Name: R. Saranya
Reg No: 192321022
Course Code: CSA0985

Java Programming

13

Collection & objects :-

Single unit of object. collection frame work provides many Interfaces and classes :-

⇒ List
⇒ Array list
⇒ linked list

List :-

```
public class main
{
  public static void main(String[] args)
  {
    obj.add("one");
    obj.add("Two");
    obj.add("Three");
    obj.add(1000);
    obj.add(10000);
    System.out.println("Array list:" + obj);
  }
}
```

O/p: one Two Three 1000 10000

Array list

import java.util.list

```
class Main
{
  public static void main (String[] args)
  {
    List < Integer > numbers = new ArrayList<>();
    number.add (1);
    number.add (2);
    number.add (3);
    System.out.println ("List" + number);
    int getnumber = number.get (2);
    System.out.println (" element at Index 2:" + get number);
    numbers.remove (1);
    System.out.println (" List after removal :" + number);
    numbers.set (1,4);
    System.out.println ("List after update :" + numbers);
    System.out.println ("Iterating through the list:");
    for (int number : numbers)
    {
      System.out.println (number + " ");
    }
System.out.println();
}
}
```

(P :

List : [1,2,3]

elements at index 2 : 3

List after removal : [1,3]

List after update : [1,4]

Iterating through the list : 1 4

Linked list :-

```
import java.util.list;
import java.util.linkedlist;
Class Main
{
  public static void main (String[] args)
  {
    list<String> numbers = new linked list<>();
    numbers.add("Apple");
    numbers.add("orange");
    numbers.add("Mango");
    String number = number.get(2);
  System.out.println("Accesed element" + numbers);
  int index = numbers.Indexof("Apple");
System.out.println("pos 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);
}
}
}
```

o/p :

Accessed element: mango

pos of 'apple' is : 0

updated list : [apple, orange, banana]

final list : apple, banana, grape, pineapple

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("orange");
```

```
fruits.add ("mango");
System.out.println ("vector" + fruits);
String element = fruitsget(2);
System.out.println ("elem at index 2:" + element);
fruits.add [index 3, element = "Banana");
System.out.println ("vector" + fruits);
vector < String> Indianfruits = new vector <>();
Indianfruits.addAll (fruits);
System.out.println ("new vector:" + Indianfruit);
Iterator < String> Iterate = indianfruits.iterator ();
System.out.println ("vector");
Iterator < String > Iterate = indianfruits.iterator ();
while (iterate.hasNext())
{
System.out.println (iterate.next());
System.out.println (" ");
}
```

Op: Acc Elem: Mango
pos of 'apple' is: 0
upd list: [apple, orange, banana]

Sort and reverse:

```
import java.util.Arrays
import java.util.Collections
```

```
class Main
{
  public static void main (String[] args)
  {
    list <String> fruits = new linked list <> ();
    fruits.add ("Apple");
      "   "   ("orange");
      "   "   ("Mango");
      "   "   ("grape");
    System.out.println ("ori list" + fruits);
    collection.sort (fruits);
    System.out.println ("new list" + fruits);
    collection.sort (fruits);
    System.out.println (fruits, Colle.reverseOrder());
    Collection.sort (fruits, Colle.revorder());
    System.out.println ("Sort in Des order" + fruits);
    System. "   " ("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 (In rev order):");

```
for (int i = fruits.size()-1; i>=0; i++){
    System.out.println(fruits.get(i));
}
}
}
```

o/p:

ori list : [apple, orange, mango, grape]

sort list : [apple, orage, mango, grape]

Rev list : [orange, mango, grape, apple]

sort in asc order : [apple, grape, mango, orange]

sort in Desc order : [orange, mango, grape, apple]

fruits in the basket

orange

mango

grape

apple.

⇒ Stack

⇒ Queue

⇒ Dequeue.

Stack:

```
class main {
public static void main (String[] args)
{
Stack < String > fruits = new stack <> ();
fruits . push ("Apple");
" " ("orange");
fruits . push ("mango");
System. out. println ("stack :" + fruits);
String remove = fruits . pop ();
System . out . println ("stack" + rem);
fruits . push ("pineapple");
System . out . println ("stack" + fruits);
String display = fruits . peak ();
System . out . println ("stack" + display);
int pos = fruits . search ("orange");
System . out . println ("pos of the fruit:" + pos)
boolean e = fruits . empty ();
sys . out . println ("is the stack is empty" + e);
fruits . clear ();
boolean e1 = fruits . empty ();
System . out . println ("is the stack is
                        empty" + e1);
```

}

}

⇒ Dequeue

```
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 n = fruits.remove();
        System.out.println("Queue" +n);
        System.out.println("Queue:" + fruits);
        String display = fruits.peak();
        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);
}
```

```
import java.util.LinkedList;
import java.util.Dequeue;
class Main
{
  public static void main(String[] args)
  {
    Queue<String> fruits = new LinkedList<>();
     fruits.add("apple");
     fruits.add("orange");
       "   "   ("mango");
     System.out.println("Dequeue" + fruits);
     String removed = fruits.poll();
   System.out.println("Rem ele:" + rem);
   "   "   "   ("Queue after poll" + fruits);
   fruits.add("pineapple");
   System.out.println("Queue after adding pineapple" + fruits);
   String frontElement = fruits.peek();
   System.out.println("Front Elem (Peek):" + frontEle);
   boolean isEmpty = fruits.isEmpty();
  System.out.println("Is the queue Empty:" + isempty);
  fruits.clear();
  boolean isemptyAfterClear = fruits.isEmpty();
  System.out.println("Is the Query empty clearing:" +
```

O/P:

Dequeue: [apple, orange, mango]
Removed element: apple
Queue after poll: [orange, mango]
Queue after pineapple: [orange, mango, pineapple]
Front element (peak): orange
Is the Queue empty? false
Is the Queue empty after clearing? True.

Map Interface:.

It is an interface include methods of Collections interface

key, value.

```
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");
  "   "  (3, "Mango");
System.out.println ("Map" + fruits);
System.  "   "  ("keys:" + fruits.keyset());
System.out.println ("values:" + fruits.values());
  "   "   "  (" entries :" + fruits.emptyset());
boolean value = (fruits.remove (2, "orange");
System.out.println ("Rem value :" + value);
System.out.println ("new map:" + fruits);
boolean value 1 = fruits.containskey (3);
System.out.println ("Avail in the Basket :" + value);
}
}
```