

```
1:
2:
3: [1]Chris Rathman / [2]ChrisRath@aol.com
4:
5:
6: "*****
7: * Allowable characters:
8: *   - a-z
9: *   - A-Z
10: *   - 0-9
11: *   - .+/\*~<>@%|&?
12: *   - blank, tab, cr, ff, lf
13: *
14: * Variables:
15: *   - variables must be declared before use
16: *   - shared vars must begin with uppercase
17: *   - local vars must begin with lowercase
18: *   - reserved names: nil, true, false, self, super, and Smalltalk
19: *
20: * Variable scope:
21: *   - Global: defined in Dictionary Smalltalk and accessible by all
22: *       objects in system
23: *   - Special: (reserved) Smalltalk, super, self, true, false, & nil
24: *   - Method Temporary: local to a method
25: *   - Block Temporary: local to a block
26: *   - Pool: variables in a Dictionary object
27: *   - Method Parameters: automatic local vars created as a result of
28: *       message call with params
29: *   - Block Parameters: automatic local vars created as a result of
30: *       value: message call
31: *   - Class: shared with all instances of one class & its subclasses
32: *   - Class Instance: unique to each instance of a class
33: *   - Instance Variables: unique to each instance
34: * *****
35: "Comments are enclosed in quotes"
36: "Period (.) is the statement separator"
37:
38: "*****
39: * Transcript:
40: * *****
41: Transcript clear.
42: Transcript show: 'Hello World'.
43: Transcript nextPutAll: 'Hello World'.
44: Transcript nextPut: $A.
45: Transcript space.

"clear to transcript window"
"output string in transcript window"
"output string in transcript window"
"output character in transcript window"
"output space character in transcript window"
```

```

46: Transcript tab.
47: Transcript cr.
48: 'Hello' printOn: Transcript.
49: 'Hello' storeOn: Transcript.
50: Transcript endEntry.
51:
52: "*****
53: * Assignment:
54: *****
55: | x y |
56: x _ 4.
57: x := 5.
58: x := y := z := 6.
59: x := (y := 6) + 1.
60: x := Object new.
61: x := 123 class.
62: x := Integer superclass.
63: x := Object allInstances.
64: x := Integer allSuperclasses.
65: x := 1.2 hash.
66: y := x copy.
67: y := x shallowCopy.
68: y := x deepCopy.
69: y := x veryDeepCopy.
70:
71: "*****
72: * Constants:
73: *****
74: | b |
75: b := true.
76: b := false.
77: x := nil.
78: x := 1.
79: x := 3.14.
80: x := 2e-2.
81: x := 16r0F.
82: x := -1.
83: x := 'Hello'.
84: x := 'I'm here'.
85: x := $A.
86: x := $ .
87: x := #aSymbol.
88: x := #(3 2 1).
89: x := #('abc' 2 $a).
90:
"output tab character in transcript window"
"carriage return /linefeed"
"append print string into the window"
"append store string into the window"
"flush the output buffer"

"assignment (Squeak) <-"
"assignment"
"compound assignment"

"bind to allocated instance of a class"
"discover the object class"
"discover the superclass of a class"
"get an array of all instances of a class"
"get all superclasses of a class"
"hash value for object"
"copy object"
"copy object (not overridden)"
"copy object and instance vars"
"complete tree copy using a dictionary"

"true constant"
>false constant"
"nil object constant"
"integer constants"
"float constants"
"fractional constants"
"hex constant".
"negative constants"
"string constant"
"single quote escape"
"character constant"
"character constant (space)"
"symbol constants"
"array constants"
"mixing of types allowed"

```

```
91: "*****
92: * Booleans:
93: *****
94: | b x y |
95: x := 1. y := 2.
96: b := (x = y).
97: b := (x ~= y).
98: b := (x == y).
99: b := (x ~~ y).
100: b := (x > y).
101: b := (x < y).
102: b := (x >= y).
103: b := (x <= y).
104: b := b not.
105: b := (x < 5) & (y > 1).
106: b := (x < 5) | (y > 1).
107: b := (x < 5) and: [y > 1].
108: b := (x < 5) or: [y > 1].
109: b := (x < 5) eqv: (y > 1).
110: b := (x < 5) xor: (y > 1).
111: b := 5 between: 3 and: 12.
112: b := 123 isKindOf: Number.
113: b := 123 isMemberOf: SmallInteger.
114: b := 123 respondsTo: sqrt.
115: b := x isNil.
116: b := x isZero.
117: b := x positive.
118: b := x strictlyPositive.
119: b := x negative.
120: b := x even.
121: b := x odd.
122: b := x isLiteral.
123: b := x isInteger.
124: b := x isFloat.
125: b := x isNumber.
126: b := $A isUppercase.
127: b := $A isLowercase.
128:
129: "*****
130: * Arithmetic expressions:
131: *****
132: | x |
133: x := 6 + 3.
134: x := 6 - 3.
135: x := 6 * 3.

"equals"
"not equals"
"identical"
"not identical"
"greater than"
"less than"
"greater than or equal"
"less than or equal"
"boolean not"
"boolean and"
"boolean or"
"boolean and (short-circuit)"
"boolean or (short-circuit)"
"test if both true or both false"
"test if one true and other false"
"between (inclusive)"
"test if object isclass or subclass of"
"test if object istype of class"
"test if object responds to message"
"test if object isnil"
"test if number iszero"
"test if number ispositive"
"test if number isgreater than zero"
"test if number isnegative"
"test if number iseven"
"test if number isodd"
"test if literal constant"
"test if object isinteger"
"test if object isfloat"
"test if object isnumber"
"test if upper case character"
"test if lower case character"

"addition"
"subtraction"
"multiplication"
```

```
136: x := 1 + 2 * 3.
137: x := 5 / 3.
138: x := 5.0 / 3.0.
139: x := 5.0 // 3.0.
140: x := 5.0 \ 3.0.
141: x := -5.
142: x := 5 sign.
143: x := 5 negated.
144: x := 1.2 integerPart.
145: x := 1.2 fractionPart.
146: x := 5 reciprocal.
147: x := 6 * 3.1.
148: x := 5 squared.
149: x := 25 sqrt.
150: x := 5 raisedTo: 2.
151: x := 5 raisedToInteger: 2.
152: x := 5 exp.
153: x := -5 abs.
154: x := 3.99 rounded.
155: x := 3.99 truncated.
156: x := 3.99 roundTo: 1.
157: x := 3.99 truncateTo: 1.
158: x := 3.99 floor.
159: x := 3.99 ceiling.
160: x := 5 factorial.
161: x := -5 quo: 3.
162: x := -5 rem: 3.
163: x := 28 gcd: 12.
164: x := 28 lcm: 12.
165: x := 100 ln.
166: x := 100 log.
167: x := 100 log: 10.
168: x := 100 floorLog: 10.
169: x := 180 degreesToRadians.
170: x := 3.14 radiansToDegrees.
171: x := 0.7 sin.
172: x := 0.7 cos.
173: x := 0.7 tan.
174: x := 0.7 arcSin.
175: x := 0.7 arcCos.
176: x := 0.7 arcTan.
177: x := 10 max: 20.
178: x := 10 min: 20.
179: x := Float pi.
180: x := Float e.

"evaluation alwaysleft to right (1 + 2) * 3"
"division with fractional result"
"division with float result"
"integer divide"
"integer remainder"
"unary minus"
"numeric sign (1, -1 or 0)"
"negate receiver"
"integer part of number (1.0)"
"fractional part of number (0.2)"
"reciprocal function"
"auto convert to float"
"square function"
"square root"
"power function"
"power function with integer"
"exponential"
"absolute value"
"round"
"truncate"
"round to specified decimal places"
"truncate to specified decimal places"
"truncate"
"round up"
"factorial"
"integer divide rounded toward zero"
"integer remainderrounded toward zero"
"greatest common denominator"
"least common multiple"
"natural logarithm"
"base 10 logarithm"
"logarithm with specified base"
"floor of the log"
"convert degrees to radians"
"convert radians to degrees"
"sine"
"cosine"
"tangent"
"arcsine"
"arccosine"
"arctangent"
"get maximum of two numbers"
"get minimum of two numbers"
"pi"
"exp constant"
```

```

181: x := Float infinity.
182: x := Float nan.
183: x := Random new next; yourself. x next.
184: x := 100 atRandom.
185:
186: "*****
187: * Bitwise Manipulation:
188: * *****
189: | b x |
190: x := 16rFF bitAnd: 16r0F.
191: x := 16rF0 bitOr: 16r0F.
192: x := 16rFF bitXor: 16r0F.
193: x := 16rFF bitInvert.
194: x := 16r0F bitShift: 4.
195: x := 16rF0 bitShift: -4.
196: "x := 16r80 bitAt: 7."
197: x := 16r80 highbit.
198: b := 16rFF allMask: 16r0F.
199: b := 16rFF anyMask: 16r0F.
200: b := 16rFF noMask: 16r0F.
201:
202: "*****
203: * Conversion:
204: * *****
205: | x |
206: x := 3.99 asInteger.
207: x := 3.99 asFraction.
208: x := 3 asFloat.
209: x := 65 asCharacter.
210: x := $A asciiValue.
211: x := 3.99 printString.
212: x := 3.99 storeString.
213: x := 15 radix: 16.
214: x := 15 printStringBase: 16.
215: x := 15 storeStringBase: 16.
216:
217: "*****
218: * Blocks:
219: * - blocks are objects and may be assigned to a variable
220: * - value is last expression evaluated unless explicit return
221: * - blocks may be nested
222: * - specification [ arguments | localvars | expressions ]
223: * - Squeak does not currently support localvars in blocks
224: * - max of three arguments allowed
225: * - ^expression terminates block & method (exits all nested blocks) *

```

```

"infinity"
"not-a-number"
"random number stream (0.0 to 1.0)"
"quick random number"

*****
*
*****
"and bits"
"or bits"
"xor bits"
"invert bits"
"left shift"
"right shift"
"bit at position (0|1) [!Squeak]"
"position of highest bit set"
"test if all bits set in mask set in receiver"
"test if any bits set in mask set in receiver"
"test if all bits set in mask clear in receiver"

*****
*
*****
"convert number tointeger (truncates in Squeak)"
"convert number tofraction"
"convert number tofloat"
"convert integer to character"
"convert characterto integer"
"convert object toString via printOn:"
"convert object toString via storeOn:"
"convert to stringin given base"

*****
*
*****

```

```

226: * - blocks intended for long term storage should not contain ^ *
227: * *****
228: | x y z |
229: x := [ y := 1. z := 2. ]. x value.
230: x := [ :argOne :argTwo | argOne, ' and ', argTwo.].
231: Transcript show: (x value: 'First' value: 'Second'); cr.
232: "x := [ | z | z := 1.].
233:
234: "*****
235: * Method calls:
236: * - unary methods are messages with no arguments
237: * - binary methods
238: * - keyword methods are messages with selectors including colons
239: *
240: * standard categories/protocols:
241: * - initialize-release (methods called for new instance)
242: * - accessing (get/set methods)
243: * - testing (boolean tests - is)
244: * - comparing (boolean tests with parameter
245: * - displaying (gui related methods)
246: * - printing (methods for printing)
247: * - updating (receive notification of changes)
248: * - private (methods private to class)
249: * - instance-creation (class methods for creating instance) *
250: * *****
251: | x |
252: x := 2 sqrt.
253: x := 2 raisedTo: 10.
254: x := 194 * 9.
255: Transcript show: (194 * 9) printString; cr.
256: x := 2 perform: #sqrt.
257: Transcript
258:   show: 'hello ';
259:   show: 'world';
260:   cr.
261: x := 3 + 2; * 100.
262:
263: "*****
264: * Conditional Statements:
265: * *****
266: | x |
267: x > 10 ifTrue: [Transcript show: 'ifTrue'; cr].
268: x > 10 ifFalse: [Transcript show: 'ifFalse'; cr].
269: x > 10
270:   ifTrue: [Transcript show: 'ifTrue'; cr]

```

```

"simple block usage"
"set up block with argument passing"
"use block with argument passing"
  localvars not available in squeak blocks"

"unary message"
"keyword message"
"binary message"
"combination (chaining)"
"indirect method invocation"
"Cascading - send multiple messages to receiver"

"result=300. Sendmessage to same receiver (3)"

*****
*
*****

```

```
271:  ifFalse: [Transcript show: 'ifFalse'; cr].
272:  x > 10
273:    ifFalse: [Transcript show: 'ifFalse'; cr]
274:    ifTrue: [Transcript show: 'ifTrue'; cr].
275:  Transcript
276:    show:
277:      (x > 10
278:        ifTrue: ['ifTrue']
279:        ifFalse: ['ifFalse']);
280:    cr.
281:  Transcript
282:    show:
283:      (x > 10
284:        ifTrue: [x > 5
285:          ifTrue: ['A']
286:          ifFalse: ['B']]
287:        ifFalse: ['C']);
288:    cr.
289:  switch := Dictionary new.
290:  switch at: $A put: [Transcript show: 'Case A'; cr].
291:  switch at: $B put: [Transcript show: 'Case B'; cr].
292:  switch at: $C put: [Transcript show: 'Case C'; cr].
293:  result := (switch at: $B) value.
294:
295:  "*****
296:  * Iteration statements:
297:  *****"
298:  | x y |
299:  x := 4. y := 1.
300:  [x > 0] whileTrue: [x := x - 1. y := y * 2].
301:  [x >= 4] whileFalse: [x := x + 1. y := y * 2].
302:  x timesRepeat: [y := y * 2].
303:  1 to: x do: [:a | y := y * 2].
304:  1 to: x by: 2 do: [:a | y := y / 2].
305:  #(5 4 3) do: [:a | x := x + a].
306:
307:  "*****
308:  * Character:
309:  *****"
310:  | x y |
311:  x := $A.
312:  y := x isLowercase.
313:  y := x isUppercase.
314:  y := x isLetter.
315:  y := x isDigit.

"if else then"

"nested if then else"

"switch functionality"
```

```

316: y := x isAlphaNumeric.
317: y := x isSeparator.
318: y := x isVowel.
319: y := x digitValue.
320: y := x asLowercase.
321: y := x asUppercase.
322: y := x asciiValue.
323: y := x asString.
324: b := $A <= $B.
325: y := $A max: $B.
326:
327: "*****
328: * Symbol:
329: *****
330: | b x y |
331: x := #Hello.
332: y := 'String', 'Concatenation'.
333: b := x isEmpty.
334: y := x size.
335: y := x at: 2.
336: y := x copyFrom: 2 to: 4.
337: y := x indexOf: $e ifAbsent: [0].
338: x do: [:a | Transcript show: a printString; cr].
339: b := x conform: [:a | (a >= $a) & (a <= $z)].
340: y := x select: [:a | a > $a].
341: y := x asString.
342: y := x asText.
343: y := x asArray.
344: y := x asOrderedCollection.
345: y := x asSortedCollection.
346: y := x asBag.
347: y := x asSet.
348:
349: "*****
350: * String:
351: *****
352: | b x y |
353: x := 'This is a string'.
354: x := 'String', 'Concatenation'.
355: b := x isEmpty.
356: y := x size.
357: y := x at: 2.
358: y := x copyFrom: 2 to: 4.
359: y := x indexOf: $a ifAbsent: [0].
360: x := String new: 4.

"test if alphanumeric"
"test if separatorchar"
"test if vowel"
"convert to numeric digit value"
"convert to lower case"
"convert to upper case"
"convert to numeric ascii value"
"convert to string"
"comparison"

"symbol assignment"
"symbol concatenation (result is string)"
"test if symbol isempty"
"string size"
"char at location"
"substring"
"first position ofcharacter within string"
"iterate over the string"
"test if all elements meet condition"
"return all elements that meet condition"
"convert symbol toString"
"convert symbol totext"
"convert symbol toarray"
"convert symbol toordered collection"
"convert symbol tosorted collection"
"convert symbol tobag collection"
"convert symbol toset collection"

"string assignment"
"string concatenation"
"test if string isempty"
"string size"
"char at location"
"substring"
"first position ofcharacter within string"
"allocate string object"

```



```

361: x
362:   at: 1 put: $a;
363:   at: 2 put: $b;
364:   at: 3 put: $c;
365:   at: 4 put: $e.
366: x := String with: $a with: $b with: $c with: $d.
367: x do: [:a | Transcript show: a printString; cr].
368: b := x conform: [:a | (a >= $a) & (a <= $z)].
369: y := x select: [:a | a > $a].
370: y := x asSymbol.
371: y := x asArray.
372: x := 'ABCD' asByteArray.
373: y := x asOrderedCollection.
374: y := x asSortedCollection.
375: y := x asBag.
376: y := x asSet.
377: y := x shuffled.
378:
379: "***** Fixed length collection *****"
380: * Array:      Fixed length collection
381: * ByteArray:   Array limited to byte elements (0-255)
382: * WordArray:  Array limited to word elements (0-2^32)
383: *****
384: | b x y sum max |
385: x := #(4 3 2 1).
386: x := Array with: 5 with: 4 with: 3 with: 2.
387: x := Array new: 4.
388: x
389:   at: 1 put: 5;
390:   at: 2 put: 4;
391:   at: 3 put: 3;
392:   at: 4 put: 2.
393: b := x isEmpty.
394: y := x size.
395: y := x at: 4.
396: b := x includes: 3.
397: y := x copyFrom: 2 to: 4.
398: y := x indexOf: 3 ifAbsent: [0].
399: y := x occurrencesOf: 3.
400: x do: [:a | Transcript show: a printString; cr].
401: b := x conform: [:a | (a >= 1) & (a <= 4)].
402: y := x select: [:a | a > 2].
403: y := x reject: [:a | a < 2].
404: y := x collect: [:a | a + a].
405: y := x detect: [:a | a > 3] ifNone: [].

"set string elements"

"set up to 4 elements at a time"
"iterate over the string"
"test if all elements meet condition"
"return all elements that meet condition"
"convert string tosymbol"
"convert string toarray"
"convert string tobyte array"
"convert string toordered collection"
"convert string tosorted collection"
"convert string tobag collection"
"convert string toset collection"
"randomly shuffle string"

"constant array"
"create array withup to 4 elements"
"allocate an arraywith specified size"
"set array elements"

"test if array is empty"
"array size"
"get array elementat index"
"test if element is in array"
"subarray"
"first position ofelement within array"
"number of times object in collection"
"iterate over the array"
"test if all elements meet condition"
"return collectionof elements that pass test"
"return collectionof elements that fail test"
"transform each element for new collection"
"find position of first element that passes test"

```

```

406: sum := 0. x do: [:a | sum := sum + a]. sum.
407: sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)]. "sum array elements"
408: sum := x inject: 0 into: [:a :c | a + c]. "sum array elements"
409: max := x inject: 0 into: [:a :c | (a > c) "find max element in array"
410: ifTrue: [a]
411: ifFalse: [c]].
412: y := x shuffled.
413: y := x asArray.
414: "y := x asByteArray."
415: y := x asWordArray.
416: y := x asOrderedCollection.
417: y := x asSortedCollection.
418: y := x asBag.
419: y := x asSet.
420:
421: "*****
422: * OrderedCollection: acts like an expandable array *
423: *****"
424: | b x y sum max |
425: x := OrderedCollection with: 4 with: 3 with: 2 with: 1.
426: x := OrderedCollection new.
427: x add: 3; add: 2; add: 1; add: 4; yourself.
428: y := x addFirst: 5.
429: y := x removeFirst.
430: y := x addLast: 6.
431: y := x removeLast.
432: y := x addAll: #(7 8 9).
433: y := x removeAll: #(7 8 9).
434: x at: 2 put: 3.
435: y := x remove: 5 ifAbsent: [].
436: b := x isEmpty.
437: y := x size.
438: y := x at: 2.
439: y := x first.
440: y := x last.
441: b := x includes: 5.
442: y := x copyFrom: 2 to: 3.
443: y := x indexOf: 3 ifAbsent: [0].
444: y := x occurrencesOf: 3.
445: x do: [:a | Transcript show: a printString; cr].
446: b := x conform: [:a | (a >= 1) & (a <= 4)].
447: y := x select: [:a | a > 2].
448: y := x reject: [:a | a < 2].
449: y := x collect: [:a | a + a].
450: y := x detect: [:a | a > 3] ifNone: [].

"sum array elements"
"sum array elements"
"sum array elements"
"find max element in array"

"randomly shuffle collection"
"convert to array"
"note: this instruction not available on Squeak"
"convert to word array"
"convert to ordered collection"
"convert to sorted collection"
"convert to bag collection"
"convert to set collection"

"create collection with up to 4 elements"
"allocate collection"
"add element to collection"
"add element at beginning of collection"
"remove first element in collection"
"add element at end of collection"
"remove last element in collection"
"add multiple elements to collection"
"remove multiple elements from collection"
"set element at index"
"remove element from collection"
"test if empty"
"number of elements"
"retrieve element at index"
"retrieve first element in collection"
"retrieve last element in collection"
"test if element is in collection"
"subcollection"
"first position of element within collection"
"number of times object in collection"
"iterate over the collection"
"test if all elements meet condition"
"return collection of elements that pass test"
"return collection of elements that fail test"
"transform each element for new collection"
"find position of first element that passes test"

```

```

451: sum := 0. x do: [:a | sum := sum + a]. sum.
452: sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)]. "sum elements"
453: sum := x inject: 0 into: [:a :c | a + c]. "sum elements"
454: max := x inject: 0 into: [:a :c | (a > c) "find max element in collection"
455:   ifTrue: [a]
456:   ifFalse: [c]].
457: y := x shuffled.
458: y := x asArray.
459: y := x asOrderedCollection.
460: y := x asSortedCollection.
461: y := x asBag.
462: y := x asSet.
463:
464: "***** like OrderedCollection except order of elements *
465: * * determined by sorting criteria *
466: * *****"
467:
468: | b x y sum max |
469: x := SortedCollection with: 4 with: 3 with: 2 with: 1.
470: x := SortedCollection new.
471: x := SortedCollection sortBlock: [:a :c | a > c].
472: x add: 3; add: 2; add: 1; add: 4; yourself.
473: y := x addFirst: 5.
474: y := x removeFirst.
475: y := x addLast: 6.
476: y := x removeLast.
477: y := x addAll: #(7 8 9).
478: y := x removeAll: #(7 8 9).
479: y := x remove: 5 ifAbsent: [].
480: b := x isEmpty.
481: y := x size.
482: y := x at: 2.
483: y := x first.
484: y := x last.
485: b := x includes: 4.
486: y := x copyFrom: 2 to: 3.
487: y := x indexOf: 3 ifAbsent: [0].
488: y := x occurrencesOf: 3.
489: x do: [:a | Transcript show: a printString; cr].
490: b := x conform: [:a | (a >= 1) & (a <= 4)].
491: y := x select: [:a | a > 2].
492: y := x reject: [:a | a < 2].
493: y := x collect: [:a | a + a].
494: y := x detect: [:a | a > 3] ifNone: [].
495: sum := 0. x do: [:a | sum := sum + a]. sum.

"sum elements"
"sum elements"
"sum elements"
"find max element in collection"

"randomly shuffle collection"
"convert to array"
"convert to ordered collection"
"convert to sortedcollection"
"convert to bag collection"
"convert to set collection"

"create collectionwith up to 4 elements"
"allocate collection"
"set sort criteria"
"add element to collection"
"add element at beginning of collection"
"remove first element in collection"
"add element at end of collection"
"remove last element in collection"
"add multiple elements to collection"
"remove multiple elements from collection"
"remove element from collection"
"test if empty"
"number of elements"
"retrieve element at index"
"retrieve first element in collection"
"retrieve last element in collection"
"test if element is in collection"
"subcollection"
"first position ofelement within collection"
"number of times object in collection"
"iterate over the collection"
"test if all elements meet condition"
"return collectionof elements that pass test"
"return collectionof elements that fail test"
"transform each element for new collection"
"find position of first element that passes test"
"sum elements"

```

```

496: sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)]. "sum elements"
497: sum := x inject: 0 into: [:a :c | a + c]. "sum elements"
498: max := x inject: 0 into: [:a :c | (a > c) "find max element in collection"
499:   ifTrue: [a]
500:   ifFalse: [c]].
501: y := x asArray.
502: y := x asOrderedCollection.
503: y := x asSortedCollection.
504: y := x asBag.
505: y := x asSet.
506:
507: "*****
508: * Bag: like OrderedCollection except elements are in no *
509: * particular order *
510: *****"
511: | b x y sum max |
512: x := Bag with: 4 with: 3 with: 2 with: 1.
513: x := Bag new.
514: x add: 4; add: 3; add: 1; add: 2; yourself.
515: x add: 3 withOccurrences: 2.
516: y := x addAll: #(7 8 9).
517: y := x removeAll: #(7 8 9).
518: y := x remove: 4 ifAbsent: [].
519: b := x isEmpty.
520: y := x size.
521: b := x includes: 3.
522: y := x occurrencesOf: 3.
523: x do: [:a | Transcript show: a printString; cr].
524: b := x conform: [:a | (a >= 1) & (a <= 4)].
525: y := x select: [:a | a > 2].
526: y := x reject: [:a | a < 2].
527: y := x collect: [:a | a + a].
528: y := x detect: [:a | a > 3] ifNone: [].
529: sum := 0. x do: [:a | sum := sum + a]. sum.
530: sum := x inject: 0 into: [:a :c | a + c].
531: max := x inject: 0 into: [:a :c | (a > c)
532:   ifTrue: [a]
533:   ifFalse: [c]].
534: y := x asOrderedCollection.
535: y := x asSortedCollection.
536: y := x asBag.
537: y := x asSet.
538:
539: "*****
540: * Set: like Bag except duplicates not allowed *
*****"
"convert to array"
"convert to ordered collection"
"convert to sortedcollection"
"convert to bag collection"
"convert to set collection"

"create collectionwith up to 4 elements"
"allocate collection"
"add element to collection"
"add multiple copies to collection"
"add multiple elements to collection"
"remove multiple elements from collection"
"remove element from collection"
"test if empty"
"number of elements"
"test if element is in collection"
"number of times object in collection"
"iterate over the collection"
"test if all elements meet condition"
"return collectionof elements that pass test"
"return collectionof elements that fail test"
"transform each element for new collection"
"find position of first element that passes test"
"sum elements"
"sum elements"
"find max element in collection"

"convert to ordered collection"
"convert to sortedcollection"
"convert to bag collection"
"convert to set collection"

```

```

541: * IdentitySet: uses identity test (== rather than =)
542: *****
543: | b x y sum max |
544: x := Set with: 4 with: 3 with: 2 with: 1.
545: x := Set new.
546: x add: 4; add: 3; add: 1; add: 2; yourself.
547: y := x addAll: #(7 8 9).
548: y := x removeAll: #(7 8 9).
549: y := x remove: 4 ifAbsent: [].
550: b := x isEmpty.
551: y := x size.
552: x includes: 4.
553: x do: [:a | Transcript show: a printString; cr].
554: b := x conform: [:a | (a >= 1) & (a <= 4)].
555: y := x select: [:a | a > 2].
556: y := x reject: [:a | a < 2].
557: y := x collect: [:a | a + a].
558: y := x detect: [:a | a > 3] ifNone: [].
559: sum := 0. x do: [:a | sum := sum + a]. sum.
560: sum := x inject: 0 into: [:a :c | a + c].
561: max := x inject: 0 into: [:a :c | (a > c)
562:   ifTrue: [a]
563:   ifFalse: [c]].
564: y := x asArray.
565: y := x asOrderedCollection.
566: y := x asSortedCollection.
567: y := x asBag.
568: y := x asSet.
569:
570: *****
571: * Interval:
572: *****
573: | b x y sum max |
574: x := Interval from: 5 to: 10.
575: x := 5 to: 10.
576: x := Interval from: 5 to: 10 by: 2.
577: x := 5 to: 10 by: 2.
578: b := x isEmpty.
579: y := x size.
580: x includes: 9.
581: x do: [:k | Transcript show: k printString; cr].
582: b := x conform: [:a | (a >= 1) & (a <= 4)].
583: y := x select: [:a | a > 7].
584: y := x reject: [:a | a < 2].
585: y := x collect: [:a | a + a].

"create collection with up to 4 elements"
"allocate collection"
"add element to collection"
"add multiple elements to collection"
"remove multiple elements from collection"
"remove element from collection"
"test if empty"
"number of elements"
"test if element is in collection"
"iterate over the collection"
"test if all elements meet condition"
"return collection of elements that pass test"
"return collection of elements that fail test"
"transform each element for new collection"
"find position of first element that passes test"
"sum elements"
"sum elements"
"find max element in collection"

"convert to array"
"convert to ordered collection"
"convert to sorted collection"
"convert to bag collection"
"convert to set collection"

*****
*
*****
"create interval object"
"create interval object with specified increment"

"test if empty"
"number of elements"
"test if element is in collection"
"iterate over interval"
"test if all elements meet condition"
"return collection of elements that pass test"
"return collection of elements that fail test"
"transform each element for new collection"

```

```

586: y := x detect: [:a | a > 3] ifNone: [].
587: sum := 0. x do: [:a | sum := sum + a]. sum.
588: sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)].
589: sum := x inject: 0 into: [:a :c | a + c].
590: max := x inject: 0 into: [:a :c | (a > c)
591:   ifTrue: [a]
592:   ifFalse: [c]].
593: y := x asArray.
594: y := x asOrderedCollection.
595: y := x asSortedCollection.
596: y := x asBag.
597: y := x asSet.
598:
599: "*****
600: * Associations:
601: *****
602: | x y |
603: x := #myVar->'hello'.
604: y := x key.
605: y := x value.
606:
607: "*****
608: * Dictionary:
609: * IdentityDictionary: uses identity test (== rather than =)
610: *****
611: | b x y |
612: x := Dictionary new.
613: x add: #a->4; add: #b->3; add: #c->1; add: #d->2; yourself.
614: x at: #e put: 3.
615: b := x isEmpty.
616: y := x size.
617: y := x at: #a ifAbsent: [].
618: y := x keyAtValue: 3 ifAbsent: [].
619: y := x removeKey: #e ifAbsent: [].
620: b := x includes: 3.
621: b := x includesKey: #a.
622: y := x occurrencesOf: 3.
623: y := x keys.
624: y := x values.
625: x do: [:a | Transcript show: a printString; crl].
626: x keysDo: [:a | Transcript show: a printString; crl].
627: x associationsDo: [:a | Transcript show: a printString; crl].
628: x keysAndValuesDo: [:aKey aValue | Transcript
629:   show: aKey printString; space;
630:   show: aValue printString; crl].

"find position of first element that passes test"
"sum elements"
"sum elements"
"sum elements"
"find max element in collection"

"convert to array"
"convert to ordered collection"
"convert to sortedcollection"
"convert to bag collection"
"convert to set collection"

*****
*
*****

"allocate collection"
"add element to collection"
"set element at index"
"test if empty"
"number of elements"
"retrieve element at index"
"retrieve key for given value with error block"
"remove element from collection"
"test if element is in values collection"
"test if element is in keys collection"
"number of times object in collection"
"set of keys"
"bag of values"
"iterate over the values collection"
"iterate over the keys collection"
"iterate over the associations"
"iterate over keysand values"

```

```

631: b := x conform: [:a | (a >= 1) & (a <= 4)].
632: y := x select: [:a | a > 2].
633: y := x reject: [:a | a < 2].
634: y := x collect: [:a | a + a].
635: y := x detect: [:a | a > 3] ifNone: [].
636: sum := 0. x do: [:a | sum := sum + a]. sum.
637: sum := x inject: 0 into: [:a :c | a + c].
638: max := x inject: 0 into: [:a :c | (a > c)
639:   ifTrue: [a]
640:   ifFalse: [c]].
641: y := x asArray.
642: y := x asOrderedCollection.
643: y := x asSortedCollection.
644: y := x asBag.
645: y := x asSet.
646:
647: Smalltalk at: #CMRGlobal put: 'CMR entry'.
648: x := Smalltalk at: #CMRGlobal.
649: Transcript show: (CMRGlobal printString).
650: Smalltalk keys do: [:k |
651:   ((Smalltalk at: k) isKindOf: Class)
652:   ifFalse: [Transcript show: k printString; cr]].
653: Smalltalk at: #CMRDictionary put: (Dictionary new).
654: CMRDictionary at: #MyVar1 put: 'hello1'.
655: CMRDictionary add: #MyVar2->'hello2'.
656: CMRDictionary size.
657: CMRDictionary keys do: [:k |
658:   Transcript show: k printString; cr].
659: CMRDictionary values do: [:k |
660:   Transcript show: k printString; cr].
661: CMRDictionary keysAndValuesDo: [:aKey :aValue |
662:   Transcript
663:     show: aKey printString;
664:     space;
665:     show: aValue printString;
666:     cr].
667: CMRDictionary associationsDo: [:aKeyValue |
668:   Transcript show: aKeyValue printString; cr].
669: Smalltalk removeKey: #CMRGlobal ifAbsent: [].
670: Smalltalk removeKey: #CMRDictionary ifAbsent: [].
671:
672: "*****
673: * Internal Stream:
674: * *****
675: | b x ios |

"test if all elements meet condition"
"return collectionof elements that pass test"
"return collectionof elements that fail test"
"transform each element for new collection"
"find position of first element that passes test"
"sum elements"
"sum elements"
"find max element in collection"

"convert to array"
"convert to ordered collection"
"convert to sortedcollection"
"convert to bag collection"
"convert to set collection"

"put global in Smalltalk Dictionary"
"read global from Smalltalk Dictionary"
"entries are directly accessible by name"
"print out all classes"

"set up user defined dictionary"
"put entry in dictionary"
"add entry to dictionary use key->value combo"
"dictionary size"
"print out keys indictionary"

"print out values in dictionary"

"print out keys and values"

"another iterator for printing key values"

"remove entry fromSmalltalk dictionary"
"remove user dictionary from Smalltalk dictionary"
*****
*
*****

```

```
676: ios := ReadStream on: 'Hello read stream'.
677: ios := ReadStream on: 'Hello read stream' from: 1 to: 5.
678: [(x := ios nextLine) notNil]
679:   whileTrue: [Transcript show: x; cr].
680: ios position: 3.
681: ios position.
682: x := ios next.
683: x := ios peek.
684: x := ios contents.
685: b := ios atEnd.
686:
687: ios := ReadWriteStream on: 'Hello read stream'.
688: ios := ReadWriteStream on: 'Hello read stream' from: 1 to: 5.
689: ios := ReadWriteStream with: 'Hello read stream'.
690: ios := ReadWriteStream with: 'Hello read stream' from: 1 to: 10.
691: ios position: 0.
692: [(x := ios nextLine) notNil]
693:   whileTrue: [Transcript show: x; cr].
694: ios position: 6.
695: ios position.
696: ios nextPutAll: 'Chris'.
697: x := ios next.
698: x := ios peek.
699: x := ios contents.
700: b := ios atEnd.
701:
702: "*****
703: * FileStream:
704: * *****
705: | b x ios |
706: ios := FileStream newFileName: 'ios.txt'.
707: ios nextPut: $H; cr.
708: ios nextPutAll: 'Hello File'; cr.
709: 'Hello File' printOn: ios.
710: 'Hello File' storeOn: ios.
711: ios close.
712:
713: ios := FileStream oldFileName: 'ios.txt'.
714: [(x := ios nextLine) notNil]
715:   whileTrue: [Transcript show: x; cr].
716: ios position: 3.
717: x := ios position.
718: x := ios next.
719: x := ios peek.
720: b := ios atEnd.
```



```
721: ios close.
722:
723: "*****
724: * Date:
725: *****
726: | x y |
727: x := Date today.
728: x := Date dateAndTimeNow.
729: x := Date readFromTimeString: '01/02/1999'.
730: x := Date newDay: 12 month: #July year: 1999
731: x := Date fromDays: 36000.
732: y := Date dayOfWeek: #Monday.
733: y := Date indexOfMonth: #January.
734: y := Date daysInMonth: 2 forYear: 1996.
735: y := Date daysInYear: 1996.
736: y := Date nameOfDay: 1
737: y := Date nameOfMonth: 1.
738: y := Date leapYear: 1996.
739: y := x weekday.
740: y := x previous: #Monday.
741: y := x dayOfMonth.
742: y := x day.
743: y := x firstDayOfMonth.
744: y := x monthName.
745: y := x monthIndex.
746: y := x daysInMonth.
747: y := x year.
748: y := x daysInYear.
749: y := x daysLeftInYear.
750: y := x asSeconds.
751: y := x addDays: 10.
752: y := x subtractDays: 10.
753: y := x subtractDate: (Date today).
754: y := x printFormat: #(2 1 3 $ / 1 1).
755: b := (x <= Date today).
756:
757: "*****
758: * Time:
759: *****
760: | x y |
761: x := Time now.
762: x := Time dateAndTimeNow.
763: x := Time readFromTimeString: '3:47:26 pm'.
764: x := Time fromSeconds: (60 * 60 * 4).
765: y := Time millisecondClockValue.

"create date for today"
"create date from current time/date"
"create date from formatted string"
"create date from parts"
"create date from elapsed days since 1/1/1901"
"day of week as int (1-7)"
"month of year as int (1-12)"
"day of month as int (1-31)"
"days in year (365|366)"
"weekday name (#Monday,...)"
"month name (#January,...)"
"1 if leap year; 0if not leap year"
"day of week (#Monday,...)"
"date for previousday of week"
"day of month (1-31)"
"day of year (1-366)"
"day of year for first day of month"
"month of year (#January,...)"
"month of year (1-12)"
"days in month (1-31)"
"year (19xx)"
"days in year (365|366)"
"days left in year(364|365)"
"seconds elapsed since 1/1/1901"
"add days to date object"
"subtract days to date object"
"subtract date (result in days)"
"print formatted date"
"comparison"

"create time from current time"
"create time from current time/date"
"create time from formatted string"
"create time from elapsed time from midnight"
"milliseconds since midnight"
```

```
766: y := Time totalSeconds.
767: y := x seconds.
768: y := x minutes.
769: y := x hours.
770: y := x addTime: (Time now).
771: y := x subtractTime: (Time now).
772: y := x asSeconds.
773: x := Time millisecondsToRun: [
774:   1 to: 1000 do: [:index | y := 3.14 * index]].
775: b := (x <= Time now).
776:
777: "*****
778: * Point:
779: * *****
780: | x y |
781: x := 200@100.
782: y := x x.
783: y := x y.
784: x := 200@100 negated.
785: x := (-200@-100) abs.
786: x := (200.5@100.5) rounded.
787: x := (200.5@100.5) truncated.
788: x := 200@100 + 100.
789: x := 200@100 - 100.
790: x := 200@100 * 2.
791: x := 200@100 / 2.
792: x := 200@100 // 2.
793: x := 200@100 \\ 3.
794: x := 200@100 + 50@25.
795: x := 200@100 - 50@25.
796: x := 200@100 * 3@4.
797: x := 200@100 // 3@4.
798: x := 200@100 max: 50@200.
799: x := 200@100 min: 50@200.
800: x := 20@5 dotProduct: 10@2.
801:
802: "*****
803: * Rectangle:
804: * *****
805: Rectangle fromUser.
806:
807: "*****
808: * Pen:
809: * *****
810: | myPen |

"total seconds since 1/1/1901"
"seconds past minute (0-59)"
"minutes past hour(0-59)"
"hours past midnight (0-23)"
"add time to time object"
"subtract time to time object"
"convert time to seconds"
"timing facility"

"comparison"

*****
*
*****

"obtain a new point"
"x coordinate"
"y coordinate"
"negates x and y"
"absolute value ofx and y"
"round x and y"
"truncate x and y"
"add scale to bothx and y"
"subtract scale from both x and y"
"multiply x and y by scale"
"divide x and y byscale"
"divide x and y byscale"
"remainder of x and y by scale"
"add points"
"subtract points"
"multiply points"
"divide points"
"max x and y"
"min x and y"
"sum of product (x1*x2 + y1*y2)"

*****
*
*****

*****
*
*****

*****
* Pen:
*****
myPen |
```

```
811: Display restoreAfter: [
812:   Display fillWhite.
813:
814:   myPen := Pen new.
815:   myPen squareNib: 1.
816:   myPen color: (Color blue).
817:   myPen home.
818:   myPen up.
819:   myPen down.
820:   myPen north.
821:   myPen turn: -180.
822:   myPen direction.
823:   myPen go: 50.
824:   myPen location.
825:   myPen goto: 200@200.
826:   myPen place: 250@250.
827:   myPen print: 'Hello World' withFont: (TextStyle default fontAt: 1).
828:   Display extent.
829:   Display width.
830:   Display height.
831:
832: ].
833:
834: "*****
835:  * Dynamic Message Calling/Compiling:
836:  *****
837: | receiver message result argument keyword1 keyword2 argument1 argument2 |
838: "unary message"
839: receiver := 5.
840: message := 'factorial' asSymbol.
841: result := receiver perform: message.
842: result := Compiler evaluate: ((receiver storeString), ' ', message).
843: result := (Message new setSelector: message arguments: #()) sentTo: receiver.
844:
845: "binary message"
846: receiver := 1.
847: message := '+' asSymbol.
848: argument := 2.
849: result := receiver perform: message withArguments: (Array with: argument).
850: result := Compiler evaluate: ((receiver storeString), ' ', message, ' ', (argument storeString)).
851: result := (Message new setSelector: message arguments: (Array with: argument)) sentTo: receiver.
852:
853: "keyword messages"
854: receiver := 12.
855: keyword1 := 'between:' asSymbol.
```

```
856: keyword2 := 'and:' asSymbol.
857: argument1 := 10.
858: argument2 := 20.
859: result := receiver
860:   perform: (keyword1, keyword2) asSymbol
861:   withArguments: (Array with: argument1 with: argument2).
862: result := Compiler evaluate:
863:   ((receiver storeString), ' ', keyword1, (argument1 storeString), ' ', keyword2, (argument2 storeString)).
864: result := (Message
865:   new
866:     setSelector: (keyword1, keyword2) asSymbol
867:     arguments: (Array with: argument1 with: argument2))
868:   sentTo: receiver.
869:
870: "*****"
871: * class/meta-class:
872: *****
873: | b x |
874: x := String name.
875: x := String category.
876: x := String comment.
877: x := String kindOfSubclass.
878: x := String definition.
879: x := String instVarNames.
880: x := String allInstVarNames.
881: x := String classVarNames.
882: x := String allClassVarNames.
883: x := String sharedPools.
884: x := String allSharedPools.
885: x := String selectors.
886: x := String sourceCodeAt: #size.
887: x := String allInstances.
888: x := String superclass.
889: x := String allSuperclasses.
890: x := String withAllSuperclasses.
891: x := String subclasses.
892: x := String allSubclasses.
893: x := String withAllSubclasses.
894: b := String instSize.
895: b := String isFixed.
896: b := String isVariable.
897: b := String isPointers.
898: b := String isBits.
899: b := String isBytes.
900: b := String isWords.

"class name"
"organization category"
"class comment"
"subclass type - subclass: variableSubclass, etc"
"class definition"
"immediate instance variable names"
"accumulated instance variable names"
"immediate class variable names"
"accumulated classvariable names"
"immediate dictionaries used as shared pools"
"accumulated dictionaries used as shared pools"
"message selectorsfor class"
"source code for specified method"
"collection of allinstances of class"
"immediate superclass"
"accumulated superclasses"
"receiver class and accumulated superclasses"
"immediate subclasses"
"accumulated subclasses"
"receiver class and accumulated subclasses"
"number of named instance variables"
"true if no indexed instance variables"
"true if has indexed instance variables"
"true if index instance vars contain objects"
"true if index instance vars contain bytes/words"
"true if index instance vars contain bytes"
true if index instance vars contain words"
```

```

901: Object withAllSubclasses size.
902:
903: "*****
904: * debugging:
905: *****
906: | a b x |
907: x yourself.
908: String browse.
909: x inspect.
910: x confirm: 'Is this correct?'.
911: x halt.
912: x halt: 'Halt message'.
913: x notify: 'Notify text'.
914: x error: 'Error string'.
915: x doesNotUnderstand: #cmrMessage.
916: x shouldNotImplement.
917: x subclassResponsibility.
918: x errorImproperStore.
919: x errorNonIntegerIndex.
920: x errorSubscriptBounds.
921: x primitiveFailed.
922:
923: a := 'A1'. b := 'B2'. a become: b.
924: Transcript show: a, b; cr.
925:
926: "*****
927: * Misc.
928: *****
929: | x |
930: "Smalltalk condenseChanges."
931: x := FillInTheBlank request: 'Prompt Me'.
932: Utilities openCommandKeyHelp
933:
934:
935:
936:
937:
938: References
939:
940: 1. http://www.angelfire.com/tx4/cus/index.html
941: 2. mailto:ChrisRath@aol.com
942: 3. http://www.angelfire.com/tx4/cus/index.html
943: 4. mailto:ChrisRath@aol.com

```

```

"get total number of class entries"

*****
*
*****

"returns receiver"
"browse specified class"
"open object inspector window"

"breakpoint to open debugger window"

"open up error window with title"
"flag message is not handled"
"flag message should not be implemented"
"flag message as abstract"
"flag an improper store into indexable object"
"flag only integers should be used as index"
"flag subscript out of bounds"
"system primitive failed"

"switch two objects"

*****
*
*****

"compress the change file"
"prompt user for input"

```