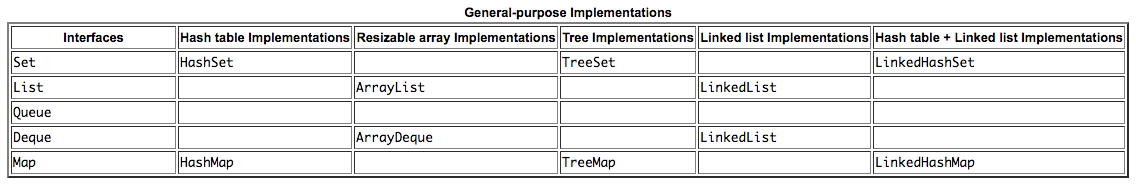
**Collection Framework**

# Interfaces

# 

# Implementations



# Algorithms

# 

\* Collection Interface

- basic operations

- int size()

- boolean isEmpty()

- boolean contains(Object element)

- boolean add(E element)

- boolean remove(Object element)

- Iterator<E> iterator()

\* Set

- A Collection that cannot contain duplicate elements

\* SortedSet

- ADD INFO HERRE

\* List

- An ordered Collection (sometimes called a sequence). Lists may contain duplicate elements.

Positional access

Search

Iteration

Range-view

\* Queue

- A collection designed for LIFO

Type of Operation Throws exception Returns special value

Insert add(e) offer(e)

Remove remove() poll()

Examine element() peek()

\* Deque (deck)

- A double-ended-queue, a linear collection

- Insertion and removal of elements at both end points

- Implements both stacks and queues at the same time

Type of First Element Last Element

Operation (Beginning of the Deque instance) (End of the Deque instance)

Insert addFirst(e) addLast(e)

offerFirst(e) offerLast(e)

Remove removeFirst() removeLast()

pollFirst() pollLast()

Examine getFirst() getLast()

peekFirst() peekLast()

Declaration

- Deque<Integer> stack = new ArrayDeque<Integer>();

\* Map

- A Map is an object that maps keys to values

- A map cannot contain duplicate keys

basic operations: put, get, remove, containsKey, containsValue, size, and empty

bulk operations: putAll and clear

collection views: keySet, entrySet, and values

\* SortedMap

- ADD INFO HERE

ArrayList An indexed sequence that grows and shrinks dynamically

LinkedList An ordered sequence that allows efficient insertions and removal at any location

ArrayDeque A double-ended queue that is implemented as a circular array

HashSet An unordered collection that rejects duplicates

TreeSet A sorted set

EnumSet A set of enumerated type values

LinkedHashSet A set that remembers the order in which elements were inserted

PriorityQueue A collection that allows efficient removal of the smallest element

HashMap A data structure that stores key/value associations

TreeMap A map in which the keys are sorted

EnumMap A map in which the keys belong to an enumerated type

LinkedHashMap A map that remembers the order in which entries were added

WeakHashMap A map with values that can be reclaimed by the garbage collector if they are not

used elsewhere

IdentityHashMap A map with keys that are compared by ==, not equals