P. Reddy Ponuka
192311062

Assignment-3

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, PRDQ.

Program1: 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:

Program2:

```
Import java.util.Arraylist;
    class accin {
    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 removed number = numbers.remove(1);
system.out.println("removed elements:"+removed
                                    numbers);

  }
}
```

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("Elements at Index 2 :" + element);
fruits.add(3, "Banana");
System.out.println("vector : " + fruits);
vector <string> indian fruits = new vector<>();
indianfruits.add("pomogranate");
indianfruits.addALL(fruits);
System.out.println("New vector =" + Indian fruits);
Iterator <String> Iterate = Indianfruits.iterator();
System.out.println("vector:");
while (iteration has next()) {
System.out.println(Iterate.next());
System.out.print(", ");
}
}
}
```

Output: Accessed element: Orange
position of 'apple' is: 0
updated list: [apple, orange, banana]
final list: apple, banana, grape, pomo granate.

*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 = {6, 3, 8, 1, 9, 2};
List <Integer> list = Arrays.asList (arrays);
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 linked list.

```
import java.util.List;
import java.util.linked list;
class main {
public static void main (String[] args) {
List <String> fruits = new linked list <> ();
fruits.add ("apple");
fruits.add ("orange");
fruits.add ("mango");
fruits.add ["Grapes");
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 descending 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 = new Stack();
    fruits.push("apple");
    fruits.push("orange");
    fruits.push("mango");
    System.out.println("stack:"+fruits);
    String remove = fruits.pop();
    System.out.println("stack:" +fruits);
    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 e fruits.empty();
System.out.println(" is the stack is empty "
}
}
```

Output:

stack after pushing pineapple : [apple, orange, pine apple].

Top element (peek): pineapple

position of 'orange' = 2

is stack 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 el = fruits.isEmpty();
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

DEQUE:

```
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 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();
System.out.println("is the queue empty after clearing"
                   +isEmpty after clear);
}}
```

output: Dequeue:[apple, orange, mango]
Removed element = apple
Queue after poll: [orange, mango]
queue after adding pineapple: [orange, mango, pineapple]
front element (peek): orange
false
true

ap interface:

It is an interface include methods of collection 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");
  fruits.put (3, "Mango");
  System.out.println("map" +fruits)
System.out.println ("keys:" +fruits.keyset ();
System.out.println("values" +fruits.values(3);
  "  "  ("  ("entries: +fruits.emptySet);
boolean value: (fruits.remove (2, "orange"));
System.out.println("removed value "+value)
  "  "  "  ("New map:" +fruits);
boolean value = fruits.containsKey(5);
System.out.println("Available in the Basket:"+value
}
}
```