Java API

A compilation of selected constructors and methods in some common Java classes.

Miscellaneous

`class java.lang.Object`:

`protected Object` **`clone`**`()throws CloneNotSupportedException`
Creates and returns a copy of this object.

`boolean` **`equals`**`(Object obj)`
Indicates whether some other object is "equal to" this one.

`Class` **`getClass`**`()`
Returns the runtime class of this `Object`.

`int` **`hashCode`**`()`
Returns a hash code value for the object.

`void` **`notify`**`()`
Wakes up a single thread that is waiting on this object's monitor.

`void` **`notifyAll`**`()`
Wakes up all threads that are waiting on this object's monitor.

`String` **`toString`**`()`
Returns a string representation of the object.

`void` **`wait`**`()throws InterruptedException`
Causes the current thread to wait until another thread invokes the `notify()` method or the `notifyAll()` method for this object.

`void` **`wait`**`(long timeout) throws InterruptedException`
Causes the current thread to wait until either another thread invokes the `notify()` method or the `notifyAll()` method for this object, or a specified amount of time has elapsed.

`interface java.lang.Comparable<T>`

`int` **`compareTo`**`(T obj)`
Compares this object with the specified object for order.

`interface java.util.Comparator<T>`

`int` **`compare`**`(T obj1, T obj2)`
Compares its two arguments for order.

`class java.lang.Integer`: *analogous for long, float, double, char, boolean, etc.*

`static int` **`parseInt`**`(String s) throws NumberFormatException`
Parses the string argument as a signed decimal integer.

`class java.lang.Math`:

`int` **`max`**`(int a,int b)`
`long` **`max`**`(long a,long b)`
`float` **`max`**`(float a,float b)`
`double` **`max`**`(double a,double b)`
Returns the greater of `a` and `b`.

`int` **`min`**`(int a,int b)`
`long` **`min`**`(long a,long b)`
`float` **`min`**`(float a,float b)`
`double` **`min`**`(double a,double b)`
Returns the smaller of `a` and `b`.

`class java.util.Random`:

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

`int` **`nextInt`**`(int bound)`
Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

`class java.util.Date`:

`Date``()`
Allocates a `Date` object and initializes it so that it represents the time at which it was allocated.

`String` **`toString`**`()`
Converts this `Date` object to a `String` of the form:
`dow mon dd hh:mm:ss zzz yyyy` (where dow = day of week)

`class java.util.Scanner`:

`Scanner``(File source) throws FileNotFoundException`
Constructs a new `Scanner` that produces values scanned from the specified file.

`Scanner``(InputStream source)`
Constructs a new `Scanner` that produces values scanned from the specified input stream.

`Scanner``(String source)`
Constructs a new `Scanner` that produces values scanned from the specified string.

`boolean` **`hasNext`**`()`
Returns true if this scanner has another token in its input.

`boolean` **`hasNext`**`(String pattern)`
Returns true if the next token matches the pattern constructed from the specified string.

`boolean` **`hasNextInt`**`()` *analogous for long, float, double, char, boolean, etc.*
Returns true if the next token in this scanner's input can be interpreted as an int value in the default radix using the `nextInt()` method.

`boolean` **`hasNextLine`**`()`
Returns true if there is another line in the input of this scanner.

`String` **`next`**`()`
Finds and returns the next complete token from this scanner.

`String` **`nextLine`**`()`
Advances this scanner past the current line and returns the input that was skipped.

Strings

`class java.lang.String`:

`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``(byte[] bytes)`
Constructs a new `String` by decoding the specified array of bytes using the platform's default charset.

`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.

`char` **`charAt`**`(int index)`
Returns the character 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.

`boolean` **`contains`**`(String str)`
Returns true if and only if this string contains the specified string.

`boolean` **`contentEquals`**`(StringBuffer sb)`
Returns true if and only if this String represents the same sequence of characters as the specified StringBuffer.

`static String` **`copyValueOf`**`(char[] data)`
Returns a String that represents the character sequence in the array specified.

`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 another `String`, ignoring case considerations.

`byte[]` **`getBytes`**`()`
Encodes this `String` into a sequence of bytes, storing the result into a new byte array.

`int` **`hashCode`**`()`
Returns a hash code for this string.

`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` **`indexOf`**`(String str)`
Returns the index within this string of the first occurrence of the specified substring.

`int` **`indexOf`**`(String str, int fromIndex)`
Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

`int` **`lastIndexOf`**`(int ch)`
Returns the index within this string of the last occurrence of the specified character.

`int` **`lastIndexOf`**`(int ch, int fromIndex)`
Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.

`int` **`lastIndexOf`**`(String str)`
Returns the index within this string of the rightmost occurrence of the specified substring.

`int` **`lastIndexOf`**`(String str, int fromIndex)`
Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.

`int` **`length`**`()`
Returns the length of this string.

`boolean` **`matches`**`(String regex)`
Tells whether or not this string matches the given regular expression.

`String` **`replace`**`(char oldChar, char newChar)`
Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.

`String` **`replaceAll`**`(String regex, String replacement)`
Replaces each substring of this string that matches the given regular expression with the given replacement.

`String` **`replaceFirst`**`(String regex, String replacement)`
Replaces the first substring of this string that matches the given regular expression with the given replacement.

`String[]` **`split`**`(String regex)`
Splits this string around matches of the given regular expression.

`boolean` **`startsWith`**`(String prefix)`
Tests if this string starts with the specified prefix.

`String` **`substring`**`(int beginIndex, int endIndex)`
Returns a new string that is a substring of this string.

`char[]` **`toCharArray`**`()`
Converts this string to a new character array.

`String` **`toLowerCase`**`()`
Converts all of the characters in this `String` to lower case.

`String` **`toUpperCase`**`()`
Converts all of the characters in this `String` to upper case.

`String` **`trim`**`()`
Returns a copy of the string, with leading and trailing whitespace omitted.

`class java.lang.StringBuilder`:

`StringBuilder``()`
Constructs a string builder with no characters in it.

`StringBuilder``(String str)`
Constructs a string builder initialized to the contents of the specified string.

`StringBuilder` **`append`**`(char c)`
Appends the string representation of the char argument to this sequence.

`StringBuilder` **`append`**`(String s)`
Appends the specified string argument to this sequence.

`int` **`length`**`()`
Returns the length (character count).

`char` **`charAt`**`(int index)`
Returns the `char` value in this sequence at the specified index.

`StringBuilder` **`insert`**`(int offset,char c)`
Inserts the string representation of the `char` argument into this sequence.

`StringBuilder` **`insert`**`(int offset,String str)`
Inserts the string into this character sequence.

`StringBuilder` **`delete`**`(int start,int end)`
Removes the characters in a substring of this sequence.

`StringBuilder` **`deleteCharAt`**`(int index)`
Removes the `char` at the specified position in this sequence.

`String` **`toString`**`();`
Returns a string representing the data in this sequence.

Data Structures

`interface java.util.Iterator<E>`:

`boolean` **`hasNext`**`()`
Returns true if the iteration has more elements. (In other words, returns true if next would return an element rather than throwing an exception.)

`E` **`next`**`()`
Returns the next element in the iteration.
Throws: `NoSuchElementException` - iteration has no more elements.

`void` **`remove`**`()`
Removes from the underlying collection the last element returned by the iterator. This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

`interface java.lang.Iterable<E>`:

`Iterator<E>` **`iterator`**`()`
Returns an iterator over a set of elements of type T.

`interface java.util.Collection<E> extends java.util.Iterable<E>`:

`boolean` **`add`**`(E o)`
Appends the specified element to the end of this collection.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Adds all of the elements in the specified collection to this collection.

`void` **`clear`**`()`
Removes all of the elements from this collection.

`boolean` **`contains`**`(E elem)`
Returns true if this collection contains the specified element.

`boolean` **`equals`**`(Object o)`
Returns `true` if and only if the specified object is also a collection, both collections have the same size, and all corresponding pairs of elements in the two collections are *equal*.

`int` **`hashCode`**`()`
Returns the hash code value for this collection.

`boolean` **`isEmpty`**`()`
Tests if this collection has no elements.

`Iterator` **`iterator`**`()`
Returns an iterator over the elements in this collection in proper sequence.

`E` **`remove`**`(Object o)`
Removes the first occurrence of the specified element from this collection, if it is present.

`int` **`size`**`()`
Returns the number of elements in this collection.

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this collection.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array.

`interface java.util.List<E> extends java.util.Collection<E>``:`

`boolean` **`add`**`(E o)`
Appends the specified element to the end of this list.

`void` **`add`**`(int index, E element)`
Inserts the specified element at the specified position in this list.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.

`void` **`clear`**`()`
Removes all of the elements from this list.

`boolean` **`contains`**`(E elem)`
Returns true if this list contains the specified element.

`boolean` **`equals`**`(Object o)`
Returns `true` if and only if the specified object is also a list, both lists have the same size, and all corresponding pairs of elements in the two lists are *equal*.

`E` **`get`**`(int index)`
Returns the element at the specified position in this list.

`int` **`hashCode`**`()`
Returns the hash code value for this list.

`int` **`indexOf`**`(E elem)`
Searches for the first occurence of the given argument, testing for equality using the equals method.

`boolean` **`isEmpty`**`()`
Tests if this list has no elements.

`Iterator` **`iterator`**`()`
Returns an iterator over the elements in this list in proper sequence.

`E` **`remove`**`(int index)`
Removes the element at the specified position in this list.

`E` **`remove`**`(Object o)`
Removes the first occurrence of the specified element from this list, if it is present.

`E` **`set`**`(int index, E element)`
Replaces the element at the specified position in this list with the specified element.

`int` **`size`**`()`
Returns the number of elements in this list.

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this list in the correct order.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

`class java.util.ArrayList<E> implements java.util.List<E>``:`

`ArrayList``()`
Constructs an empty array list.

`boolean` **`add`**`(E o)`
Appends the specified element to the end of this list.

`Void` **`add`**`(int index, E element)`
Inserts the specified element at the specified position in this list.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Adds all of the elements in the specified collection to this list.

`void` **`clear`**`()`
Removes all of the elements from this list.

`Object` **`clone`**`()`
Returns a shallow copy of this list.

`boolean` **`contains`**`(E elem)`
Returns true if this list contains the specified element.

`boolean` **`equals`**`(Object o)`
Returns `true` if and only if the specified object is also a list, both lists have the same size, and all corresponding pairs of elements in the two lists are *equal*.

`E` **`get`**`(int index)`
Returns the element at the specified position in this list.

`int` **`hashCode`**`()`
Returns the hash code value for this list.

`int` **`indexOf`**`(E elem)`
Searches for the first occurence of the given argument, testing for equality using the equals method.

`boolean` **`isEmpty`**`()`
Tests if this list has no elements.

`Iterator` **`iterator`**`()`
Returns an iterator over the elements in this list in proper sequence.

`E` **`remove`**`(int index)`
Removes the element at the specified position in this list.

`E` **`remove`**`(Object o)`
Removes the first occurrence of the specified element from this list, if it is present.

`E` **`set`**`(int index, E element)`
Replaces the element at the specified position in this list with the specified element.

`int` **`size`**`()`
Returns the number of elements in this list.

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this list in the correct order.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

`class java.util.LinkedList<E> implements java.util.List<E>``:`

`LinkedList``()`
Constructs an empty list.

`boolean` **`add`**`(E o)`
Appends the specified element to the end of this list.

`Void` **`add`**`(int index, E element)`
Inserts the specified element at the specified position in this list.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Adds all of the elements in the specified collection to this list.

`boolean` **`addFirst`**`(E o)`
Inserts the specified element at the beginning of this list..

`boolean` **`addLast`**`(E o)`
Appends the specified element to the end of this list.

`void` **`clear`**`()`
Removes all of the elements from this list.

`Object` **`clone`**`()`
Returns a shallow copy of this list.

`boolean` **`contains`**`(E elem)`
Returns true if this list contains the specified element.

`boolean` **`equals`**`(Object o)`
Returns `true` if and only if the specified object is also a list, both lists have the same size, and all corresponding pairs of elements in the two lists are *equal*.

`E` **`get`**`(int index)`
Returns the element at the specified position in this list.

`E` **`getFirst`**`()`
Returns the first element in this list.

`E` **`getLast`**`()`
Returns the last element in this list.

`int` **`hashCode`**`()`
Returns the hash code value for this list.

`int` **`indexOf`**`(E elem)`
Searches for the first occurence of the given argument, testing for equality using the equals method.

`boolean` **`isEmpty`**`()`
Tests if this list has no elements.

`Iterator` **`iterator`**`()`
Returns an iterator over the elements in this list in proper sequence.

`E` **`remove`**`(int index)`
Removes the element at the specified position in this list.

`E` **`remove`**`(Object o)`
Removes the first occurrence of the specified element from this list, if it is present.

`E` **`removeFirst`**`()`
Removes and returns the first element from this list.

`E` **`removeLast`**`()`
Removes and returns the last element from this list.

`E` **`set`**`(int index, E element)`
Replaces the element at the specified position in this list with the specified element.

`int` **`size`**`()`
Returns the number of elements in this list.

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this list in the correct order.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.

`interface java.util.Set<E> extends java.util.Collection<E>``:`

`boolean` **`add`**`(E o)`
Adds the specified element to this set if it is not already present.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Adds all of the elements in the specified collection to this set if they're not already present.

`void` **`clear`**`()`
Removes all of the elements from this set.

`boolean` **`contains`**`(E elem)`
Returns true if this set contains the specified element.

`boolean` **`equals`**`(Object o)`
Returns true if the specified object is also a set, the two sets have the same size, and every member of the specified set is contained in this set (or equivalently, every member of this set is contained in the specified set).

`int` **`hashCode`**`()`
Returns the hash code value for this set.

`boolean` **`isEmpty`**`()`
Returns true if this set contains no elements.

`Iterator` **`iterator`**`()`
Returns an iterator over the elements in this set.

`E` **`remove`**`(Object o)`
Removes the specified element from this set, if it is present.

`boolean` **`removeAll`**`(Collection<?> c)`
Removes from this set all of its elements that are contained in the specified collection.

`boolean` **`retainAll`**`(Collection<?> c)`
Retains only the elements in this set that are contained in the specified collection.

`int` **`size`**`()`
Returns the number of elements in this set (its cardinality).

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this set.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this set;
the runtime type of the returned array is that of the specified array.

`interface java.util.SortedSet<E> extends java.util.Set<E>:`

`boolean` **`add`**`(E o)`
Adds the specified element to this set if it is not already present.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Adds all of the elements in the specified collection to this set if they're not already present.

`void` **`clear`**`()`
Removes all of the elements from this set.

`boolean` **`contains`**`(E elem)`
Returns true if this set contains the specified element.

`Comparator<? super E>` **`comparator`**`()`
Returns the comparator used to order the elements in this set, or null if this set uses the natural ordering of its elements.

`boolean` **`equals`**`(Object o)`
Returns true if the specified object is also a set, the two sets have the same size, and every member of the specified set is contained in this set (or equivalently, every member of this set is contained in the specified set).

`E` **`first`**`()`
Returns the first (lowest) element currently in this set.

`int` **`hashCode`**`()`
Returns the hash code value for this set.

`SortedSet<E>` **`headSet`**`(E toElement)`
Returns a view of the portion of this set whose elements are strictly less than toElement.

`boolean` **`isEmpty`**`()`
Returns true if this set contains no elements.

`Iterator` **`iterator`**`()`
Returns an iterator over the elements in this set.

`E` **`last`**`()`
Returns the last (highest) element currently in this set.

`E` **`remove`**`(Object o)`
Removes the specified element from this set, if it is present. `boolean` **`removeAll`**`(Collection<?> c)`
Removes from this set all of its elements that are contained in the specified collection.

`boolean` **`retainAll`**`(Collection<?> c)`
Retains only the elements in this set that are contained in the specified collection.

`int` **`size`**`()`
Returns the number of elements in this set (its cardinality).

`SortedSet<E>` **`subSet`**`(E fromElement, E toElement)`
Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive.

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this set.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this set;
the runtime type of the returned array is that of the specified array.

`SortedSet<E>` **`tailSet`**`(E fromElement)`
Returns a view of the portion of this set whose elements are greater than or equal to fromElement. a new, empty TreeSet sorted according to the ordering defined by the compareTo method in class E.

`class java.util.HashSet<E> implements java.util.Set<E>:`

`HashSet``()`
Constructs a new, empty HashSet.

`boolean` **`add`**`(E o)`
Adds the specified element to this set if it is not already present.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Adds all of the elements in the specified collection to this set if they're not already present.

`void` **`clear`**`()`
Removes all of the elements from this set.

`Object` **`clone`**`()`
Returns a shallow copy of this set.

`boolean` **`contains`**`(E o)`
Returns true if this set contains the specified element.

`boolean` **`equals`**`(Object o)`
Returns true if the specified object is also a set, the two sets have the same size, and every member of the specified set is contained in this set (or equivalently, every member of this set is contained in the specified set).

`int` **`hashCode`**`()`
Returns the hash code value for this set.

`boolean` **`isEmpty`**`()`
Returns true if this set contains no elements.

`Iterator<E>` **`iterator`**`()`
Returns an iterator over the elements in this set.

`boolean` **`remove`**`(E o)`
Removes the specified element from this set if it is present.

`boolean` **`removeAll`**`(Collection<?> c)`
Removes from this set all of its elements that are contained in the specified collection.

`boolean` **`retainAll`**`(Collection<?> c)`
Retains only the elements in this set that are contained in the specified collection.

`int` **`size`**`()`
Returns the number of elements in this set (its cardinality).

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this set.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this set;
the runtime type of the returned array is that of the specified array.

`class java.util.TreeSet<E> implements java.util.SortedSet<E>`:

`TreeSet``()`
Constructs a new, empty TreeSet sorted according to the natural ordering of its elements.

`TreeSet``(Comparator<? super E> comparator)`
Constructs a new, empty tree set, sorted according to the specified comparator.

`boolean` **`add`**`(E o)`
Adds the specified element to this set if it is not already present.

`boolean` **`addAll`**`(Collection<? extends E> c)`
Adds all of the elements in the specified collection to this set if they're not already present.

`void` **`clear`**`()`
Removes all of the elements from this set.

`Object` **`clone`**`()`
Returns a shallow copy of this set.

`boolean` **`contains`**`(E elem)`
Returns true if this set contains the specified element.

`Comparator<? super E>` **`comparator`**`()`
Returns the comparator used to order the elements in this set, or null if this set uses the natural ordering of its elements.

`boolean` **`equals`**`(Object o)`
Returns true if the specified object is also a set, the two sets have the same size, and every member of the specified set is contained in this set (or equivalently, every member of this set is contained in the specified set).

`E` **`first`**`()`
Returns the first (lowest) element currently in this set.

`int` **`hashCode`**`()`
Returns the hash code value for this set.

`SortedSet<E>` **`headSet`**`(E toElement)`
Returns a view of the portion of this set whose elements are strictly less than toElement.

`boolean` **`isEmpty`**`()`
Returns true if this set contains no elements.

`Iterator` **`iterator`**`()`
Returns an iterator over the elements in this set.

`E` **`last`**`()`
Returns the last (highest) element currently in this set.

`E` **`pollFirst`**`()`
Retrieves and removes the first (lowest) element, or returns null if this set is empty.

`E` **`pollLast`**`()`
Retrieves and removes the last (highest) element, or returns null if this set is empty.

`E` **`remove`**`(Object o)`
Removes the specified element from this set, if it is present. `boolean`

`removeAll``(Collection<?> c)`
Removes from this set all of its elements that are contained in the specified collection.

`boolean` **`retainAll`**`(Collection<?> c)`
Retains only the elements in this set that are contained in the specified collection.

`int` **`size`**`()`
Returns the number of elements in this set (its cardinality).

`SortedSet<E>` **`subSet`**`(E fromElement, E toElement)`
Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive.

`E[]` **`toArray`**`()`
Returns an array containing all of the elements in this set.

`E[]` **`toArray`**`(E[] a)`
Returns an array containing all of the elements in this set;
the runtime type of the returned array is that of the specified array.

`SortedSet<E>` **`tailSet`**`(E fromElement)`
Returns a view of the portion of this set whose elements are greater than or equal to fromElement. a new, empty TreeSet sorted according to the ordering defined by the compareTo method in class E.

`class java.util.HashMap<K,V>`, **`class java.util.TreeMap<K,V>`**:

`HashMap``()`
Constructs an empty HashMap.

`TreeMap``()`
Constructs an empty TreeMap.

Methods (both classes)

`void` **`clear`**`()`
Removes all mappings from this map.

`boolean` **`containsKey`**`(Object key)`
Returns true if this map contains a mapping for the specified key.

`boolean` **`containsValue`**`(Object value)`
Returns true if this map maps one or more keys to the specified value.

`Set<Map.Entry<K,V>>` **`entrySet`**`()`
Returns a collection view of the mappings contained in this map.

`V` **`get`**`(Object key)`
Returns the value to which the specified key is mapped in this hash map,
or null if the map contains no mapping for this key.

`boolean` **`isEmpty`**`()`
Returns true if this map contains no key-value mappings.

`Set<K>` **`keySet`**`()`
Returns a set view of the keys contained in this map.

`V` **`put`**`(K key, V value)`
Associates the specified value with the specified key in this map.

`V` **`remove`**`(Object key)`
Removes the mapping for this key from this map if present.

`int` **`size`**`()`
Returns the number of key-value mappings in this map.

`String` **`toString`**`()`

Returns a string representation of this map. The string representation consists of a list of key-value mappings enclosed in braces (`"{}"`). Adjacent mappings are separated by the

characters `", "` (comma and space). Each key-value mapping is rendered as the key followed by an equals sign (`"="`) followed by the associated value.

`Collection<V>` **`values`**`()`
Returns a collection view of the values contained in this map.

`interface java.util.Map.Entry<K,V>`:

`K` **`getKey`**`()`
Returns the key corresponding to this entry.

`V` **`getValue`**`()`
Returns the value corresponding to this entry.

Streams and Files

General

`void` **`close`**`()`
Closes the stream.

`class java.io.FileReader`:

`FileReader``(String fileName) throws FileNotFoundException`
Creates a new `FileReader`, given the name of the file to read from.

`int` **`read`**`() throws IOException`
Reads a single character.

`int` **`read`**`(char[] cbuf, int off, int len) throws IOException`
Reads characters into a portion of an array

`long` **`skip`**`(long n) throws IOException`
Skips n characters.

`class java.io.BufferedReader`:

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

`int` **`read`**`() throws IOException`
Reads a single character.

`int` **`read`**`(char[] cbuf, int off, int len) throws IOException`
Reads characters into a portion of an array.

`String` **`readLine`**`() throws IOException`
Reads a line of text.
`long` **`skip`**`(long n) throws IOException`
Skips characters.

`class java.io.FileWriter:`

`FileWriter``(String fileName) throws IOException`
Constructs a FileWriter object given a file name.

`FileWriter``(String filename,boolean append) throws IOException`
Constructs a FileWriter object given a file name, with a boolean indicating whether or not to append the data written to the end of the file rather than the beginning.

`void` **`write`**`(char[] cbuf, int off, int len) throws IOException`
Writes a portion of an array of characters.

`void` **`write`**`(int c) throws IOException`
Writes a single character.

`void` **`write`**`(String str, int off, int len) throws IOException`
Writes a portion of a string.

`void` **`flush`**`() throws IOException`
Flushes the stream.

`class java.io.BufferedWriter`:

`BufferedWriter``(Writer out)`
Creates a buffered character-output stream that uses a default-sized output buffer.

`void` **`flush`**`() throws IOException`
Flushes the stream.

`void` **`newLine`**`() throws IOException`
Writes a line separator.

`void` **`write`**`(char[] cbuf, int off, int len) throws IOException`
Writes a portion of an array of characters.

`void` **`write`**`(int c) throws IOException`
Writes a single character.

`void` **`write`**`(String s, int off, int len) throws IOException`
Writes a portion of a String.

`class java.io.PrintWriter:`

`PrintWriter``(Writer out)`
Creates a new PrintWriter.

`void` **`print`**`(String s)`
Prints a string.

`void` **`println`**`(String s)`
Prints a String and then terminates the line.

`class java.io.FileInputStream`:

`FileInputStream``(String name) throws FileNotFoundException`
Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

`int` **`read`**`() throws IOException`
Reads a byte of data from this input stream.

`int` **`read`**`(byte[] b, int off, int len) throws IOException`
Reads up to `len` bytes of data from this input stream into an array of bytes.

`long` **`skip`**`(long n) throws IOException`
Skips over and discards `n` bytes of data from the input stream.

`class java.io.DataInputStream`:

`DataInputStream``(InputStream in)`
Creates a DataInputStream that uses the specified underlying InputStream.

`int` **`read`**`(byte[] b) throws IOException`
Reads some number of bytes from the contained input stream and stores them into the buffer array `b`.

`int` **`read`**`(byte[] b, int off, int len) throws IOException`
Reads up to `len` bytes of data from the contained input stream into an array of bytes.

`boolean` **`readBoolean`**`() throws IOException`
Reads one input byte and returns `true` if that byte is nonzero, `false` if that byte is zero.

`byte` **`readByte`**`() throws IOException`
Reads and returns one input byte.

`char` **`readChar`**`() throws IOException`
Reads two input bytes and returns a `char` value.

`double` **`readDouble`**`() throws IOException`
Reads eight input bytes and returns a `double` value.

`float` **`readFloat`**`() throws IOException`
Reads four input bytes and returns a `float` value.

`int` **`readInt`**`() throws IOException`
Reads four input bytes and returns an `int` value.

`long` **`readLong`**`() throws IOException`
Reads eight input bytes and returns a `long` value.

`short` **`readShort`**`()throws IOException`
Reads two input bytes and returns a `short` value.

`int` **`skipBytes`**`(int n) throws IOException`
Makes an attempt to skip over `n` bytes of data from the input stream, discarding the skipped bytes.

`class java.io.ObjectInputStream`:

`ObjectInputStream``(InputStream in) throws IOException`
Creates an ObjectInputStream that reads from the specified InputStream.

`Object` **`readObject`**`()throws IOException,ClassNotFoundException`
Read an object from the ObjectInputStream.

`class java.io.BufferedInputStream`:

`BufferedInputStream``(InputStream in)`
Creates a BufferedInputStream that reads from the specified InputStream.

`int` **`read`**`()throws IOException`
Reads the next byte of data from the input stream.

`int` **`read`**`(byte[] b, int off, int len) throws IOException`
Reads bytes from this byte-input stream into the specified byte array, starting at the given offset.

`void` **`reset`**`()throws IOException`
Repositions this stream to the position at the time the `mark` method was last called on this input stream.

`long` **`skip`**`(long n) throws IOException`
Skips over and discards `n` bytes of data from this input stream.

`class java.io.InputStreamReader`:

`InputStreamReader``(InputStream in)`
Creates an InputStreamReader that uses the default charset.

`int` **`read`**`()throws IOException`
Reads a single character.

`int` **`read`**`(char[] cbuf, int offset, int length) throws IOException`
Reads characters into a portion of an array.

`class java.io.FileOutputStream`:

`FileOutputStream``(String name) throws FileNotFoundException`
Creates a file output stream to write to the file with the specified name.

`void` **`write`**`(byte[] b, int off, int len) throws IOException`
Writes `len` bytes from the specified byte array starting at offset `off` to this file output stream.

`void` **`write`**`(int b) throws IOException`
Writes the specified byte to this file output stream.

`class java.io.DataOutputStream`:

`DataOutputStream``(OutputStream out)`
Creates a new data output stream to write data to the specified underlying output stream.

`void` **`write`**`(byte[] b, int off, int len) throws IOException`
Writes `len` bytes from the specified byte array starting at offset `off` to the underlying output stream.

`void` **`write`**`(int b) throws IOException`
Writes the specified byte (the low eight bits of the argument `b`) to the underlying output stream.

`void` **`writeBoolean`**`(boolean v) throws IOException`
Writes a `boolean` to the underlying output stream as a 1-byte value.

`void` **`writeByte`**`(int v) throws IOException`
Writes out a `byte` to the underlying output stream as a 1-byte value.

`void` **`writeBytes`**`(String s) throws IOException`
Writes out the string to the underlying output stream as a sequence of bytes.

`void` **`writeChar`**`(int v) throws IOException`
Writes a `char` to the underlying output stream as a 2-byte value, high byte first.

`void` **`writeChars`**`(String s) throws IOException`
Writes a string to the underlying output stream as a sequence of characters.

`void` **`writeDouble`**`(double v) throws IOException`
Converts the double argument to a `long` using the `doubleToLongBits` method in class `Double`, and then writes that `long` value to the underlying output stream as an 8-byte quantity, high byte first.

`void` **`writeFloat`**`(float v) throws IOException`
Converts the float argument to an `int` using the `floatToIntBits` method in class `Float`, and then writes that `int` value to the underlying output stream as a 4-byte quantity, high byte first.

`void` **`writeInt`**`(int v) throws IOException`
Writes an `int` to the underlying output stream as four bytes, high byte first.

`void` **`writeLong`**`(long v) throws IOException`
Writes a `long` to the underlying output stream as eight bytes, high byte first.

`void` **`writeShort`**`(int v) throws IOException`
Writes a `short` to the underlying output stream as two bytes, high byte first.

`class java.io.ObjectOutputStream`:

`ObjectOutputStream``(OutputStream out) throws IOException`
Creates an ObjectOutputStream that writes to the specified OutputStream.

`void` **`writeObject`**`(Object obj) throws IOException, NotSerializableException`
Write the specified object to the ObjectOutputStream.

`class java.io.BufferedOutputStream`:

`BufferedOutputStream``(OutputStream out)`
Creates a new buffered output stream to write data to the specified underlying output stream.

`void` **`flush`**`()throws IOException`
Flushes this buffered output stream.

`void` **`write`**`(byte[] b, int off, int len) throws IOException`
Writes `len` bytes from the specified byte array starting at offset `off` to this buffered output stream.

`void` **`write`**`(int b) throws IOException`
Writes the specified byte to this buffered output stream.

`class java.io.File`:

`File(String pathname)`
Creates a new `File` instance representing the given pathname.

`boolean` **`createNewFile`**`() throws IOException`
Creates a new, empty file named by this pathname if and only if a file with this name does not yet exist. Returns true if the named file does not exist and was successfully created; false if the named file already exists.

`boolean` **`exists`**`()`
Tests whether the file or directory denoted by this abstract pathname exists.

`String` **`getName`**`()`
Returns the name of the file or directory denoted by this abstract pathname.

`boolean` **`isDirectory`**`()`
Tests whether the file denoted by this abstract pathname is a directory.

`boolean` **`isFile`**`()`
Tests whether the file denoted by this abstract pathname is a normal file.

`File[]` **`listFiles`**`()`
Returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname.

Observer

`class java.util.Observable`:

`void` **`addObserver`**`(Observer o)`
Adds an observer to the set of observers for this object.

`protected void` **`setChanged`**`()`
Marks this Observable object as having been changed.

`void` **`notifyObservers`**`(Object arg)`
If this object has changed, then notify all of its observers.

`interface java.util.Observer`:

`void` **`update`**`(Observable o, Object arg)`
This method is called whenever the observed object is changed.