Collections JAVA

scorcism	Abhishek Pathak
Status	

List

ArrayList

```
package Collection;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ArrayList1 {
    public static void main(String[] args){
        /* Like a normal array which has the property of changing size dynamically
        This is the part of List
        Time Complexity: O(n) - for remove and insert

        */

        ArrayList<Integer> array = new ArrayList<>();
        array.add(12);
        array.add(34);
        array.remove(0);
        System.out.println(array);
        System.out.println(array.contains(12));
        array.add(1,50);
        // Removing the element with value if it exists
        array.remove(Integer.valueOf(50));
        System.out.println(array);
        System.out.println(array.isEmpty());
        System.out.println(array.size());
        System.out.println(array.get(0));
        // array.clear(); // Used to remove all the elements form the array list

        // We can also add a new entire List into the old list as:
        List<Integer> list = new ArrayList<>();
       list.add(9);
       list.add(8);
       list.add(7);
        // Set function is used to update the existing element
        list.set(2,5);
        System.out.println(list.size());
        array.addAll(list);
        System.out.println(array);

        // Iterating through the loop
        for(int i = 0; i < list.size(); i++){
            System.out.println(list.get(i));
        }
```

```
        // Using iterator
        Iterator<Integer> it = list.iterator();

        // Jab tak koi next element hai tab tak print karna
        while(it.hasNext()){
            System.out.println("Iterator: " + it.next());
        }

    }
}
```

LinkedList

```
package Collection;

import java.util.LinkedList;
import java.util.Queue;

public class Queue1 {
    public static void main(String[] args) {
        // First in First out - FIFO

        Queue<Integer> queue = new LinkedList<>();

        // Adding elements to the queue
        queue.offer(12);
        queue.offer(2);
        queue.offer(24);
        queue.offer(21);
        System.out.println(queue);

        // Removing element
        queue.poll(); // After removing this also returns the element
        System.out.println(queue);

        // Peek the element jo hi poll hone wala hai
        System.out.println(queue.peek());
    }
}
```

Stack

```
package Collection;

import java.util.Stack;

public class Stack1 {
    public static void main(String[] args) {
        // This is the part of List
        // Last in First out type of structure LIFO
```

```
        Stack<String> name = new Stack<>();

        // For adding elements we use push
        name.push("Lion");
        name.push("Tiger");
        name.push("Dog");
        name.push("Cat");
        name.push("Zebra");
        System.out.println(name);

        // Checking which element is at the top of stack
        System.out.println(name.peek());

        // Removing last element
        name.pop();
        System.out.println(name);
    }
}
```

Queue

LinkedList

```
package Collection;

import java.util.LinkedList;
import java.util.Queue;

public class Queue1 {
    public static void main(String[] args) {
        // First in First out - FIFO

        Queue<Integer> queue = new LinkedList<>();

        // Adding elements to the queue
        queue.offer(12);
        queue.offer(2);
        queue.offer(24);
        queue.offer(21);
        System.out.println(queue);

        // Removing element
        queue.poll(); // After removing this also returns the element
        System.out.println(queue);

        // Peek the element jo hi poll hone wala hai
        System.out.println(queue.peek());
    }
}
```

PriorityQueue

```
package Collection;

import java.util.Queue;
import java.util.PriorityQueue;

public class PriorityQueue1 {
    public static void main(String[] args) {
        // Queue with priority
        Queue<Integer> pq = new PriorityQueue<>();

        // This arranges the array in priority base -> Here by default priority is
        // set to the lowest element

        pq.offer(23);
        pq.offer(12);
        pq.offer(2);
        pq.offer(67);
        System.out.println(pq);

        // Remove the priority element
        pq.poll();
        System.out.println(pq);
    }
}
```

ArrayDequeue

```
package Collection;

import java.util.ArrayDeque;

public class ArrayDeque1 {
    public static void main(String[] args) {
        ArrayDeque<Integer> dq = new ArrayDeque<>();

        // Insert and remove element form front and back
        // Insert element in front
        dq.offer(12);
        dq.offerFirst(4);
        dq.offerLast(56);
        dq.offer(3);
        System.out.println(dq);

        System.out.println(dq.peek());
        System.out.println(dq.peekFirst());
        System.out.println(dq.peekLast());

        System.out.println(dq.poll());

        System.out.println(dq.pollFirst());
        System.out.println("Poll First" + dq);

        System.out.println(dq.pollLast());
        System.out.println("Poll Last: " + dq);
```

```
    }
}
```

Set

HashSet

```
package Collection;

import java.util.HashSet;
import java.util.Set;

public class HashSet1 {
    public static void main(String[] args) {

        Set<Integer> set = new HashSet<>();
        // No duplicated elements allowed
        // Order is not defined
        // O(1)

        set.add(12);
        set.add(34);
        set.add(56);
        set.add(78);
        set.add(90);
        System.out.println(set);

        set.remove(1);
        System.out.println(set);

        System.out.println(set.contains(78));

        System.out.println(set.isEmpty());

        System.out.println(set.size());
    }
}
```

LinkedHashSet

```
package Collection;

import java.util.LinkedHashSet;
import java.util.Set;

public class LinkedHashSet1 {
    public static void main(String[] args) {
```

```
        Set<Integer> lhs = new LinkedHashSet<>();

        // With set property it also inherits the property of LinkedHashSet

        lhs.add(12);
        lhs.add(34);
        lhs.add(56);
        lhs.add(78);
        lhs.add(90);
        System.out.println(lhs);

        lhs.remove(1);
        System.out.println(lhs);

        System.out.println(lhs.contains(78));

        System.out.println(lhs.isEmpty());

        System.out.println(lhs.size());

    }
}
```

TreeSet

```
package Collection;

import java.util.Set;
import java.util.TreeSet;

public class TreeSet1 {
    public static void main(String[] args) {
        Set<Integer> set = new TreeSet<>();

        // With set, it also has the property of Tree - Binary Tree so, all the elements are in sorted format
        // In O(log n)

        set.add(12);
        set.add(34);
        set.add(56);
        set.add(78);
        set.add(90);
        System.out.println(set);

        set.remove(1);
        System.out.println(set);

        System.out.println(set.contains(78));

        System.out.println(set.isEmpty());

        System.out.println(set.size());
    }
}
```

Map

TreeMap

```
package Collection;

import java.util.HashMap;
import java.util.Map;
import java.util.TreeMap;

public class TreeMap1 {
    public static void main(String[] args) {
        Map<String, Integer> map = new TreeMap<>();
        // Yeh keys ko under the hood binary trees me dalta hai
        // Keys ko sort karte chalta hai
        // O(log n)

        map.put("Abhishek",18);
        map.put("Jash",19);
        map.put("Aman",22);
        map.put("Tanmay",20);
        System.out.println(map);

        // Aagr hogi toh remove ho jayiegi
        map.remove("Tanmay");
        System.out.println(map);
    }
}
```

HashMap

```
package Collection;

import java.util.HashMap;
import java.util.Map;

public class HashMap1 {
    public static void main(String[] args) {
        // Maps are based on the key : value pair
        // Key and value both can have different datatype
        // All the keys are unique
        // If the keys is present in the map it will override the old one
        // O(1)

        Map<String,Integer> hm  = new HashMap<>();

        hm.put("Abhishek",18);
        hm.put("Jash",19);
        hm.put("Yash",1);
        hm.put("Tanmay",20);
        System.out.println(hm);

        System.out.println(hm.containsValue(20));
```

```
        // Avoid the upgrade of key

        /*
        hm.putIfAbsent("Jash",19);
        if(!hm.containsKey("Jash")){
            hm.put("Jash",19);
        }
        */

        // Iterating through the map
        for(Map.Entry<String, Integer> e: hm.entrySet()){
            System.out.println(e.getKey());
            System.out.println(e.getValue());
        }

        // Getting the array of keys
        System.out.println(hm.keySet());

        // Getting the values
        System.out.println(hm.values());

    }
}
```

Class

CollectionClass

```
package Collection;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class CollectionClass1 {
    public static void main(String[] args) {
        // Function like min, max, frequency ...

        ArrayList<Integer> array = new ArrayList<>();
        array.add(12);
        array.add(34);
        array.add(78);

        System.out.println(Collections.min(array));
        System.out.println(Collections.max(array));
        System.out.println(Collections.frequency(array,12));

        // Sorting
        Collections.sort(array);
        System.out.println(array);

        // In reverse order
        Collections.sort(array, Comparator.reverseOrder());
        System.out.println(array);
```

```
    }
}
```

ArrayClass

```
package Collection;
import java.util.Arrays;

public class ArrayClass1 {
    public static void main(String[] args) {
        // Used for the manipulation of array - primitive array
        // Searching , indexing, remove, ... converting array to arrayList

        int[] array = {1,2,3,4,5,6,7,8,9};
        int[] array2 = {2,45,6,4,32,345,35,435,5,54,};
        int index =Arrays.binarySearch(array,9); // Only works in sorted array O(log n)
        System.out.println(index);

        // Sorting array
        Arrays.sort(array2); // This uses quick sort in backend
        for(int e: array2){
            System.out.print(e + " ");
        }

    }
}
```