P. Bala Narasimhulu
192372019
C.S.A 0985

ASSIGNMENT 10

Collections of Java as follows :-

1. Array list :- An array list is resizable array implementation.

```
import Java . util *;
Class Arraylist ex{
  Public Static void main (String[] args) {
    Arraylist <String> List . new Array list <>();
      List . add ("Apple");
      List . add ("Banana");
      List . add ("Cherry");
      System. out. println (List );
  }
}.
```

Output :- [Apple, Banana, cherry].

2. Linked List :-

A Linked list is a doubly - Linked List implementation of List interface.

Program :-

```
import Java . util *;
Class Linked List ex {.
  Public Static Void main (String args[]){
    LinkedList < String > List - new Linked List<>();
      List . add ("Apple");
      List . add ("Cherry");
  }
}
```

Output :- [Apple , cherry]

3. Hash Set :-
A Hash Set is a Set implementation that used a hash-table for storage.

```
import. Java. util *;
Class Hashset {
    Public Static Void main (string args[]) {
        Hashset <string> Set -new Hashset <>();
            Set . add ("Apple");
            Set. ad d ("IceCream");
            System.out. println (set);
    }
}
```

Output :-

```
[Apple, Icecream]
```

4. Tree Set :
A TreeSet is a Set implementation that Use for Strage.

```
import. Java. util *;
Class treeSet ex {
    Public Static Void main (string args[]) {
        TreeSet <String> set = new treeset <>();
            Set . add ("Apple");
            Set. ad d ("Banana");
            Set . add ("Cherry");
        System. out. println (set);
    }
}
```

output :- {Apple, Banana, Cherry}

5. Hashmap :- a map implementation that uses a. Hash table For storage.

```
import : Java. util *;
Class Hashmap ex {
    Public Static void main (String args[]) {
  Hashmap < String . integer > map. new Hashmapex();
    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.

```
import . Java. util *;
Class treemapex {
   Public Static void main (String args[]) {
   Treemap < String, integer map. new treemap();
    Map. put ("Apple", 1);
    Map. put ("Banana", 2);
    Map. put ("cherry", 3);
   System. out. println (map);
}}
```

that

Output :-

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

7. **LinkedHashset :-**

A LinkedHashset is a Set implementation that uses a hashtable and LinkedList for a Storage.

```
import Java.util.*;
Class LinkedHashsetex {
    public Static void main (String args[]){
     LinkedHashset <String> set = new LinkedHashex};
      set.add ("Apple");
       Set.add ("Banana");
        Set.add ("cherry");
      System.out.println (set);
  }
}
```

Output :-

```
{Apple, Banana, cherry)
```

8. **priority Queue :-**

```
import Java.util.*;
Class priorityQueueex {
  Public Stable Void main (String[]args()] {
  priorityQueue<String> Queue=new priority Queue<>();
Queue . add ("Apple");
Queue. add ("Banana");
Queue. add ("Cherry");
System. out. println (Queue);
}
}
```

Output :

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

9. **Array Dequeue :-**

~~An~~ ArrayDeQueue is a De-Queue of implementation ~~that~~ uses array of Storage.

```
import. Java . util.*;
Class Array Dequeuer {
Public Static void main (String args []){
Array dequeue <String> dequeue = new Array dequeue <>();
dequeue. add ("Apple");
dequeue. add ("Banana");
System. out. println (dequeue);
}
}
```

Output :-

```
[Apple, Banana]
```

10. **Stack :-**

LIFO implementation of the List Interface.

```
import ·Java. util *;
Class Stackex {
    Public Static void main (String args[])
        Stack <String> Stack . new Stack <> ();
        Stack · push ("Apple");
        Stack · push ("cherry");
        Stack · push ("Banana")
    }   System out · println (stack);
}
```

Output :-

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

11. Vector :-

A vector is a synchorized implementation of List interface

```
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, ].
```