JAVA PROGRAMMING

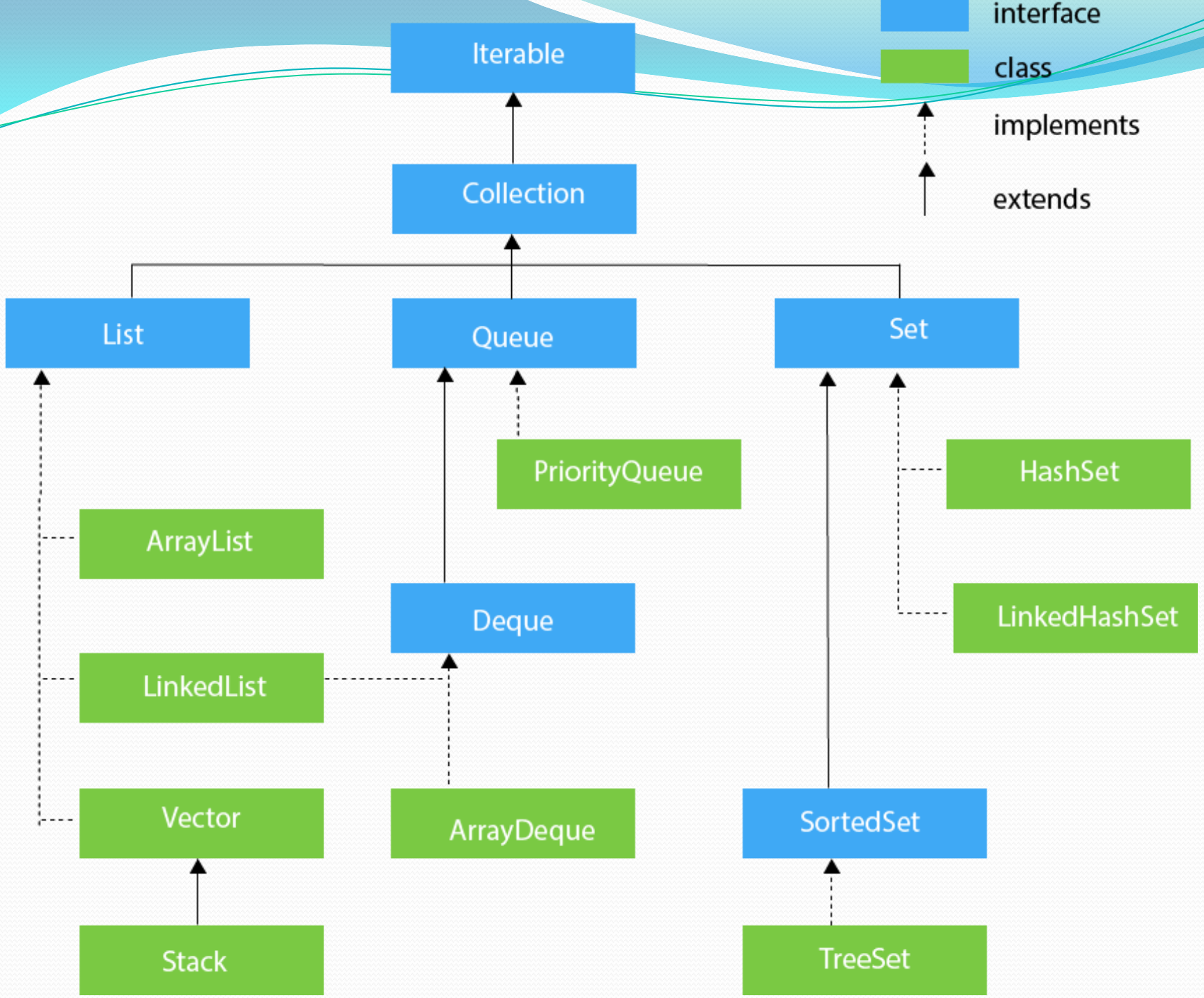
Chapter 3

Collection

Chaskar R. R.

Introduction

- The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.


interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet

List Interface

- 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.
- List <data-type> list1= **new** ArrayList();
- List <data-type> list2 = **new** LinkedList();

ArrayList

The ArrayList class implements the List interface.

It uses a dynamic array to store the duplicate element of different data types.

The ArrayList class maintains the insertion order and is non-synchronized.

E.g.

```
ArrayList<String> list=new ArrayList<String>();//Creating arraylist

list.add("Ravi");//Adding object in arraylist

list.add("Vijay");
```

LinkedList

LinkedList implements the Collection interface.

It uses a doubly linked list internally to store the elements.

It can store the duplicate elements.

It maintains the insertion order and is not synchronized.

E.g.

```
LinkedList<String> al=new LinkedList<String>();

al.add("Ravi");

al.add("Vijay");
```

VECTOR

Vector uses a dynamic array to store the data elements.

It is similar to ArrayList.

However, It is synchronized and contains many methods that are not the part of Collection framework.

E.g.

```
Vector<String> v=new Vector<String>();

v.add("Ayush");

v.add("Amit");
```

Advantages

Vector is synchronized.

Vector contain many legacy method that are not part of the collection framework.

Interface

Iterator interface provides the facility of iterating the elements in a forward direction only.

METHOD OF COLLECTION INTERFACE

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.

public boolean removeAll(Collection<?> c)

It is used to delete all the elements of the specified collection from the invoking collection.

5. public int size()

It returns the total number of elements in the collection.

6. public void clear()

It removes total number of elements from collection.

set Interface

- 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.

1. Set<data-type> s1 = **new** HashSet<data-type>();
2. Set<data-type> s2 = **new** LinkedHashSet<data-type>();
3. Set<data-type> s3 = **new** TreeSet<data-type>();

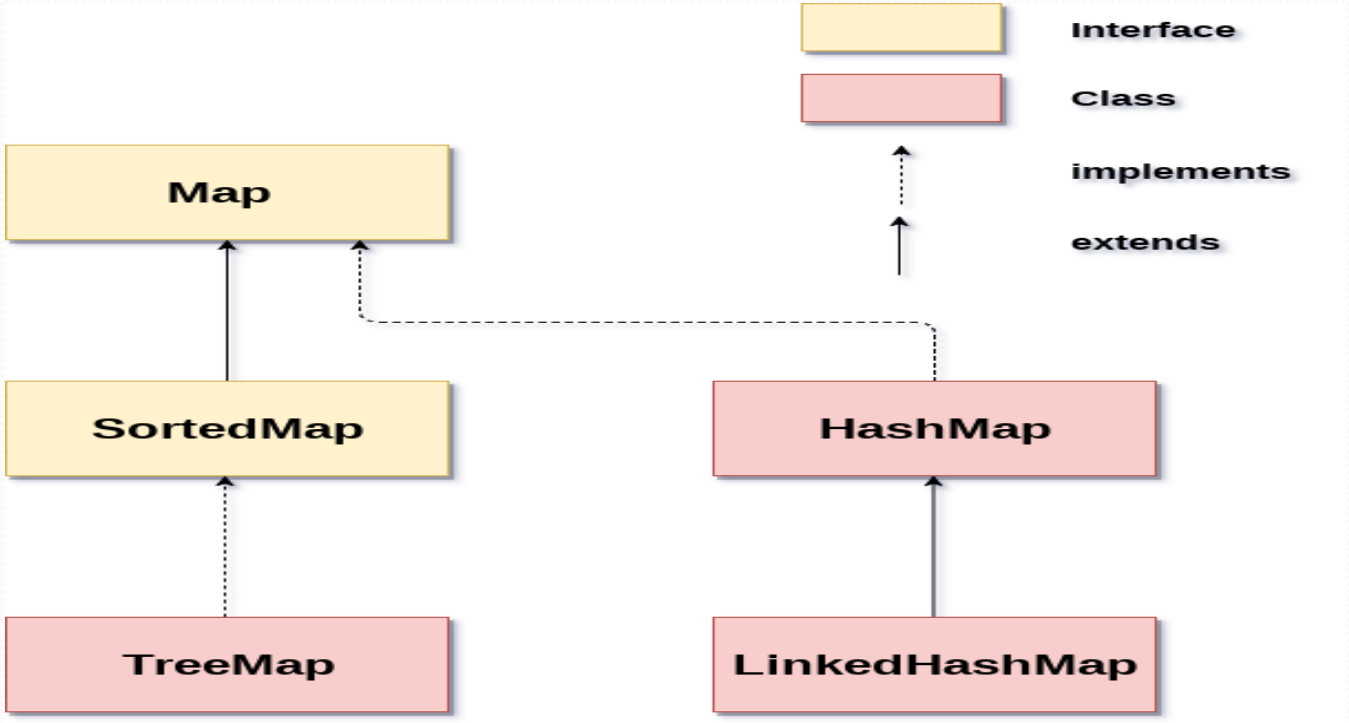
Working with Maps

A map contains values on the basis of key, i.e. key and value pair.

Each key and value pair is known as an entry.

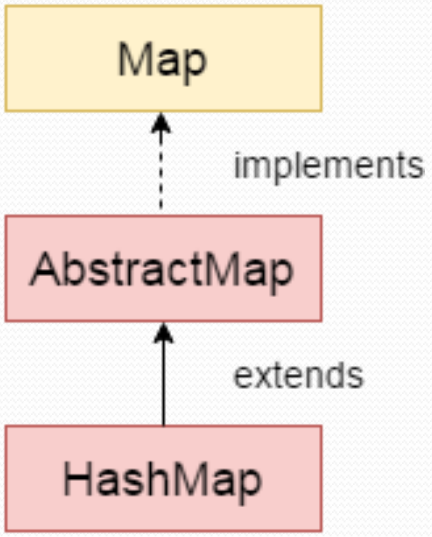
A Map contains unique keys.

A Map is useful if you have to search, update or delete elements on the basis of a key.



Java HashMap

- Java **HashMap** class implements the Map interface which allows us *to store key and value pair*, where keys should be unique.
- If you try to insert the duplicate key, it will replace the element of the corresponding key.
- It is easy to perform operations using the key index like updation, deletion, etc.
- HashMap class is found in the java.util package.



Constructor

1.HashMap() :- It is used to construct a default HashMap.

2. HashMap(int capacity) :- It is used to initializes the capacity of the hash map to the given integer value, capacity.

3. HashMap(int capacity, float loadFactor) :- It is used to initialize both the capacity and load factor of the hash map by using its arguments

Methods

- **void clear**():- It is used to remove all of the mappings from this map.
- **boolean isEmpty**():- It is used to return true if this map contains no ke:- y-value mappings.
- **Object clone**():- It is used to return a shallow copy of this HashMap instance: the keys and values themselves are not cloned.
- **Set entrySet**():- It is used to return a collection view of the mappings contained in this map.
- **Set keySet**():- It is used to return a set view of the keys contained in this map.

Java TreeMap class

- Java TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
- Java TreeMap contains only unique elements.
- Java TreeMap cannot have a null key but can have multiple null values.
- Java TreeMap is non synchronized.
- Java TreeMap maintains ascending order.
- **public class** TreeMap<K,V> **extends** AbstractMap<K,V> **implements** NavigableMap<K,V>, Cloneable, Serializable

Constructor

1. TreeMap()

It is used to construct an empty tree map that will be sorted using the natural order of its key.

1. TreeMap(Comparator<? super K> comparator)

It is used to construct an empty tree-based map that will be sorted using the comparator comp.

Methods

- void clear():- It removes all the key-value pairs from a map.
- Object clone():- It returns a shallow copy of TreeMap instance.
- K firstKey():- It is used to return the first (lowest) key currently in this sorted map.
- K lastKey():- It is used to return the last (highest) key currently in the sorted map.
- V remove(Object key):- It removes the key-value pair of the specified key from the map.