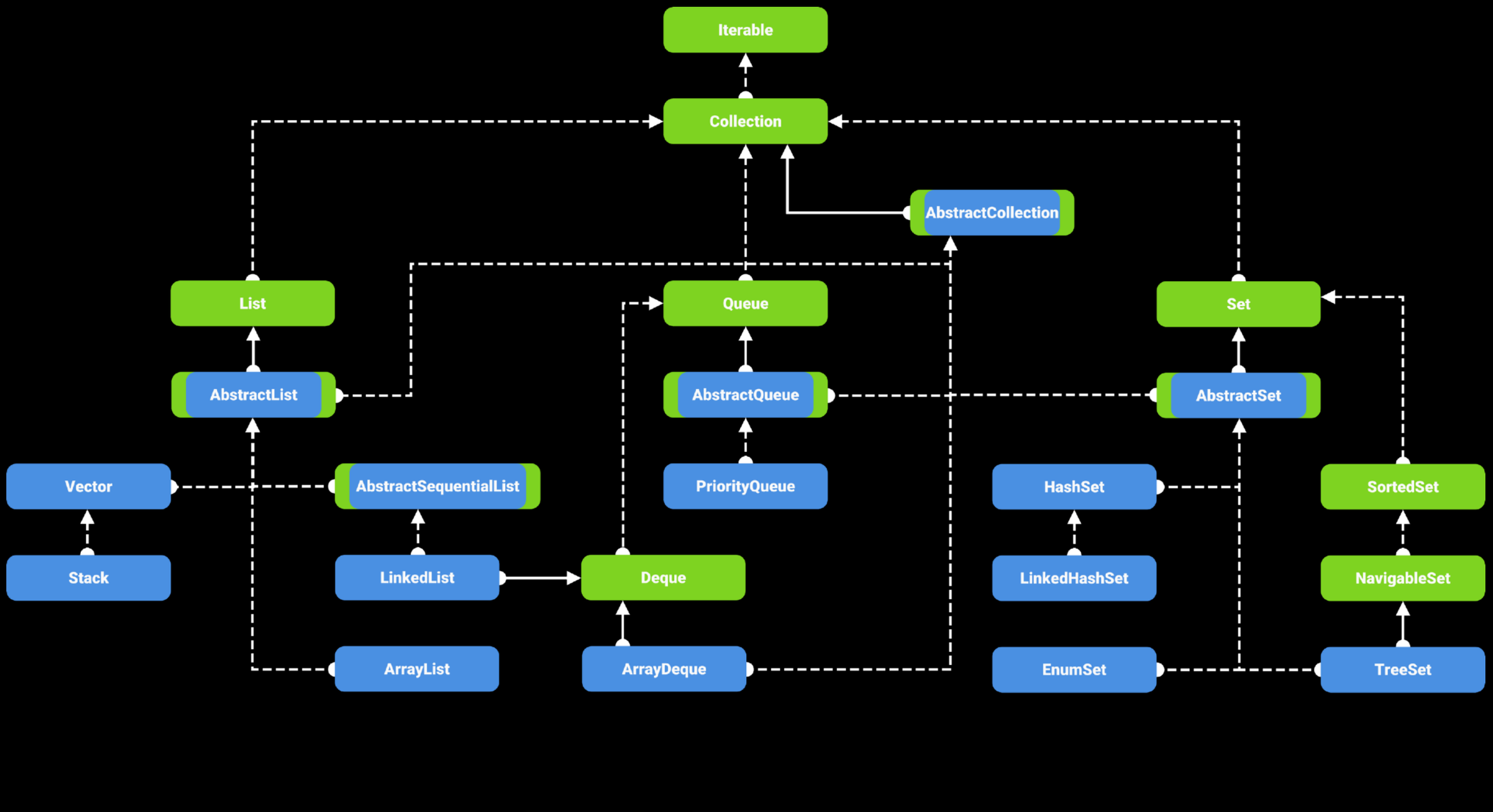
Anand K Parmar

# JAVA COLLECTION CHEATSHEET

A Guide to Master
Java Collection Framework

# java.util. COLLECTION AND ITS HIERARCHY

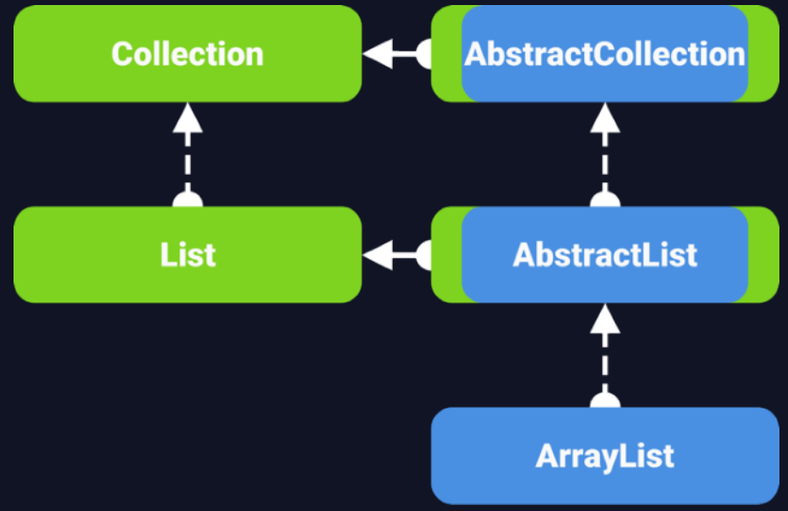


## FREQUENTLY USED FUNCTIONS

- **add(Object o):** Appends the specified element to the end of this list.
- **addAll(Collection 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.
- **get(int index):** Returns the element at the specified position in this list.
- **remove(int index):** Removes the element at the specified position in this list.
- **remove(Object o):** Removes the first occurrence of the specified element.
- **removeAll(Collection c):** Removes from this list all of its elements that are contained in the specified collection.
- **set(int index, Object o):** Replaces the element at the specified position in this list with the specified element.
- **isEmpty():** Returns true if this list contains no elements.
- **size():** Returns the number of elements in this list.
- **clear():** Removes all of the elements from this list.
- **contains(Object o):** Returns true if this list contains the specified element.
- **indexOf(Object o):** Returns the index of the first occurrence of the specified element in this list, or -1.
- **sort(Comparator c):** Sorts this list according to the comparator.
- **toArray():** Returns an array containing all of the elements in this list.
- **iterator():** Returns an iterator over the elements in this list in proper sequence.
- **listIterator():** Returns a list iterator over the elements in this list.

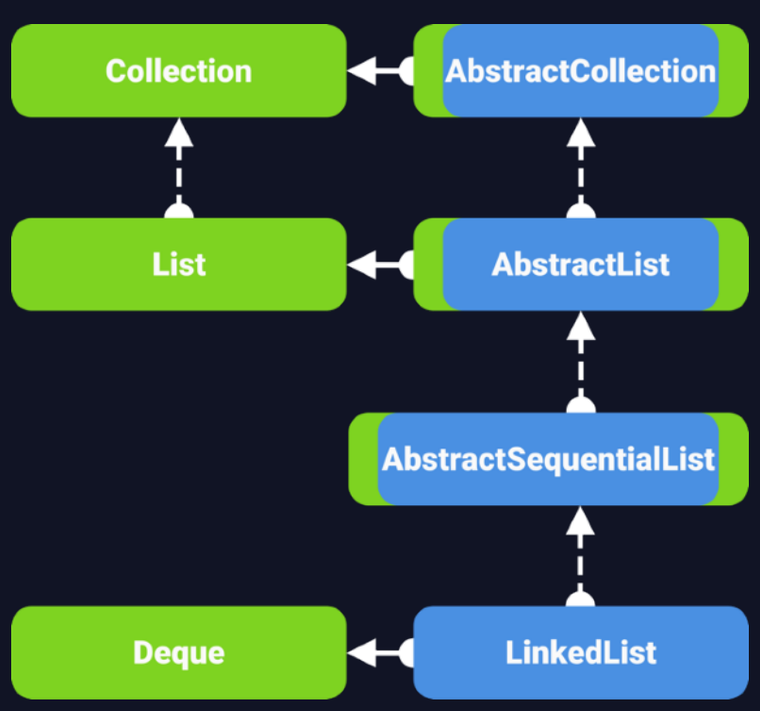
**Java.util.**

# ARRAYLIST



Java.util.

# LINKEDLIST




## FREQUENTLY USED FUNCTIONS

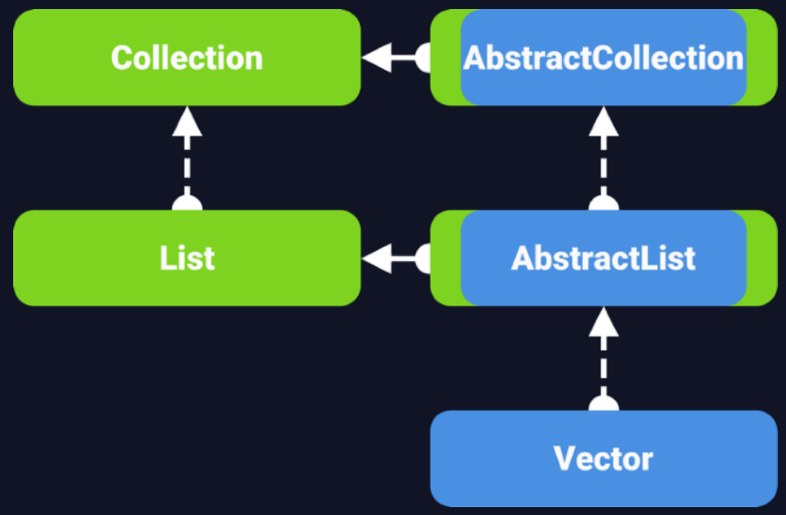
- **add(Object o):** Appends the specified element to the end of this list.
- **addFirst(Object o):** Inserts the specified element at the beginning of this list.
- **addLast(Object o):** Appends the specified element to the end of this list.
- **addAll(Collection c):** Appends all element of a given collection.
- **get(int index):** Returns the element at the specified position in this list.
- **getFirst():** Returns the first element in this list.
- **getLast():** Returns the last element in this list.
- **remove(int index):** Removes the element at the specified position in this list.
- **remove(Object o):** Removes the first occurrence of the specified element.
- **removeFirst():** Removes and returns the first element from this list.
- **removeLast():** Removes and returns the last element from this list.
- **set(int index, Object o):** Replaces the element at the specified position in this list with the specified element.
- **isEmpty():** Returns true if this list contains no elements.
- **size():** Returns the number of elements in this list.
- **clear():** Removes all of the elements from this list.
- **contains(Object o):** Returns true if this list contains the specified element.
- **indexOf(Object o):** Returns the index of the first occurrence of the specified element in this list, or -1.
- **listIterator(int index):** Returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list.
- **descendingIterator():** Returns an iterator over the elements in reverse sequential order.

## FREQUENTLY USED FUNCTIONS

- **add(Object o):** Appends the specified element to the end of this vector.
- **get(int index):** Returns the element at the specified position in this vector.
- **firstElement():** Returns the first component (the item at index 0) of this vector.
- **lastElement():** Returns the last component of this vector.
- **remove(int index):** Removes the element at the specified position in this vector.
- **remove(Object o):** Removes the first occurrence of the specified element.
- **removeAll(Collection c):** Removes from this vector all of its elements that are contained in the specified collection.
- **set(int index, Object o):** Replaces the element at the specified position in this vector with the specified element.
- **isEmpty():** Tests if this vector has no components.
- **size():** Returns the number of components in this vector.
- **clear():** Removes all of the elements from this vector.
- **contains(Object o):** Returns true if this vector contains the specified element.
- **indexOf(Object o):** Returns the index of the first occurrence of the specified element in this vector, or -1.
- **sort(Comparator c):** Sorts this vector according to the comparator.
- **toArray():** Returns an array containing all of the elements in this vector in the correct order.
- **iterator():** Returns an iterator over the elements in this list in proper sequence.
- **listIterator():** Returns a list iterator over the elements in this vector.
- **elements():** Returns an enumeration of the components of this vector.

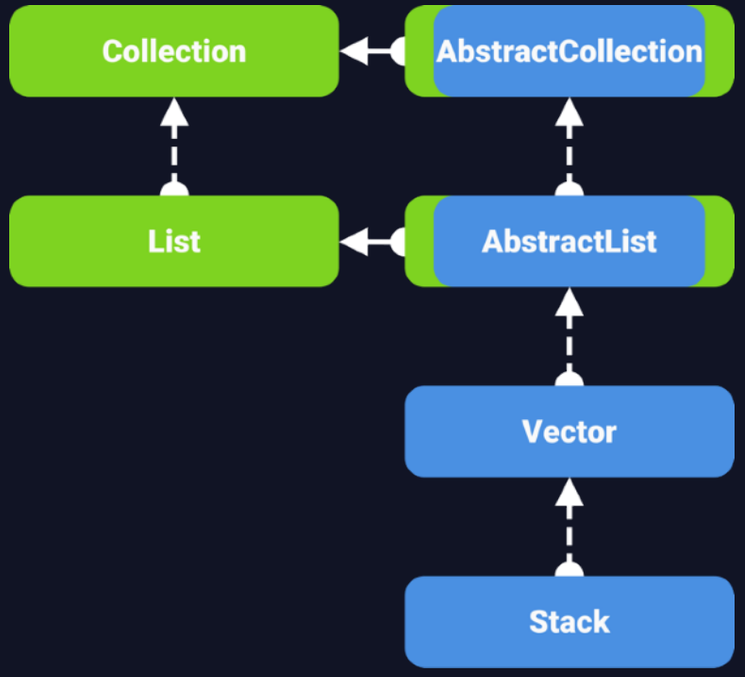
**Java.util.**

# VECTOR



Java.util.

# STACK




## FREQUENTLY USED FUNCTIONS

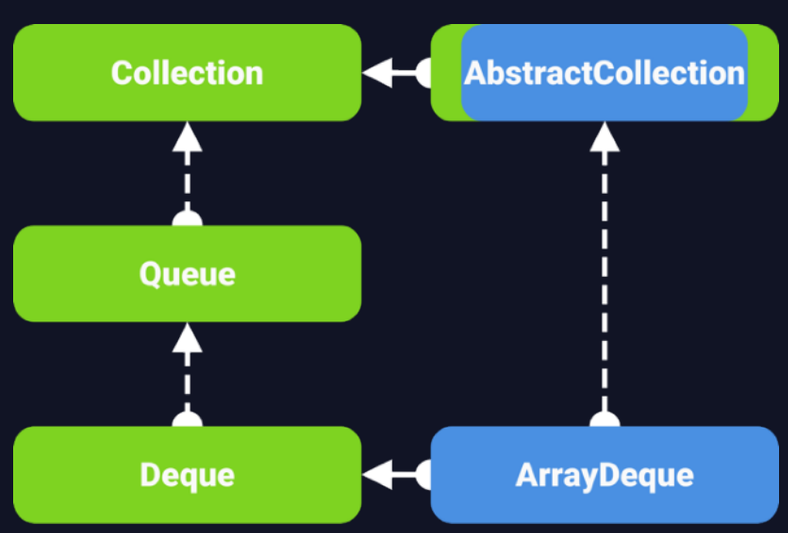
- **empty():** Tests if this stack is empty.
- **peek():** Looks at the object at the top of this stack without removing it from the stack.
- **pop():** Removes the object at the top of this stack and returns that object as the value of this function.
- **push(Object o):** Pushes an item onto the top of this stack.
- **search(Object o):** Returns the 1-based position where an object is on this stack.

## FREQUENTLY USED FUNCTIONS

- **add(Object o):** Appends the specified element to the end of this deque.
- **addFirst(Object o):** Inserts the specified element at the beginning of this deque.
- **addLast(Object o):** Appends the specified element to the end of this deque.
- **getFirst():** Retrieves, but does not remove, the first element of this deque.
- **getLast():** Retrieves, but does not remove, the last element of this deque.
- **remove():** Retrieves and removes the head of the queue represented by this deque.
- **remove(Object o):** Removes a single instance of the specified element from this deque.
- **removeFirst():** Retrieves and removes the first element of this deque.
- **removeLast():** Retrieves and removes the last element of this deque.
- **isEmpty():** Returns true if this deque contains no elements.
- **size():** Returns the number of elements in this deque.
- **clear():** Removes all of the elements from this deque.
- **contains(Object o):** Returns true if this deque contains the specified element.
- **iterator():** Returns an iterator over the elements in this deque.
- **descendingIterator():** Returns an iterator over the elements in this deque in reverse sequential order.
- **push(Object o):** Pushes an element onto the stack represented by this deque.
- **pop():** Pops an element from the stack represented by this deque.
- **peek():** Retrieves, but does not remove, the head of the queue represented by this deque, or returns null if this deque is empty.

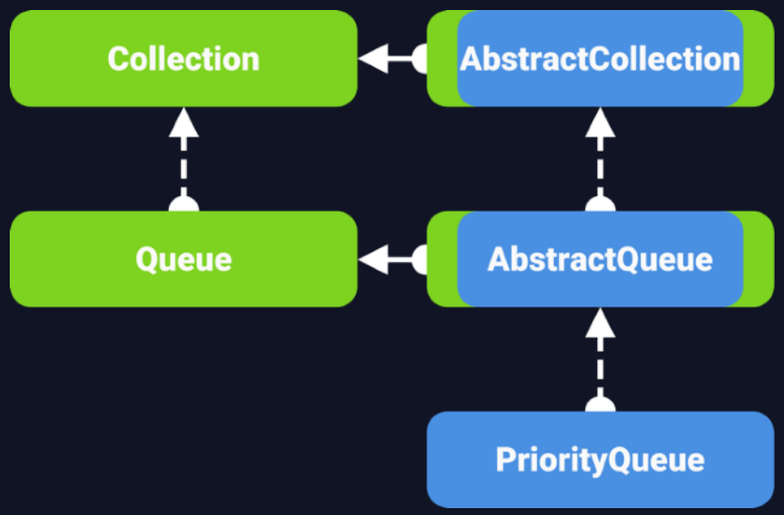
**Java.util.**

# ARRAYDEQUE



Java.util.

# PRIORITYQUEUE

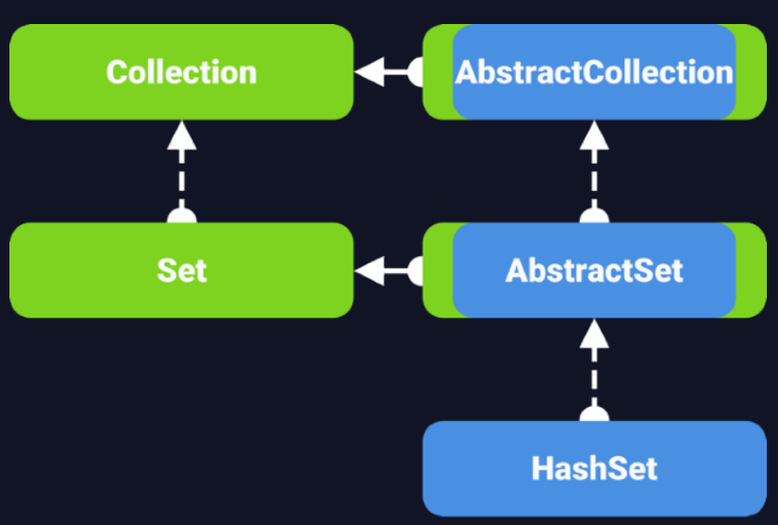



## FREQUENTLY USED FUNCTIONS

- **add(Object o):** Inserts the specified element into this priority queue.
- **offer(Object o):** Inserts the specified element into this priority queue.
- **peek():** Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
- **poll():** Retrieves and removes the head of this queue, or returns null if this queue is empty.
- **remove(Object o):** Removes a single instance of the specified element from this queue, if it is present.
- **contains(Object o):** Returns true if this queue contains the specified element.
- **size():** Returns the number of elements in this collection.
- **iterator():** Returns an iterator over the elements in this queue.
- **comparator():** Returns the comparator used to order the elements in this queue, or null if this queue is sorted according to the natural ordering of its elements.

Java.util.

# HASHSET




## FREQUENTLY USED FUNCTIONS

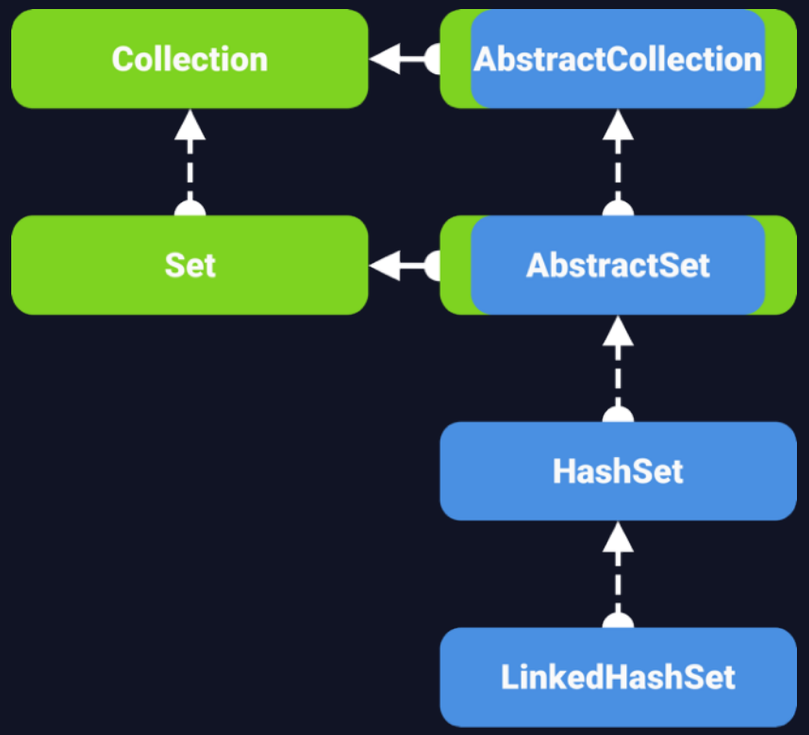
- **add(Object o):** Adds the specified element to this set if it is not already present.
- **remove(Object o):** Removes the specified element from this set if it is present.
- **contains(Object o):** Returns true if this set contains the specified element.
- **size():** Returns the number of elements in this set (its cardinality).
- **clear():** Removes all of the elements from this set.
- **isEmpty():** Returns true if this set contains no elements.
- **iterator():** Returns an iterator over the elements in this set.
- **containsAll(Collection c):** Returns true if this set contains all of the elements of the specified collection. If the specified collection is also a set, this method returns true if it is a ***subset*** of this set.
- **retainAll(Collection c):** Retains only the elements in this set that are contained in the specified collection. If the specified collection is also a set, this operation effectively modifies this set so that its value is the ***intersection*** of the two sets.
- **removeAll(Collection c):** Removes from this set all of its elements that are contained in the specified collection. If the specified collection is also a set, this operation effectively modifies this set so that its value is the *asymmetric set* ***difference*** of the two sets.

## FREQUENTLY USED FUNCTIONS

- **add(Object o):** Adds the specified element to this set if it is not already present.
- **remove(Object o):** Removes the specified element from this set if it is present.
- **contains(Object o):** Returns true if this set contains the specified element.
- **size():** Returns the number of elements in this set (its cardinality).
- **clear():** Removes all of the elements from this set.
- **isEmpty():** Returns true if this set contains no elements.
- **iterator():** Returns an iterator over the elements in this set.
- **containsAll(Collection c):** Returns true if this set contains all of the elements of the specified collection. If the specified collection is also a set, this method returns true if it is a ***subset*** of this set.
- **retainAll(Collection c):** Retains only the elements in this set that are contained in the specified collection. If the specified collection is also a set, this operation effectively modifies this set so that its value is the ***intersection*** of the two sets.
- **removeAll(Collection c):** Removes from this set all of its elements that are contained in the specified collection. If the specified collection is also a set, this operation effectively modifies this set so that its value is the *asymmetric set* ***difference*** of the two sets.

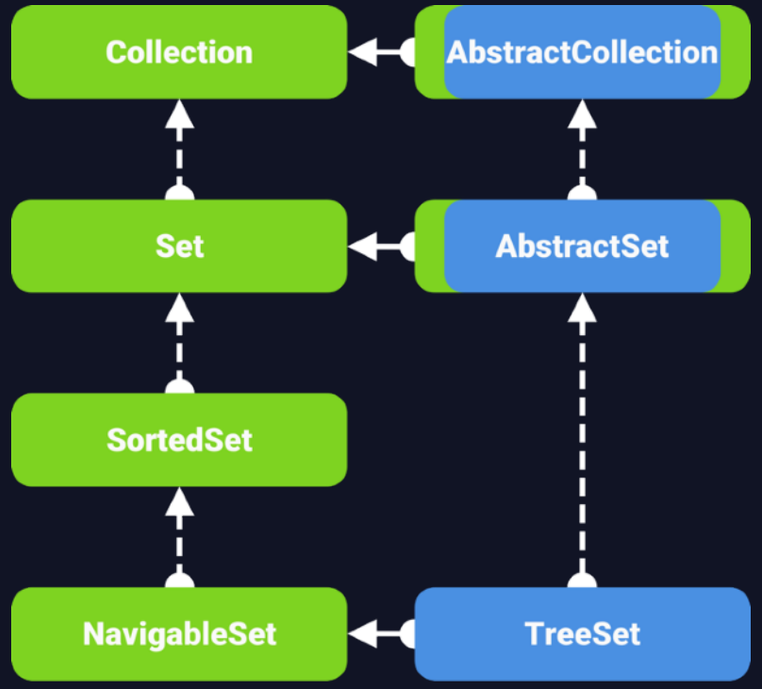
**Java.util.**

# LINKEDHASHSET



Java.util.

# TREESET




## FREQUENTLY USED FUNCTIONS

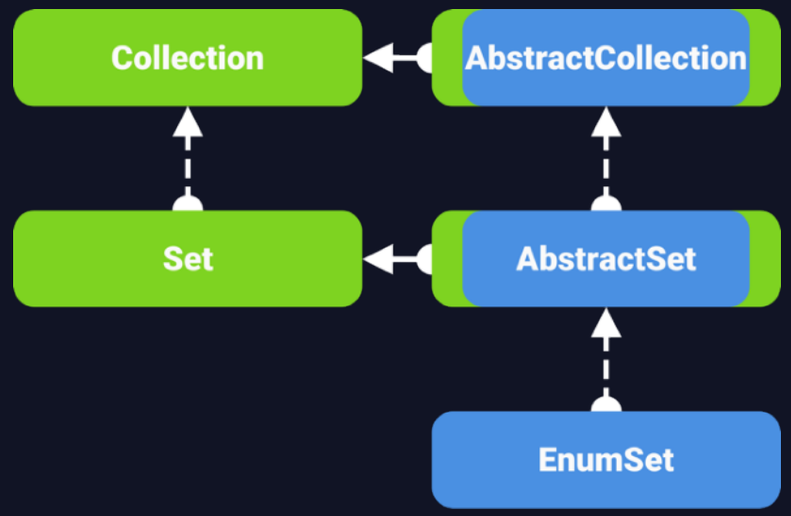
- **add(Object o):** Adds the specified element to this set if it is not already present.
- **remove(Object o):** Removes the specified element from this set if it is present.
- **pollFirst():** Retrieves and removes the first (lowest) element, or returns null.
- **pollLast():** Retrieves and removes the last (highest) element, or returns null.
- **contains(Object o):** Returns true if this set contains the specified element.
- **first():** Returns the first (lowest) element currently in this set.
- **last():** Returns the last (highest) element currently in this set.
- **size():** Returns the number of elements in this set (its cardinality).
- **clear():** Removes all of the elements from this set.
- **isEmpty():** Returns true if this set contains no elements.
- **iterator():** Returns an iterator over the elements in this set in ascending order.
- **descendingIterator():** Returns an iterator over the elements in this set in descending order.
- **ceiling(Object o):** Returns the least element in this set greater than or equal to the given element, or null if there is no such element.
- **floor(Object o):** Returns the greatest element in this set less than or equal to the given element, or null if there is no such element.
- **headSet(Object toElement, boolean inclusive):** Returns a view of the portion of this set whose elements are less than (or equal to, if inclusive is true) toElement.
- **tailSet(Object toElement, boolean inclusive):** Returns a view of the portion of this set whose elements are greater than (or equal to, if inclusive is true) toElement.

## FREQUENTLY USED FUNCTIONS

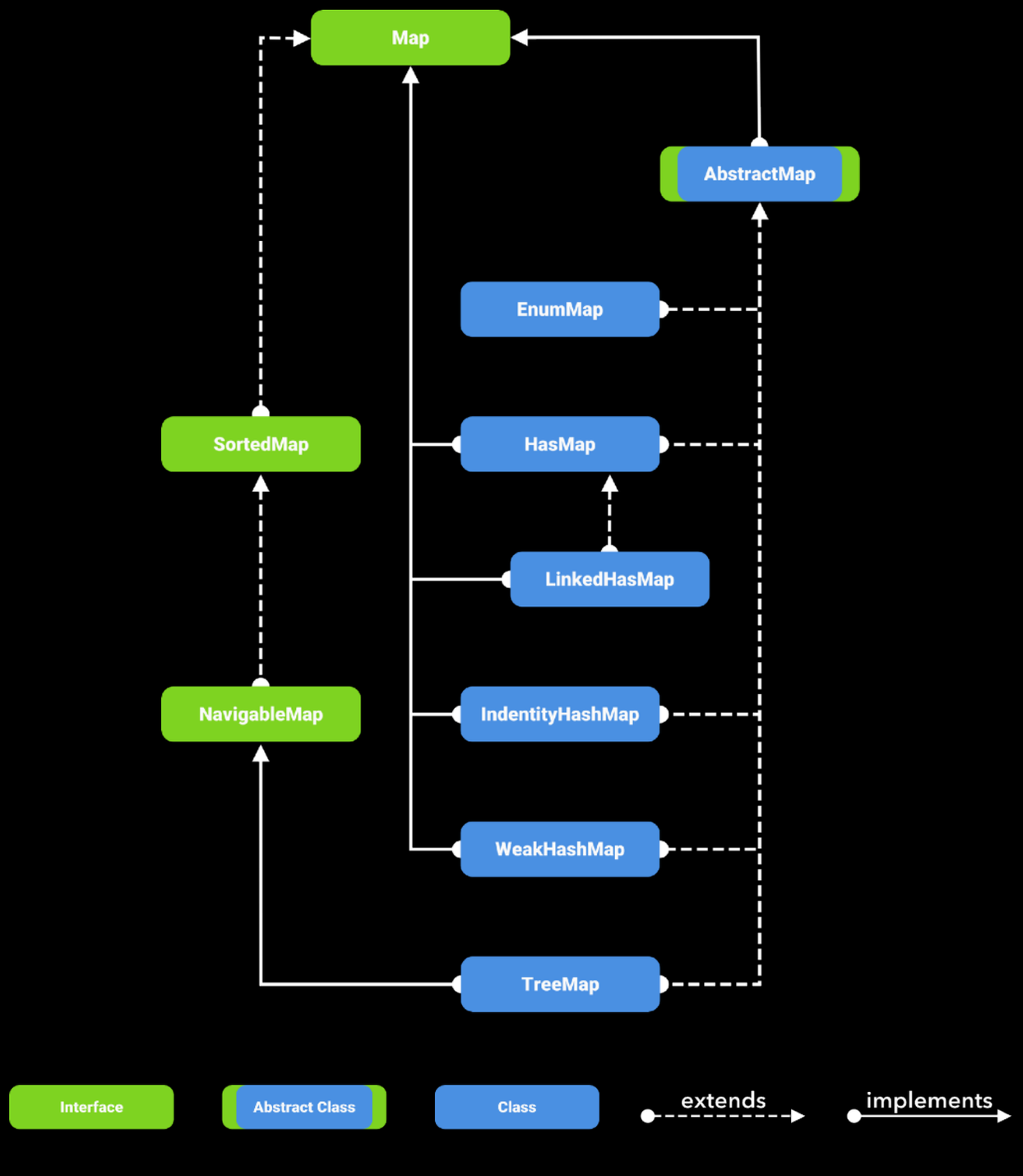
- **allOf(Class elementType):** Creates an enum set containing all of the elements in the specified element type.
- **complementOf(EnumSet e):** Creates an enum set with the same element type as the specified enum set, initially containing all the elements of this type that are not contained in the specified set.
- **noneOf(Class elementType):** Creates an empty enum set with the specified element type.
- **of(Enum e):** Creates an enum set initially containing the specified element.
- **of(Enum first, Enum... rest):** Creates an enum set initially containing the specified elements.
- **range(Enum from, Enum to):** Creates an enum set initially containing all of the elements in the range defined by the two specified endpoints.
- **add(Object o):** Adds the specified element to this set if it is not already present.
- **remove(Object o):** Removes the specified element from this set if it is present.
- **contains(Object o):** Returns true if this set contains the specified element.
- **size():** Returns the number of elements in this set (its cardinality).
- **clear():** Removes all of the elements from this set.
- **isEmpty():** Returns true if this set contains no elements.

# Java.util.

# ENUMSET

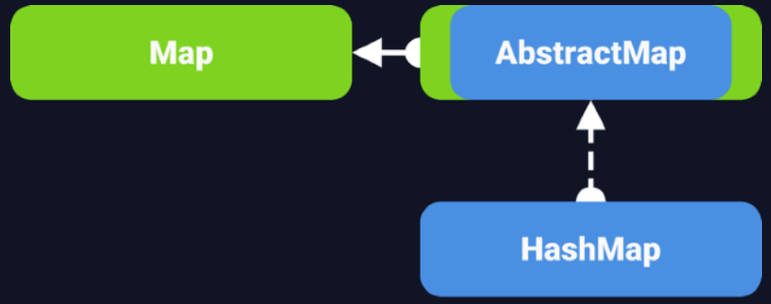


# java.util. MAP AND ITS HIERARCHY



Java.util.

# HASHMAP




## FREQUENTLY USED FUNCTIONS

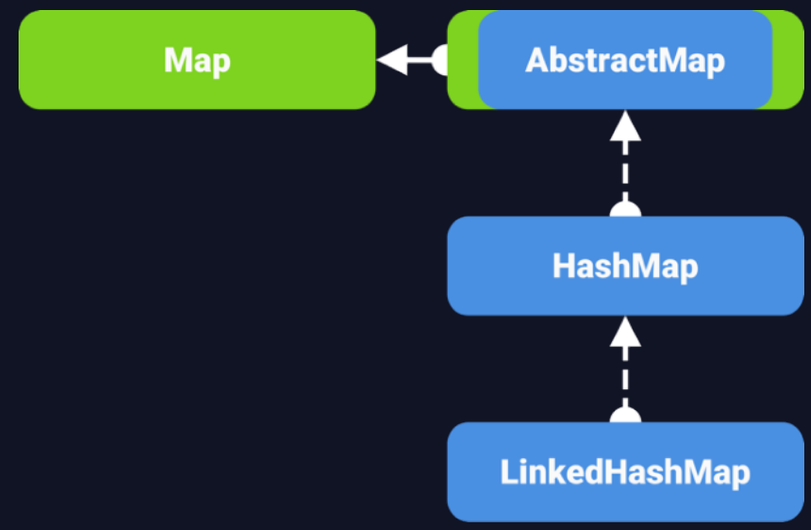
- **put(K key, V value):** Adds the specified value with the specified key in this map.
- **putIfAbsent(K key, V value):** If the specified key is not already associated with a value (or is mapped to null) associates it with the given value and returns null, else returns the current value.
- **remove(Object o):** Removes the mapping for the specified key from this map
- **remove(Object key, Object value):** Removes the entry for the specified key only if it is currently mapped to the specified value.
- **replace(K key, V oldValue, V newValue):** Replaces the entry for the specified key only if currently mapped to the specified value.
- **get(Object key):** Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- **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.
- **containsKey(Object key):** Returns true if this map contains a mapping for the specified key.
- **containsValue(Object value):** Returns true if this map maps one or more keys to the specified value.
- **size():** Returns the number of key-value mappings in this map.
- **clear():** Removes all of the mappings from this map.
- **isEmpty():** Returns true if this map contains no key-value mappings.
- **entrySet():** Returns a Set view of the mappings contained in this map.
- **keySet():** Returns a Set view of the keys contained in this map.
- **values():** Returns a Collection view of the values contained in this map.

## FREQUENTLY USED FUNCTIONS

- **put(K key, V value):** Adds the specified value with the specified key in this map.
- **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.
- **remove(Object o):** Removes the mapping for the specified key from this map
- **remove(Object key, Object value):** Removes the entry for the specified key only if it is currently mapped to the specified value.
- **replace(K key, V oldValue, V newValue):** Replaces the entry for the specified key only if currently mapped to the specified value.
- **get(Object key):** Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- **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.
- **containsKey(Object key):** Returns true if this map contains a mapping for the specified key.
- **containsValue(Object value):** Returns true if this map maps one or more keys to the specified value.
- **size():** Returns the number of key-value mappings in this map.
- **clear():** Removes all of the mappings from this map.
- **isEmpty():** Returns true if this map contains no key-value mappings.
- **entrySet():** Returns a Set view of the mappings contained in this map.
- **keySet():** Returns a Set view of the keys contained in this map.
- **values():** Returns a Collection view of the values contained in this map.

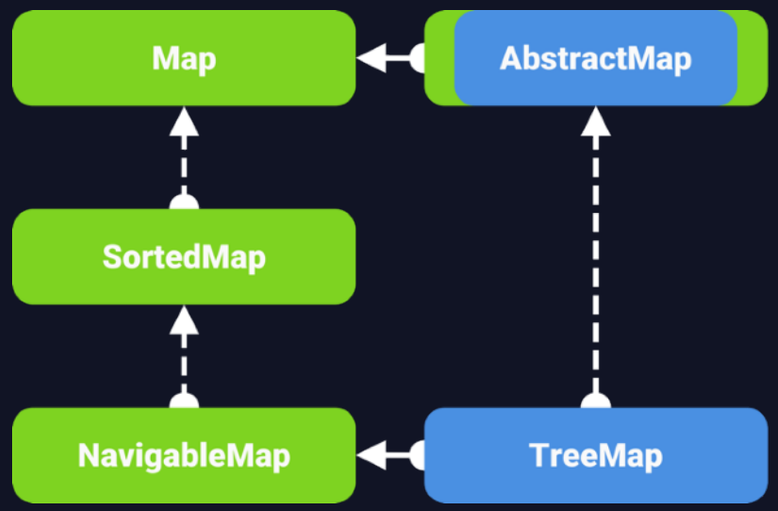
**Java.util.**

# LINKEDHASHMAP



Java.util.

# TREEMAP




## FREQUENTLY USED FUNCTIONS

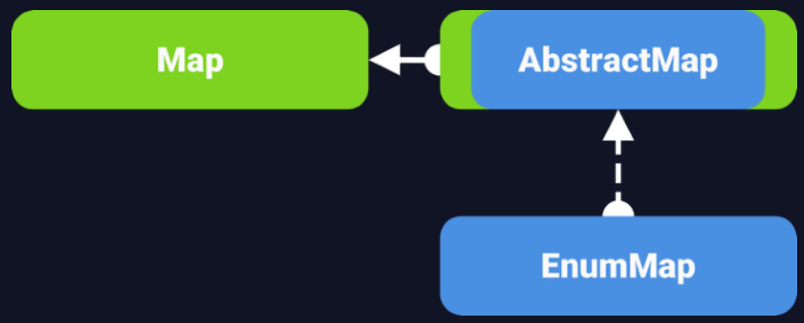
- **put(K key, V value):** Adds the specified value with the specified key in this map.
- **remove(Object o):** Removes the mapping for this key from this map if present.
- **get(Object key):** Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- **containsKey(Object key):** Returns true if this map contains a mapping for the specified key.
- **containsValue(Object value):** Returns true if this map maps one or more keys to the specified value.
- **size():** Returns the number of key-value mappings in this map.
- **clear():** Removes all of the mappings from this map.
- **entrySet():** Returns a Set view of the mappings contained in this map.
- **keySet():** Returns a Set view of the keys contained in this map.
- **values():** Returns a Collection view of the values contained in this map.
- **ceilingEntry(Object key):** Returns a key-value mapping associated with the least key greater than or equal to the given key, or null if there is no such key.
- **floorEntry(Object key):** Returns a key-value mapping associated with the greatest key less than or equal to the given key, or null if there is no such key.
- **firstEntry():** Returns a key-value mapping associated with the least key in this map, or null if the map is empty.
- **lastEntry():** Returns a key-value mapping associated with the greatest key in this map, or null if the map is empty.
- **descendingMap():** Returns a reverse order view of the mappings.
- **comparator():** Returns the comparator used to order the keys in this map, or null if this map uses the natural ordering of its keys.

## FREQUENTLY USED FUNCTIONS

- **put(K key, V value):** Associates the specified value with the specified key in this map.
- **putAll(Map m):** Copies all of the mappings from the specified map to this map.
- **remove(Object o):** Removes the mapping for a key from this map if it is present.
- **get(Object key):** Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- **containsKey(Object key):** Returns true if this map contains a mapping for the specified key.
- **containsValue(Object value):** Returns true if this map maps one or more keys to the specified value.
- **size():** Returns the number of key-value mappings in this map.
- **clear():** Removes all of the mappings from this map.
- **isEmpty():** Returns true if this map contains no key-value mappings.
- **entrySet():** Returns a Set view of the mappings contained in this map.
- **keySet():** Returns a Set view of the keys contained in this map.
- **values():** Returns a Collection view of the values contained in this map.

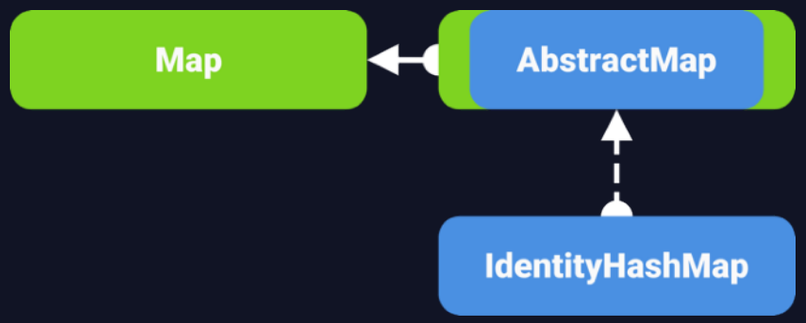
**Java.util.**

# ENUMMAP



Java.util.

# INDETITYHASHMAP




## FREQUENTLY USED FUNCTIONS

- **put(K key, V value):** Adds the specified value with the specified key in this map.
- **putAll(Map m):** Copies all of the mappings from the specified map to this map.
- **remove(Object o):** Removes the mapping for this key from this map if present.
- **get(Object key):** Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- **containsKey(Object key):** Tests whether the specified object reference is a key in this identity hash map.
- **containsValue(Object value):** Tests whether the specified object reference is a value in this identity hash map.
- **size():** Returns the number of key-value mappings in this identity hash map.
- **clear():** Removes all of the mappings from this map.
- **isEmpty():** Returns true if this identity hash map contains no key-value mappings.
- **entrySet():** Returns a Set view of the mappings contained in this map.
- **keySet():** Returns an identity-based set view of the keys contained in this map.
- **values():** Returns a Collection view of the values contained in this map.

## FREQUENTLY USED FUNCTIONS

- **put(K key, V value):** Adds the specified value with the specified key in this map.
- **putAll(Map m):** Copies all of the mappings from the specified map to this map.
- **remove(Object o):** Removes the mapping for a key from this weak hash map if it is present.
- **get(Object key):** Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- **containsKey(Object key):** Returns true if this map contains a mapping for the specified key.
- **containsValue(Object value):** Returns true if this map maps one or more keys to the specified value.
- **size():** Returns the number of key-value mappings in this map.
- **clear():** Removes all of the mappings from this map.
- **isEmpty():** Returns true if this map contains no key-value mappings.
- **entrySet():** Returns a Set view of the mappings contained in this map.
- **keySet():** Returns a Set view of the keys contained in this map.
- **values():** Returns a Collection view of the values contained in this map.

**Java.util.**

# WEAKHASHMAP

# THANK YOU

*Created By*

**ANAND K PARMAR**

*Designed By*

**COLOREDFEATHER**

*Stay in touch*