DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
Potentially confusable characters		
1Il|iO0oB8	1Il|iO0oB8	1Il|iO0oB8
Coding meta characters		
<>&"'^$(){}[]|?+*/\,.	<>&"'^$(){}[]|?+*/\,.	<>&"'^$(){}[]|?+*/\,.
Allowable characters		
abcdefghijklmnopqrstuvwxyz	abcdefghijklmnopqrstuvwxyz	abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ	ABCDEFGHIJKLMNOPQRSTUVWXYZ	ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890	1234567890	1234567890
.+/*~<>@%|&?	.+/*~<>@%|&?	.+/*~<>@%|&?
Transcript		
Transcript show: 'Hello World'.	Transcript show: 'Hello World'.	Transcript show: 'Hello World'.
Transcript nextPutAll: 'Hello World'.	Transcript nextPutAll: 'Hello World'.	Transcript nextPutAll: 'Hello World'.
Transcript nextPut: $A.	Transcript nextPut: $A.	Transcript nextPut: $A.
Transcript space.	Transcript space.	Transcript space.
Transcript tab.	Transcript tab.	Transcript tab.
Transcript cr.	Transcript cr.	Transcript cr.
'Hello' printOn: Transcript.	Hello' printOn: Transcript.	'Hello' printOn: Transcript.
'Hello' storeOn: Transcript.	Hello' storeOn: Transcript.	'Hello' storeOn: Transcript.
Assignment		
| x y |	| x y |	| x y |
x := 5.	x := 5.	x := 5.
x := y := z := 6.	x := y := z := 6.	x := y := z := 6.
x := (y := 6) + 1.	x := (y := 6) + 1.	x := (y := 6) + 1.
x := Object new.	x := Object new.	x := Object new.
x := 123 class.	x := 123 class.	x := 123 class.
x := 1.2 hash.	x := 1.2 hash.	x := 1.2 hash.
y := x copy.	y := x copy.	y := x copy.
y := x shallowCopy.	y := x shallowCopy.	y := x shallowCopy.
y := x deepCopy.	y := x deepCopy.	y := x deepCopy.
y := x veryDeepCopy.	y := x veryDeepCopy.	y := x veryDeepCopy.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
Constants		
| b |	| b |	| b |
b := true.	b := true.	b := true.
b := false.	b := false.	b := false.
x := nil.	x := nil.	x := nil.
x := 1.	x := 1.	x := 1.
x := 3.14.	x := 3.14.	x := 3.14.
x := 2e-2.	x := 2e-2.	x := 2e-2.
x := 16r0F.	x := 16r0F.	x := 16r0F.
x := -1.	x := -1.	x := -1.
x := 'Hello'.	x := 'Hello'.	x := 'Hello'.
x := 'I''m here'.	x := 'I''m here'.	x := 'I''m here'.
x := $A.	x := $A.	x := $A.
x := $.	x := $.	x := $.
x := #aSymbol.	x := #aSymbol.	x := #aSymbol.
x := #(3 2 1).	x := #(3 2 1).	x := #(3 2 1).
x := #('abc' 2 $a).	x := #('abc' 2 $a).	x := #('abc' 2 $a).
Booleans		
| b x y |	| b x y |	| b x y |
x := 1. y := 2.	x := 1. y := 2.	x := 1. y := 2.
b := (x = y).	b := (x = y).	b := (x = y).
b := (x ~= y).	b := (x ~= y).	b := (x ~= y).
b := (x == y).	b := (x == y).	b := (x == y).
b := (x ~~ y).	b := (x ~~ y).	b := (x ~~ y).
b := (x > y).	b := (x > y).	b := (x > y).
b := (x < y).	b := (x < y).	b := (x < y).
b := (x >= y).	b := (x >= y).	b := (x >= y).
b := (x <= y).	b := (x <= y).	b := (x <= y).
b := b not.	b := b not.	b := b not.
b := (x < 5) & (y > 1).	b := (x < 5) & (y > 1).	b := (x < 5) & (y > 1).

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
b := (x < 5) | (y > 1).	b := (x < 5) | (y > 1).	b := (x < 5) | (y > 1).
b := (x < 5) and: [y > 1].	b := (x < 5) and: [y > 1].	b := (x < 5) and: [y > 1].
b := (x < 5) or: [y > 1].	b := (x < 5) or: [y > 1].	b := (x < 5) or: [y > 1].
b := (x < 5) eqv: (y > 1).	b := (x < 5) eqv: (y > 1).	b := (x < 5) eqv: (y > 1).
b := (x < 5) xor: (y > 1).	b := (x < 5) xor: (y > 1).	b := (x < 5) xor: (y > 1).
b := 5 between: 3 and: 12.	b := 5 between: 3 and: 12.	b := 5 between: 3 and: 12.
b := 123 isKindOf: Number.	b := 123 isKindOf: Number.	b := 123 isKindOf: Number.
b := 123 isMemberOf: SmallInteger.	b := 123 isMemberOf: SmallInteger.	b := 123 isMemberOf: SmallInteger.
b := 123 respondsTo: sqrt.	b := 123 respondsTo: sqrt.	b := 123 respondsTo: sqrt.
b := x isNil.	b := x isNil.	b := x isNil.
b := x isZero.	b := x isZero.	b := x isZero.
b := x positive.	b := x positive.	b := x positive.
b := x strictlyPositive.	b := x strictlyPositive.	b := x strictlyPositive.
b := x negative.	b := x negative.	b := x negative.
b := x isLiteral.	b := x isLiteral.	b := x isLiteral.
b := x isInteger.	b := x isInteger.	b := x isInteger.
b := x isFloat.	b := x isFloat.	b := x isFloat.
b := x isNumber.	b := x isNumber.	b := x isNumber.
b := $A isUppercase.	b := $A isUppercase.	b := $A isUppercase.
b := $A isLowercase.	b := $A isLowercase.	b := $A isLowercase.
Arithmetic expressions		
| x |	| x |	| x |
x := 6 + 3.	x := 6 + 3.	x := 6 + 3.
x := 6 - 3.	x := 6 - 3.	x := 6 - 3.
x := 6 * 3.	x := 6 * 3.	x := 6 * 3.
x := 1 + 2 * 3.	x := 1 + 2 * 3.	x := 1 + 2 * 3.
x := 5 / 3.	x := 5 / 3.	x := 5 / 3.
x := 5.0 / 3.0.	x := 5.0 / 3.0.	x := 5.0 / 3.0.
x := 5.0 // 3.0.	x := 5.0 // 3.0.	x := 5.0 // 3.0.
x := 5.0 \\ 3.0.	x := 5.0 \\ 3.0.	x := 5.0 \\ 3.0.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x := -5.	x := -5.	x := −5.
x := 5 sign.	x := 5 sign.	x := 5 sign.
x := 5 negated.	x := 5 negated.	x := 5 negated.
x := 1.2 integerPart.	x := 1.2 integerPart.	x := 1.2 integerPart.
x := 1.2 fractionPart.	x := 1.2 fractionPart.	x := 1.2 fractionPart.
x := 5 reciprocal.	x := 5 reciprocal.	x := 5 reciprocal.
x := 6 * 3.1.	x := 6 * 3.1.	x := 6 * 3.1.
x := 5 squared.	x := 5 squared.	x := 5 squared.
x := 25 sqrt.	x := 25 sqrt.	x := 25 sqrt.
x := 5 raisedTo: 2.	x := 5 raisedTo: 2.	x := 5 raisedTo: 2.
x := 5 raisedToInteger: 2.	x := 5 raisedToInteger: 2.	x := 5 raisedToInteger: 2.
x := 5 exp.	x := 5 exp.	x := 5 exp.
x := -5 abs.	x := -5 abs.	x := −5 abs.
x := 3.99 rounded.	x := 3.99 rounded.	x := 3.99 rounded.
x := 3.99 truncated.	x := 3.99 truncated.	x := 3.99 truncated.
x := 3.99 roundTo: 1.	x := 3.99 roundTo: 1.	x := 3.99 roundTo: 1.
x := 3.99 truncateTo: 1.	x := 3.99 truncateTo: 1.	x := 3.99 truncateTo: 1.
x := 3.99 floor.	x := 3.99 floor.	x := 3.99 floor.
x := 3.99 ceiling.	x := 3.99 ceiling.	x := 3.99 ceiling.
x := 5 factorial.	x := 5 factorial.	x := 5 factorial.
x := -5 quo: 3.	x := -5 quo: 3.	x := −5 quo: 3.
x := -5 rem: 3.	x := -5 rem: 3.	x := −5 rem: 3.
x := 28 gcd: 12.	x := 28 gcd: 12.	x := 28 gcd: 12.
x := 28 lcm: 12.	x := 28 lcm: 12.	x := 28 lcm: 12.
x := 100 ln.	x := 100 ln.	x := 100 ln.
x := 100 log.	x := 100 log.	x := 100 log.
x := 100 log: 10.	x := 100 log: 10.	x := 100 log: 10.
x := 100 floorLog: 10.	x := 100 floorLog: 10.	x := 100 floorLog: 10.
x := 180 degreesToRadians.	x := 180 degreesToRadians.	x := 180 degreesToRadians.
x := 3.14 radiansToDegrees.	x := 3.14 radiansToDegrees.	x := 3.14 radiansToDegrees.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x := 0.7 sin.	x := 0.7 sin.	x := 0.7 sin.
x := 0.7 cos.	x := 0.7 cos.	x := 0.7 cos.
x := 0.7 tan.	x := 0.7 tan.	x := 0.7 tan.
x := 0.7 arcSin.	x := 0.7 arcSin.	x := 0.7 arcSin.
x := 0.7 arcCos.	x := 0.7 arcCos.	x := 0.7 arcCos.
x := 0.7 arcTan.	x := 0.7 arcTan.	x := 0.7 arcTan.
x := 10 max: 20.	x := 10 max: 20.	x := 10 max: 20.
x := 10 min: 20.	x := 10 min: 20.	x := 10 min: 20.
x := Float pi.	x := Float pi.	x := Float pi.
x := Float e.	x := Float e.	x := Float e.
x := Float infinity.	x := Float infinity.	x := Float infinity.
x := Float nan.	x := Float nan.	x := Float nan.
x := Random new next; yourself. x next.	x := Random new next; yourself. x next.	x := Random new next; yourself. x next.
Bitwise Manipulation		
| b x |	| b x |	| b x |
x := 16rFF bitAnd: 16r0F.	x := 16rFF bitAnd: 16r0F.	x := 16rFF bitAnd: 16r0F.
x := 16rF0 bitOr: 16r0F.	x := 16rF0 bitOr: 16r0F.	x := 16rF0 bitOr: 16r0F.
x := 16rFF bitXor: 16r0F.	x := 16rFF bitXor: 16r0F.	x := 16rFF bitXor: 16r0F.
x := 16rFF bitInvert.	x := 16rFF bitInvert.	x := 16rFF bitInvert.
x := 16r0F bitShift: 4.	x := 16r0F bitShift: 4.	x := 16r0F bitShift: 4.
x := 16rF0 bitShift: -4.	x := 16rF0 bitShift: -4.	x := 16rF0 bitShift: -4.
x := 16r80 bitAt: 7.	x := 16r80 bitAt: 7.	x := 16r80 bitAt: 7.
x := 16r80 highbit.	x := 16r80 highbit.	x := 16r80 highbit.
b := 16rFF allMask: 16r0F.	b := 16rFF allMask: 16r0F.	b := 16rFF allMask: 16r0F.
b := 16rFF anyMask: 16r0F.	b := 16rFF anyMask: 16r0F.	b := 16rFF anyMask: 16r0F.
b := 16rFF noMask: 16r0F.	b := 16rFF noMask: 16r0F.	b := 16rFF noMask: 16r0F.
Conversion		
| x |	| x |	| x |
x := 3.99 asInteger.	x := 3.99 asInteger.	x := 3.99 asInteger.
x := 3.99 asFraction.	x := 3.99 asFraction.	x := 3.99 asFraction.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x := 3 asFloat.	x := 3 asFloat.	x := 3 asFloat.
x := 65 asCharacter.	x := 65 asCharacter.	x := 65 asCharacter.
x := $A asciiValue.	x := $A asciiValue.	x := $A asciiValue.
x := 3.99 printString.	x := 3.99 printString.	x := 3.99 printString.
x := 3.99 storeString.	x := 3.99 storeString.	x := 3.99 storeString.
x := 15 radix: 16.	x := 15 radix: 16.	x := 15 radix: 16.
x := 15 printStringBase: 16.	x := 15 printStringBase: 16.	x := 15 printStringBase: 16.
x := 15 storeStringBase: 16.	x := 15 storeStringBase: 16.	x := 15 storeStringBase: 16.
Blocks		
| x y z |	| x y z |	| x y z |
x := [y := 1. z := 2.]. x value.	x := [y := 1. z := 2.]. x value.	x := [y := 1. z := 2.]. x value.
x := [:argOne :argTwo | argOne, ' and ' , argTwo.].	x := [:argOne :argTwo | argOne, ' and ' , argTwo.].	x := [:argOne :argTwo | argOne, ' and ' , argTwo.].
Transcript show: (x value: 'First' value: 'Second'); cr.	Transcript show: (x value: 'First' value: 'Second'); cr.	Transcript show: (x value: 'First' value: 'Second'); cr.
x := [| z | z := 1.].	x := [| z | z := 1.].	x := [| z | z := 1.].
Method calls		
| x |	| x |	| x |
x := 2 sqrt.	x := 2 sqrt.	x := 2 sqrt.
x := 2 raisedTo: 10.	x := 2 raisedTo: 10.	x := 2 raisedTo: 10.
x := 194 * 9.	x := 194 * 9.	x := 194 * 9.
Transcript show: (194 * 9) printString; cr.	Transcript show: (194 * 9) printString; cr.	Transcript show: (194 * 9) printString; cr.
x := 2 perform: #sqrt.	x := 2 perform: #sqrt.	x := 2 perform: #sqrt.
Transcript show: 'hello '; show: 'world'; cr.	Transcript show: 'hello '; show: 'world'; cr.	Transcript show: 'hello '; show: 'world'; cr.
x := 3 + 2; * 100.	x := 3 + 2; * 100.	x := 3 + 2; * 100.
Conditional Statements		
| x |	| x |	| x |
x > 10 ifTrue: [Transcript show: 'ifTrue'; cr].	x > 10 ifTrue: [Transcript show: 'ifTrue'; cr].	x > 10 ifTrue: [Transcript show: 'ifTrue'; cr].

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x > 10 ifFalse: [Transcript show: 'ifFalse'; cr].	x > 10 ifFalse: [Transcript show: 'ifFalse'; cr].	x > 10 ifFalse: [Transcript show: 'ifFalse'; cr].
x > 10 ifTrue: [Transcript show: 'ifTrue'; cr] ifFalse: [Transcript show: 'ifFalse'; cr].	x > 10 ifTrue: [Transcript show: 'ifTrue'; cr] ifFalse: [Transcript show: 'ifFalse'; cr].	x > 10 ifTrue: [Transcript show: 'ifTrue'; cr] ifFalse: [Transcript show: 'ifFalse'; cr].
x > 10 ifFalse: [Transcript show: 'ifFalse'; cr] ifTrue: [Transcript show: 'ifTrue'; cr]. Transcript show: (x > 10 ifTrue: ['ifTrue'] ifFalse: ['ifFalse']); cr. Transcript show: (x > 10 ifTrue: [x > 5 ifTrue: ['A'] ifFalse: ['B']] ifFalse: ['C']); cr.	x > 10 ifFalse: [Transcript show: 'ifFalse'; cr] ifTrue: [Transcript show: 'ifTrue'; cr]. Transcript show: (x > 10 ifTrue: ['ifTrue'] ifFalse: ['ifFalse']); cr. Transcript show: (x > 10 ifTrue: [x > 5 ifTrue: ['A'] ifFalse: ['B']] ifFalse: ['C']); cr.	x > 10 ifFalse: [Transcript show: 'ifFalse'; cr] ifTrue: [Transcript show: 'ifTrue'; cr]. Transcript show: (x > 10 ifTrue: ['ifTrue'] ifFalse: ['ifFalse']); cr. Transcript show: (x > 10 ifTrue: [x > 5 ifTrue: ['A'] ifFalse: ['B']] ifFalse: ['C']); cr.
switch := Dictionary new.	switch := Dictionary new.	switch := Dictionary new.
switch at: $A put: [Transcript show: 'Case A'; cr].	switch at: $A put: [Transcript show: 'Case A'; cr].	switch at: $A put: [Transcript show: 'Case A'; cr].
switch at: $B put: [Transcript show: 'Case B'; cr].	switch at: $B put: [Transcript show: 'Case B'; cr].	switch at: $B put: [Transcript show: 'Case B'; cr].
switch at: $C put: [Transcript show: 'Case C'; cr].	switch at: $C put: [Transcript show: 'Case C'; cr].	switch at: $C put: [Transcript show: 'Case C'; cr].
result := (switch at: $B) value.	result := (switch at: $B) value.	result := (switch at: $B) value.
Iteration statements		
| x y |	| x y |	| x y |
x := 4. y := 1.	x := 4. y := 1.	x := 4. y := 1.
[x > 0] whileTrue: [x := x - 1. y := y * 2].	[x > 0] whileTrue: [x := x - 1. y := y * 2].	[x > 0] whileTrue: [x := x - 1. y := y * 2].
[x >= 4] whileFalse: [x := x + 1. y := y * 2].	[x >= 4] whileFalse: [x := x + 1. y := y * 2].	[x >= 4] whileFalse: [x := x + 1. y := y * 2].
x timesRepeat: [y := y * 2].	x timesRepeat: [y := y * 2].	x timesRepeat: [y := y * 2].

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
1 to: x do: [:a | y := y * 2].	1 to: x do: [:a | y := y * 2].	1 to: x do: [:a | y := y * 2].
1 to: x by: 2 do: [:a | y := y / 2].	1 to: x by: 2 do: [:a | y := y / 2].	1 to: x by: 2 do: [:a | y := y / 2].
#(5 4 3) do: [:a | x := x + a].	#(5 4 3) do: [:a | x := x + a].	#(5 4 3) do: [:a | x := x + a].
Character		
| x y |	| x y |	| x y |
x := $A.	x := $A.	x := $A.
y := x isLowercase.	y := x isLowercase.	y := x isLowercase.
y := x isUppercase.	y := x isUppercase.	y := x isUppercase.
y := x isLetter.	y := x isLetter.	y := x isLetter.
y := x isDigit.	y := x isDigit.	y := x isDigit.
y := x isAlphaNumeric.	y := x isAlphaNumeric.	y := x isAlphaNumeric.
y := x isSeparator.	y := x isSeparator.	y := x isSeparator.
y := x isVowel.	y := x isVowel.	y := x isVowel.
y := x digitValue.	y := x digitValue.	y := x digitValue.
y := x asLowercase.	y := x asLowercase.	y := x asLowercase.
y := x asUppercase.	y := x asUppercase.	y := x asUppercase.
y := x asciiValue.	y := x asciiValue.	y := x asciiValue.
y := x asString.	y := x asString.	y := x asString.
b := $A <= $B.	b := $A <= $B.	b := $A <= $B.
y := $A max: $B.	y := $A max: $B.	y := $A max: $B.
Symbol		
| b x y |	| b x y |	| b x y |
x := #Hello.	x := #Hello.	x := #Hello.
y := 'String', 'Concatenation'.	y := 'String', 'Concatenation'.	y := 'String', 'Concatenation'.
b := x isEmpty.	b := x isEmpty.	b := x isEmpty.
y := x size.	y := x size.	y := x size.
y := x at: 2.	y := x at: 2.	y := x at: 2.
y := x copyFrom: 2 to: 4.	y := x copyFrom: 2 to: 4.	y := x copyFrom: 2 to: 4.
y := x indexOf: $e ifAbsent: [0].	y := x indexOf: $e ifAbsent: [0].	y := x indexOf: $e ifAbsent: [0].
x do: [:a | Transcript show: a printString; cr].	x do: [:a | Transcript show: a printString; cr].	x do: [:a | Transcript show: a printString; cr].

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
b := x conform: [:a | (a >= $a) & (a <= $z)].	b := x conform: [:a | (a >= $a) & (a <= $z)].	b := x conform: [:a | (a >= $a) & (a <= $z)].
y := x select: [:a | a > $a].	y := x select: [:a | a > $a].	y := x select: [:a | a > $a].
y := x asString.	y := x asString.	y := x asString.
y := x asText.	y := x asText.	y := x asText.
y := x asArray.	y := x asArray.	y := x asArray.
y := x asOrderedCollection.	y := x asOrderedCollection.	y := x asOrderedCollection.
y := x asSortedCollection.	y := x asSortedCollection.	y := x asSortedCollection.
y := x asBag.	y := x asBag.	y := x asBag.
y := x asSet.	y := x asSet.	y := x asSet.
String		
| b x y |	| b x y |	| b x y |
x := 'This is a string'.	x := 'This is a string'.	x := 'This is a string'.
x := 'String', 'Concatenation'.	x := 'String', 'Concatenation'.	x := 'String', 'Concatenation'.
b := x isEmpty.	b := x isEmpty.	b := x isEmpty.
y := x size.	y := x size.	y := x size.
y := x at: 2.	y := x at: 2.	y := x at: 2.
y := x copyFrom: 2 to: 4.	y := x copyFrom: 2 to: 4.	y := x copyFrom: 2 to: 4.
y := x indexOf: $a ifAbsent: [0].	y := x indexOf: $a ifAbsent: [0].	y := x indexOf: $a ifAbsent: [0].
x := String new: 4.	x := String new: 4.	x := String new: 4.
x at: 1 put: $a; at: 2 put: $b; at: 3 put: $c; at: 4 put: $e.	x at: 1 put: $a; at: 2 put: $b; at: 3 put: $c; at: 4 put: $e.	x at: 1 put: $a; at: 2 put: $b; at: 3 put: $c; at: 4 put: $e.
x := String with: $a with: $b with: $c with: $d.	x := String with: $a with: $b with: $c with: $d.	x := String with: $a with: $b with: $c with: $d.
x do: [:a | Transcript show: a printString; cr].	x do: [:a | Transcript show: a printString; cr].	x do: [:a | Transcript show: a printString; cr].
b := x conform: [:a | (a >= $a) & (a <= $z)].	b := x conform: [:a | (a >= $a) & (a <= $z)].	b := x conform: [:a | (a >= $a) & (a <= $z)].
y := x select: [:a | a > $a].	y := x select: [:a | a > $a].	y := x select: [:a | a > $a].
y := x asSymbol.	y := x asSymbol.	y := x asSymbol.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
y := x asArray.	y := x asArray.	y := x asArray.
x := 'ABCD' asByteArray.	x := 'ABCD' asByteArray.	x := 'ABCD' asByteArray.
y := x asOrderedCollection.	y := x asOrderedCollection.	y := x asOrderedCollection.
y := x asSortedCollection.	y := x asSortedCollection.	y := x asSortedCollection.
y := x asBag.	y := x asBag.	y := x asBag.
y := x asSet.	y := x asSet.	y := x asSet.
y := x shuffled.	y := x shuffled.	y := x shuffled.
OrderedCollection		
| b x y sum max |	| b x y sum max |	| b x y sum max |
x := OrderedCollection with: 4 with: 3 with: 2 with: 1.	x := OrderedCollection with: 4 with: 3 with: 2 with: 1.	x := OrderedCollection with: 4 with: 3 with: 2 with: 1.
x := OrderedCollection new.	x := OrderedCollection new.	x := OrderedCollection new.
x add: 3; add: 2; add: 1; add: 4; yourself.	x add: 3; add: 2; add: 1; add: 4; yourself.	x add: 3; add: 2; add: 1; add: 4; yourself.
y := x addFirst: 5.	y := x addFirst: 5.	y := x addFirst: 5.
y := x removeFirst.	y := x removeFirst.	y := x removeFirst.
y := x addLast: 6.	y := x addLast: 6.	y := x addLast: 6.
y := x removeLast.	y := x removeLast.	y := x removeLast.
y := x addAll: #(7 8 9).	y := x addAll: #(7 8 9).	y := x addAll: #(7 8 9).
y := x removeAll: #(7 8 9).	y := x removeAll: #(7 8 9).	y := x removeAll: #(7 8 9).
x at: 2 put: 3.	x at: 2 put: 3.	x at: 2 put: 3.
y := x remove: 5 ifAbsent: [].	y := x remove: 5 ifAbsent: [].	y := x remove: 5 ifAbsent: [].
b := x isEmpty.	b := x isEmpty.	b := x isEmpty.
y := x size.	y := x size.	y := x size.
y := x at: 2.	y := x at: 2.	y := x at: 2.
y := x first.	y := x first.	y := x first.
y := x last.	y := x last.	y := x last.
b := x includes: 5.	b := x includes: 5.	b := x includes: 5.
y := x copyFrom: 2 to: 3.	y := x copyFrom: 2 to: 3.	y := x copyFrom: 2 to: 3.
y := x indexOf: 3 ifAbsent: [0].	y := x indexOf: 3 ifAbsent: [0].	y := x indexOf: 3 ifAbsent: [0].
y := x occurrencesOf: 3.	y := x occurrencesOf: 3.	y := x occurrencesOf: 3.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x do: [:a | Transcript show: a printString; cr].	x do: [:a | Transcript show: a printString; cr].	x do: [:a | Transcript show: a printString; cr].
b := x conform: [:a | (a >= 1) & (a <= 4)].	b := x conform: [:a | (a >= 1) & (a <= 4)].	b := x conform: [:a | (a >= 1) & (a <= 4)].
y := x select: [:a | a > 2].	y := x select: [:a | a > 2].	y := x select: [:a | a > 2].
y := x reject: [:a | a < 2].	y := x reject: [:a | a < 2].	y := x reject: [:a | a < 2].
y := x collect: [:a | a + a].	y := x collect: [:a | a + a].	y := x collect: [:a | a + a].
y := x detect: [:a | a > 3] ifNone: [].	y := x detect: [:a | a > 3] ifNone: [].	y := x detect: [:a | a > 3] ifNone: [].
sum := 0. x do: [:a | sum := sum + a]. sum.	sum := 0. x do: [:a | sum := sum + a]. sum.	sum := 0. x do: [:a | sum := sum + a]. sum.
sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)].	sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)].	sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)].
sum := x inject: 0 into: [:a :c | a + c].	sum := x inject: 0 into: [:a :c | a + c].	sum := x inject: 0 into: [:a :c | a + c].
max := x inject: 0 into: [:a :c | (a > c) ifTrue: [a] ifFalse: [c]].	max := x inject: 0 into: [:a :c | (a > c) ifTrue: [a] ifFalse: [c]].	max := x inject: 0 into: [:a :c | (a > c) ifTrue: [a] ifFalse: [c]].
y := x shuffled.	y := x shuffled.	y := x shuffled.
y := x asArray.	y := x asArray.	y := x asArray.
y := x asOrderedCollection.	y := x asOrderedCollection.	y := x asOrderedCollection.
y := x asSortedCollection.	y := x asSortedCollection.	y := x asSortedCollection.
y := x asBag.	y := x asBag.	y := x asBag.
y := x asSet.	y := x asSet.	y := x asSet.
Interval		
| b x y sum max |	| b x y sum max |	| b x y sum max |
x := Interval from: 5 to: 10.	x := Interval from: 5 to: 10.	x := Interval from: 5 to: 10.
x := 5 to: 10.	x := 5 to: 10.	x := 5 to: 10.
x := Interval from: 5 to: 10 by: 2.	x := Interval from: 5 to: 10 by: 2.	x := Interval from: 5 to: 10 by: 2.
x := 5 to: 10 by: 2.	x := 5 to: 10 by: 2.	x := 5 to: 10 by: 2.
b := x isEmpty.	b := x isEmpty.	b := x isEmpty.
y := x size.	y := x size.	y := x size.
x includes: 9.	x includes: 9.	x includes: 9.
x do: [:k | Transcript show: k printString; cr].	x do: [:k | Transcript show: k printString; cr].	x do: [:k | Transcript show: k printString; cr].
b := x conform: [:a | (a >= 1) & (a <= 4)].	b := x conform: [:a | (a >= 1) & (a <= 4)].	b := x conform: [:a | (a >= 1) & (a <= 4)].

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
y := x select: [:a | a > 7].	y := x select: [:a | a > 7].	y := x select: [:a | a > 7].
y := x reject: [:a | a < 2].	y := x reject: [:a | a < 2].	y := x reject: [:a | a < 2].
y := x collect: [:a | a + a].	y := x collect: [:a | a + a].	y := x collect: [:a | a + a].
y := x detect: [:a | a > 3] ifNone: [].	y := x detect: [:a | a > 3] ifNone: [].	y := x detect: [:a | a > 3] ifNone: [].
sum := 0. x do: [:a | sum := sum + a]. sum.	sum := 0. x do: [:a | sum := sum + a]. sum.	sum := 0. x do: [:a | sum := sum + a]. sum.
sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)].	sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)].	sum := 0. 1 to: (x size) do: [:a | sum := sum + (x at: a)].
sum := x inject: 0 into: [:a :c | a + c].	sum := x inject: 0 into: [:a :c | a + c].	sum := x inject: 0 into: [:a :c | a + c].
y := x asArray.	y := x asArray.	y := x asArray.
y := x asOrderedCollection.	y := x asOrderedCollection.	y := x asOrderedCollection.
y := x asSortedCollection.	y := x asSortedCollection.	y := x asSortedCollection.
y := x asBag.	y := x asBag.	y := x asBag.
y := x asSet.	y := x asSet.	y := x asSet.
Associations		
| x y |	| x y |	| x y |
x := #myVar->'hello'.	x := #myVar->'hello'.	x := #myVar->'hello'.
y := x key.	y := x key.	y := x key.
y := x value.	y := x value.	y := x value.
Dictionary		
| b x y |	| b x y |	| b x y |
x := Dictionary new.	x := Dictionary new.	x := Dictionary new.
x add: #a->4; add: #b->3; add: #c->1; add: #d->2; yourself.	x add: #a->4; add: #b->3; add: #c->1; add: #d->2; yourself.	x add: #a->4; add: #b->3; add: #c->1; add: #d->2; yourself.
x at: #e put: 3.	x at: #e put: 3.	x at: #e put: 3.
b := x isEmpty.	b := x isEmpty.	b := x isEmpty.
y := x size.	y := x size.	y := x size.
y := x at: #a ifAbsent: [].	y := x at: #a ifAbsent: [].	y := x at: #a ifAbsent: [].
y := x keyAtValue: 3 ifAbsent: [].	y := x keyAtValue: 3 ifAbsent: [].	y := x keyAtValue: 3 ifAbsent: [].
y := x removeKey: #e ifAbsent: [].	y := x removeKey: #e ifAbsent: [].	y := x removeKey: #e ifAbsent: [].
b := x includes: 3.	b := x includes: 3.	b := x includes: 3.
b := x includesKey: #a.	b := x includesKey: #a.	b := x includesKey: #a.

DejaVu Sans 9 1x space

```
y := x occurrencesOf: 3.
y := x keys.
y := x values.
x do: [:a | Transcript show: a printString; cr].
x keysDo: [:a | Transcript show: a printString; cr].
x associationsDo: [:a | Transcript show: a printString; cr].
x keysAndValuesDo: [:aKey :aValue | Transcript
  show: aKey printString; space;
  show: aValue printString; cr].
b := x conform: [:a | (a >= 1) & (a <= 4)].
y := x select: [:a | a > 2].
y := x reject: [:a | a < 2].
y := x collect: [:a | a + a].
y := x detect: [:a | a > 3] ifNone: [].
sum := 0. x do: [:a | sum := sum + a]. sum.
sum := x inject: 0 into: [:a :c | a + c].
y := x asArray.
y := x asOrderedCollection.
y := x asSortedCollection.
y := x asBag.
y := x asSet.
Smalltalk at: #CMRGlobal put: 'CMR entry'.
x := Smalltalk at: #CMRGlobal.
Transcript show: (CMRGlobal printString).
Smalltalk keys do: [ :k |
  ((Smalltalk at: k) isKindOf: Class)
    ifFalse: [Transcript show: k printString; cr]].
Smalltalk at: #CMRDictionary put: (Dictionary new).
```

DejaVu Sans 9 2x space

```
y := x occurrencesOf: 3.
y := x keys.
y := x values.
x do: [:a | Transcript show: a printString; cr].
x keysDo: [:a | Transcript show: a printString; cr].
x associationsDo: [:a | Transcript show: a
printString; cr].
x keysAndValuesDo: [:aKey :aValue | Transcript
    show: aKey printString; space;
    show: aValue printString; cr].
b := x conform: [:a | (a >= 1) & (a <= 4)].
y := x select: [:a | a > 2].
y := x reject: [:a | a < 2].
y := x collect: [:a | a + a].
y := x detect: [:a | a > 3] ifNone: [].
sum := 0. x do: [:a | sum := sum + a]. sum.
sum := x inject: 0 into: [:a :c | a + c].
y := x asArray.
y := x asOrderedCollection.
y := x asSortedCollection.
y := x asBag.
y := x asSet.
Smalltalk at: #CMRGlobal put: 'CMR entry'.
x := Smalltalk at: #CMRGlobal.
Transcript show: (CMRGlobal printString).
Smalltalk keys do: [ :k |
    ((Smalltalk at: k) isKindOf: Class)
        ifFalse: [Transcript show: k printString; cr]].
Smalltalk at: #CMRDictionary put: (Dictionary new).
```

Source Code Pro 9

```
y := x occurrencesOf: 3.
y := x keys.
y := x values.
x do: [:a | Transcript show: a printString;
cr].
x keysDo: [:a | Transcript show: a
printString; cr].
x associationsDo: [:a | Transcript show: a
printString; cr].
x keysAndValuesDo: [:aKey :aValue | Transcript
   show: aKey printString; space;
   show: aValue printString; cr].
b := x conform: [:a | (a >= 1) & (a <= 4)].
y := x select: [:a | a > 2].
y := x reject: [:a | a < 2].
y := x collect: [:a | a + a].
y := x detect: [:a | a > 3] ifNone: [].
sum := 0. x do: [:a | sum := sum + a]. sum.
sum := x inject: 0 into: [:a :c | a + c].
y := x asArray.
y := x asOrderedCollection.
y := x asSortedCollection.
y := x asBag.
y := x asSet.
Smalltalk at: #CMRGlobal put: 'CMR entry'.
x := Smalltalk at: #CMRGlobal.
Transcript show: (CMRGlobal printString).
Smalltalk keys do: [ :k |
   ((Smalltalk at: k) isKindOf: Class)
      ifFalse: [Transcript show: k
printString; cr]].
Smalltalk at: #CMRDictionary put: (Dictionary
new).
```

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
CMRDictionary at: #MyVar1 put: 'hello1'.	CMRDictionary at: #MyVar1 put: 'hello1'.	CMRDictionary at: #MyVar1 put: 'hello1'.
CMRDictionary add: #MyVar2->'hello2'.	CMRDictionary add: #MyVar2->'hello2'.	CMRDictionary add: #MyVar2->'hello2'.
CMRDictionary size.	CMRDictionary size.	CMRDictionary size.
CMRDictionary keys do: [:k |	CMRDictionary keys do: [:k |	CMRDictionary keys do: [:k |
Transcript show: k printString; cr].	Transcript show: k printString; cr].	Transcript show: k printString; cr].
CMRDictionary values do: [:k |	CMRDictionary values do: [:k |	CMRDictionary values do: [:k |
Transcript show: k printString; cr].	Transcript show: k printString; cr].	Transcript show: k printString; cr].
CMRDictionary keysAndValuesDo: [:aKey :aValue |	CMRDictionary keysAndValuesDo: [:aKey :aValue |	CMRDictionary keysAndValuesDo: [:aKey :aValue |
Transcript	Transcript	Transcript
show: aKey printString;	show: aKey printString;	show: aKey printString;
space;	space;	space;
show: aValue printString;	show: aValue printString;	show: aValue printString;
cr].	cr].	cr].
CMRDictionary associationsDo: [:aKeyValue |	CMRDictionary associationsDo: [:aKeyValue |	CMRDictionary associationsDo: [:aKeyValue |
Transcript show: aKeyValue printString; cr].	Transcript show: aKeyValue printString; cr].	Transcript show: aKeyValue printString; cr].
Smalltalk removeKey: #CMRGlobal ifAbsent: [].	Smalltalk removeKey: #CMRGlobal ifAbsent: [].	Smalltalk removeKey: #CMRGlobal ifAbsent: [].
Smalltalk removeKey: #CMRDictionary ifAbsent: [].	Smalltalk removeKey: #CMRDictionary ifAbsent: [].	Smalltalk removeKey: #CMRDictionary ifAbsent: [].
Internal Stream		
| b x ios |	| b x ios |	| b x ios |
ios := ReadStream on: 'Hello read stream'.	ios := ReadStream on: 'Hello read stream'.	ios := ReadStream on: 'Hello read stream'.
ios := ReadStream on: 'Hello read stream' from: 1 to: 5.	ios := ReadStream on: 'Hello read stream' from: 1 to: 5.	ios := ReadStream on: 'Hello read stream' from: 1 to: 5.
[(x := ios nextLine) notNil]	[(x := ios nextLine) notNil]	[(x := ios nextLine) notNil]
whileTrue: [Transcript show: x; cr].	whileTrue: [Transcript show: x; cr].	whileTrue: [Transcript show: x; cr].
ios position: 3.	ios position: 3.	ios position: 3.
ios position.	ios position.	ios position.
x := ios next.	x := ios next.	x := ios next.
x := ios peek.	x := ios peek.	x := ios peek.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x := ios contents.	x := ios contents.	x := ios contents.
b := ios atEnd.	b := ios atEnd.	b := ios atEnd.
ios := ReadWriteStream on: 'Hello read stream'.	ios := ReadWriteStream on: 'Hello read stream'.	ios := ReadWriteStream on: 'Hello read stream'.
ios := ReadWriteStream on: 'Hello read stream' from: 1 to: 5.	ios := ReadWriteStream on: 'Hello read stream' from: 1 to: 5.	ios := ReadWriteStream on: 'Hello read stream' from: 1 to: 5.
ios := ReadWriteStream with: 'Hello read stream'.	ios := ReadWriteStream with: 'Hello read stream'.	ios := ReadWriteStream with: 'Hello read stream'.
ios := ReadWriteStream with: 'Hello read stream' from: 1 to: 10.	ios := ReadWriteStream with: 'Hello read stream' from: 1 to: 10.	ios := ReadWriteStream with: 'Hello read stream' from: 1 to: 10.
ios position: 0.	ios position: 0.	ios position: 0.
[(x := ios nextLine) notNil]	[(x := ios nextLine) notNil]	[(x := ios nextLine) notNil]
whileTrue: [Transcript show: x; cr].	whileTrue: [Transcript show: x; cr].	whileTrue: [Transcript show: x; cr].
ios position: 6.	ios position: 6.	ios position: 6.
ios position.	ios position.	ios position.
ios nextPutAll: 'Chris'.	ios nextPutAll: 'Chris'.	ios nextPutAll: 'Chris'.
x := ios next.	x := ios next.	x := ios next.
x := ios peek.	x := ios peek.	x := ios peek.
x := ios contents.	x := ios contents.	x := ios contents.
b := ios atEnd.	b := ios atEnd.	b := ios atEnd.
FileStream		
| b x ios |	| b x ios |	| b x ios |
ios := FileStream newFileNamed: 'ios.txt'.	ios := FileStream newFileNamed: 'ios.txt'.	ios := FileStream newFileNamed: 'ios.txt'.
ios nextPut: $H; cr.	ios nextPut: $H; cr.	ios nextPut: $H; cr.
ios nextPutAll: 'Hello File'; cr.	ios nextPutAll: 'Hello File'; cr.	ios nextPutAll: 'Hello File'; cr.
'Hello File' printOn: ios.	Hello File' printOn: ios.	'Hello File' printOn: ios.
'Hello File' storeOn: ios.	Hello File' storeOn: ios.	'Hello File' storeOn: ios.
ios close.	ios close.	ios close.
ios := FileStream oldFileNamed: 'ios.txt'.	ios := FileStream oldFileNamed: 'ios.txt'.	ios := FileStream oldFileNamed: 'ios.txt'.
[(x := ios nextLine) notNil]	[(x := ios nextLine) notNil]	[(x := ios nextLine) notNil]
whileTrue: [Transcript show: x; cr].	whileTrue: [Transcript show: x; cr].	whileTrue: [Transcript show: x; cr].
ios position: 3.	ios position: 3.	ios position: 3.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x := ios position.	x := ios position.	x := ios position.
x := ios next.	x := ios next.	x := ios next.
x := ios peek.	x := ios peek.	x := ios peek.
b := ios atEnd.	b := ios atEnd.	b := ios atEnd.
ios close.	ios close.	ios close.
Date		
| x y |	| x y |	| x y |
x := Date today.	x := Date today.	x := Date today.
x := Date dateAndTimeNow.	x := Date dateAndTimeNow.	x := Date dateAndTimeNow.
x := Date readFromString: '01/02/1999'.	x := Date readFromString: '01/02/1999'.	x := Date readFromString: '01/02/1999'.
x := Date newDay: 12 month: #July year: 1999	x := Date newDay: 12 month: #July year: 1999	x := Date newDay: 12 month: #July year: 1999
x := Date fromDays: 36000.	x := Date fromDays: 36000.	x := Date fromDays: 36000.
y := Date dayOfWeek: #Monday.	y := Date dayOfWeek: #Monday.	y := Date dayOfWeek: #Monday.
y := Date indexOfMonth: #January.	y := Date indexOfMonth: #January.	y := Date indexOfMonth: #January.
y := Date daysInMonth: 2 forYear: 1996.	y := Date daysInMonth: 2 forYear: 1996.	y := Date daysInMonth: 2 forYear: 1996.
y := Date daysInYear: 1996.	y := Date daysInYear: 1996.	y := Date daysInYear: 1996.
y := Date nameOfDay: 1	y := Date nameOfDay: 1	y := Date nameOfDay: 1
y := Date nameOfMonth: 1.	y := Date nameOfMonth: 1.	y := Date nameOfMonth: 1.
y := Date leapYear: 1996.	y := Date leapYear: 1996.	y := Date leapYear: 1996.
y := x weekday.	y := x weekday.	y := x weekday.
y := x previous: #Monday.	y := x previous: #Monday.	y := x previous: #Monday.
y := x dayOfMonth.	y := x dayOfMonth.	y := x dayOfMonth.
y := x day.	y := x day.	y := x day.
y := x firstDayOfMonth.	y := x firstDayOfMonth.	y := x firstDayOfMonth.
y := x monthName.	y := x monthName.	y := x monthName.
y := x monthIndex.	y := x monthIndex.	y := x monthIndex.
y := x daysInMonth.	y := x daysInMonth.	y := x daysInMonth.
y := x year.	y := x year.	y := x year.
y := x daysInYear.	y := x daysInYear.	y := x daysInYear.
y := x daysLeftInYear.	y := x daysLeftInYear.	y := x daysLeftInYear.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
y := x asSeconds.	y := x asSeconds.	y := x asSeconds.
y := x addDays: 10.	y := x addDays: 10.	y := x addDays: 10.
y := x subtractDays: 10.	y := x subtractDays: 10.	y := x subtractDays: 10.
y := x subtractDate: (Date today).	y := x subtractDate: (Date today).	y := x subtractDate: (Date today).
y := x printFormat: #(2 1 3 $/ 1 1).	y := x printFormat: #(2 1 3 $/ 1 1).	y := x printFormat: #(2 1 3 $/ 1 1).
b := (x <= Date today).	b := (x <= Date today).	b := (x <= Date today).
Time		
| x y |	| x y |	| x y |
x := Time now.	x := Time now.	x := Time now.
x := Time dateAndTimeNow.	x := Time dateAndTimeNow.	x := Time dateAndTimeNow.
x := Time readFromString: '3:47:26 pm'.	x := Time readFromString: '3:47:26 pm'.	x := Time readFromString: '3:47:26 pm'.
x := Time fromSeconds: (60 * 60 * 4).	x := Time fromSeconds: (60 * 60 * 4).	x := Time fromSeconds: (60 * 60 * 4).
y := Time millisecondClockValue.	y := Time millisecondClockValue.	y := Time millisecondClockValue.
y := Time totalSeconds.	y := Time totalSeconds.	y := Time totalSeconds.
y := x seconds.	y := x seconds.	y := x seconds.
y := x minutes.	y := x minutes.	y := x minutes.
y := x hours.	y := x hours.	y := x hours.
y := x addTime: (Time now).	y := x addTime: (Time now).	y := x addTime: (Time now).
y := x subtractTime: (Time now).	y := x subtractTime: (Time now).	y := x subtractTime: (Time now).
y := x asSeconds.	y := x asSeconds.	y := x asSeconds.
x := Time millisecondsToRun: [1 to: 1000 do: [:index | y := 3.14 * index]].	x := Time millisecondsToRun: [1 to: 1000 do: [:index | y := 3.14 * index]].	x := Time millisecondsToRun: [1 to: 1000 do: [:index | y := 3.14 * index]].
b := (x <= Time now).	b := (x <= Time now).	b := (x <= Time now).
Point		
| x y |	| x y |	| x y |
x := 200@100.	x := 200@100.	x := 200@100.
y := x x.	y := x x.	y := x x.
y := x y.	y := x y.	y := x y.
x := 200@100 negated.	x := 200@100 negated.	x := 200@100 negated.
x := (-200@-100) abs.	x := (-200@-100) abs.	x := (-200@-100) abs.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x := (200.5@100.5) rounded.	x := (200.5@100.5) rounded.	x := (200.5@100.5) rounded.
x := (200.5@100.5) truncated.	x := (200.5@100.5) truncated.	x := (200.5@100.5) truncated.
x := 200@100 + 100.	x := 200@100 + 100.	x := 200@100 + 100.
x := 200@100 - 100.	x := 200@100 - 100.	x := 200@100 - 100.
x := 200@100 * 2.	x := 200@100 * 2.	x := 200@100 * 2.
x := 200@100 / 2.	x := 200@100 / 2.	x := 200@100 / 2.
x := 200@100 // 2.	x := 200@100 // 2.	x := 200@100 // 2.
x := 200@100 \\ 3.	x := 200@100 \\ 3.	x := 200@100 \\ 3.
x := 200@100 + 50@25.	x := 200@100 + 50@25.	x := 200@100 + 50@25.
x := 200@100 - 50@25.	x := 200@100 - 50@25.	x := 200@100 - 50@25.
x := 200@100 * 3@4.	x := 200@100 * 3@4.	x := 200@100 * 3@4.
x := 200@100 // 3@4.	x := 200@100 // 3@4.	x := 200@100 // 3@4.
x := 200@100 max: 50@200.	x := 200@100 max: 50@200.	x := 200@100 max: 50@200.
x := 200@100 min: 50@200.	x := 200@100 min: 50@200.	x := 200@100 min: 50@200.
x := 20@5 dotProduct: 10@2.	x := 20@5 dotProduct: 10@2.	x := 20@5 dotProduct: 10@2.
Dynamic message calling/compiling		
| receiver message result argument keyword1 keyword2 argument1 argument2 |	| receiver message result argument keyword1 keyword2 argument1 argument2 |	| receiver message result argument keyword1 keyword2 argument1 argument2 |
unary message		
receiver := 5.	receiver := 5.	receiver := 5.
message := 'factorial' asSymbol.	message := 'factorial' asSymbol.	message := 'factorial' asSymbol.
result := receiver perform: message.	result := receiver perform: message.	result := receiver perform: message.
result := Compiler evaluate: ((receiver storeString), ' ', message).	result := Compiler evaluate: ((receiver storeString), ' ', message).	result := Compiler evaluate: ((receiver storeString), ' ', message).
result := (Message new setSelector: message arguments: #()) sentTo: receiver.	result := (Message new setSelector: message arguments: #()) sentTo: receiver.	result := (Message new setSelector: message arguments: #()) sentTo: receiver.
binary message		
receiver := 1.	receiver := 1.	receiver := 1.
message := '+' asSymbol.	message := '+' asSymbol.	message := '+' asSymbol.
argument := 2.	argument := 2.	argument := 2.
result := receiver perform: message withArguments: (Array with: argument).	result := receiver perform: message withArguments: (Array with: argument).	result := receiver perform: message withArguments: (Array with: argument).

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
result := Compiler evaluate: ((receiver storeString), ' ', message, ' ', (argument storeString)).	result := Compiler evaluate: ((receiver storeString), ' ', message, ' ', (argument storeString)).	result := Compiler evaluate: ((receiver storeString), ' ', message, ' ', (argument storeString)).
result := (Message new setSelector: message arguments: (Array with: argument)) sentTo: receiver.	result := (Message new setSelector: message arguments: (Array with: argument)) sentTo: receiver.	result := (Message new setSelector: message arguments: (Array with: argument)) sentTo: receiver.
keyword messages		
receiver := 12.	receiver := 12.	receiver := 12.
keyword1 := 'between:' asSymbol.	keyword1 := 'between:' asSymbol.	keyword1 := 'between:' asSymbol.
keyword2 := 'and:' asSymbol.	keyword2 := 'and:' asSymbol.	keyword2 := 'and:' asSymbol.
argument1 := 10.	argument1 := 10.	argument1 := 10.
argument2 := 20.	argument2 := 20.	argument2 := 20.
result := receiver	result := receiver	result := receiver
perform: (keyword1, keyword2) asSymbol	perform: (keyword1, keyword2) asSymbol	perform: (keyword1, keyword2) asSymbol
withArguments: (Array with: argument1 with: argument2).	withArguments: (Array with: argument1 with: argument2).	withArguments: (Array with: argument1 with: argument2).
result := Compiler evaluate:	result := Compiler evaluate:	result := Compiler evaluate:
((receiver storeString), ' ', keyword1, (argument1 storeString) , ' ', keyword2, (argument2 storeString)).	((receiver storeString), ' ', keyword1, (argument1 storeString) , ' ', keyword2, (argument2 storeString)).	((receiver storeString), ' ', keyword1, (argument1 storeString) , ' ', keyword2, (argument2 storeString)).
result := (Message new	result := (Message new	result := (Message new
setSelector: (keyword1, keyword2) asSymbol	setSelector: (keyword1, keyword2) asSymbol	setSelector: (keyword1, keyword2) asSymbol
arguments: (Array with: argument1 with: argument2))	arguments: (Array with: argument1 with: argument2))	arguments: (Array with: argument1 with: argument2))
sentTo: receiver.	sentTo: receiver.	sentTo: receiver.
Class/meta-class		
| b x |	| b x |	| b x |
x := String name.	x := String name.	x := String name.
x := String category.	x := String category.	x := String category.
x := String comment.	x := String comment.	x := String comment.
x := String kindOfSubclass.	x := String kindOfSubclass.	x := String kindOfSubclass.
x := String definition.	x := String definition.	x := String definition.
x := String instVarNames.	x := String instVarNames.	x := String instVarNames.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x := String allInstVarNames.	x := String allInstVarNames.	x := String allInstVarNames.
x := String classVarNames.	x := String classVarNames.	x := String classVarNames.
x := String allClassVarNames.	x := String allClassVarNames.	x := String allClassVarNames.
x := String sharedPools.	x := String sharedPools.	x := String sharedPools.
x := String allSharedPools.	x := String allSharedPools.	x := String allSharedPools.
x := String selectors.	x := String selectors.	x := String selectors.
x := String sourceCodeAt: #size.	x := String sourceCodeAt: #size.	x := String sourceCodeAt: #size.
x := String allInstances.	x := String allInstances.	x := String allInstances.
x := String superclass.	x := String superclass.	x := String superclass.
x := String allSuperclasses.	x := String allSuperclasses.	x := String allSuperclasses.
x := String withAllSuperclasses.	x := String withAllSuperclasses.	x := String withAllSuperclasses.
x := String subclasses.	x := String subclasses.	x := String subclasses.
x := String allSubclasses.	x := String allSubclasses.	x := String allSubclasses.
x := String withAllSubclasses.	x := String withAllSubclasses.	x := String withAllSubclasses.
b := String instSize.	b := String instSize.	b := String instSize.
b := String isPointers.	b := String isPointers.	b := String isPointers.
b := String isBits.	b := String isBits.	b := String isBits.
b := String isBytes.	b := String isBytes.	b := String isBytes.
b := String isWords.	b := String isWords.	b := String isWords.
Object withAllSubclasses size.	Object withAllSubclasses size.	Object withAllSubclasses size.
Debugging		
| a b x |	| a b x |	| a b x |
x yourself.	x yourself.	x yourself.
String browse.	String browse.	String browse.
x inspect.	x inspect.	x inspect.
x confirm: 'Is this correct?'.	x confirm: 'Is this correct?'.	x confirm: 'Is this correct?'.
x halt.	x halt.	x halt.
x halt: 'Halt message'.	x halt: 'Halt message'.	x halt: 'Halt message'.
x notify: 'Notify text'.	x notify: 'Notify text'.	x notify: 'Notify text'.
x error: 'Error string'.	x error: 'Error string'.	x error: 'Error string'.

DejaVu Sans 9 1x space	DejaVu Sans 9 2x space	Source Code Pro 9
x doesNotUnderstand: #cmrMessage.	x doesNotUnderstand: #cmrMessage.	x doesNotUnderstand: #cmrMessage.
x shouldNotImplement.	x shouldNotImplement.	x shouldNotImplement.
x subclassResponsibility.	x subclassResponsibility.	x subclassResponsibility.
x errorImproperStore.	x errorImproperStore.	x errorImproperStore.
x errorNonIntegerIndex.	x errorNonIntegerIndex.	x errorNonIntegerIndex.
x errorSubscriptBounds.	x errorSubscriptBounds.	x errorSubscriptBounds.
x primitiveFailed.	x primitiveFailed.	x primitiveFailed.