Collections - Collections, Lists, Sets, Maps

A Collection is an arbitrary "bunch" of objects

- There may be duplicate objects in a collection
- Removing an object from a collection leaves behind any duplicates of the object

A List is an ordered collection of objects (sometimes called a sequence)

- The ordering of the objects is determined by the user of the list
- Stacks and Queues can be made from lists

A Set is a collection of objects without duplicates

- A duplicate object cannot be added to a set

A Map is an object that maps keys to values

- There cannot be duplicate keys in a map - the keys form a set
- A map can find the corresponding value for any of its keys or tell if an object does not match any of its keys
Collection

boolean add(Object o)
    Ensures that this collection contains the specified element (optional).

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.

boolean isEmpty()
    Returns true if this collection contains no elements.

Iterator iterator()
    Returns an iterator over the elements in this collection.

boolean remove(Object o)
    Removes a single instance of the specified element from this collection, if it is present (optional).

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

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

Object[] toArray(Object[] a)
    Returns an array containing all of the elements in this collection whose runtime type is that of the specified array.
List (extends Collection)

void add(int index, Object element)
    Inserts the specified element at the specified position in this list (optional).

Object get(int index)
    Returns the element at the specified position in this list.

int indexOf(Object o)
    Returns the index in this list of the first occurrence of the specified element, or -1 if this list does not contain this element.

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

int lastIndexOf(Object o)
    Returns the index in this list of the last occurrence of the specified element, or -1 if this list does not contain this element.

ListIterator listIterator()
    Returns a list iterator of the elements in this list (in proper sequence).

ListIterator listIterator(int index)
    Returns a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list.

Object remove(int index)
    Removes the element at the specified position in this list (optional).

Object set(int index, Object element)
    Replaces the element at the specified position in this list with the specified element (optional).

int size()
    Returns the number of elements in this list.
Set (extends Collection)

Same operations as Collection but prevents duplicates from appearing in the set
SortedSet (extends Set)

Comparator comparator()
Returns the comparator associated with this sorted set, or null if it uses its elements' natural ordering.

Object first()
Returns the first (lowest) element currently in this sorted set.

Object last()
Returns the last (highest) element currently in this sorted set.

SortedSet subSet(Object fromElement, Object toElement)
Returns a view of the portion of this sorted set whose elements range from fromElement, inclusive, to toElement, exclusive.
Comparator

    int compare(Object o1, Object o2)
    Compares its two arguments for order.

    boolean equals(Object obj)
    Indicates whether some other object is "equal to" this Comparator.
Iterator

boolean hasNext()
    Returns true if the iteration has more elements.

Object next()
    Returns the next element in the iteration.

void remove()
    Removes from the underlying collection the last element returned by the iterator (optional).
Map

void clear()
   Removes all mappings from this map (optional).

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 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.

Object get(Object key)
   Returns the value to which this map maps the specified key.

int hashCode()
   Returns the hash code value for this map.

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

Object put(Object key, Object value)
   Associates the specified value with the specified key in this map (optional).

Object remove(Object key)
   Removes the mapping for this key from this map if present (optional).

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

boolean equals(Object o)
   Compares the specified object with this entry for equality.

Object getKey()
   Returns the key corresponding to this entry.

Object getValue()
   Returns the value corresponding to this entry.

int hashCode()
   Returns the hash code value for this map entry.

Object setValue(Object value)
   Replaces the value corresponding to this entry with the specified value (optional).
Collection Classes

AbstractCollection implements Collection
  AbstractList extends AbstractCollection implements List
    AbstractSequentialList extends AbstractList
      LinkedList extends AbstractSequentialList implements List, Cloneable, Serializable
    ArrayList extends AbstractList implements List, Cloneable, Serializable
    Vector extends AbstractList implements List, Cloneable, Serializable
      Stack extends Vector
  AbstractSet extends AbstractCollection implements Set
    HashSet extends AbstractSet implements Set, Cloneable, Serializable
    TreeSet extends AbstractSet implements SortedSet, Cloneable, Serializable
AbstractMap implements Map
  HashMap extends AbstractMap implements Map, Cloneable, Serializable
  TreeMap extends AbstractMap implements SortedMap, Cloneable, Serializable
WeakHashMap extends AbstractMap implements.Map

Arrays
BitSet implements Cloneable, Serializable

Collections

Dictionary
  Hashtable extends Dictionary implements Map, Cloneable, Serializable
Collection Interfaces

Collection
public interface Collection

List
public interface List
extends Collection

Set
public interface Set
extends Collection

SortedSet
public interface SortedSet
extends Set

Comparator
public interface Comparator

Enumeration
public interface Enumeration

Iterator
public interface Iterator

ListIterator
public interface ListIterator
extends Iterator

Map
public interface Map

SortedMap
public interface SortedMap
extends Map

Map.Entry
public static interface Map.Entry
Properties of implementations

Hashtables ()
  O(1) average access times
  Unsorted

Trees
  O(log n) access times
  Sorted - ordering maintained at all times

Linked Lists
  O(1) access time to first/next/last element
  User-controlled ordering
  O(n) search times
  Can "splice" elements in the middle in O(1)

Arrays
  Random access
  Insertions in middle can be O(n)