COP 3330 Java Docs Reference

Input/Output Classes

Scanner

Constructor	Description
`Scanner(File source)`	Constructs a new Scanner from source.
`Scanner(InputStream source)`	Constructs a new Scanner from source.

Modifier and Type	Method	Description
`void`	`close()`	Closes this Scanner.
`boolean`	`hasNext()`	Returns true if this Scanner has more tokens.
`boolean`	`hasNextDouble()`	Returns true if this Scanner has a next token which is a double. (Similar methods exist for all primitive types.)
`String`	`next()`	Returns the next String from this Scanner
`double`	`nextDouble()`	Returns the next double from this Scanner
`int`	`nextInt()`	Returns the next int from this Scanner
`String`	`nextLine()`	Returns the next line from this Scanner

File

Constructor	Description
`File(String pathname)`	Creates a new file object (for reading) from the file with the path and name given in pathname.

FileWriter

Constructor	Description
`FileWriter(String fileName)`	Creates a new file object (for rwriting) from the file with the path and name given in pathname.

BufferedReader

Constructor	Description
`BufferedReader(Reader in)`	Creates a buffering character input stream that uses a default input buffer.

Modifier and Type	Method	Description
`void`	`close()`	`Closes the stream.`
`String`	`readLine()`	`Reads a line of text and returns it.`
`boolean`	`ready()`	`Tells whether the stream is ready to be read or not.`

PrintWriter

Constructor	Description
`PrintWriter(File file)`	Creates a PrintWriter to file.

Modifier and Type	Method	Description
`PrintWriter`	`append(char c)`	Appends c to this writer.
`void`	`close()`	Closes the stream.
`void`	`flush()`	Flushes the stream.
`void`	`print(String s)`	Prints a string. Similar methods exist for all primitive types.
`void`	`println()`	Terminates current line by writing '\n' to the stream.
`void`	`println(String s)`	Prints string and terminates the line.

String Related Classes

String

Constructor	Description
`String()`	Initializes a newly created String object so that it represents an empty character sequence.
`String(char[] value)`	Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.
`String(String original)`	Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.
`String(StringBuffer buffer)`	Allocates a new string that contains the sequence of characters currently contained in the string buffer argument.

Modifier and Type	Method	Description
`char`	`charAt()`	Returns the char value at the specified index.
`int`	`compareTo(String anotherString)`	Compares two strings lexicographically.
`int`	`compareToIgnoreCase(String str)`	Compares two strings lexicographically, ignoring case differences.
`String`	`concat(String str)`	Concatenates the specified string to the end of this string, returning a reference to the newly created String object.
`boolean`	`contains(CharSequence s)`	Returns true if and only if this string contains the specified sequence of char values.
`boolean`	`endsWith(String suffix)`	Tests if this string ends with the specified suffix.
`boolean`	`equals(Object anObject)`	Compares this string to the specified object.
`boolean`	`equalsIgnoreCase(String anotherString)`	Compares this String to anotherString, ignoring case considerations.
`int`	`indexOf(int ch)`	Returns the index within this string of the first occurrence of the specified character.

int	indexOf(int ch, int fromIndex)	Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.
int	length()	Returns the length of this string.
String	replace(char oldChar, char newChar)	Returns a string resulting from replacing all occurrences of oldChar in this string with newChar.
String	substring(int beginIndex)	Returns a string that is a substring of this string, from beginIndex to the end of the string.
String	substring(int beginIndex, int endIndex)	Returns a string that is a substring of this string, starting at beginIndex and ending at endIndex-1, of this String object.
char[]	toCharArray()	Converts this string to a new character array.
String	toLowerCase()	Converts all of the characters in this String to lower case using the rules of the default locale.
String	toUpperCase()	Converts all of the characters in this String to upper case using the rules of the default locale.

StringTokenizer

Constructor	Description
StringTokenizer(String str)	Constructs a StringTokenizer for str using whitespace delimiters.
StringTokenizer(String str, String delim)	Constructs a StringTokenizer for str using only the characters in delim as delimiters.

Modifier and Type	Method	Description
int	countTokens()	Retruns the number of times nextToken() can be called without generating an Exception.
boolean	hasMoreTokens()	Returns true if the next call to nextToken() will not generate an Exception.
String	nextToken()	Returns the next token from this StringTokenizer.

Wrapper Classes

Integer

Constructor	Description
Integer(int value)	Constructs an Integer object equal to value.
Integer(String s)	Constructs an Integer object equal to the numeric value of s, assuming that s forms a valid integer.

Modifier and Type	Method	Description
static int	compare(int x, int y)	Retruns a negative integer if x<y, positive integer if x>y and 0 if they are equal.
boolean	equals(Object obj)	Returns true iff this and obj are equal Integers in value.
static int	parseInt(String s)	Returns the integer represented by the String s.

Double

Constructor	Description
`Double(double value)`	Constructs a Double object equal to value.
`Double(String s)`	Constructs a Double object equal to the numeric value of s.

Modifier and Type	Method	Description
`static int`	`compare(int x, int y)`	Retruns a negative integer if x<y, positive integer if x>y and 0 if they are equal.
`boolean`	`equals(Object obj)`	Returns true iff this and obj are equal Doubles in value.
`static double`	`parseDouble(String s)`	Returns the double represented by the String s.

Common Classes

Math

Modifier and Type	Field	Description
`static double`	`E`	The base of the natural logarithm
`static double`	`PI`	Ratio of the circumference to the diameter of a circle

Modifier and Type	Method	Description
`static double`	`abs(double a)`	Returns the absolute value of a, similar methods exist for all numeric types.
`static double`	`acos(double a)`	Returns the arc cosine of a radians.
`static double`	`asin(double a)`	Returns the arc sine of a radians.
`static double`	`atan(double a)`	Returns the arc tangent of a radians.
`static double`	`cbrt(double a)`	Returns the cube root of a.
`static double`	`ceil(double a)`	Returns the smallest integer >= a.
`static double`	`cos(double a)`	Returns the cosine of a radians.
`static double`	`exp(double a)`	Returns e raised to the power a.
`static double`	`floor(double a)`	Returns the greatest integer <= a.
`static double`	`hypot(double x, double y)`	Returns the hypotenuse of a right triangle with legs x and y.
`static double`	`log(double a)`	Returns the natural logarithm of a.
`static double`	`log10(double a)`	Returns the base 10 logarithm of a.
`static double`	`max(double a, double b)`	Returns the larger of a and b. Similar methods exist for all numeric types.
`static double`	`min(double a, double b)`	Returns the smaller of a and b. Similar methods exist for all numeric types.
`static double`	`pow(double a, double b)`	Returns a raised to the power b.
`static double`	`random()`	Returns a random real in the range [0.0, 1.0)
`static int`	`round(double a)`	Returns the nearest integer to a.
`static double`	`sin(double a)`	Returns the sine of a radians.
`static double`	`sqrt(double a)`	Returns the square root of a.
`static double`	`tan(double a)`	Returns the tangent of a radians.
`static double`	`toDegrees(double angRad)`	Returns the equivalent angle measure in degrees of angRad radians.
`static double`	`toRadians(double angDeg)`	Returns the equivalent angle measure in radians of angDeg degrees.

Note: All math methods return NaN if the operation isn't well-defined.

Random

Constructor	Description
`Random()`	Creates a new random number generator.

Modifier and Type	Method	Description
`double`	`nextDouble()`	Returns the next pseudorandom double in the range [0.0, 1.0)
`int`	`nextInt()`	Returns the next pseudorandom int. Each possible int value is equally likely.
`int`	`nextInt(int bound)`	Returns the next pseudorandom int in between 0 and bound-1, inclusive.

Built In Data Structures

ArrayList

Constructor	Description
`ArrayList<E>()`	Creates an empty ArrayList.

Modifier and Type	Method	Description
`boolean`	`add(E e)`	Adds e to the end of this ArrayList
`void`	`add(int index, E e)`	Inserts e at index index in this ArrayList
`void`	`clear()`	Empties the contents of this ArrayList.
`E`	`get(int index)`	Returns the item at index index of this ArrayList.
`int`	`indexOf(Object o)`	Returns the first index where o is located in this ArrayList or -1 if not.
`E`	`remove(int index)`	Removes and returns the item at index index in this ArrayList.
`boolean`	`remove(Object o)`	Returns the first occurrence of o in this ArrayList. Returns false if o isn't in the list.
`int`	`size()`	Returns the number of elements in this ArrayList.
`Object[]`	`toArray()`	Returns an array containing all elements of this ArrayList in the same order.

Stack

Constructor	Description
`Stack<E>()`	Creates an empty Stack.

Modifier and Type	Method	Description
`boolean`	`empty()`	Returns true iff this Stack is empty.
`E`	`peek()`	Returns the top of this Stack (without removing it.)
`E`	`pop()`	Returns and removes the top of this Stack.
`E`	`push(E item)`	Pushes item onto the top of this Stack.
`int`	`size()`	Returns the number of elements in this Stack.

Queue

Constructor	Description
`Queue<E>()`	Creates an empty Queue.

Modifier and Type	Method	Description
`boolean`	`add(E e)`	Inserts e to the back of this Queue.
`E`	`peek()`	Returns the front of this Queue (without removing it.)
`E`	`poll()`	Returns and removes the top of this Stack.
`int`	`size()`	Returns the number of elements in this Queue.

PriorityQueue

Constructor	Description
`PriorityQueue()`	Creates an empty PriorityQueue.

Modifier and Type	Method	Description
`boolean`	`add(E e)`	Adds e to this PriorityQueue.
`void`	`clear()`	Clears the contents of this PriorityQueue.
`E`	`peek()`	Returns the minimum element currently in this PriorityQueue (but doesn't remove it)
`E`	`poll()`	Removes and returns the minimum element currently in this PriorityQueue.
`int`	`size()`	Returns the number of elements currently in this PriorityQueue.

ArrayDeque

Constructor	Description
`ArrayDeque<E>()`	Creates an empty ArrayDeque.

Modifier and Type	Method	Description
`boolean`	`addFirst(E e)`	Adds e to the front of this ArrayDeque.
`void`	`addLast(E e)`	Adds e to the backof this ArrayDeque.
`void`	`clear()`	Empties the contents of this ArrayDeque.
`E`	`getFirst()`	Returns the first item of this ArrayDeque.
`E`	`getLast()`	Returns the last item of this ArrayDeque.
`E`	`pollFirst()`	Returns and removes the first item from this ArrayDeque.
`E`	`pollFirst()`	Returns and removes the last item from this ArrayDeque.
`int`	`size()`	Returns the number of elements in this ArrayDeque.
`Object[]`	`toArray()`	Returns an array containing all elements of this ArrayList in the same order.

HashSet

Constructor	Description
`HashSet<E>()`	Creates an empty HashSet.

Modifier and Type	Method	Description
`boolean`	`add(E e)`	Adds e to this HashSet.
`void`	`clear()`	Empties the contents of this HashSet.
`boolean`	`contains(E e)`	Returns true iff e is in this HashSet.
`boolean`	`remove(E e)`	Removes e from this HashSet, returns false only if e wasn't in the set.
`int`	`size()`	Returns the number of elements in this HashSet.

TreeSet

Constructor	Description
`TreeSet()`	Creates an empty TreeSet.

Modifier and Type	Method	Description
`boolean`	`add(E e)`	Adds e to this TreeSet.
`E`	`ceiling(E e)`	Returns the least element in this TreeSet >= e or null if there's none.
`void`	`clear()`	Empties the contents of this TreeSet.
`boolean`	`contains(E e)`	Returns true iff e is in this TreeSet.
`E`	`first()`	Returns the first element in this TreeSet or null if there's none.
`E`	`floor(E e)`	Returns the greatest element in this TreeSet <= e or null if there's none.
`E`	`higher(E e)`	Returns the least element in this TreeSet > e or null if there's none.
`E`	`last()`	Returns the last element in this TreeSet or null if there's none.
`E`	`lower(E e)`	Returns the greatest element in this TreeSet < e or null if there's none.
`E`	`pollFirst()`	`Returns and removes the first element in this TreeSet, or null if there's none.`
`E`	`pollLast()`	`Returns and removes the last element in this TreeSet, or null if there's none.`
`boolean`	`remove(E e)`	Removes e from this HashSet, returns false only if e wasn't in the set.
`int`	`size()`	Returns the number of elements in this HashSet.

HashMap

Constructor	Description
`HashMap<K,V>()`	Creates an empty HashMap.

Modifier and Type	Method	Description
`void`	`clear()`	Empties the contents of this HashMap.
`boolean`	`containsKey(E e)`	Returns true iff e is a key in this HashMap.
`boolean`	`containsValue(E e)`	Returns true iff e is a value in this HashMap.
`V`	`get(Object key)`	Returns the mapped value for key in this HashMap.
`Set<K>`	`keySet()`	Returns the set of keys in this HashMap.
`V`	`put(K key, V value)`	Maps key to value in this HashMap.
`V`	`remove(K key)`	Removes the mapping associated with key and returns the corresponding value.
`int`	`size()`	Returns the number of elements in this HashMap.

TreeMap

Constructor	Description
`TreeMap()`	Creates an empty TreeMap.

Modifier and Type	Method	Description
`E`	`ceilingKey(E e)`	Returns the least key in this TreeMap >= e or null if there's none.
`void`	`clear()`	Empties the contents of this TreeMap.
`boolean`	`containsKey(E e)`	Returns true iff e is a key in this TreeMap.
`boolean`	`containsValue(E e)`	Returns true iff e is a value in this TreeMap.
`E`	`firstKey()`	Returns the first key in this TreeMap or null if there's none.
`E`	`floorKey()`	Returns the greatest key in this TreeMap <= e or null if there's none.
`V`	`get(Object key)`	Returns the mapped value for this key in this TreeMap.
`E`	`higherKey(E e)`	Returns the least key in this TreeMap > e or null if there's none.
`Set<K>`	`keySet()`	Returns the set of keys in this TreeMap.
`E`	`lastKey()`	Returns the last key in this TreeMap or null if there's none.
`E`	`lowerKey(E e)`	Returns the greatest key in this TreeMap < e or null if there's none.
`V`	`put(K key, V value)`	Maps key to value in this TreeMap.
`V`	`remove(K key)`	Removes the mapping associated with key and returns the corresponding value.
`int`	`size()`	Returns the number of elements in this TreeMap.