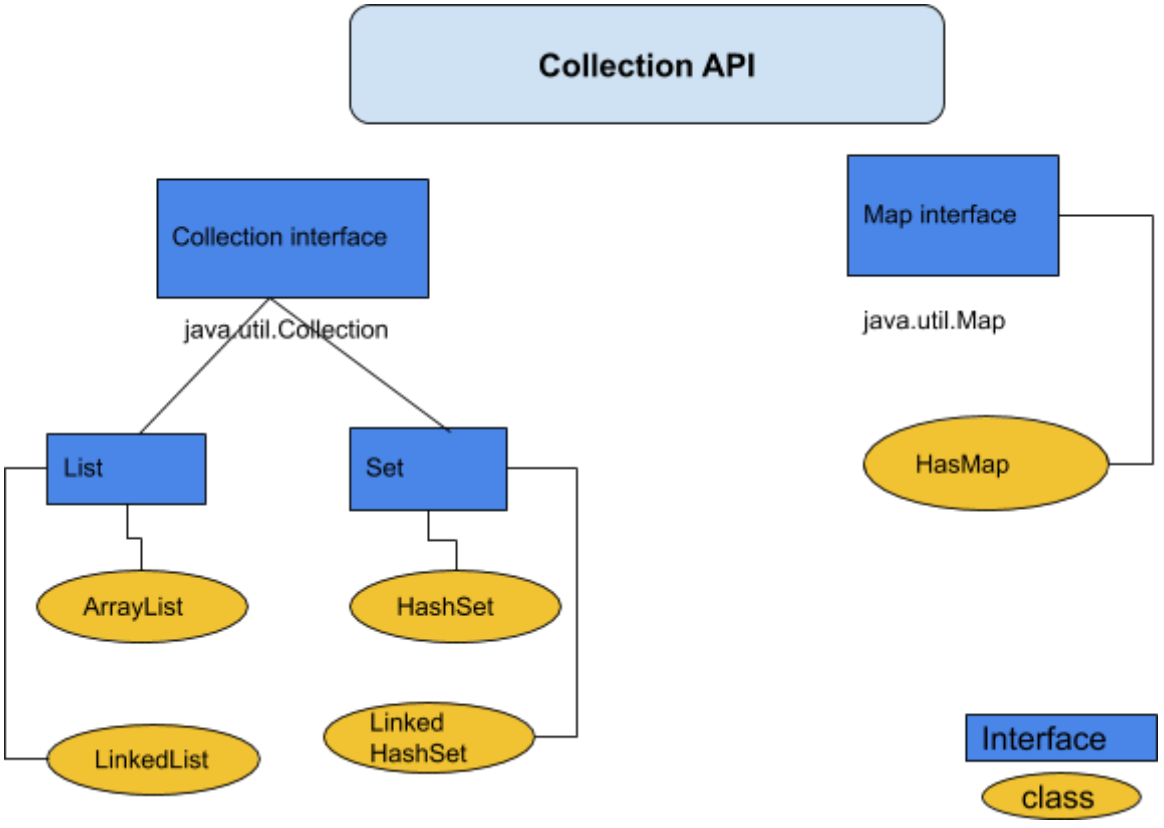
Collections in Java

- The Collection in Java is a framework that provides a facility to store and manipulate the group of objects.
- A Collection is a group of individual objects represented as a single unit
- Collection Framework is a set of classes and interfaces that implement commonly reusable collection data structures.
- It works in the manner of a library.
- The ‘java.util’ package contains all the classes and interfaces for the Collection framework.
- It provided methods to perform all type of operations on data such as searching, sorting, insertion, manipulation, and deletion.
- Java Collection framework provides many interfaces such as Set, List, etc. and classes such as ArrayList, etc.



Framework in Java

- It provides readymade architecture.
- It represents a set of classes and interfaces.
- It is optional.

Collection framework

The Collection framework represents a unified architecture for storing and manipulating a group of objects.
The various interfaces in the Java Collection Framework are:

Set

- Set Interface in Java is present in java.util package. It extends the Collection interface.
- It represents the unordered set of elements which doesn't allow us to store the duplicate items.
- We can store at most one null value in Set.
- Set is implemented by HashSet, LinkedHashSet, and TreeSet.

List

- List interface is the child interface of Collection interface.
- It inhibits a list type data structure in which we can store the ordered collection of objects.
- It can have duplicate values.
- List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

Methods of Collection interface

- add(Object o)
- addAll(Collection c)
- contains(Object o)
- isEmpty()
- remove(Object o)
- removeAll
- removeAll

Interfaces and its implementations, i.e., classes Algorithm

1. public boolean add(E e)
   It is used to insert an element in this collection.
2. public boolean addAll(Collection<? extends E> c)
   It is used to insert the specified collection elements in the invoking collection.

3. public boolean remove(Object element)
   It is used to delete an element from the collection.
4. public boolean removeAll(Collection<?> c)
   It is used to delete all the elements of the specified collection from the invoking collection.
5. default boolean remove(Predicate<? super E> filter)
   It is used to delete all the elements of the collection that satisfy the specified predicate.
6. public boolean retainAll(Collection<?> c)
   It is used to delete all the elements of invoking collection except the specified collection.
7. public int size()
   It returns the total number of elements in the collection.
8. public void clear()
   It removes the total number of elements from the collection.
9. public boolean contains(Object element)
   It is used to search for an element.
10. public boolean containsAll(Collection<?> c)
    It is used to search the specified collection in the collection.
11. public Iterator iterator()
    It returns an iterator.
12. public Object[] toArray()
    It converts the collection into an array.
13. public <T> T[] toArray(T[] a)
    It converts the collection into an array. Here, the runtime type of the returned array is that of the specified array.
14. public boolean isEmpty()
    It checks if the collection is empty.
15. default Stream parallelStream()
    It returns a possibly parallel Stream with the collection as its source.
16. default Stream stream()
    It returns a sequential Stream with the collection as its source.
17. default Spliterator spliterator()
    It generates a Spliterator over the specified elements in the collection.
18. public boolean equals(Object element)
    It matches two collections.
19. public int hashCode()
    It returns the hash code number of the collection.

ArrayList

- For storing elements it uses dynamic array.
- It implements List interface.
- ArrayList in java can contain duplicate value.
- Its size can is adjustable.
- It mains order in which values are inserted.

Methods of Java ArrayList

1. void add(int index, E element)
   It is used to insert the specified element at the specified position in a list.
2. boolean add(E e)
   It is used to append the specified element at the end of a list.
3. void clear()
   It is used to remove all of the elements from this list.
4. get(int index)
   It is used to fetch the element from the particular position of the list.
5. boolean isEmpty()
   It returns true if the list is empty, otherwise false.
6. boolean contains(Object o)
   It returns true if the list contains the specified element
7. int indexOf(Object o)
   It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
8. E remove(int index) It is used to remove the element present at the specified position in the list.
9. boolean remove(Object o) It is used to remove the first occurrence of the specified element.
10. boolean removeAll(Collection<?> c)
    It is used to remove all the elements from the list.
11. set(int index, E element)
    It is used to replace the specified element in the list, present at the specified position.
12. int size()
    It returns the number of elements present in the list.

LinkedList

- To store elements it uses a doubly linked list.
- It can contain duplicate elements.
- It maintains insertion order.
- ArrayList is slow and better for storing and accessing data while LinkedList is fast for manipulating data.

Methods of Java LinkedList

1. boolean add(E e)
   It is used to append the specified element to the end of a list.
2. void add(int index, E element)
   It is used to insert the specified element at the specified position index in a list.
3. void addFirst(E e)
   It is used to insert the given element at the beginning of a list.
4. void addLast(E e)
   It is used to append the given element to the end of a list.
5. void clear()
   It is used to remove all the elements from a list.
6. boolean contains(Object o)
   It is used to return true if a list contains a specified element.
7. get(int index)
   It is used to return the element at the specified position in a list.