Map

Modifier and Type	Method	Description
`void`	`clear()`	Removes all of the mappings from this map (optional operation).
`default V`	`compute(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)`	Attempts to compute a mapping for the specified key and its current mapped value (or `null` if there is no current mapping).
`default V`	`computeIfAbsent(K key, Function<? super K,? extends V> mappingFunction)`	If the specified key is not already associated with a value (or is mapped to `null`), attempts to compute its value using the given mapping function and enters it into this map unless `null`.
`default V`	`computeIfPresent(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)`	If the value for the specified key is present and non-null, attempts to compute a new mapping given the key and its current mapped value.
`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.
`static <K,V> Map.Entry<K,V>`	`entry(K k, V v)`	Returns an immutable `Map.Entry` containing the given key and value.
`Set<Map.Entry<K,V>>`	`entrySet()`	Returns a `Set` view of the mappings contained in this map.
`boolean`	`equals(Object o)`	Compares the specified object with this map for equality.
`default void`	`forEach(BiConsumer<? super K,? super V> action)`	Performs the given action for each entry in this map until all entries have been processed or the action throws an exception.
`V`	`get(Object key)`	Returns the value to which the specified key is mapped, or `null` if this map contains no mapping for the key.
`default V`	`getOrDefault(Object key, V defaultValue)`	Returns the value to which the specified key is mapped, or `defaultValue` if this map contains no mapping for the key.
`int`	`hashCode()`	Returns the hash code value for this map.
`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.
`default V`	`merge(K key, V value, BiFunction<? super V,? super V,? extends V> remappingFunction)`	If the specified key is not already associated with a value or is associated with null, associates it with the given non-null value.
`static <K,V> Map<K,V>`	`of()`	Returns an immutable map containing zero mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1)`	Returns an immutable map containing a single mapping.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2)`	Returns an immutable map containing two mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3)`	Returns an immutable map containing three mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4)`	Returns an immutable map containing four mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5)`	Returns an immutable map containing five mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6)`	Returns an immutable map containing six mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7)`	Returns an immutable map containing seven mappings.

`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7, K k8, V v8)`	Returns an immutable map containing eight mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7, K k8, V v8, K k9, V v9)`	Returns an immutable map containing nine mappings.
`static <K,V> Map<K,V>`	`of(K k1, V v1, K k2, V v2, K k3, V v3, K k4, V v4, K k5, V v5, K k6, V v6, K k7, V v7, K k8, V v8, K k9, V v9, K k10, V v10)`	Returns an immutable map containing ten mappings.
`static <K,V> Map<K,V>`	`ofEntries(Map.Entry<? extends K,? extends V>... entries)`	Returns an immutable map containing keys and values extracted from the given entries.
`V`	`put(K key, V value)`	Associates the specified value with the specified key in this map (optional operation).
`void`	`putAll(Map<? extends K,? extends V> m)`	Copies all of the mappings from the specified map to this map (optional operation).
`default V`	`putIfAbsent(K key, V value)`	If the specified key is not already associated with a value (or is mapped to `null`) associates it with the given value and returns `null`, else returns the current value.
`V`	`remove(Object key)`	Removes the mapping for a key from this map if it is present (optional operation).
`default boolean`	`remove(Object key, Object value)`	Removes the entry for the specified key only if it is currently mapped to the specified value.
`default V`	`replace(K key, V value)`	Replaces the entry for the specified key only if it is currently mapped to some value.
`default boolean`	`replace(K key, V oldValue, V newValue)`	Replaces the entry for the specified key only if currently mapped to the specified value.
`default void`	`replaceAll(BiFunction<? super K,? super V,? extends V> function)`	Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception.
`int`	`size()`	Returns the number of key-value mappings in this map.
`Collection<V>`	`values()`	Returns a `Collection` view of the values contained in this map.

•

List

Modifier and Type	Method	Description
void	add(int index, E element)	Inserts the specified element at the specified position in this list (optional operation).
boolean	add(E e)	Appends the specified element to the end of this list (optional operation).
boolean	addAll(int index, Collection<? extends E> c)	Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
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 (optional operation).
void	clear()	Removes all of the elements from this list (optional operation).
boolean	contains(Object o)	Returns `true` if this list contains the specified element.
boolean	containsAll(Collection<?> c)	Returns `true` if this list contains all of the elements of the specified collection.
boolean	equals(Object o)	Compares the specified object with this list for equality.
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(Object o)	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	isEmpty()	Returns `true` if this list contains no elements.
Iterator<E>	iterator()	Returns an iterator over the elements in this list in proper sequence.
int	lastIndexOf(Object o)	Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
ListIterator<E>	listIterator()	Returns a list iterator over the elements in this list (in proper sequence).
ListIterator<E>	listIterator(int index)	Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.
static <E> List<E>	of()	Returns an immutable list containing zero elements.
static <E> List<E>	of(E e1)	Returns an immutable list containing one element.
static <E> List<E>	of(E... elements)	Returns an immutable list containing an arbitrary number of elements.
static <E> List<E>	of(E e1, E e2)	Returns an immutable list containing two elements.
static <E> List<E>	of(E e1, E e2, E e3)	Returns an immutable list containing three elements.
static <E> List<E>	of(E e1, E e2, E e3, E e4)	Returns an immutable list containing four elements.
static <E> List<E>	of(E e1, E e2, E e3, E e4, E e5)	Returns an immutable list containing five elements.
static <E> List<E>	of(E e1, E e2, E e3, E e4, E e5, E e6)	Returns an immutable list containing six elements.
static <E> List<E>	of(E e1, E e2, E e3, E e4, E e5, E e6, E e7)	Returns an immutable list containing seven elements.
static <E> List<E>	of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8)	Returns an immutable list containing eight elements.
static <E> List<E>	of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9)	Returns an immutable list containing nine elements.
static <E> List<E>	of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9, E e10)	Returns an immutable list containing ten elements.

E	remove(int index)	Removes the element at the specified position in this list (optional operation).
boolean	remove(Object o)	Removes the first occurrence of the specified element from this list, if it is present (optional operation).
boolean	removeAll(Collection<?> c)	Removes from this list all of its elements that are contained in the specified collection (optional operation).
default void	replaceAll(UnaryOperator<E> operator)	Replaces each element of this list with the result of applying the operator to that element.
boolean	retainAll(Collection<?> c)	Retains only the elements in this list that are contained in the specified collection (optional operation).
E	set(int index, E element)	Replaces the element at the specified position in this list with the specified element (optional operation).
int	size()	Returns the number of elements in this list.
default void	sort(Comparator<? super E> c)	Sorts this list according to the order induced by the specified `Comparator`.
default Spliterator<E>	spliterator()	Creates a `Spliterator` over the elements in this list.
List<E>	subList(int fromIndex, int toIndex)	Returns a view of the portion of this list between the specified `fromIndex`, inclusive, and `toIndex`, exclusive.
Object[]	toArray()	Returns an array containing all of the elements in this list in proper sequence (from first to last element).
<T> T[]	toArray(T[] a)	Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array.

Set

Modifier and Type	Method	Description
`boolean`	`add(E e)`	Adds the specified element to this set if it is not already present (optional operation).
`boolean`	`addAll(Collection<? extends E> c)`	Adds all of the elements in the specified collection to this set if they're not already present (optional operation).
`void`	`clear()`	Removes all of the elements from this set (optional operation).
`boolean`	`contains(Object o)`	Returns `true` if this set contains the specified element.
`boolean`	`containsAll(Collection<?> c)`	Returns `true` if this set contains all of the elements of the specified collection.
`boolean`	`equals(Object o)`	Compares the specified object with this set for equality.
`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.
`static <E> Set<E>`	`of()`	Returns an immutable set containing zero elements.
`static <E> Set<E>`	`of(E e1)`	Returns an immutable set containing one element.
`static <E> Set<E>`	`of(E... elements)`	Returns an immutable set containing an arbitrary number of elements.
`static <E> Set<E>`	`of(E e1, E e2)`	Returns an immutable set containing two elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3)`	Returns an immutable set containing three elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3, E e4)`	Returns an immutable set containing four elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3, E e4, E e5)`	Returns an immutable set containing five elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3, E e4, E e5, E e6)`	Returns an immutable set containing six elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3, E e4, E e5, E e6, E e7)`	Returns an immutable set containing seven elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8)`	Returns an immutable set containing eight elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9)`	Returns an immutable set containing nine elements.
`static <E> Set<E>`	`of(E e1, E e2, E e3, E e4, E e5, E e6, E e7, E e8, E e9, E e10)`	Returns an immutable set containing ten elements.
`boolean`	`remove(Object o)`	Removes the specified element from this set if it is present (optional operation).
`boolean`	`removeAll(Collection<?> c)`	Removes from this set all of its elements that are contained in the specified collection (optional operation).
`boolean`	`retainAll(Collection<?> c)`	Retains only the elements in this set that are contained in the specified collection (optional operation).
`int`	`size()`	Returns the number of elements in this set (its cardinality).
`default Spliterator<E>`	`spliterator()`	Creates a `Spliterator` over the elements in this set.
`Object[]`	`toArray()`	Returns an array containing all of the elements in this set.
`<T> T[]`	`toArray(T[] 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.

Collection

Modifier and Type	Method	Description
boolean	add(E e)	Ensures that this collection contains the specified element (optional operation).
boolean	addAll(Collection<? extends E> c)	Adds all of the elements in the specified collection to this collection (optional operation).
void	clear()	Removes all of the elements from this collection (optional operation).
boolean	contains(Object o)	Returns `true` if this collection contains the specified element.
boolean	containsAll(Collection<?> c)	Returns `true` if this collection contains all of the elements in the specified collection.
boolean	equals(Object o)	Compares the specified object with this collection for equality.
int	hashCode()	Returns the hash code value for this collection.
boolean	isEmpty()	Returns `true` if this collection contains no elements.
Iterator<E>	iterator()	Returns an iterator over the elements in this collection.
default Stream<E>	parallelStream()	Returns a possibly parallel `Stream` with this collection as its source.
boolean	remove(Object o)	Removes a single instance of the specified element from this collection, if it is present (optional operation).
boolean	removeAll(Collection<?> c)	Removes all of this collection's elements that are also contained in the specified collection (optional operation).
default boolean	removeIf(Predicate<? super E> filter)	Removes all of the elements of this collection that satisfy the given predicate.
boolean	retainAll(Collection<?> c)	Retains only the elements in this collection that are contained in the specified collection (optional operation).
int	size()	Returns the number of elements in this collection.
default Spliterator<E>	spliterator()	Creates a `Spliterator` over the elements in this collection.
default Stream<E>	stream()	Returns a sequential `Stream` with this collection as its source.
Object[]	toArray()	Returns an array containing all of the elements in this collection.
<T> T[]	toArray(T[] 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.