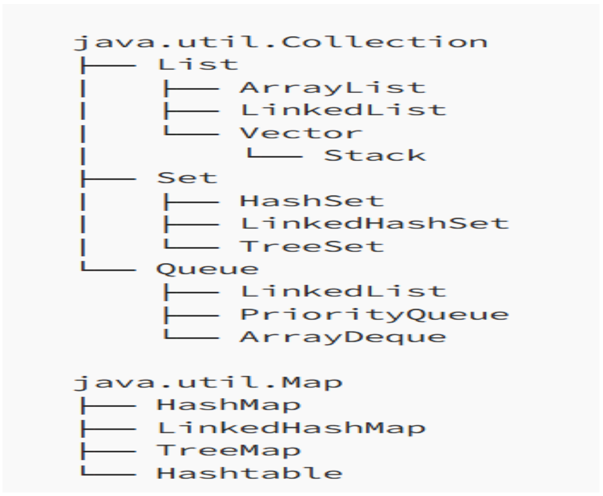
Unit-4



```
java.util.Collection
├── List
│     ├── ArrayList
│     ├── LinkedList
│     └── Vector
│           └── Stack
├── Set
│     ├── HashSet
│     ├── LinkedHashSet
│     └── TreeSet
└── Queue
      ├── LinkedList
      ├── PriorityQueue
      └── ArrayDeque

java.util.Map
├── HashMap
├── LinkedHashMap
├── TreeMap
└── Hashtable
```

1. **HashMap**:

- Does not maintain order.
- Allows null keys and values.
- Not thread-safe.

2. **LinkedHashMap**:

- Maintains insertion order.
- Allows null keys and values.
- Not thread-safe.

3. **TreeMap**:

- Maintains natural order or custom order defined by a Comparator.
- Does not allow null keys.
- Not thread-safe.

4. **Hashtable**:

- Does not allow null keys or values.
- Synchronized, hence thread-safe, but slower.

5. **ConcurrentHashMap**:

- Does not allow null keys or values.
- Thread-safe, designed for concurrent access.

Ex 1 :List Example: LinkedList

```java
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

public class LinkedListExample {
    public static void main(String[] args) {
        List<String> linkedList = new LinkedList<>();
        linkedList.add("Apple");
        linkedList.add("Banana");
        linkedList.add("Cherry");

        // Enhanced for loop
        for (String fruit : linkedList) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = linkedList.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Ex 2: List Example: Vector

```
import java.util.Iterator;
import java.util.List;
import java.util.Vector;

public class VectorExample {
    public static void main(String[] args) {
        List<String> vector = new Vector<>();
        vector.add("Apple");
        vector.add("Banana");
        vector.add("Cherry");

        // Enhanced for loop
        for (String fruit : vector) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = vector.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

List Example: Stack

```java
import java.util.Iterator;
import java.util.Stack;

public class StackExample {
    public static void main(String[] args) {
        Stack<String> stack = new Stack<>();
        stack.push("Apple");
        stack.push("Banana");
        stack.push("Cherry");

        // Enhanced for loop
        for (String fruit : stack) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = stack.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Set Example: HashSet

```java
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class HashSetExample {
    public static void main(String[] args) {
        Set<String> hashSet = new HashSet<>();
        hashSet.add("Apple");
        hashSet.add("Banana");
        hashSet.add("Cherry");

        // Enhanced for loop
        for (String fruit : hashSet) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = hashSet.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Set Example: LinkedHashSet

```
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.Set;

public class LinkedHashSetExample {
    public static void main(String[] args) {
        Set<String> linkedHashSet = new LinkedHashSet<>();
        linkedHashSet.add("Apple");
        linkedHashSet.add("Banana");
        linkedHashSet.add("Cherry");

        // Enhanced for loop
        for (String fruit : linkedHashSet) {
            System.out.println(fruit);
        }
        // Iterator
        Iterator<String> iterator = linkedHashSet.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Set Example: TreeSet

```
import java.util.Iterator;
import java.util.Set;
import java.util.TreeSet;

public class TreeSetExample {
    public static void main(String[] args) {
        Set<String> treeSet = new TreeSet<>();
        treeSet.add("Apple");
        treeSet.add("Banana");
        treeSet.add("Cherry");

        // Enhanced for loop
        for (String fruit : treeSet) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = treeSet.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Queue Example: LinkedList

```java
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Queue;

public class LinkedListQueueExample {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<>();
        queue.add("Apple");
        queue.add("Banana");
        queue.add("Cherry");

        // Enhanced for loop
        for (String fruit : queue) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = queue.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Queue Example: PriorityQueue

```java
import java.util.Iterator;
import java.util.PriorityQueue;
import java.util.Queue;

public class PriorityQueueExample {
    public static void main(String[] args) {
        Queue<String> priorityQueue = new PriorityQueue<>();
        priorityQueue.add("Apple");
        priorityQueue.add("Banana");
        priorityQueue.add("Cherry");

        // Enhanced for loop
        for (String fruit : priorityQueue) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = priorityQueue.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Deque Example: ArrayDeque

```
import java.util.ArrayDeque;
import java.util.Deque;
import java.util.Iterator;

public class ArrayDequeExample {
    public static void main(String[] args) {
        Deque<String> deque = new ArrayDeque<>();
        deque.add("Apple");
        deque.add("Banana");
        deque.add("Cherry");

        // Enhanced for loop
        for (String fruit : deque) {
            System.out.println(fruit);
        }

        // Iterator
        Iterator<String> iterator = deque.iterator();
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

Code Examples of java.util.Map package

In Java, a Map is a data structure that is used to store key-value pairs. Understanding how to iterate over the elements of a map plays a very important role. There are 5 ways to iterate over the elements of a map,

Note: We cannot iterate over the elements of a map directly with the help of iterators because a map is not a collection.

We use different views like entrySet(), keySet(), or other utility methods to iterate.

Map Example: HashMap

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

public class HashMapExample {
   public static void main(String[] args)


{
       Map<Integer, String> hashMap = new HashMap<>();
       hashMap.put(1, "Apple");
       hashMap.put(2, "Banana");
       hashMap.put(3, "Cherry");

       // Iterating using enhanced for loop
       for (Map.Entry<Integer, String> entry :
hashMap.entrySet()) {
           System.out.println("Key: " + entry.getKey() + ", Value:
" + entry.getValue());
       }

       // Iterating using iterator
       Iterator<Map.Entry<Integer, String>> iterator =
hashMap.entrySet().iterator();
       while (iterator.hasNext()) {
           Map.Entry<Integer, String> entry = iterator.next();
           System.out.println("Key: " + entry.getKey() + ", Value:
" + entry.getValue());
```

```
        }
    }
}
```

Map Example: LinkedHashMap

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;

public class LinkedHashMapExample {
    public static void main(String[] args) {
        Map<Integer, String> linkedHashMap = new
LinkedHashMap<>();
        linkedHashMap.put(1, "Apple");
        linkedHashMap.put(2, "Banana");
        linkedHashMap.put(3, "Cherry");

        // Iterating using enhanced for loop
        for (Map.Entry<Integer, String> entry :
linkedHashMap.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value:
" + entry.getValue());
        }

        // Iterating using iterator
        Iterator<Map.Entry<Integer, String>> iterator =
linkedHashMap.entrySet().iterator();
        while (iterator.hasNext()) {
            Map.Entry<Integer, String> entry = iterator.next();
            System.out.println("Key: " + entry.getKey() + ", Value:
" + entry.getValue());
```

```
        }
    }
}
```

Map Example: TreeMap

```
import java.util.Iterator;
import java.util.Map;
import java.util.TreeMap;

public class TreeMapExample {
    public static void main(String[] args) {
        Map<Integer, String> treeMap = new TreeMap<>();
        treeMap.put(3, "Cherry");
        treeMap.put(1, "Apple");
        treeMap.put(2, "Banana");

        // Iterating using enhanced for loop
        for (Map.Entry<Integer, String> entry :
treeMap.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value:
" + entry.getValue());
        }

        // Iterating using iterator
        Iterator<Map.Entry<Integer, String>> iterator =
treeMap.entrySet().iterator();
        while (iterator.hasNext()) {
            Map.Entry<Integer, String> entry = iterator.next();
            System.out.println("Key: " + entry.getKey() + ", Value:
```

```
" + entry.getValue());
        }
    }
}
```

Map Example: HashTable

```
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Map;

public class HashtableExample {
    public static void main(String[] args) {
        Map<Integer, String> hashtable = new Hashtable<>();
        hashtable.put(1, "Apple");
        hashtable.put(2, "Banana");
        hashtable.put(3, "Cherry");

        // Iterating using enhanced for loop
        for (Map.Entry<Integer, String> entry :
hashtable.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value:
" + entry.getValue());
        }

        // Iterating using iterator
        Iterator<Map.Entry<Integer, String>> iterator =
hashtable.entrySet().iterator();
        while (iterator.hasNext()) {
            Map.Entry<Integer, String> entry = iterator.next();
            System.out.println("Key: " + entry.getKey() + ", Value: 
" + entry.getValue());
        }
    }
}
```

Or (using lamda expression and forEach())

```
import java.util.Hashtable;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Map<Integer, String> hashtable = new Hashtable<>();
        hashtable.put(1, "Apple");
        hashtable.put(2, "Banana");
        hashtable.put(3, "Cherry");

        hashtable.forEach((k,v)->{


         System.out.println(k+"\t"+v);


        });
    }
}
```

Ex: comparable

```
import java.util.ArrayList;
import java.util.Collections;
class Emp implements Comparable<Emp>
{
        int eid,sal;
        String ename;
        public Emp(int eid,String ename, int sal)
        {
                this.eid=eid;
                this.ename=ename;
                this.sal=sal;
        }


public int compareTo(Emp e)
{
        if(sal==e.sal)
                return 0;
        else if(sal>e.sal)
                return 1;
        else
                return -1;
}

Or
public int compareTo(Emp e)
{
        return ename.compareTo(e.name);
}
public class Main
{
    public static void main(String[] args)
        {
         ArrayList<Emp> p=new ArrayList<>();
         p.add(new Emp(1,"ravi",10000));
        p.add(new Emp(12,"rashi",5000));
```

```
		p.add(new Emp(13,"ravj",15000));

		p.add(new Emp(1,"raju",2000));
		Collections.sort(p);

		for (Emp e:p)
		{
			System.out.println(e.eid+"\t"+e.ename+"\t"+e.sal);
		}

		}
	}
```