K. Meghana
192321126
Java Programming
CSA0935

(18)

Assignment -03

collection of objects

A collection is a group of objects, known as its elements
It is a single unit of object
collection frame work provides many interfaces and classes

Interfaces - list, set, queue, Dequeue

classes - Arraylist, vector, Linked list, PRD2

Program-1 -> Arraylist

```
Import java.util.Arraylist
    Public class main{
        Public static void main (string[] args){
        Arraylist<string> obj = new Arraylist<>();
            obj.add ("one");
            obj.add ("two");
            obj.add ("three");
            obj.add ("1000");
            obj.add (10000);
        system.out.Println ("Arraylist:" +obj);
    }
}
```

output

```
One
Two
Three
1000
10000
```

Program-2

```
import java.util.ArrayList;
class main{
    Public static void main (string[] args){
ArrayList< Integer > numbers = new Array list <> ();
numbers.add (1);
numbers.add(2);
numbers.add(3);
int index = numbers.index of (2);
system.out.Println("position of 2 is" + index);
int removed Number = numbers.remove(1);
system.out println ("Removed element:" + removed Number);
}
}
```

Output

Position of 2 is 1
Removed element: 2

Program-3

```
import java.util.List;
import java.util.LinkedList;
class main{
Public static void main(string[] args){
  List<string>numbers = new LinkedList<>();
numbers.add("Apple");
numbers.add("Orange");
numbers.add("Mango");
string number = numbers.get(2);
system.out.Println("Accessed element:number");
Int index = numbers.Index of("Apple");
system.out.Println("Position of 2 is" + index);
```

```
numbers.set (2, "Banana");
System.out.println("Updated List:" +numbers);
    numbers.remove ("Orange");
System.out.println ("Final List:");
    for(String fruit :numbers){
    System.out.println(fruits);
    }
}
}
```

Output

```
Accessed element: mango
Position of 'apple' is:0
updated list :[apple, orange, banana]
final list : apple banana
```

Program-4

```
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("Orange");
  fruits.add("Mango");
System.out.Println("Vector:" + fruits);
  String element = fruits.get(2);
System.out.println("Element at Index 2:" + element);
    fruits.add(3,"Banana");
System.out.println("Vector:" + fruites);
    Vector<String> indian fruits = new Vector<>();
  indian fruits.add("Promogranate");
  indian fruits.addALL("fruits");
```

```
System.out.println("new vector=" + indianfruits);
Iterator<String> iterate = indianfruits.iterator();
    System.out.Print("vector:");
    while(iterate.hasNext()){
  System.out.Print(iterate.next());
  System.out.Print(",");
    }
  }
}
```

output

```
Accessed Element : Mango
Position of 'apple' is: 0
Updated List: [apple, orange, banana]
final List: apple, banana grape pomogranate
```

Sort the elements

```
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
Public class SortArray{
Public static void main(String[] args){
    Integer[] array = {5,3,8,1,9,2};
List<Integer> list = Arrays.asList(array);
  Collections.sort(list);
  array = list.toArray(new Integer[0]);
System.out.println(Arrays.toString(array));
  }
}
```

output

```
[1,2,3,5,8,9]
```

Fruits using List and LinkedList

```
import java.util.List;
import java.util.LinkedList;
class main{
    Public static void main (string[] args){
    List<string> fruits = new LinkedList <>();
    fruites.add ("apple");
    fruites.add ("orange");
    fruites.add ("Mango");
    fruites.add ("Grape");
    System.out Println ("original List :"+ fruits);
      collections.reverse (fruits);
    system.out. Println ("Reversed List:"+ fruits);
      Collections. sort (fruits);
    system out Println ("sorted List :"+ fruits);
      collections sort (fruits);
    system out println ("sorted in ascending order:"+ fruits);
      collections. sort (fruits, Collections. reverse order());
    system.out. Println ("sorted in Descending order," fruits);
    system out Println ("Fruits in Basket:");
      for (int i=0 ; i< fruits size(), i++){
        system out println (fruits. get (i));
    }
    system .out Println ("Fruits in the basket (in reverse order):");
    for(int i= fruits size() -1; i>=0; i--){
      system.out println (fruits.get(i));
    }
}
```

output

Original list : [apple, orange, Mango, grape]

sorted list: [apple, grape, Mango, orange]

Reversed list: [Orange, Mango, grape, apple]

sorted in ascending order: [apple, grape, Mango, orange]

Sorted in descending order: [orange, Mango, grape, apple]

fruits in the Basket

Orange Mango Grape Apple

fruits in Basket (in reversed order)

Apple Grape Mango Orange

Stack

```
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 ("stack:" + fruits);
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 the fruit" + Position");
```

```
boolean e = fruits.empty();
System.out.println("is the stack is empty" + e);
fruit.clear();
boolean e1 fruits.empty();
System.out.Println("is the stack is empty" + e1);
}
}
```

Output

```
stack after Pushing Pineapple : [apple, orange Pineapple);
Top element (Peek) : Pineapple
Position of 'orange' : 2
is stack is empty : False
is stack empty after clearing : true.
```

Queue

```
import.java.util.Queue;
class Main {
Public static void main(string[] args){
Queue<string> fruits = LinkedList<>();
fruits.add("Apple");
fruits.add("orange");
fruits.add("Mango");
System.out.println("Queue:" + fruits);
string r = fruits.remove();
System.out.println("Queue:" + r);
System.out.println("Queue:" + fruits);
string display = fruits.peek();
System.out.println("stack:" + display);
```

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

Output

Queue : Apple, Orange, Mango
Remove : apple
Queue after : Orange, Mango
Empty Queue : False
is the Queue is empty. False

DeQueue

```
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.println("Queue:" + fruits);
    String removed = fruits.poll();
System.out.println("Removed element:" + removed);
System.out.println("Queue after poll:" +fruits);
    fruit.add("Pineapple");
```

```
System.out.println("Queue after adding Pineapple:" + fruits);
String frontElement = fruits.peek();
System.out.println("Front element (peek):" + frontElement);
boolean isEmpty = fruits.isEmpty();
System.out.println("Is the queue empty?" + isEmpty);
fruits.clear();
boolean isEmptyAfterClear = fruits.isEmpty();
System.out.println("Is the queue empty after clearing?" + isEmptyAfte
}
}
```

Output

DeQueue : [apple, orange, mango]
Removed element : apple
Queue after poll : [orange, mango]
Queue after adding Pineapple : [orange, mango, Pineapple]
Front element (peek) : orange
Is the queue empty : false
Is the queue empty after clearing ? true.

Map Interface

It is an interface include methods of collection.

```
import java.util.map;
import java.util.HashMap;
class main{
  Public static void main (String[] args){
map<Integer, String> fruits = new Map<>();
  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 emptyset ());
  Sys
  boolean Value = (fruits.remove (2, "Orange"));
system.out. Println ("Removed value" + value);
  system " " ("New Map:" + fruits);
boolean value = fruits containskey (5);
System.out. Println ("Available in the Basket:" + value);
  }
}
```