14

12/8/21

ASSIGNMENT 3

Name :- M. chandu naga sowmya

Reg. NO :- 19237-2064

Course code :- CSA0985

Course Name :- Java Programming

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, LinkedList, PRDQ

Program 1 - Array List

```
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

```
Importjava.util.ArrayList;
class Main{
    Public static void main (string[] args) {
    ArrayList <Integer >numbers = new ArrayList<>();
    numbers.add(1);
```

```
numbers.add(2);
numbers.add(3);
int index = numbers.indexOf(2);
System.out.println("position of 2 is"+index);
int removedNumber = 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.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 pomogranete

* **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.to string(array));
}}
```

output:- [1, 2, 3, 5, 8, 9]

* **fruits using List and Linked List**

```
import java.util.List;
Import java.util.LinkedList;
class Main{
  public static void main (string[] args){
    List<string> fruits =new LinkedList<>();
```

```
        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:" +fruits);
    vector<string> indianfruits = new vector<>();
 indianfruits.add("pomogrante");
indianfruits.addALL(fruits);
System.out.println("New vector =" + Indianfruits);
Iterator<string> iterate = indianfruits.iterator();
System.out.print("vector:");
while(iterate.hasNext()){
```

```
fruits.add("apple");
fruits.add("orange");
fruits.add("Mango");
fruits.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.reverseorder());
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 dscending order: [orange, mango, grape, apple]
fruits in the Basket:
orange Mango Grape Apple
fruits In Basket (in reverse order):
apple Grape Mango orange

Stack

```
import java.util.stack;
 class Main {
   Public Static void main [String[] args){
Stack<String>fruits = newstack<>();
   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.seach("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 e= 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 is 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 el = fruits.isemptyer;
System.out.println("is the Queue is empty:" + el);
}
}
```

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);
  fruits.add("Pineapple");
System.out.println("Queue after ~~poll~~ 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?"
                    +isEmptyAfterClear);
  }
}
```

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 includes the methods of collection. Interface, key, value

Program

```
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(3));
System.out.println ("values:" +fruits.Values (3));
System.out.println("Entries:"+fruits.emptyset(3));
Boolean value=(fruits.remove (2, "Orange"));
System.out.println ("Removed Value" +value);
System.out.println ("NewMap:" +fruits);
Boolean Value1 = fruits.containskey (s);
System.out.println("Available in the Basket :" +Value1);
    }
}
```