Map

Map is an interface which stores the objects as a key value pair, it is present in java.util package.
Each key,value pair is called as entry.
Whenever we want to represents objects as key value pair we go for maps.
Implementing classes of map is

1. Hashtable
2. HashMap
3. LinkdHashMap
4. TreeMap

Characteristics
A map can cannot contain duplicate keys.
Map can contain duplicate values.
Each key can contain atmost one value, and not more than one.

Methods :-
put(k,v)
putAll(Map)
get(key)
remove(key)
keySet()
values()
entrySet()

Hashtable :-
HashTable is a class which implements Map Interface.

Characteristics :-
Hashtable introduced in JDK 1.0v (legacy class)
It allows heterogeneous keys and values.
Default size is 11.
duplicate keys are not allowed but values can be duplicate.
Hashtable it self a D.S.
Random order based on hashcode.
Null values are not allowed as key as well as value.
These are thread safe and syncronised.
It has 4 constructors.

ex:-

```
public class CollFrWork {
	public static void main(String[] args) {
		Hashtable h=new Hashtable();
		h.put("1", "mango");
```

```
                h.put("1", "pomegranate");
                h.putIfAbsent("2", "kiwi");
                h.put("3", "banana");
                h.put(2, 3);
                h.put(9, 3);
                h.put("5", "strawberry");
                h.put(null, 69); //CTE
                h.put(null, null); //CTE
                System.out.println(h);
        }
}
```

HashMap :-

HashMap is a class which implements Map interface.

Characteristics

introduced in 1.2v.
heterogenous data allowed.
Data structure is Hashtable.
Default size is 16.
duplicate keys are not allowed but values can be duplicate.
random order based on hashcode.
only one null key is allowed and multiple null values are allowed.
These are not thread safe and non syncronised.
It has 4 constructors.

ex:-

```
public class MapProg {
        public static void main(String[] args) {
                HashMap<Integer,String> h=new HashMap();
                h.put(1, "Redmi");
                h.put(1, "Iphone");
                h.putIfAbsent(2, "Oppo");
                h.put(3, "OnePlus");
                h.put(null, null);
                h.put(null, null);
                System.out.println(h);
                Set<Integer> set = h.keySet();
                for(Integer s:set) {
                        System.out.println(s);
                }
                Collection<String> val = h.values();
                for(String v:val) {
                        System.out.println(v);
                }
                for(Entry<Integer, String> es:h.entrySet()) {
```

```
                System.out.println(es);
            }
        }
}
```

ex2:-

```
public class CollFrWork {
    public static void main(String[] args) {
        HashMap<Integer,String> h=new HashMap();
        h.put(1, "Redmi");
        h.put(1, "Iphone");
        h.putIfAbsent(2, "Oppo");
        h.put(3, "OnePlus");
        h.put(null, null);
        h.put(null, null);
        System.out.println(h);

        System.out.println("Printing key set using iterator");
        Set<Integer> set = h.keySet();
        Iterator i=set.iterator();
        while(i.hasNext()) {
            System.out.println(i.next());
        }

        System.out.println("Printing values using iterator");
        Collection<String> val = h.values();
        Iterator i1=val.iterator();
        while(i1.hasNext()) {
            System.out.println(i1.next());
        }

        System.out.println("Printing entry set using Iterator");
        Set<Entry<Integer, String>> es = h.entrySet();
        Iterator<Entry<Integer, String>> i2 = es.iterator();
        while(i2.hasNext()) {
            System.out.println(i2.next());
        }

        System.out.println("Printing entry set using for each loop");
        for(Entry<Integer, String> eset:h.entrySet()) {
            System.out.println(eset.getKey()+"==="+eset.getValue());
        }
    }
}
```

LinkedHashMap :-

LinkedHashMap is a class which extends HashMap and implements Map interface.

Characteristics

Introduced in 1.4 v

Heterogeneous data is allowed

DataStructure is HashTable and LinkedList.

Default size is 16.

Duplicate keys are not allowed but values can be duplicate

LinkedHashMap is a class which extends HashMap and implements Map interface.

We will get output as per insertion order.

It has 5 constructors.

ex:-

```
public class MapProg {
	public static void main(String[] args) {
		LinkedHashMap<Integer,String> h=new LinkedHashMap();
		h.put(1, "mango");
		h.put(1, "pomegranate");
		h.putIfAbsent(2, "kiwi");
		h.put(3, "banana");
		h.put(null, null);
		h.put(5, null);
		System.out.println(h);
		Set<Integer> set = h.keySet();
		for(Integer s:set) {
			System.out.println(s);
		}
		Collection<String> val = h.values();
		for(String v:val) {
			System.out.println(v);
		}
		for(Entry<Integer, String> es:h.entrySet()) {
			System.out.println(es);
		}
	}
}
```

ex2 :-

```
public class MapProg {
	public static void main(String[] args) {
		LinkedHashMap<Integer,String> h=new LinkedHashMap();
		h.put(1, "Redmi");
		h.put(1, "Iphone");
		h.putIfAbsent(2, "Oppo");
		h.put(3, "OnePlus");
		h.put(null, null);
		h.put(null, null);
```

```
                System.out.println(h);
                Set<Integer> set = h.keySet();
                System.out.println("Iterating over iterator");
                Iterator i=set.iterator();
                while(i.hasNext()) {
                        System.out.println(i.next());
                }
                Collection<String> val = h.values();
                Iterator i1=val.iterator();
                while(i1.hasNext()) {
                        System.out.println(i1.next());
                }
                for(Entry<Integer, String> es:h.entrySet()) {
                        System.out.println(es.getKey()+"==="+es.getValue());
                }
        }
}
```

TreeMap :-

TreeMap is a class which implements Map Interface.

Characteristics :-

Introduced in1.2v.

Only homogeneous data is allowed, if heterogeneous data is added then we will get classcast exception.

DataStructure is red-black tree.

Default size is 16.

Duplicate keys are not allowed but duplicate values are allowed.

Gives output in sorted order.

Null keys are not allowed.

It has 4 constructors.

Methods :-

lastKey() :- it gives max key

firstKey() :- it gives min key

headmap() :- it gives keys lessthan the key we mentioned

tailMap() :- it gives keys greaterthan the key we mentioned

descendingKeySet() :- Gives keys in descending order

descendingMap() :-Gives entry in descending order

submap() :- gives portion of collection.

ex:-

```
public class MapProg {
        public static void main(String[] args) {
                TreeMap<String,String> h=new TreeMap();
                h.put("1", "mango");
                h.put("1", "pomegranate");
```

```
		h.putIfAbsent("2", "kiwi");
		h.put("3", "banana");
		h.put("6", null);
		h.put("5", "strawberry");
		System.out.println(h);
		Set<String> set = h.keySet();
		for(String s:set) {
			System.out.println(s);
		}
		Collection<String> val = h.values();
		for(String v:val) {
			System.out.println(v);
		}
		for(Entry es:h.entrySet()) {
			System.out.println(es);
		}
	}
}
```

ex2:-

```
public class CollFrWork {
	public static void main(String[] args) {
		TreeMap<String,String> h=new TreeMap();
		h.put("1", "mango");
		h.put("1", "pomegranate");
		h.putIfAbsent("2", "kiwi");
		h.put("3", "banana");
		h.put("2", null);
		h.put("5", "strawberry");
		System.out.println(h);

		System.out.println("Printing values using Iterator");
		Collection<String> val = h.values();
		Iterator<String> vi = val.iterator();
		while(vi.hasNext()) {
			System.out.println(vi.next());
		}

		System.out.println("Printing keys using Iterator");
		 Set<String> kset = h.keySet();
		Iterator<String> ksi = kset.iterator();
		while(ksi.hasNext()) {
			System.out.println(ksi.next());
		}

		System.out.println("Printing entry set using Iterator");
		Collection<Entry<String, String>> eset = h.entrySet();
```

```
		Iterator<Entry<String, String>> eseti = eset.iterator();
		while(eseti.hasNext()) {
			System.out.println(eseti.next());
		}
	}
}

ex3:-
public class CollFrWork {
	public static void main(String[] args) {
		TreeMap<String,String> h=new TreeMap();
		h.put("1", "mango");
		h.put("1", "pomegranate");
		h.putIfAbsent("2", "kiwi");
		h.put("3", "banana");
		h.put("2", null);
		h.put("5", "strawberry");
		System.out.println(h);

		System.out.println(h.lastKey());
		System.out.println(h.firstKey());

		SortedMap<String, String> lessThan3 = h.headMap("3");
		System.out.println(lessThan3); //output in key value pair

		SortedMap<String, String> greThan3 = h.tailMap("3");
		System.out.println(greThan3); //output in key value pair

		Collection<String> valueslessThan3 = h.headMap("3").values();
		System.out.println(valueslessThan3); //gives values as output

		Collection<String> valuesgrthanThan3 = h.tailMap("3").values();
		System.out.println(valueslessThan3); //gives values as output

		Set<String> keysgreThan3 = h.headMap("3").keySet();
		System.out.println(keysgreThan3);

		Set<String> keyslessThan3 = h.tailMap("3").keySet();
		System.out.println(keyslessThan3);

		System.out.println(h.descendingKeySet()); //gives keys in descending order
		System.out.println(h.descendingMap()); //gives entry in descending order

		SortedMap<String, String> subM = h.subMap("2", "4"); //gives portion of
collection
		System.out.println(subM);
	}	}
```

Hashtable	HashMap
Introduced in 1.0V	Introduced in 1.2v
Heterogeneous data is allowed	Heterogeneous data is allowed
Default size is 11	Default size is 16
Hashtable is threadsafe and synchronised.	Hashtable is not threadsafe and non synchronised.
Hashtable is slow due to added synchronisation	HashMap is fast.
Null keys or values are not allowed.	only one null key is allowed and multiple null values are allowed.
duplicate keys are not allowed but values can be duplicate.	duplicate keys are not allowed but values can be duplicate.
It has 4 constructors	It has 4 constructors