(2)

12/04/24

ASSIGNMENT - 3

JAVA PROGRAMMING.

CSA0985

V. Aadharsh Vishal
192311009.
B.E-CSE

Collections framework in Java.util Package.

```
import Java.util.Arraylist;
    public class main{
        public static void main (String [] args) {
            Array list<String> obj = new Array list <> ();
            obj.add ("One");
            obj.add ("Two");
            obj.add ("Three");
    System.out.println ("Array list:" + obj);
}
}
```

ArrayList in Collecton

```
import java util. List;
import java.util.Arraylist;
class Main {
    public static void main (String [] args) {
    List <Integer> numbers = new ArrayList <> ();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
    System.out.println("List:" + numbers);
int get number = numbers.get(2);
System.out.println("Accessed Element:" + get Number);
```

int removeNumber = numbers.remove(1);
System.out.println("Removed element: " + removeNumber);

}

3.

Linked list:-

```
import java.util.List;
import java.util.LinkedList;
class Main {
    public static void main(String[] args) {
        List<Integer> numbers = new LinkedList<>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        System.out.println("List: " + numbers);
        int number = numbers.get(2);
        System.out.println("Accessed Element: " + number);
        int index = numbers.indexOf(2);
        System.out.println("Position of 2 is " + index);
        int removedNumber = numbers.remove(1);
        System.out.println("Removed element: " + removedNumber);
```

Vector:-

```
import java.util.Iterator;
import java.util.Vector;
class Main {
```

```
public static void main(String[] args) {
Vector<String> fruits = new Vector<>();
  fruits.add("Apple");
  fruits.add("Orang");
  fruits.add("mango");
  System.out.println("Vector:"+fruits);
String element = fruits.get(2);
System.out.println("Elemental at index 2:" + element);
fruits.add(3,"Banana");
System.out.println("Vector:"+ fruits);
Vector<String> indianfruits = new Vector<>();
Indianfruits.add("Pomogranate");
System.out.println("New Vector:"+Indianfruits);
Indianfruits.iterator();
System.out.println("Vector:");
while(iforate.hasNext()){
      System.out.println(iforate.next());
System.out.println(",");
  }
}
}
```

Collection in reverse

```
import.java.util.*;
import.java.util.Iterator;
import java.util.LinkedList;
import java.util.Collections;

class Main{
    public static Void main(String[] args){
    List<String> fruits = new LinkedList<>();
    fruits.add("Apple");
    fruits.add("Orange");
    fruits.add("Mango");
    System.out.println("Linked list:"+fruits);
    fruits.add(2,"Banana");
    System.out.println("Linked list:"+fruits);
    Collections.reverse(fruits);
    System.out.println("Fruits in reverse order:"+fruits);
    Collections.sort(fruits);
    System.out.println("Fruits is Ascending order:"+fruits);
    Collections.sort(fruits, Collections.reverseOrder());
    System.out.println("Fruits in Descending order:"+fruits);
    System.out.println("Fruits in the Basket:");
    for(int i=0; i<fruits.size(); i++){
    System.out.println(fruits.get(i));
}.
```

```
        System.out.println("Fruits in the Basket:");
        for(int i=fruits.size()-1; i>=0; i--){
            System.out.println(fruits.get(i));
        }
    }
}
```

Stack (Push, Pop, peek, empty).

```
import.java.lang.*;
import.java.util.Stack;
class Main{
    public static void main(String[] args){
        Stack<String> fruits = new Stack<>();
        fruits.push("Apple");
        fruits.push("Orange");
        fruits.push("Mango");
        System.out.println("~~Linked list~~ Stack:" + fruits);
        ~~fruits.add(2, "Banana");~~
        ~~System.out.println("Fruits in Re~~
        String remove = fruits.pop();
        System.out.println("Stack:" + remove);
        fruits.push("Pineapple");
        System.out.println("Stack:" + fruits);
        String display = fruits.peek();
        System.out.println("Stack:" + display);
```

```
int position = fruits.search("orange");
  System.out.println("Position of fruit:" + position);
boolean e = fruits.empty();
System.out.println("Is the Stack is Empty:" + e);
fruits.clear();
boolean e1 = fruits.empty();
System.out.println("Is the stack is Empty:" + e1);
  }
}
```

Queue:-

```
import java.lang.*;
import java.util.LinkedList;
import java.util.Queue;
class Main{
  public static void main(String[] args){
    Queue<String> fruits = new LinkedList<>();
    fruits.add("Apple");
    fruits.add("orange");
    fruits.add("Mango");
  System.out.print("Queue:" + fruits);
  String r = fruits.remove();
  System.out.println("Queue:" + r);
  System.out.println("Queue:" + fruits);
  String r = fruits.peek();
```

```
System.out.println("Stack: "+fruits);
boolean e = fruits.isEmpty();
System.out.println("Is the Queue is empty: "+e);
fruits.clear();
boolean e1 = fruits.isempty();
System.out.println("Is the Queue is Empty: "+e1);
}
}
```

Dequeue:-

```
import java.long.*;
import java.util.LinkedList;
import java.util.ArrayDeque;

class Main {
    public static void main(String[] args){
        ArrayDeque<String> fruits = new ArrayDeque();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.addFirst("Orange");
        fruits.addLast("Mango");
        System.out.println("Dequeue:"+fruits);
        String r = fruits.remove();
        System.out.println("Dequeue:"+r);
        System.out.println("Dequeue:"+fruits);
        String display = fruits.peek();
```

```
System.out.println("Dequeue: " + display);
boolean e = fruits.isEmpty();
System.out.println("Is the Dequeue is Empty: " + e);
fruits.clear();
System.out.println("Empty Dequeue: " + fruits);
boolean e1 = fruits.isEmpty();
System.out.println("Is the Dequeue is Empty: " + e1);
}
}
```

HashMap :-

```
import java.util.Map;
import java.util.HashMap;
class Main {
    public static void main(String[] args) {
        Map<Integer, String> fruits = new HashMap<>();
        fruits.put(1, "Apple");
        fruits.put(2, "Orange");
        fruits.put(3, "Mango");
        System.out.println("Map: " + fruits);
        System.out.println("Keys: " + fruits.keySet());
        System.out.println("Values: " + fruits.values());
        System.out.println("Entries: " + fruits.entrySet());
        boolean value = fruits.remove(2, "Orange");
        System.out.println("Removed Value: " + value);
        System.out.println("New Map: " + fruits);
```

```
boolean Value1 = Fruits.containsKey(1);
System.out.println("Available in the Basket:", Value1);
Fruits.replace(Key:3, old Value: "Mango", New Value: "Banana");
System.out.println("After replacing the item:" + Fruits);
}
}
```

O/P: Map: {1=Apple, 2=Orange, 3=Mango}.

Keys: [1, 2, 3]

Values: [Apple, Orange, Mango]

Entries: [1=Apple, 2=Orange, 3=Mango]

Removed value: True

New Map: {1=Apple, 3=Mango}

Available in the Basket: False

After replacing the item: {1=Apple, 3=Banana}.