 2.
    v append: 3.

    c := Array with: 4 with: 5 with: 6.

    v appendAll: c.
    i := 1.

    v do: [:e |
      self assert: i equals: e.
      i := i + 1 ]
  )

  testAsArray = (
    | v arr |
    v := Vector new.
    self assert: 0 equals: v asArray length.

    v append: 1.
    v append: 2.

    arr := v asArray.
    self assert: 2 equals: arr length.
    self assert: 1 equals: (arr at: 1).
    self assert: 2 equals: (arr at: 2).
  )

  testAsArrayAfterRemoveFirst = (
    | v arr |
    v := Vector new.
    v append: 1.
    v append: 2.
    v append: 5.

    v removeFirst.

    arr := v asArray.
    self assert: 2 equals: arr length.
    self assert: 2 equals: (arr at: 1).
    self assert: 5 equals: (arr at: 2).
  )

  testAsSet = (
    | v set |
```

```
    v := Vector new.
    v append: 1.
    v append: 2.
    v append: 3.
    v append: 4.
    self assert: 4 equals: v size.

    set := v asSet.
    self assert: 4 equals: set size.

    v append: 1.
    v append: 1.
    v append: 1.

    self assert: 4 + 3 equals: v size.

    set := v asSet.
    self assert: 4 equals: set size.
  )

  testIsEmpty = (
    | v |
    v := Vector new.
    self assert: v isEmpty.

    v append: 1.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.

    v append: #ee.
    self deny: v isEmpty.

    v removeFirst.
    self assert: v isEmpty.
  )

  testRemoveObj = (
    | v |
    v := Vector new.
    v append: #a.
    v append: #b.
    v append: #c.
    v append: #d.
    v append: #e.
    v append: #f.
    v append: #g.
    v append: #h.

    self assert: 8 equals: v size.

    self deny: (v remove: #aa).
    self assert: (v remove: #e).
    self assert: 7 equals: v size.
  )

  testAppendComma = (
    | v |
    v := Vector new.
    v, #a.
```

```
    v, #b.

    self assert: 2 equals: v size.
    self assert: (v contains: #a).
    self assert: (v contains: #b).
  )

  testDoIndexes = (
    | i v |
    v := Vector new.
    v doIndexes: [:j |
      self assert: false ].

    v appendAll: #(1 2 3 4 5).
    i := 1.
    v doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )

  testDoIndexesAfterRemoveFirst = (
    | i v |
    v := Vector new.
    v appendAll: #(1 2 3 4 5).

    v removeFirst.

    i := 1.
    v doIndexes: [:j |
      self assert: i equals: j.
      i := i + 1 ].

    self assert: 5 equals: i.
  )

  testDo = (
    | i v |
    v := Vector new.
    v appendAll: #(1 2 3 4 5).
    i := 1.
    v do: [:v |
      self assert: i equals: v.
      i := i + 1.
    ].
    self assert: 6 equals: i.
  )
)
```

codespeed.conf

```
# -*- mode: yaml -*-
# Config file for ReBench
default_experiment: all
default_data_file: 'codespeed.data'

reporting:
    codespeed:
        url: https://som-speed.stefan-marr.de/result/add/json/

runs:
    max_invocation_time: 6000

benchmark_suites:
    macro-startup:
        gauge_adapter: RebenchLog
        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 "
        benchmarks:
            - Richards:
                extra_args: "1 0 1"
                codespeed_name: "Richards [>"
            - DeltaBlue:
                extra_args: "1 0 1000"
                codespeed_name: "DeltaBlue [>"
            - Mandelbrot:
                extra_args: "1 0 300"
                codespeed_name: "Mandelbrot [>"
            - NBody:
                extra_args: "1 0 30000"
                codespeed_name: "NBody [>"
            - Json:
                extra_args: "1 0 80"
                codespeed_name: "Json [>"
            - GraphSearch:
                extra_args: "1 0 30"
                codespeed_name: "GraphSearch [>"
            - PageRank:
                extra_args: "1 0 1400"
                codespeed_name: "PageRank [>"

    macro-steady:
        gauge_adapter: RebenchLog
        command: *MACRO_CMD
        benchmarks:
            - Richards:
                extra_args: "130 0 60"
                codespeed_name: "Richards >]"
                warmup: 30
            - DeltaBlue:
                extra_args: "120 0 20000"
                codespeed_name: "DeltaBlue >]"
                warmup: 20
            - Mandelbrot:
                extra_args: "110 0 1000"
```

```
                codespeed_name: "Mandelbrot >]"
                warmup: 10
            - NBody:
                extra_args: "120 0 500000"
                codespeed_name: "NBody >]"
                warmup: 20
            - Json:
                extra_args: "120 0 80"
                codespeed_name: "Json >]"
                warmup: 20
            - GraphSearch:
                extra_args: "250 0 30"
                codespeed_name: "GraphSearch >]"
                warmup: 100
            - PageRank:
                extra_args: "120 0 1400"
                codespeed_name: "PageRank >]"
                warmup: 20

    micro-startup-100:
        gauge_adapter: RebenchLog
        command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/BenchmarkHarness.som %(benchmark)s "
        benchmarks:
            - Fibonacci:
                extra_args: "1 0 100"
                codespeed_name: "Fibonacci 100x [>"
            - Dispatch:
                extra_args: "1 0 1000"
                codespeed_name: "Dispatch 100x [>"
            - Bounce:
                extra_args: "1 0 100"
                codespeed_name: "Bounce 100x [>"
            - Loop:
                extra_args: "1 0 500"
                codespeed_name: "Loop 100x [>"
            - Permute:
                extra_args: "1 0 50"
                codespeed_name: "Permute 100x [>"
            - Queens:
                extra_args: "1 0 50"
                codespeed_name: "Queens 100x [>"
            - List:
                extra_args: "1 0 50"
                codespeed_name: "List 100x [>"
            - Recurse:
                extra_args: "1 0 100"
                codespeed_name: "Recurse 100x [>"
            - Storage:
                extra_args: "1 0 20"
                codespeed_name: "Storage 100x [>"
            - Sieve:
                extra_args: "1 0 100"
                codespeed_name: "Sieve 100x [>"
            - BubbleSort:
                extra_args: "1 0 100"
                codespeed_name: "BubbleSort 100x [>"
            - QuickSort:
                extra_args: "1 0 20"
                codespeed_name: "QuickSort 100x [>"
```

```
                - Sum:
                    extra_args: "1 0 500"
                    codespeed_name: "Sum 100x [>"
                - Towers:
                    extra_args: "1 0 20"
                    codespeed_name: "Towers 100x [>"
                - TreeSort:
                    extra_args: "1 0 10"
                    codespeed_name: "TreeSort 100x [>"
                - IntegerLoop:
                    extra_args: "1 0 400"
                    codespeed_name: "IntegerLoop 100x [>"
                - FieldLoop:
                    extra_args: "1 0 50"
                    codespeed_name: "FieldLoop 100x [>"
                - WhileLoop:
                    extra_args: "1 0 1000"
                    codespeed_name: "WhileLoop 100x [>"

        micro-startup:
            gauge_adapter: RebenchLog
            command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/BenchmarkHarness.som %(benchmark)s "
            benchmarks:
                - Fibonacci:
                    extra_args: "1 0 3"
                    codespeed_name: "Fibonacci [>"
                - Dispatch:
                    extra_args: "1 0 20"
                    codespeed_name: "Dispatch [>"
                - Bounce:
                    extra_args: "1 0 2"
                    codespeed_name: "Bounce [>"
                - Loop:
                    extra_args: "1 0 10"
                    codespeed_name: "Loop [>"
                - Permute:
                    extra_args: "1 0 3"
                    codespeed_name: "Permute [>"
                - Queens:
                    extra_args: "1 0 2"
                    codespeed_name: "Queens [>"
                - List:
                    extra_args: "1 0 2"
                    codespeed_name: "List [>"
                - Recurse:
                    extra_args: "1 0 3"
                    codespeed_name: "Recurse [>"
                - Storage:
                    extra_args: "1 0 2"
                    codespeed_name: "Storage [>"
                - Sieve:
                    extra_args: "1 0 5"
                    codespeed_name: "Sieve [>"
                - BubbleSort:
                    extra_args: "1 0 3"
                    codespeed_name: "BubbleSort [>"
                - QuickSort:
                    extra_args: "1 0 3"
                    codespeed_name: "QuickSort [>"
```

```
            - Sum:
                extra_args: "1 0 10"
                codespeed_name: "Sum [>"
            - Towers:
                extra_args: "1 0 2"
                codespeed_name: "Towers [>"
            - TreeSort:
                extra_args: "1 0 1"
                codespeed_name: "TreeSort [>"
            - IntegerLoop:
                extra_args: "1 0 8"
                codespeed_name: "IntegerLoop [>"
            - FieldLoop:
                extra_args: "1 0 3"
                codespeed_name: "FieldLoop [>"
            - WhileLoop:
                extra_args: "1 0 30"
                codespeed_name: "WhileLoop [>"

    micro-steady-100:
        gauge_adapter: RebenchLog
        command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/BenchmarkHarness.som %(benchmark)s "
        benchmarks:
            - Fannkuch:
                extra_args: "55 0 9"
                codespeed_name: "Fannkuch 100x >]"
                warmup: 5
            - Fibonacci:
                extra_args: "60 0 1000"
                codespeed_name: "Fibonacci 100x >]"
                warmup: 10
            - Dispatch:
                extra_args: "55 0 10000"
                codespeed_name: "Dispatch 100x >]"
                warmup: 5
            - Bounce:
                extra_args: "60 0 4000"
                codespeed_name: "Bounce 100x >]"
                warmup: 10
            - Loop:
                extra_args: "55 0 10000"
                codespeed_name: "Loop 100x >]"
                warmup: 5
            - Permute:
                extra_args: "55 0 1500"
                codespeed_name: "Permute 100x >]"
                warmup: 5
            - Queens:
                extra_args: "55 0 1000"
                codespeed_name: "Queens 100x >]"
                warmup: 5
            - List:
                extra_args: "65 0 1000"
                codespeed_name: "List 100x >]"
                warmup: 15
            - Recurse:
                extra_args: "65 0 2000"
                codespeed_name: "Recurse 100x >]"
                warmup: 15
```

```
            - Storage:
                extra_args: "60 0 1000"
                codespeed_name: "Storage 100x >]"
                warmup: 10
            - Sieve:
                extra_args: "60 0 2500"
                codespeed_name: "Sieve 100x >]"
                warmup: 10
            - BubbleSort:
                extra_args: "60 0 3000"
                codespeed_name: "BubbleSort 100x >]"
                warmup: 10
            - QuickSort:
                extra_args: "60 0 2000"
                codespeed_name: "QuickSort 100x >]"
                warmup: 10
            - Sum:
                extra_args: "55 0 10000"
                codespeed_name: "Sum 100x >]"
                warmup: 5
            - Towers:
                extra_args: "55 0 1000"
                codespeed_name: "Towers 100x >]"
                warmup: 5
            - TreeSort:
                extra_args: "60 0 1000"
                codespeed_name: "TreeSort 100x >]"
                warmup: 10
            - IntegerLoop:
                extra_args: "55 0 8000"
                codespeed_name: "IntegerLoop 100x >]"
                warmup: 5
            - FieldLoop:
                extra_args: "55 0 900"
                codespeed_name: "FieldLoop 100x >]"
                warmup: 5
            - WhileLoop:
                extra_args: "55 0 9000"
                codespeed_name: "WhileLoop 100x >]"
                warmup: 5
    micro-steady:
        gauge_adapter: RebenchLog
        command: "-cp Smalltalk:/home/smarr/.local/SOM/Examples/Benchmarks/LanguageFeatures /home/smarr/.local/SOM/Examples/Benchmarks/BenchmarkHarness.som %(benchmark)s "
        benchmarks:
            - Fannkuch:
                extra_args: "14 0 6"
                codespeed_name: "Fannkuch >]"
                warmup: 4
            - Fibonacci:
                extra_args: "15 0 3"
                codespeed_name: "Fibonacci >]"
                warmup: 5
            - Dispatch:
                extra_args: "12 0 20"
                codespeed_name: "Dispatch >]"
                warmup: 2
            - Bounce:
                extra_args: "22 0 2"
                codespeed_name: "Bounce >]"
```

```yaml
            warmup: 12
        - Loop:
            extra_args: "14 0 10"
            codespeed_name: "Loop >]"
            warmup: 4
        - Permute:
            extra_args: "16 0 3"
            codespeed_name: "Permute >]"
            warmup: 6
        - Queens:
            extra_args: "13 0 2"
            codespeed_name: "Queens >]"
            warmup: 3
        - List:
            extra_args: "16 0 2"
            codespeed_name: "List >]"
            warmup: 6
        - Recurse:
            extra_args: "14 0 3"
            codespeed_name: "Recurse >]"
            warmup: 4
        - Storage:
            extra_args: "17 0 2"
            codespeed_name: "Storage >]"
            warmup: 7
        - Sieve:
            extra_args: "18 0 5"
            codespeed_name: "Sieve >]"
            warmup: 8
        - BubbleSort:
            extra_args: "16 0 3"
            codespeed_name: "BubbleSort >]"
            warmup: 6
        - QuickSort:
            extra_args: "15 0 3"
            codespeed_name: "QuickSort >]"
            warmup: 5
        - Sum:
            extra_args: "20 0 10"
            codespeed_name: "Sum >]"
            warmup: 10
        - Towers:
            extra_args: "20 0 2"
            codespeed_name: "Towers >]"
            warmup: 10
        - TreeSort:
            extra_args: "15 0 1"
            codespeed_name: "TreeSort >]"
            warmup: 5
        - IntegerLoop:
            extra_args: "14 0 8"
            codespeed_name: "IntegerLoop >]"
            warmup: 4
        - FieldLoop:
            extra_args: "12 0 3"
            codespeed_name: "FieldLoop >]"
            warmup: 2
        - WhileLoop:
            extra_args: "13 0 30"
            codespeed_name: "WhileLoop >]"
            warmup: 3
```

```yaml

executors:
    SOM:
        path: .
        executable: som.sh

    TruffleSOM-interpreter:
        path: .
        executable: som.sh
    TruffleSOM-graal:
        path: .
        executable: som
        args: "-E"

    TruffleSOM-interpreter-exp:
        path: .
        executable: som.sh
    TruffleSOM-graal-exp:
        path: .
        executable: som
        args: "-E"

    CSOM:
        path: .
        executable: CSOM
    SOMpp:
        path: .
        executable: som.sh
    PySOM:
        path: .
        executable: som.sh
    RPySOM-interpreter:
        path: .
        executable: RPySOM-no-jit
    RPySOM-jit:
        path: .
        executable: RPySOM-jit
    RTruffleSOM-interpreter:
        path: .
        executable: RTruffleSOM-no-jit
    RTruffleSOM-jit:
        path: .
        executable: RTruffleSOM-jit

# define the benchmarks to be executed for a re-executable benchmark run
experiments:
    SOM:
        description: All benchmarks on SOM (Java, bytecode-based)
        suites:
            - micro-startup-100
            - micro-steady-100
            - micro-startup
            - micro-steady
            - macro-startup
            - macro-steady
        executions:
            - SOM
    TruffleSOM:
        description: All benchmarks on TruffleSOM (Java, AST Interpreter)
        suites:
            - micro-startup-100
```

```
            - micro-steady-100
            - macro-startup
            - macro-steady
        executions:
            #- TruffleSOM-interpreter
            - TruffleSOM-graal
    TruffleSOM-exp:
        description: All benchmarks on TruffleSOM (Java, AST Interpreter)
        suites:
            - micro-startup-100
            - micro-steady-100
            - macro-startup
            - macro-steady
        executions:
            #- TruffleSOM-interpreter
            - TruffleSOM-graal-exp

    CSOM:
        description: All benchmarks on CSOM
        suites:
            - micro-startup
            - macro-startup
        executions:
            - CSOM
    SOMpp:
        description: All benchmarks on SOM++
        suites:
            - micro-startup
            - micro-startup-100
            - macro-startup
        executions:
            - SOMpp
    PySOM:
        description: All benchmarks on PySOM
        suites:
            - micro-startup
            - micro-steady
            - macro-startup
        executions:
            - PySOM
    RPySOM:
        description: All benchmarks on RPySOM
        suites:
            #- micro-startup
            #- micro-steady
            - micro-startup-100
            - micro-steady-100
            - macro-startup
            - macro-steady
        executions:
            #- RPySOM-interpreter
            - RPySOM-jit
    RTruffleSOM:
        description: All benchmarks on RTruffleSOM
        suites:
            #- micro-startup
            #- micro-steady
            - micro-startup-100
            - micro-steady-100
            - macro-startup
            - macro-steady
```

```
        executions:
            #- RTruffleSOM-interpreter
            - RTruffleSOM-jit
    RTruffleSOM-OMOP:
        description: All benchmarks on RTruffleSOM
        suites:
            #- micro-startup
            #- micro-steady
            - micro-startup-100
            - micro-steady-100
            - macro-startup
            - macro-steady
        executions:
            #- RTruffleSOM-interpreter
            - RTruffleSOM-jit
    TruffleSOM-OMOP:
        suites:
            - micro-startup-100
            - micro-steady-100
            - macro-startup
            - macro-steady
        executions:
            - TruffleSOM-graal
```

```markdown
# SMOG (2)

SOM and _A Little Smalltalk_ (which has always fascinated me, since about 1983) are small, OOP languages, in a pretty _pure_ sense.

**Smog** is small Go implementation of SOM.

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).

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.


### Step 0

_What's the go version of hierachical classes?_ Well, because of the _composition_ way of doing things, one can nest _structs_. (but not interfaces.)
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).


### LSP support for SOM

There is an LSP support for _vscode_ [effortless-language-servers](https://marketplace.visualstudio.com/items?itemName=MetaConcProject.effortless-language-servers)

### Why did you organize the SOM code differently?

See #4 in[Structuring Applications in Go](https://www.gobeyond.dev/structuring-applications/).
We all need to be golang coders like Ben Johnson.

```

smog/cmd/main.go

```
package main

import (
’&÷2

’&v—F‡V"æ6öÒ÷‡C fW"÷6öÒ÷6Öör
)

func main() {
—R £Ò g6ÖöråVæ—fW'6W·Ð
– &w3" £Ò ÷2ä &w5³ ¥Ð
—Rä–çFW' &WB† &w3"•
—RäW†—Bƒ •
}

// Main = (
// —'Vã¢  &w2 Ò €
// ’  Â R  &w3" À
// ’  R £Ò Væ—fW'6R æWrà
// ’   &w3" £Ò  &w2 6÷ ”g&öÓ¢ "à
// ’  R –çFW' &WC¢  &w3"à
// ’  R W†—C¢  à
// ’•
//   )

```

smog/compiler.go

```
package smog
```

smog/go.mod

```
module github.com/xt0fer/som/smog

go 1.20

```

smog/interpreter.go

```go
package smog

type Interpreter struct {
—Væ—fW'6R ¥Væ—fW'6P
–g& ÖR    ¤g& ÖP
}

func NewInterpreter(u *Universe) *Interpreter {
––’ £Ò d–çFW' &WFW'·Ð
––’çVæ—fW'6R Ò P
—&WGW&â –•
}

// "
// Frame layout:
// +-----------------+
// | Arguments       | 1
// +-----------------+
// | Local Variables | <-- localOffset
// +-----------------+
// | Stack           | <-- stackPointer
// | ...             |
// +-----------------+
// "
// |
//   "Points at the top element"
//   stackPointer
//   bytecodeIndex

//   "the offset at which local variables start"
//   localOffset

//   method
//   context
//   previousFrame
//   stack
// |

type Frame struct {
—7F 6µ ö–çFW" ¤ôôö&¦V7@
–'—FV6öFT–æFW, –çB
–Æö6 Äöfg6WB –çB
–ÖWF†öB ¤ôôö&¦V7@
–6öçFW‡B ¤ôôö&¦V7@
— &Wf–÷W4g& ÖR ¤ôôö&¦V7@
—7F 6² ¤ôôö&¦V7@
}

func NewFrame() *Frame {
–b £Ò dg& ÖW·Ð

—&WGW&â `
}
```

smog/primitives.go

```
package smog

```

smog/vm.go

```
package smog

import "os"

// | symbolTable globals classPath dumpBytecodes interpreter

// avoidExit
// lastExitCode
// exitBlock

// nilObject
// trueObject
// falseObject

// objectClass
// classClass
// metaclassClass

// nilClass
// integerClass
// arrayClass
// methodClass
// symbolClass
// primClass
// stringClass
// systemClass
// blockClass
// doubleClass

// trueClass
// falseClass
// |

// initialize = (
// symbolTable := Dictionary new.
// globals := Dictionary new.
// interpreter := Interpreter new: self.
// dumpBytecodes := false.
// avoidExit := false
// )

type ObjToObjMap map[*OOObject]*OOObject

type Universe struct {
–7–Ö&öÅF &ÆR   ö&¥Fôö&¤Ö
–vÆö& Ç2       ö&¥Fôö&¤Ö
––çFW' &WFW"   ¤–çFW' &WFW
–GV× '–FV6öFW2 &ööÀ
– fö–DW†–B     &ööÀ
}

// UNIVERSE

func (u *Universe) Exit(code int) {
–÷2äW†–B†6öFR•
}
func (u *Universe) Interpret(args []string) {

```

```
}

```

smog/vmobjects.go

```
package smog

// OOObject - an attempt to make All objects in this project this struct tied to its interface
type OOObject struct {
}

type Object interface {
—6öÔ6Æ 72,’ ¥46Æ 70
—6WE6öÔ6Æ 72† 46Æ 72 ¥46Æ 72•
}

type Sender interface {
—6VæB‡6VÆV7F÷%7G&–ær 7G&–ærÂ  &wVÖVçG2 µÒ¤ôôö&¦V7BÂ Væ—fW'6R ¥Væ—fW'6RÂ interpreter *Interpreter)
—6VæDFöW4æ÷EVæFW'7F æB‡6VÆV7F÷" 7G&–ærÂ Væ—fW'6R ¥Væ—fW'6RÂ –çFW' &WFW" *Interpreter)
—6VæEVæ¶æ÷väVÆö& Â†vÆö& Äæ ÖR ¤ôôö&¦V7BÂ Væ—fW'6R ¥Væ—fW'6RÂ –çFW' &WFW" *Interpreter)
—6VæDW66  VD&Æö6²†&Æö6² ¤ôôö&¦V7BÂ Væ—fW'6R ¥Væ—fW'6RÂ –çFW' &WFW" *Interpreter)
}

// SSymbol = SString (
// —Â çVÕ6–væ GW&T &wVÖVçG2 À

// SObject = SAbstractObject (
// —Â f–VÆG2 6Æ §¢ À

type SObject struct {
”ôôö&¦V7@
”f–VÆG2 µÔö&¦V7@
”6Æ §¢  ¥46Æ 70
}

func NewSObject(n int32, with *OOObject) *SObject {
—6ò £Ò e4ö&¦V7G·Ð
—6òäf–VÆG2 Ò Ö ¶R…µÔö&¦V7BÂ â•
•
—&WGW&â 6ð
}

// initialize: numberOfFields with: nilObject = (
//     fields := Array new: numberOfFields withAll: nilObject
//   )

// somClass = (
//
//•â 6Æ §
//
// )
func (so *SObject) somClass() *SClass {
—&WGW&â 6òä6Æ §
}

// somClass: aSClass = (
//
//–6Æ §¢ £Ò  46Æ 70
```

```
//
// )
func (so *SObject) setSomClass(aSClass *SClass) {
—6òä6Æ §¢ Ò  46Æ 70
}

// somClassIn: universe = (
//
//•â 6Æ §
//
// )
func (so *SObject) somClassIn(u *Universe) *SClass {
—&WGW&â 6òä6Æ §
}

// fieldName: index = (
//
//’$vWB F†R æ ÖR öb F†R f–VÆB v—F, F†R v—fVâ –æFW,
//•â 6Æ §¢ –ç7F æ6Tf–VÆDæ ÖS¢ –æFW€
//
// )
func (so *SObject) fieldName(index int32) string {
—&WGW&â 6òä6Æ §¢æf–VÆDæ ÖR†–æFW,•
}

// fieldIndex: name = (
//
//’$vWB F†R –æFW, f÷" F†R f–VÆB v—F, F†R v—fVâ æ ÖR
//•â 6Æ §¢ Æöö·W f–VÆD–æFWƒ¢ æ ÖP
//
// )
func (so *SObject) fieldIndex(name string) int32 {
—&WGW&â 6òä6Æ §¢æf–VÆD–æFW,†æ ÖR•
}

//   numberOfFields = (
//     "Get the number of fields in this object"
//     ^ fields length
//   )

// field: index = (
//
//’$vWB F†R f–VÆB v—F, F†R v—fVâ –æFW,
//•â f–VÆG2  C¢ –æFW€
//
// )
func (so *SObject) field(index int32) Object {
—&WGW&â 6òäf–VÆG5¶–æFW…Ð
}

// field: index put: value = (
//
//’%6WB F†R f–VÆB v—F, F†R v—fVâ –æFW, Fò F†R v—fVâ f ÇVR
//–f–VÆG2  C¢ –æFW,  WC¢ f ÇVP
//
// )
func (so *SObject) fieldPut(index int32, value Object) Object {
—6òäf–VÆG5¶–æFW…Ò Ò f ÇVP
}

//   "For using in debugging tools such as the Diassembler"
```

```
//   debugString = ( ^ 'SObject(' + clazz name string + ')' )

//   ----


// ??
type Invokable *OOObject

// SClass = SObject (
//
//—Â Væ—fW'6P
//’  7W W$6Æ 70
//’  æ ÖP
//’  –ç7F æ6T–çfö¶ &ÆW2 –ç7F æ6Tf–VÆG7À
type SClass struct {
•4ö&¦V7@
•Væ—fW'6R       ¥Væ—fW'6P
•7W W$6Æ 72     ¥46Æ 70
”æ ÖR           ¥57–Ö&öÀ
”–ç7F æ6Tf–VÆG2 µÔ–çfö¶ &ÆP
}

// SSymbol = SString (
//
//—Â çVÕ6–væ GW&T &wVÖVçG2 À
type SSymbol struct {
•57G&–æp
”çVÕ6–væ GW&T &wVÖVçG2 –ç@
}

// SString = SAbstractObject (
//
//—Â 7G&–ær À
type SString struct {
•4ö&¦V7@
•2 7G&–æp
}

func (S *SString) string() string { return S.S }

// "For using in debugging tools such as the Diassembler"
func (S *SString) debugString() string {
—B £Ò %57G&–ær," ² 2å2 ² "’
—&WGW&â @
}

// somClassIn: universe = (
//
//’  â Væ—fW'6R 7G&–æt6Æ 70
//’•
func (S *SString) somClassIn(u *Universe) *SClass {
—&WGW&â 2å4ö&¦V7Bä6Æ §
}

// initializeWith: aString = (
func NewString(aString string) *SString {
—2 £Ò e57G&–æw·Ð
—2å2 Ò  7G&–æp
—&WGW&â 0
}

```

specification/Makefile

```
SHELL = '/bin/bash'

all: SOMParser.class

antlr.jar:
—vvWB Ôò  çFÇ"æ¦ " ‡GG 3¢ò÷wwræ çFÇ"æ÷&röF÷væÆö Bö çFÇ"ÓBãrã"Ö6ö× ÆWFRæ¦

SOMParser.java: antlr.jar SOM.g4
–¦ f  Ö7   çFÇ"æ¦ " ÷&ræ çFÇ"çcBåFööÂ 4ôÒæs@

SOMParser.class: SOMParser.java
–¦ f 2 Ö7   çFÇ"æ¦ " ¢æ¦ f

test: are-we-fast-yet SOMParser.class
—6WB ÖS² À
–f–æB ââ Öæ ÖR r¢ç6öÒr × &–çC  Â v†–ÆR &V B ÖB BBuÃ r “² Fò À
™–V6†ò "BG¶—Ò#² À
™”õUCÖ ¦ f  Ö7   çFÇ"æ¦ #¢â ÷&ræ çFÇ"çcBæwV’åFW7E&–r 4ôÒ 6Æ 76FVb Ð
diagnostics "$${i}" 2>&1`; \
™––b ²   ×¢ "BDõUB" Ó² F†Vâ À
™™echo "$$OUT"; \
™™exit 1; \
™–f“² À
–FöæP

are-we-fast-yet:
–v—B 6ÆöæR ÒÖFW FƒÓ  ‡GG 3¢òöv—F‡V"æ6öÒ÷6Ö '"ö &R×vRÖf 7B×–WB  &R×vRÖf 7BÐ
yet

clean:
—&Ò ¢æ¦ f  ¢æ6Æ 72 ¢çFö¶Vç2 ¢æ–çFW'

clobber: clean
—&Ò  çFÇ"æ¦
—&Ò Õ&b  &R×vRÖf 7B×–W@

```

specification/SOM.g4

```
grammar SOM;

/* This parser accepts valid programs adhering to the following grammar. Comments
and white space are not dealt with in the grammar. Names of non-terminals begin
with a lower-case letter; terminals, with an upper-case one. */

classdef:
    Identifier Equal superclass
    instanceFields method*
    ( Separator classFields method* )?
    EndTerm;

superclass:
    Identifier? NewTerm;

instanceFields:
    ( Or variable* Or )?;

classFields:
    ( Or variable* Or )?;

method:
   pattern Equal ( Primitive | methodBlock );

pattern:
    unaryPattern | keywordPattern | binaryPattern;

unaryPattern:
    unarySelector;

binaryPattern:
    binarySelector argument;

keywordPattern:
    ( keyword argument )+;

methodBlock:
    NewTerm blockContents? EndTerm;

unarySelector:
    identifier;

binarySelector:
    Or | Comma | Minus | Equal | Not | And | Star | Div | Mod | Plus | More |
    Less | At | Per | OperatorSequence;

identifier:
    Primitive | Identifier;

keyword:
    Keyword;

argument:
    variable;

```

```
blockContents:
    ( Or localDefs Or )?
    blockBody;

localDefs:
    variable*;

blockBody:
      Exit result
    | expression ( Period blockBody? )?;

result:
    expression Period?;

expression:
    assignation | evaluation;

assignation:
    assignments evaluation;

assignments:
    assignment+;

assignment:
    variable Assign;

evaluation:
    primary messages?;

primary:
    variable | nestedTerm | nestedBlock | literal;

variable:
    identifier;

messages:
      unaryMessage+ binaryMessage* keywordMessage?
    | binaryMessage+ keywordMessage?
    | keywordMessage;

unaryMessage:
    unarySelector;

binaryMessage:
    binarySelector binaryOperand;

binaryOperand:
    primary unaryMessage*;

keywordMessage:
    ( keyword formula )+;

formula:
    binaryOperand binaryMessage*;

nestedTerm:
    NewTerm expression EndTerm;

literal:
    literalArray | literalSymbol | literalString | literalNumber;

```

```
literalArray:
    Pound NewTerm
    literal*
    EndTerm;

literalNumber:
    negativeDecimal | literalDecimal;

literalDecimal:
    literalInteger | literalDouble;

negativeDecimal:
    Minus literalDecimal;

literalInteger:
    Integer;

literalDouble:
    Double;

literalSymbol:
    Pound ( string | selector );

literalString:
    string;

selector:
    binarySelector | keywordSelector | unarySelector;

keywordSelector:
    Keyword | KeywordSequence;

string:
    STString;

nestedBlock:
    NewBlock blockPattern? blockContents? EndBlock;

blockPattern:
    blockArguments Or;

blockArguments:
    ( Colon argument )+;

/* Lexer */

Comment:   '"' ~["]* '"' -> skip;
Whitespace : [ \t\r\n]+ -> skip ;

Primitive: 'primitive';
Identifier: [\p{Alpha}] [\p{Alpha}0-9_]*;

Equal: '=';

Separator: '----' '-'*;

NewTerm: '(';
EndTerm: ')';
Or: '|';
```

```
Comma: ',';
Minus: '-';
Not:   '~';
And:   '&';
Star:  '*';
Div:   '/';
Mod:   '\\';
Plus:  '+';
More:  '>';
Less:  '<';
At:    '@';
Per:   '%';

OperatorSequence: (
    Not | And | Or | Star | Div |
    Mod | Plus | Equal | More | Less |
    Comma | At | Per | Minus )+;

Colon: ':';

NewBlock: '[';
EndBlock: ']';

Pound:  '#';
Exit:   '^';
Period: '.';
Assign: ':=';

Integer: [0-9]+;
Double: [0-9]+ '.' [0-9]+;

Keyword: Identifier Colon;

KeywordSequence: Keyword+;

STString:
    '\''
    (   '\\t'
      | '\\b'
      | '\\n'
      | '\\r'
      | '\\f'
      | '\\0'
      | '\\\''
      | '\\\\'
      |  ~('\''| '\\')
    )*
    '\'';

```