CSA0985

9 18/10/24

Name: C.K.Afroz
Regno: 192311108

Collections of Java as follows:-

1. Arraylist:- An Analysis is resizabla array implementation

```
import Java util *;
  Class Arraylistex{
    Public Static Void main (String []args){
  Arraylist <String >List New ArrayList<>();
        List.add ("Apple");
        List.add ("Banana");
        List.add ("Cherry");
     System out printn(List);
   }
  }
```

Output:-

```
[Apple, Banana, Cherry]
```

2. Linkedlist:-
   A Linkedlist is a doubly-Linked list implementation of list interface

Program:

```
 import Java util *;
Class LinkedListex {
 Public static void man (String args []){
   Linkedlist <String >List New LinkedList<>();
          List.add ("Apple");
          List.add ("Cherry");
     }
    }
```

Output:

```
[Apple, Cherry]
```

3. Hashset:-

Is a set implementation that used as hashtable for strong.

Code:-

```
  import Java.util *;
 Class Hashex {
   Public static void main(String args[]){
     Set.add ("Apple");
     Set.add("Icecream");
    system out print (set);
  }
 }
```

Output:-

```
     [Apple, Icecream]
```

4. Tree set

A Treeset is a set implementation that uses a tree for storage.

Code:-

```
 Import Java.util *;
 Class Treesetex {
  Public static void main (String args[]){
   Treeset<String>set = New Treeset<>();
          Set.add ("Apple");
          Set.add("Banana");
          Set.add(" cherry");
 System out printn (set);
 }
}
```

Output:-

```
[Apple ,Banana ,cherry]
```

5. Hashmap:- a map implementation that uses a hash import Java util *;

```
Class Hashmapx {
        Public static void main(String arg [])
        map.put ("Apple", 1);
        map.put ("Banana", 2);
        map.put ("cherry", 3);
    System.out.println (map);
  }
 }
```

Output:-

```
{ Apple = 1, Banana = 2, cherry = 3 }
```

6 TreeMap:-

A TreeMap is a map implementation that uses a tree for storage

Code:-

```
  import Java util * ;
Class Treemapex {
   Public static void main (String args []){
Tree Map <String, Integer map = new Treemap<>();
         map. put ("Apple", 1);
         map. put ("Banana", 2);
         map. put ("Cherry", 3);
         system . out . print ln (map);
   }
 }
```

Output:-

{Apple 1, Banana= 2, cherry=3}.

7. Linked Hashset

A Linkedhash set is a set implementation that uses a hashtable and Linked List for Storage.

Code:

```
import Java util *;
Hash Linked Hashsetex {
      Public static void main(String args){
Linced Hash set <String> set = new Linked Hashset
                                          <>()
         Set.add ("Apple");
         set.add ("Banana");
         Set.add ("Cherry");
System.out.println(set);
 }
}
```

Output:-

```
[Apple, Banana, Cherry]
```

8. Priority Queue:-

A priority Queue is a Queue implementation that orders elements Based on their natural ordering or a custom comparatior.

Code:-

```
  Import Java.util *;
Class priority Queusex {
```

```
Public Stable void main (String() arg())ꜟ
Priority Queue <String> Queue new priority Queue
                                           <>()?
        Queue add ("Apple");
        queue add ("Banana");
        queue.add ("Cherry");
      System out. print ln(queue);
  }
}
```

Output:-

```
[Apple , Banana ,cherry]
```

9. Array Dequeue:-

An Array Dequeue is adeque implementation that users an Array for storage

Code:-

```
  Import java util *;
 Class Array Dequen {
    Public static void main (String args ())ꜟ
  Array Deque <String> deque = New Array Deque
                                          <>();
       deque.add ("Apple");
       deque.add ("Banana");
     System out println (dequeue);
  }
  }
```

Output:-

```
[Apple, Banana].
```

10. Stack:-

LIFO implementation of the List Interface.

```
import Java util *;
class Stacker {
Public static void main(String args[]){
    Stack <String> Stack "new Stack<>();
    Stack.push ("Apple");
    Stack.push("Banana");
     Stack.push("Cherry");
    System out println(Stack);
  }
}
```

Output:-

```
[Apple, Banana, cherry]
```

11. Vector:

A vector is a synchronized implementation of the List interface.

Code:-

```
import Java.util *;
Class vectorex {
    Public static void main (string args []){
  Vector <String> vector =new vector <>();
          vector.add ("Apple");
            Vector.add ("Custard apple");
       System out println(vector);
    }
  }
```

Output:

```
[Apple , custard Apple]
```