Tue Sep 18 15:27:01 PDT 2007

Contents

Smalltalk Methods Initially defined in Our Smalltalk.

Array methods

1. **lessthan**::= *theArray < aCollection*::= *Test for lexicographic ordering*.
2. **equals**::= *theArray = aCollection*::= *Test for all elements equal*.
3. **atput**::= *theArray at: index put: value*::= *returns an array with an element change*.
4. **binaryDo**::= *theArray binaryDo: aBlock*::=*Sends aBlock subscript and element pairs*.
5. **collect**::= *theArray collect: aBlock* ::= *TheArray is fed thru aBlock to become a new array`*.
6. **copyFromlowto**::= *theArray copyFrom: low to: high* ::= *returns a subarray low..hig*.
7. **deepCopy**::= *theArray deepCopy*.
8. **deepCopyFromto**::= *theArray deepCopyFrom: low to: high* .
9. **do**::= *theArray do: aBlock*::= *sending its elements to the block one by one*.
10. **exchange**::= *theArray exchange: a and: b* ::=*returns object with elements interchanged*.
11. **grow**::= *theArray grow: aValue* ::= *produces a larger array from theArray`*.
12. **includesKey**::= *theArray includesKey: index* ::=*returns a Boolea*.
13. **new**::= *theArray new* ::= *produces a new empty array object`*.
14. **reverseDo**::= *theArray reverseDo: aBlock* ::=*sends elements from last to first into aBlock`*.
15. **select**::= *theArray select: aCond* ::= *selects elements in theArray that fit aCond giving an array`*.
16. **shallowCopy**::= *theArray shallowCopy*.
17. **size**::= *theArray size* ::=*returns an Integer equal to the number of elements*.
18. **withdo**::= *theArray with: aCollection do: aBlock* ::=*sends pairs from theArray and aCollection to aBlock*.
19. **with**::= *theArray with: aCollection ifAbsent: z do: aBlock* .
20. **with**::= *theArray with: newElement Returns a new array with newElement added to it`*.

Block methods

21. **blockContext**::= *theBlock blockContext: aContext*.
22. **checkArgumentCount**::= *theBlock checkArgumentCount: count* ::=*returns True object if number arguments in theBlock is equal to count*.
23. **value**::= *theBlock value* ::=*returns the object returned by the last statement in the bloc*.
24. **value**::= *theBlock value: x Evaluate using x as an argument and return value of last statement*.
25. **valuevalue**::= *theBlock value: x value: y Evaluate using x an y as arguments...*
valuevaluevalue:= *theBlock value: x value: y value: z Evaluate using x,y and z as argument*.
26. **whileFalse**::= *theBlock whileFalse: aBlock repeats theBlock until aBlock is true`*.
27. **whileTrue**::= *theBlock whileTrue* ::= *repeating the block if its value is Tru*.
28. **whileTrue**::= *theBlock whileTrue: aBlock repeating aBlock and while theBlock returns true*.

Boolean Methods

29. **and**::= *theBoolean and: aBlock if theBoolean is true then evaluates block. If both true returns true, else returns false*.
30. **ifFalse**::= *theBoolean ifFalse: falseBlock if theBoolean is false evaluate the block*.
31. **ifFalse**::= *theBoolean ifFalse: falseBlock ifTrue: trueBlock ifthenelse, in fact handled by either*

True or False.

32. **ifTrue**::= *theBoolean ifTrue: trueBlock if theBoolean is true then evaluate the trueBlock*.
33. **or**::= *theBoolean or: aBlock If theBoolean is false evaluates aBlock and returns True if aBlock returns true*.

ByteArray methods

34. **asString**::= *theByteArray asString* ::=*returns converted array as a string*.
35. **basicAt**::= *theByteArray basicAt: index* ::=*returns byte at index position in the ByteArray*.
36. **basicAt**::= *theByteArray basicAt: index put: value* ::=*returns array of bytes with element at index changed to be value*.
37. **size**::= *theByteArray size* ::=*returns an integer equal number of elements in array*.
38. **size**::= *theByteArray size: value ?*.

Char methods

39. **<**::= *theChar < aValue* ::=*returns Boolean true/false depending on comparison*.
40. **=**::= *theChar = aValue* ::=*returns Boolean true/false depending on comparison*.
41. **==**::= *theChar == aValue* ::=*returns true iff same objects*.
42. **asInteger**::= *theChar asInteger* ::=*returns numeric equivalent of character*.
43. **asString**::= *theChar asString* ::=*returns the string with theChar as its single element*.
44. **digitValue**::= *theChar digitValue* ::=*return result of converting theChar to a digit(decimal?)*.
45. **isAlphaNumeric**::= *theChar isAlphaNumeric* ::=*returns Boolean, true when letter or digit*.
46. **isAlphabetic**::= *theChar isAlphabetic* ::=*returns Boolean, true when letter*.
47. **isBlank**::= *theChar isBlank* ::=*returns a Boolean, true when theChar is blank*.
48. **isChar**::= *theChar isChar*.
49. **isDigit**::= *theChar isDigit* ::=*returns Boolean, true when theCahr is a digit*.
50. **isLowercase**::= *theChar isLowercase* ::=*returns a Boolean, true when theCahr is a lower case letter*.
51. **isUppercase**::= *theChar isUppercase* ::=*returns a Boolean, tru when a capital letter*.
52. **printString**::= *theChar printString* ::=*returns printable image*.
53. **value**::= *theChar value: aValue private - used for initialization`*.

Class methods

54. **addMethod**::= *theClass addMethod* ::= *Lets you add a new method to theClass - using an editor`*.
55. **addSubClass**::= *theClass addSubClass* ::= *Adds a new class with variables, name etc - interactive.`*.
56. **addSubClass**::= *theClass addSubClass: aSymbol instanceVariableNames: aString* .
57. **display**::= *theClass display*.
58. **doEdit**::= *theClass doEdit: method*.
59. **editMethod**::= *theClass editMethod: aSymbol Lets you edit method of theClass called aSymbol`*.
60. **fileOut**::= *theClass fileOut* ::=*Sends text version of theClass to open file`*.

61. **fileOutMethodsOn**::= *theClass fileOutMethodsOn: aFile*.
62. **fileOutOn**::= *theClass fileOutOn: aFile*.
63. **initialize**::= *theClass initialize ?*.
64. **instanceSize**::= *theClass instanceSize ?*.
65. **methodNamed**::= *theClass methodNamed: aSymbol ?`*.
66. **methods**::= *theClass methods ::= generates a dictionary of methods of theClass*.
67. **name**::= *theClass name ::=returns the name of theClass*.
68. **name**::= *theClass name: aString ?`*.
69. **new**::= *theClass new ::=returns a new object as an instance of class and also sends it a* new' message if the Class has one'.
70. **new**::= *theClass new: size hack out block the right size and class `*.
71. **printString**::= *theClass printString ::=returns printable image*.
72. **readInstanceVariables**::= *theClass readInstanceVariables*.
73. **readMethods**::= *theClass readMethods*.
74. **respondsTo**::= *theClass respondsTo ::=returns a collection of methods that theClass responds to*.
75. **respondsTo**::= *theClass respondsTo: theSymbol ::=returns Boolean, true if theClass responds to theSymbol*.
76. **subClasses**::= *theClass subClasses ?*.
77. **superClass**::= *theClass superClass ::=returns the Class theClass is a subclass of*.
78. **superClass**::= *theClass superClass: aClass ::=returns Boolean, true if aClass is a super class of theClass*.
79. **upSuperclassChain**::= *theClass upSuperclassChain: aBlock ?*.
80. **variables**::= *theClass variables ::=returns a Collection of the instance variables of theClass*.
81. **variables**::= *theClass variables: nameArray ?*.
82. **viewMethod**::= *theClass viewMethod: methodNameSymbol*.
83. **watch**::= *theClass watch: name* .

Collection methods

84. **<**::= *theCollection < aCollection ::=returns Boolean true/false depending on comparison*.
85. **=**::= *theCollection = aCollection ::= Test for equality*.
86. **asArray**::= *theCollection asArray ::=return an array of elements in theCollection*.
87. **asByteArray**::= *theCollection asByteArray ::=return an array of elements in theCollection*.
88. **asSet**::= *theCollection asSet ::=return a set of elements in theCollection with noduplicates*.
89. **asString**::= *theCollection asString ::=return a string representing the collection*.
90. **display**::= *theCollection display ::= output the elements in the array to the termina*.
91. **includes**::= *theCollection includes: value*.
92. **includes**::= *theCollection includes: value ::=return Boolean, True if valu is in theCollection*.
93. **inject**::= *theCollection inject: thisValue into: binaryBlock* .
94. **inject**::= *theCollection inject: thisValue into: binaryBlock ?*.
95. **isEmpty**::= *theCollection isEmpty ::=returns Boolean, true whne theCollection is empty*.
96. **occurrencesOf**::= *theCollection occurrencesOf: anObject ::=returns an Integer count of the number of times anObject occurs in theCollection*.
97. **printString**::= *theCollection printString ::=returns printable image*.
98. **size**::= *theCollection size ::=returns integer equal to number of elements in theCollection*.

99. **sort**::= *theCollection sort* ::=*returns a collection of elements in lexographic order*.
100. **sort**::= *theCollection sort: aBlock* ::=*returns sorted collection using block to compare pairs*.

Context methods

101. **arguments**::= *theContext arguments: value ?*.
102. **atput**::= *theContext at: key put: value*.
103. **blockReturn**::= *theContext blockReturn*.
104. **copy**::= *theContext copy*.
105. **executeFromcreator**::= *theContext executeFrom: value creator: interp ?*.
106. **method**::= *theContext method: value ?*.
107. **returnToBlock**::= *theContext returnToBlock: bytePtr*.
108. **temporaries**::= *theContext temporaries ?*.
109. **temporaries**::= *theContext temporaries: theTemporary ?`*.

Dictinary methods

110. **at**::= *theDictionary at: aKey ifAbsent: exceptionBlock* ::=*returns element with aKey or returns value of exceptionBlock*.
111. **at**::= *theDictionary at: aKey put: aValue Places aValue in theDictionary with key Akay*.
112. **basicRemoveKey**::= *theDictionary basicRemoveKey: aKey*::= *removes aKey from theDictionary*.
113. **binaryDo**::= *theDictionary binaryDo: aBlock* ::=*Sends pairs of subscripts and relevant values to aBlock*.
114. **display**::= *theDictionary display* ::= *show what keys are in theDictionary*.
115. **hash**::= *theDictionary hash: aKey ?*.
116. **includesKey**::= *theDictionary includesKey: aKey True if aKey is in theDictionar*.
117. **new**::= *theDictionary new* ::=*returns an empty dictionary - automatic done when any dictionary is created*.
118. **removeKey**::= *theDictionary removeKey: aKey*::= *removes aKey from theDictionary*.
119. **removeKey**::= *theDictionary removeKey: aKey ifAbsent: exceptionBlock*::= *removes aKey or evaluates the excptionBlock*.

False Methods

120. **ifTrue**::= *theFalse ifTrue: trueBlock ifFalse: falseBlock Evaluates the trueBlock*.
121. **not**::= *theFalse not* ::=*returns True*.
122. **printString**::= *theFalse printString*.
123. **xor**::= *theFalse xor: aBoolean*.

File methods

124. **asString**::= *theFile asString* .
125. **close**::= *theFile close*.

126. **delete**::= *theFile delete*.
127. **fileIn**::= *theFile fileIn*.
128. **fileIn**::= *theFile fileIn: name*.
129. **getNumber**::= *theFile getNumber*.
130. **getString**::= *theFile getString*.
131. **mode**::= *theFile mode: m*.
132. **name**::= *theFile name*.
133. **name**::= *theFile name: string*.
134. **open**::= *theFile open*.
135. **open**::= *theFile open: m*.
136. **print**::= *theFile print: aString*.
137. **printNoReturn**::= *theFile printNoReturn: aString*.
138. **readUntil**::= *theFile readUntil: conditionBlock doing: actionBlock* .
139. **saveImage**::= *theFile saveImage*.
140. **scratchFile**::= *theFile scratchFile*.

Float methods

141. *****::= *theFloat * value* ::=*returns product as floating point*.
142. **+**::= *theFloat + value* ::=*returns sum*.
143. **-**::= *theFloat - value* ::=*returns the difference*. /::= *theFloat / value* ::=*returns the result of floating point division*.
144. **<**::= *theFloat < value* ::=*returns Boolean true/false depending on comparison*.
145. **=**::= *theFloat = value* ::=*returns Boolean true/false depending on comparison*.
146. **ceiling**::= *theFloat ceiling* ::=*returns next integer equal or above theFloat*.
147. **coerce**::= *theFloat coerce: value* ::=*returns a value in a different class(?)*.
148. **exp**::= *theFloat exp* ::=*returns e to the power (theFloat)*.
149. **floor**::= *theFloat floor* ::=*returns largest integer less than of equal to theFloat*.
150. **fractionalPart**::= *theFloat fractionalPart* ::=*return the fractional part of theFloat*.
151. **generality**::= *theFloat generality* ::=*returns an integer indicating direction of coercions*.
152. **integerPart**::= *theFloat integerPart* ::=*returns integer*.
153. **isFloat**::= *theFloat isFloat*.
154. **ln**::= *theFloat ln* ::=*returns the natural logarithm of theFloat*.
155. **new**::= *theFloat new*.
156. **printString**::= *theFloat printString* ::=*returns printable image*.
157. **quo**::= *theFloat quo: value*.
158. **rounded**::= *theFloat rounded* ::=*returns an Integer closest to theFloat*.
159. **sqrt**::= *theFloat sqrt* ::=*returns the square root of theFloat*.
160. **truncated**::= *theFloat truncated* ::=*returns Integer below(?) or closer to zero(?) theFloat*.

Fraction methods

161. *****::= *theFraction * f* ::= *multiplication`*.
162. **+**::= *theFraction + f* ::= *sum, addition`*.
163. **-**::= *theFraction - f* ::= *difference, subtraction`*. /::= *theFraction / f*.
164. **<**::= *theFraction < f* ::= *Returns Boolean comparison, less than`*.

165. **=**::= *theFraction = f*.
166. **abs**::= *theFraction abs*.
167. **asFloat**::= *theFraction asFloat*.
168. **bottom**::= *theFraction bottom*.
169. **coerce**::= *theFraction coerce: x*.
170. **generality**::= *theFraction generality*.
171. **isFraction**::= *theFraction isFraction*.
172. **ln**::= *theFraction ln*.
173. **printString**::= *theFraction printString*.
174. **raisedTo**::= *theFraction raisedTo: x*.
175. **reciprocal**::= *theFraction reciprocal*.
176. **top**::= *theFraction top*.
177. **truncated**::= *theFraction truncated*.
178. **with**::= *theFraction with: t over: b*.

IndexedCollection methods

179. **addAll**::= *theIndexedCollection addAll: aCollection Adds whole of aCollection to theIndexedCollection*.
180. **asArray**::= *theIndexedCollection asArray ::=returns equivalent array*.
181. **asDictionary**::= *theIndexedCollection asDictionary ::=returns equivalent dictionary object*.
182. **at**::= *theIndexedCollection at: aKey ::=return item with aKey as key*.
183. **at**::= *theIndexedCollection at: index ifAbsent: exceptionBlock Place aValue in theIndexedCollection with index value index*.
184. **binaryInject**::= *theIndexedCollection binaryInject: thisValue into: aBlock ?*.
185. **collect**::= *theIndexedCollection collect: aBlock ::=returns a similar collection with each element the value returned by the block when given an element in theCollection as an argument*.
186. **do**::= *theIndexedCollection do: aBlock For each item in theIndexedCollection in turn apply the block to the item*.
187. **indexOf**::= *theIndexedCollection indexOf: aBlock ?*.
188. **indexOf**::= *theIndexedCollection indexOf: aBlock ifAbsent: exceptionBlock ?*.
189. **keys**::= *theIndexedCollection keys ::=returns a collection of keys identifying objects in theIndexedCollection*.
190. **select**::= *theIndexedCollection select: aBlock ::=returns a list of items which when passed to aBlock reult in a True value being returned*.
191. **values**::= *theIndexedCollection values ::=returns a set of values in theIndexedCollection*.

Integer methods

192. *****::= *theInteger * value ::=returns product as fixed point*.
193. **+**::= *theInteger + value ::=returns sum*. ,::= *theInteger , value*.
194. **-**::= *theInteger - value ::=returns the difference*. /::= *theInteger / value ::=returns the fraction*. //::= *theInteger // value ::=returns result of integer division*.
195. **<**::= *theInteger < value ::=returns Boolean true/false depending on comparison*.
196. **=**::= *theInteger = value ::=returns Boolean true/false depending on comparison*.
197. **>**::= *theInteger > value ::=returns Boolean true/false depending on comparison*.

198. ****::= *theInteger \\ value* ::=*returns result of integer division*.
199. **allMask**::= *theInteger allMask: value ?*.
200. **anyMask**::= *theInteger anyMask: value ?*.
201. **asCharacter**::= *theInteger asCharacter* ::=*returns equivalent character*.
202. **asDigit**::= *theInteger asDigit ?*.
203. **asFloat**::= *theInteger asFloat* ::=*returns floating point version of theInteger*.
204. **asFraction**::= *theInteger asFraction*.
205. **asLongInteger**::= *theInteger asLongInteger* .
206. **asString**::= *theInteger asString* ::=*returns a string representing theInteger as a decima*.
207. **bitAnd**::= *theInteger bitAnd: value rturns result of bitwise operation on theInteger and the value*.
208. **bitAt**::= *theInteger bitAt: value* ::=*returns integer(?) bit in position value in theInteger*.
209. **bitInvert**::= *theInteger bitInvert rturns result of complementing each bit in theInteger*.
210. **bitOr**::= *theInteger bitOr: value rturns result of bitwise operation on theInteger and the value*.
211. **bitShift**::= *theInteger bitShift: value* ::=*returns the result of shifting bits in theInteger*.
212. **bitXor**::= *theInteger bitXor: value rturns result of complementing each bit in theInteger*.
213. **even**::= *theInteger even* ::=*return true if theInteger is divisible by 2*.
214. **factorial**::= *theInteger factorial* ::=*returns factorial of theInteger*.
215. **gcd**::= *theInteger gcd: value* ::=*return the greatest common divisor of theInteger and value*.
216. **generality**::= *theInteger generality* ::=*returns an integer indicating direction of coercions*..
217. **isShortInteger**::= *theInteger isShortInteger*.
218. **lcm**::= *theInteger lcm: value* ::=*returns the least common multiple of theInteger and value*.
219. **new**::= *theInteger new*.
220. **odd**::= *theInteger odd* ::=*returns Boolean, true when division by 2 lease a remainder*.
221. **printString**::= *theInteger printString* ::=*returns printable image*.
222. **quo**::= *theInteger quo: value* ::=*returns the quotient of when theInteger is divided by value*.
223. **radix**::= *theInteger radix: base ?*.
224. **rem**::= *theInteger rem: value* ::=*returns the remainder when theInteger is divided by value*.
225. **timesRepeat**::= *theInteger timesRepeat: aBlock repats the evaluation of the block theInteger number of times*.
226. **truncated**::= *theInteger truncated*.

Interval methods

227. **do**::= *theInterval do: aBlock* .
228. **do**::= *theInterval do: aBlock* ::=*send each item in turn as agument to aBlock*.
229. **first**::= *theInterval first* ::= *Starts to generate items in interval, return first*.
230. **inRange**::= *theInterval inRange: value* ::=*returns Boolean: true if value is in theInterval`*.
231. **lower**::= *theInterval lower: aValue changes the lowest value in theInterval*.
232. **next**::= *theInterval next* ::= *generates the next item in interval after the last one*.
233. **reset**::= *theInterval reset*.
234. **step**::= *theInterval step: aValue specifies the step in theInterval*.
235. **upper**::= *theInterval upper: aValue changes the upper end of an interval*.

Link methods

236. **add**::= *theLink add: newValue whenFalse: aBlock ?*.
237. **at**::= *theLink at: aKey ifAbsent: exceptionBlock* ::=*return item with aKey as key*.
238. **at**::= *theLink at: aKey put: aValue ?*.
239. **binaryDo**::= *theLink binaryDo: aBlock* ::=*Sends pairs of subscripts and relevant values to aBlock*.
240. **includesKey**::= *theLink includesKey: aKey ?*.
241. **key**::= *theLink key: aKey ?*.
242. **link**::= *theLink link: aLink ?*.
243. **next**::= *theLink next*.
244. **removeKey**::= *theLink removeKey: aKey ?*.
245. **removeValue**::= *theLink removeValue: aValue ?*.
246. **reverseDo**::= *theLink reverseDo: aBlock*.
247. **size**::= *theLink size* ::=*returns integer equal to number of elements*.
248. **value**::= *theLink value ?*.
249. **value**::= *theLink value: aValue ?*.

List methods

250. **add**::= *theList add: aValue adds aValue at the head of theList*.
251. **add**::= *theList add: aValue ordered: aBlock adds aValue in the correct place to preserve order in theList*.
252. **addAll**::= *theList addAll: aValue ?*.
253. **addFirst**::= *theList addFirst: aValue add aValue before the first element of theList*.
254. **addLast**::= *theList addLast: aValue add aValue after the end of theList*.
255. **collect**::= *theList collect: aBlock elements in theList are passed to aBlock giving a list of block values*.
256. **do**::= *theList do: aBlock* ::=*send each item in turn as agument to aBlock*.
257. **first**::= *theList first* ::=*returns the first item in the list*.
258. **links**::= *theList links*.
259. **reject**::= *theList reject: aBlock* ::=*returns a list of items that cause the block to return a False value when given items as arguments*.
260. **remove**::= *theList remove: value* ::=*the value is removed from theList*.
261. **removeFirst**::= *theList removeFirst* ::=*the first value in theList is removed*.
262. **reverseDo**::= *theList reverseDo: aBlock*.
263. **select**::= *theList select: aBlock* ::=*returns list of items that aBlock gives a True value*.
264. **size**::= *theList size* ::=*returns integer equal to number of elements*.

LongInteger methods

265. *****::= *theLongInteger * n ::= multiplication`*.
266. **+**::= *theLongInteger + n ::= sum, addition`*.
267. **-**::= *theLongInteger - n ::= difference, subtraction`*.
268. **<**::= *theLongInteger < n ::= Returns Boolean comparison, less than`*.
269. **=**::= *theLongInteger = n*.
270. **abs**::= *theLongInteger abs*.
271. **asFloat**::= *theLongInteger asFloat* .

272. **bitShift**::= *theLongInteger bitShift: n*.
273. **coerce**::= *theLongInteger coerce: n*.
274. **digits**::= *theLongInteger digits*.
275. **generality**::= *theLongInteger generality*.
276. **isLongInteger**::= *theLongInteger isLongInteger*.
277. **isShortInteger**::= *theLongInteger isShortInteger*.
278. **negated**::= *theLongInteger negated*.
279. **negative**::= *theLongInteger negative*.
280. **new**::= *theLongInteger new*.
281. **printString**::= *theLongInteger printString* .
282. **quo**::= *theLongInteger quo: value* .
283. **sign**::= *theLongInteger sign: s digits: d*.
284. **timesShort**::= *theLongInteger timesShort: value* .
285. **with**::= *theLongInteger with: n bitDo: aBlock* .

Magnitude methods

286. **<**::= *theMagnitude < value* ::=*returns Boolean true/false depending on comparison*.
287. **<=**::= *theMagnitude <= value* ::=*returns Boolean true/false depending on comparison*.
288. **=**::= *theMagnitude = value* ::=*returns Boolean true/false depending on comparison*.
289. **>**::= *theMagnitude > value* ::=*returns Boolean true/false depending on comparison*.
290. **>=**::= *theMagnitude >= value* ::=*returns Boolean true/false depending on comparison*.
291. **between**::= *theMagnitude between: low and: high* ::=*returns Boolean true if theMagnitude is >=high and <= low*.
292. **isChar**::= *theMagnitude isChar*.
293. **max**::= *theMagnitude max: value* ::=*returns the largest of theMagnitude and value*.
294. **min**::= *theMagnitude min: value* ::=*returns the lower of theMagnitude and cvalue*.
295. **~=**::= *theMagnitude ~= value* ::=*returns Boolean value, true if not equal*.

Method methods

296. **compileWithClass**::= *theMethod compileWithClass: aClass ?*.
297. **display**::= *theMethod display outputs source code and compiled code of theMethod*.
298. **executeWith**::= *theMethod executeWith: arguments*.
299. **message**::= *theMethod message: aSymbol ?*.
300. **name**::= *theMethod name* ::=*returns the name of theMethod*.
301. **printString**::= *theMethod printString* ::=*returns printable image*.
302. **signature**::= *theMethod signature*.
303. **text**::= *theMethod text ?*.
304. **text**::= *theMethod text: aString ?*.
305. **watch**::= *theMethod watch: aBlock*.
306. **watchWith**::= *theMethod watchWith: arguments*.

Number methods

307. *****::= *theNumber * value* ::=*returns product with max generallity*.
308. **+**::= *theNumber + value* ::=*returns sum*.
309. **-**::= *theNumber - value* ::=*returns the difference*. /::= *theNumber / value* ::=*returns the result of floating point division*. //::= *theNumber // value* ::= *integer division, truncate towards negative infinity`*.
310. **<**::= *theNumber < value* ::=*returns Boolean true/false depending on comparison*.
311. **=**::= *theNumber = value* ::=*returns Boolean true/false depending on comparison*.
312. ****::= *theNumber \\ value* ::= *remainder after integer division `*.
313. **abs**::= *theNumber abs* ::=*returns absolute value*.
314. **ceiling**::= *theNumber ceiling* ::= *the smallest integer greater than or equal this number`*.
315. **copy**::= *theNumber copy*.
316. **exp**::= *theNumber exp* ::=*returns e to the power (theFloat)*.
317. **floor**::= *theNumber floor* ::= *The largest integer that is less than or equal to me`*.
318. **fractionalPart**::= *theNumber fractionalPart*.
319. **isInteger**::= *theNumber isInteger*.
320. **isNumber**::= *theNumber isNumber*.
321. **ln**::= *theNumber ln* ::=*returns a Floating point natural logarithm of theNumber*.
322. **log**::= *theNumber log: value Common logarithm (base 10)*.
323. **maxgen**::= *theNumber maxgen: value ?*.
324. **negated**::= *theNumber negated* ::=*return the nagative of theNumber*.
325. **negative**::= *theNumber negative* ::=*return Boolean, true if less than zero?*.
326. **positive**::= *theNumber positive* ::=*returns a Boolean, true when theNumber is greater than zero*.
327. **quo**::= *theNumber quo: value*.
328. **raisedTo**:::= *theNumber raisedTo: n* ::=*returns Number that is theNumber to the nth powe*.
329. **reciprocal**::= *theNumber reciprocal 1.0/theNumeber*.
330. **rem**:::= *theNumber rem: value*.
331. **roundTo**:::= *theNumber roundTo: value ?*.
332. **sign**::= *theNumber sign* ::=*returns an Integer sign of theNumber*.
333. **sqrt**::= *theNumber sqrt* ::=*returns the square root of theNumber as a Float*.
334. **squared**::= *theNumber squared* ::=*returns the square of the number*.
335. **strictlyPositive**::= *theNumber strictlyPositive* ::=*returns Boolean, true when greater than zero*.
336. **to**:::= *theNumber to: value* ::=*returns and Interval of Numbers in range theNumber to value*.
337. **to**:::= *theNumber to: value by: step* ::=*returns and Interval of Numbers in range theNumber to value with step between items*.
338. **trucateTo**:::= *theNumber trucateTo: value ?*.

Object methods

339. **=**::= *theObject = aValue* ::=*returns Boolean true/false depending on comparison*.
340. **==**::= *theObject == aValue* ::=*returns Boolean , true iff same object*.
341. **asString**::= *theObject asString*.
342. **assign**::= *theObject assign: name value: val*.
343. **basicAt**::= *theObject basicAt: index* ::=*returns item with index index*.
344. **basicAt**::= *theObject basicAt: index put: value* ::= *change the item at position index in theObject*.
345. **basicSize**::= *theObject basicSize* ::= *rteurns integer describing size of theObject*.

346. **class**::= *theObject class* ::= *Return the Class in which theObject is an instance*.
347. **copy**::= *theObject copy*.
348. **deepCopy**::= *theObject deepCopy* .
349. **display**::= *theObject display* ::= *outputs suitable information about object*.
350. **hash**::= *theObject hash ?*.
351. **isFloat**::= *theObject isFloat*.
352. **isFraction**::= *theObject isFraction*.
353. **isInteger**::= *theObject isInteger*.
354. **isKindOf**::= *theObject isKindOf: aClass* ::=*returns a Boolean, true if theObject is a memebr of a sublass of a subclass...of aClass*.
355. **isLongInteger**::= *theObject isLongInteger*.
356. **isMemberOf**::= *theObject isMemberOf: aClass* ::=*returns a Boolean, true if theObject is an instance of aClass*.
357. **isNil**::= *theObject isNil Boolean, true when theObject is the nill object*.
358. **isNumber**::= *theObject isNumber*.
359. **isShortInteger**::= *theObject isShortInteger*.
360. **message**::= *theObject message: m notRecognizedWithArguments: a*.
361. **new**::= *theObject new default intialisation routine for all objects*.
362. **notNil**::= *theObject notNil ?*.
363. **print**::= *theObject print outputs theObject*s printString'.
364. **printString**::= *theObject printString* ::=*returns printable image*.
365. **respondsTo**::= *theObject respondsTo: message*.
366. **shallowCopy**::= *theObject shallowCopy* .
367. **~~**::= *theObject ~~ aValue* ::=*returns Boolean, true if not the same object`*.

Random methods

368. **between**::= *theRandom between: low and: high* ::=*returns a random number uniformly distributed in low..high*.
369. **next**::= *theRandom next* ::=*returns a random Float between 0.0 and 1.0*.
370. **next**::= *theRandom next: n reurns an array of n next random values in 0.0..1.0*.
371. **randInteger**::= *theRandom randInteger: n* ::=*returns a random integer in range 1 to n*.
372. **set**::= *theRandom set: value ?*.

Scheduler methods

373. **initialize**::= *theScheduler initialize* .

Set methods

374. **add**::= *theSet add: value adds a new (nonduplicated) value*.

Smalltalk methods

375. **load**::= *theSMalltalk load: fileName (local) read in new methods, classes, etc from file fileName*.
376. **cantFindGlobal**::= *theSmalltalk cantFindGlobal: name Handles exception, outputs message*.
377. **class**::= *theSmalltalk class: aClass doesNotRespond: aMessage Handles exception, outputs message*.
378. **echo**::= *theSmalltalk echo ::= toggle whether interpretter echoes its input or not`*.
379. **error**::= *theSmalltalk error: aString*.
380. **flushMessageCache**::= *theSmalltalk flushMessageCache ?*.
381. **getPrompt**::= *theSmalltalk getPrompt: aString*.
382. **inquire**::= *theSmalltalk inquire: aString* .
383. **perform**::= *theSmalltalk perform: message withArguments: args*.
384. **perform**::= *theSmalltalk perform: message withArguments: args ifError: aBlock* .
385. **saveImage**::= *theSmalltalk saveImage*.
386. **saveImage**::= *theSmalltalk saveImage: name* .
387. **watch**::= *theSmalltalk watch*.

String methods

*,::= theString , value ::=returns string made up of TheString and value*s printString'.
388. **<**::= *theString < value ::=returns Boolean true/false depending on comparison*.
389. **=**::= *theString = value ::=returns Boolean true/false depending on comparison*.
390. **asByteArray**::= *theString asByteArray* .
391. **asInteger**::= *theString asInteger ::=returns result of interpretting theString as an integer*..
392. **asSymbol**::= *theString asSymbol ::=returns a Symbol with string as its representation*.
393. **basicAt**::= *theString basicAt: index ::=returns character at position index*.
394. **basicAt**::= *theString basicAt: index put: aValue change character at position index in theString*.
395. **copy**::= *theString copy ::=returns a copy of the string*.
396. **copyFrom**::= *theString copyFrom: low to: high ::=returns a substring low..high*.
397. **edit**::= *theString edit* .
398. **execute**::= *theString execute* .
399. **hash**::= *theString hash*.
400. **input**::= *theString input ::= (local) print theString as a prompt and input from terminal*.
401. **load**::= *theString load ::= (local) read in a file of definitions with name theString*.
402. **print**::= *theString print*.
403. **printNoReturn**::= *theString printNoReturn (local) ::=send theString to stdout with no end-of-line*.
404. **printString**::= *theString printString ::=returns printable image*.
405. **size**::= *theString size ::=returns an integer equal to the number of characters in theString*.
406. **unixCommand**::= *theString unixCommand*.
407. **value**::= *theString value*.
408. **words**::= *theString words: aBlock ?*.
409. **else**::= *theSwitch else: block case statements default case*.
410. **ifMatch**::= *theSwitch ifMatch: key do: block part of a case statement*.
411. **key**::= *theSwitch key: value Start of cases*.

Symbol methods

412. **asString**::= *theSymbol asString* ::=*returns a string representing theSymbol*.
413. **assign**::= *theSymbol assign: value*.
414. **copy**::= *theSymbol copy*.
415. **printString**::= *theSymbol printString* ::=*returns printable image*.
416. **respondsTo**::= *theSymbol respondsTo* ::=*returns a collection of classes that respond to theSymbol*.
417. **value**::= *theSymbol value*.
418. **apply**::= *theSymbol apply: args*.
419. **apply**::= *theSymbol apply: args ifError: aBlock*.

True methods

420. **ifTrue**::= *theTrue ifTrue: trueBlock ifFalse: falseBlock evaluates the falseBlock*.
421. **not**::= *theTrue not* ::=*returns a False*.
422. **printString**::= *theTrue printString*.
423. **xor**::= *theTrue xor: aBoolean*.

UndefinedObject methods

424. **createGlobals**::= *theUndefinedObject createGlobals* .
425. **initialize**::= *theUndefinedObject initialize*.
426. **isNil**::= *theUndefinedObject isNil* ::=*returns True*.
427. **notNil**::= *theUndefinedObject notNil* ::=*returns True if not a nil*.
428. **printString**::= *theUndefinedObject printString*.

. . . . . (end of section **Smalltalk Methods Initially defined in Our Smalltalk.**) **<<**Contents | End**>>**

End