| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Map.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/Locale.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Map.Entry.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/Map.html)    [**NO FRAMES**](http://docs.google.com/Map.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | CONSTR | [METHOD](#2et92p0) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

## **java.util**

Interface Map<K,V>

**Type Parameters:**K - the type of keys maintained by this mapV - the type of mapped values **All Known Subinterfaces:** [Bindings](http://docs.google.com/javax/script/Bindings.html), [ConcurrentMap](http://docs.google.com/java/util/concurrent/ConcurrentMap.html)<K,V>, [ConcurrentNavigableMap](http://docs.google.com/java/util/concurrent/ConcurrentNavigableMap.html)<K,V>, [LogicalMessageContext](http://docs.google.com/javax/xml/ws/handler/LogicalMessageContext.html), [MessageContext](http://docs.google.com/javax/xml/ws/handler/MessageContext.html), [NavigableMap](http://docs.google.com/java/util/NavigableMap.html)<K,V>, [SOAPMessageContext](http://docs.google.com/javax/xml/ws/handler/soap/SOAPMessageContext.html), [SortedMap](http://docs.google.com/java/util/SortedMap.html)<K,V> **All Known Implementing Classes:** [AbstractMap](http://docs.google.com/java/util/AbstractMap.html), [Attributes](http://docs.google.com/java/util/jar/Attributes.html), [AuthProvider](http://docs.google.com/java/security/AuthProvider.html), [ConcurrentHashMap](http://docs.google.com/java/util/concurrent/ConcurrentHashMap.html), [ConcurrentSkipListMap](http://docs.google.com/java/util/concurrent/ConcurrentSkipListMap.html), [EnumMap](http://docs.google.com/java/util/EnumMap.html), [HashMap](http://docs.google.com/java/util/HashMap.html), [Hashtable](http://docs.google.com/java/util/Hashtable.html), [IdentityHashMap](http://docs.google.com/java/util/IdentityHashMap.html), [LinkedHashMap](http://docs.google.com/java/util/LinkedHashMap.html), [PrinterStateReasons](http://docs.google.com/javax/print/attribute/standard/PrinterStateReasons.html), [Properties](http://docs.google.com/java/util/Properties.html), [Provider](http://docs.google.com/java/security/Provider.html), [RenderingHints](http://docs.google.com/java/awt/RenderingHints.html), [SimpleBindings](http://docs.google.com/javax/script/SimpleBindings.html), [TabularDataSupport](http://docs.google.com/javax/management/openmbean/TabularDataSupport.html), [TreeMap](http://docs.google.com/java/util/TreeMap.html), [UIDefaults](http://docs.google.com/javax/swing/UIDefaults.html), [WeakHashMap](http://docs.google.com/java/util/WeakHashMap.html)

public interface **Map<K,V>**

An object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value.

This interface takes the place of the Dictionary class, which was a totally abstract class rather than an interface.

The Map interface provides three *collection views*, which allow a map's contents to be viewed as a set of keys, collection of values, or set of key-value mappings. The *order* of a map is defined as the order in which the iterators on the map's collection views return their elements. Some map implementations, like the TreeMap class, make specific guarantees as to their order; others, like the HashMap class, do not.

Note: great care must be exercised if mutable objects are used as map keys. The behavior of a map is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is a key in the map. A special case of this prohibition is that it is not permissible for a map to contain itself as a key. While it is permissible for a map to contain itself as a value, extreme caution is advised: the equals and hashCode methods are no longer well defined on such a map.

All general-purpose map implementation classes should provide two "standard" constructors: a void (no arguments) constructor which creates an empty map, and a constructor with a single argument of type Map, which creates a new map with the same key-value mappings as its argument. In effect, the latter constructor allows the user to copy any map, producing an equivalent map of the desired class. There is no way to enforce this recommendation (as interfaces cannot contain constructors) but all of the general-purpose map implementations in the JDK comply.

The "destructive" methods contained in this interface, that is, the methods that modify the map on which they operate, are specified to throw UnsupportedOperationException if this map does not support the operation. If this is the case, these methods may, but are not required to, throw an UnsupportedOperationException if the invocation would have no effect on the map. For example, invoking the [putAll(Map)](http://docs.google.com/java/util/Map.html#putAll(java.util.Map)) method on an unmodifiable map may, but is not required to, throw the exception if the map whose mappings are to be "superimposed" is empty.

Some map implementations have restrictions on the keys and values they may contain. For example, some implementations prohibit null keys and values, and some have restrictions on the types of their keys. Attempting to insert an ineligible key or value throws an unchecked exception, typically NullPointerException or ClassCastException. Attempting to query the presence of an ineligible key or value may throw an exception, or it may simply return false; some implementations will exhibit the former behavior and some will exhibit the latter. More generally, attempting an operation on an ineligible key or value whose completion would not result in the insertion of an ineligible element into the map may throw an exception or it may succeed, at the option of the implementation. Such exceptions are marked as "optional" in the specification for this interface.

This interface is a member of the  [Java Collections Framework](http://docs.google.com/technotes/guides/collections/index.html).

Many methods in Collections Framework interfaces are defined in terms of the [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) method. For example, the specification for the [containsKey(Object key)](http://docs.google.com/java/util/Map.html#containsKey(java.lang.Object)) method says: "returns true if and only if this map contains a mapping for a key k such that (key==null ? k==null : key.equals(k))." This specification should *not* be construed to imply that invoking Map.containsKey with a non-null argument key will cause key.equals(k) to be invoked for any key k. Implementations are free to implement optimizations whereby the equals invocation is avoided, for example, by first comparing the hash codes of the two keys. (The [Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()) specification guarantees that two objects with unequal hash codes cannot be equal.) More generally, implementations of the various Collections Framework interfaces are free to take advantage of the specified behavior of underlying [Object](http://docs.google.com/java/lang/Object.html) methods wherever the implementor deems it appropriate.

**Since:** 1.2 **See Also:**[HashMap](http://docs.google.com/java/util/HashMap.html), [TreeMap](http://docs.google.com/java/util/TreeMap.html), [Hashtable](http://docs.google.com/java/util/Hashtable.html), [SortedMap](http://docs.google.com/java/util/SortedMap.html), [Collection](http://docs.google.com/java/util/Collection.html), [Set](http://docs.google.com/java/util/Set.html)

| **Nested Class Summary** | |
| --- | --- |
| static interface | [**Map.Entry**](http://docs.google.com/java/util/Map.Entry.html)**<**[**K**](http://docs.google.com/java/util/Map.Entry.html)**,**[**V**](http://docs.google.com/java/util/Map.Entry.html)**>**            A map entry (key-value pair). |

| **Method Summary** | |
| --- | --- |
| void | [**clear**](http://docs.google.com/java/util/Map.html#clear())()            Removes all of the mappings from this map (optional operation). |
| boolean | [**containsKey**](http://docs.google.com/java/util/Map.html#containsKey(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns true if this map contains a mapping for the specified key. |
| boolean | [**containsValue**](http://docs.google.com/java/util/Map.html#containsValue(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) value)            Returns true if this map maps one or more keys to the specified value. |
| [Set](http://docs.google.com/java/util/Set.html)<[Map.Entry](http://docs.google.com/java/util/Map.Entry.html)<[K](http://docs.google.com/java/util/Map.html),[V](http://docs.google.com/java/util/Map.html)>> | [**entrySet**](http://docs.google.com/java/util/Map.html#entrySet())()            Returns a [Set](http://docs.google.com/java/util/Set.html) view of the mappings contained in this map. |
| boolean | [**equals**](http://docs.google.com/java/util/Map.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Compares the specified object with this map for equality. |
| [V](http://docs.google.com/java/util/Map.html) | [**get**](http://docs.google.com/java/util/Map.html#get(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key. |
| int | [**hashCode**](http://docs.google.com/java/util/Map.html#hashCode())()            Returns the hash code value for this map. |
| boolean | [**isEmpty**](http://docs.google.com/java/util/Map.html#isEmpty())()            Returns true if this map contains no key-value mappings. |
| [Set](http://docs.google.com/java/util/Set.html)<[K](http://docs.google.com/java/util/Map.html)> | [**keySet**](http://docs.google.com/java/util/Map.html#keySet())()            Returns a [Set](http://docs.google.com/java/util/Set.html) view of the keys contained in this map. |
| [V](http://docs.google.com/java/util/Map.html) | [**put**](http://docs.google.com/java/util/Map.html#put(K,%20V))([K](http://docs.google.com/java/util/Map.html) key, [V](http://docs.google.com/java/util/Map.html) value)            Associates the specified value with the specified key in this map (optional operation). |
| void | [**putAll**](http://docs.google.com/java/util/Map.html#putAll(java.util.Map))([Map](http://docs.google.com/java/util/Map.html)<? extends [K](http://docs.google.com/java/util/Map.html),? extends [V](http://docs.google.com/java/util/Map.html)> m)            Copies all of the mappings from the specified map to this map (optional operation). |
| [V](http://docs.google.com/java/util/Map.html) | [**remove**](http://docs.google.com/java/util/Map.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) key)            Removes the mapping for a key from this map if it is present (optional operation). |
| int | [**size**](http://docs.google.com/java/util/Map.html#size())()            Returns the number of key-value mappings in this map. |
| [Collection](http://docs.google.com/java/util/Collection.html)<[V](http://docs.google.com/java/util/Map.html)> | [**values**](http://docs.google.com/java/util/Map.html#values())()            Returns a [Collection](http://docs.google.com/java/util/Collection.html) view of the values contained in this map. |

| **Method Detail** |
| --- |

### size

int **size**()

Returns the number of key-value mappings in this map. If the map contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

**Returns:**the number of key-value mappings in this map

### isEmpty

boolean **isEmpty**()

Returns true if this map contains no key-value mappings.

**Returns:**true if this map contains no key-value mappings

### containsKey

boolean **containsKey**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns true if this map contains a mapping for the specified key. More formally, returns true if and only if this map contains a mapping for a key k such that (key==null ? k==null : key.equals(k)). (There can be at most one such mapping.)

**Parameters:**key - key whose presence in this map is to be tested **Returns:**true if this map contains a mapping for the specified key **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the key is of an inappropriate type for this map (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified key is null and this map does not permit null keys (optional)

### containsValue

boolean **containsValue**([Object](http://docs.google.com/java/lang/Object.html) value)

Returns true if this map maps one or more keys to the specified value. More formally, returns true if and only if this map contains at least one mapping to a value v such that (value==null ? v==null : value.equals(v)). This operation will probably require time linear in the map size for most implementations of the Map interface.

**Parameters:**value - value whose presence in this map is to be tested **Returns:**true if this map maps one or more keys to the specified value **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the value is of an inappropriate type for this map (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified value is null and this map does not permit null values (optional)

### get

[V](http://docs.google.com/java/util/Map.html) **get**([Object](http://docs.google.com/java/lang/Object.html) key)

Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

More formally, if this map contains a mapping from a key k to a value v such that (key==null ? k==null : key.equals(k)), then this method returns v; otherwise it returns null. (There can be at most one such mapping.)

If this map permits null values, then a return value of null does not *necessarily* indicate that the map contains no mapping for the key; it's also possible that the map explicitly maps the key to null. The [containsKey](http://docs.google.com/java/util/Map.html#containsKey(java.lang.Object)) operation may be used to distinguish these two cases.

**Parameters:**key - the key whose associated value is to be returned **Returns:**the value to which the specified key is mapped, or null if this map contains no mapping for the key **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the key is of an inappropriate type for this map (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified key is null and this map does not permit null keys (optional)

### put

[V](http://docs.google.com/java/util/Map.html) **put**([K](http://docs.google.com/java/util/Map.html) key,  
 [V](http://docs.google.com/java/util/Map.html) value)

Associates the specified value with the specified key in this map (optional operation). If the map previously contained a mapping for the key, the old value is replaced by the specified value. (A map m is said to contain a mapping for a key k if and only if [m.containsKey(k)](http://docs.google.com/java/util/Map.html#containsKey(java.lang.Object)) would return true.)

**Parameters:**key - key with which the specified value is to be associatedvalue - value to be associated with the specified key **Returns:**the previous value associated with key, or null if there was no mapping for key. (A null return can also indicate that the map previously associated null with key, if the implementation supports null values.) **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the put operation is not supported by this map [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified key or value prevents it from being stored in this map [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified key or value is null and this map does not permit null keys or values [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified key or value prevents it from being stored in this map

### remove

[V](http://docs.google.com/java/util/Map.html) **remove**([Object](http://docs.google.com/java/lang/Object.html) key)

Removes the mapping for a key from this map if it is present (optional operation). More formally, if this map contains a mapping from key k to value v such that (key==null ? k==null : key.equals(k)), that mapping is removed. (The map can contain at most one such mapping.)

Returns the value to which this map previously associated the key, or null if the map contained no mapping for the key.

If this map permits null values, then a return value of null does not *necessarily* indicate that the map contained no mapping for the key; it's also possible that the map explicitly mapped the key to null.

The map will not contain a mapping for the specified key once the call returns.

**Parameters:**key - key whose mapping is to be removed from the map **Returns:**the previous value associated with key, or null if there was no mapping for key. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the remove operation is not supported by this map [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the key is of an inappropriate type for this map (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified key is null and this map does not permit null keys (optional)

### putAll

void **putAll**([Map](http://docs.google.com/java/util/Map.html)<? extends [K](http://docs.google.com/java/util/Map.html),? extends [V](http://docs.google.com/java/util/Map.html)> m)

Copies all of the mappings from the specified map to this map (optional operation). The effect of this call is equivalent to that of calling [put(k, v)](http://docs.google.com/java/util/Map.html#put(K,%20V)) on this map once for each mapping from key k to value v in the specified map. The behavior of this operation is undefined if the specified map is modified while the operation is in progress.

**Parameters:**m - mappings to be stored in this map **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the putAll operation is not supported by this map [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of a key or value in the specified map prevents it from being stored in this map [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified map is null, or if this map does not permit null keys or values, and the specified map contains null keys or values [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of a key or value in the specified map prevents it from being stored in this map

### clear

void **clear**()

Removes all of the mappings from this map (optional operation). The map will be empty after this call returns.

**Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the clear operation is not supported by this map

### keySet

[Set](http://docs.google.com/java/util/Set.html)<[K](http://docs.google.com/java/util/Map.html)> **keySet**()

Returns a [Set](http://docs.google.com/java/util/Set.html) view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation), the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the Iterator.remove, Set.remove, removeAll, retainAll, and clear operations. It does not support the add or addAll operations.

**Returns:**a set view of the keys contained in this map

### values

[Collection](http://docs.google.com/java/util/Collection.html)<[V](http://docs.google.com/java/util/Map.html)> **values**()

Returns a [Collection](http://docs.google.com/java/util/Collection.html) view of the values contained in this map. The collection is backed by the map, so changes to the map are reflected in the collection, and vice-versa. If the map is modified while an iteration over the collection is in progress (except through the iterator's own remove operation), the results of the iteration are undefined. The collection supports element removal, which removes the corresponding mapping from the map, via the Iterator.remove, Collection.remove, removeAll, retainAll and clear operations. It does not support the add or addAll operations.

**Returns:**a collection view of the values contained in this map

### entrySet

[Set](http://docs.google.com/java/util/Set.html)<[Map.Entry](http://docs.google.com/java/util/Map.Entry.html)<[K](http://docs.google.com/java/util/Map.html),[V](http://docs.google.com/java/util/Map.html)>> **entrySet**()

Returns a [Set](http://docs.google.com/java/util/Set.html) view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the Iterator.remove, Set.remove, removeAll, retainAll and clear operations. It does not support the add or addAll operations.

**Returns:**a set view of the mappings contained in this map

### equals

boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) o)

Compares the specified object with this map for equality. Returns true if the given object is also a map and the two maps represent the same mappings. More formally, two maps m1 and m2 represent the same mappings if m1.entrySet().equals(m2.entrySet()). This ensures that the equals method works properly across different implementations of the Map interface.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**o - object to be compared for equality with this map **Returns:**true if the specified object is equal to this map**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

int **hashCode**()

Returns the hash code value for this map. The hash code of a map is defined to be the sum of the hash codes of each entry in the map's entrySet() view. This ensures that m1.equals(m2) implies that m1.hashCode()==m2.hashCode() for any two maps m1 and m2, as required by the general contract of [Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()).

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**the hash code value for this map**See Also:**[Map.Entry.hashCode()](http://docs.google.com/java/util/Map.Entry.html#hashCode()), [Object.equals(Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [equals(Object)](http://docs.google.com/java/util/Map.html#equals(java.lang.Object))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Map.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/Locale.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Map.Entry.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/Map.html)    [**NO FRAMES**](http://docs.google.com/Map.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | CONSTR | [METHOD](#2et92p0) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).