Praise to Allah

The Lord of the Universe, the most gracious, the most merciful, the in-finite, the king, the Judge, the one and only , the sole possessor of our life.

Humera Tariq

PhD, MS, MCS (Computer Science), B.E (Electrical)

Postdoc (Medical Image Processing, Machine Learning, Deep Neural Networks)

Email: humera@uok.edu.pk
Web: https://humera.pk/

Write the output of following program along with reasoning (1).

```
import java.util.ArrayList;
class Point {
    int x; // x-coordinate
    int y; // y-coordinate

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
public class Main {
    public static void main(String[] args) {
        ArrayList pArrayList = new ArrayList<>();
        pArrayList.add(new Point(10, 20));
        pArrayList.add(new Point(5, 10));

        // Print the x-coordinate of the 2nd element (index 1)
        System.out.println("X-coordinate of 2nd element: " + pArrayList.get(1).x);
    }
}
```

UNIVERSITY OF KARACHI

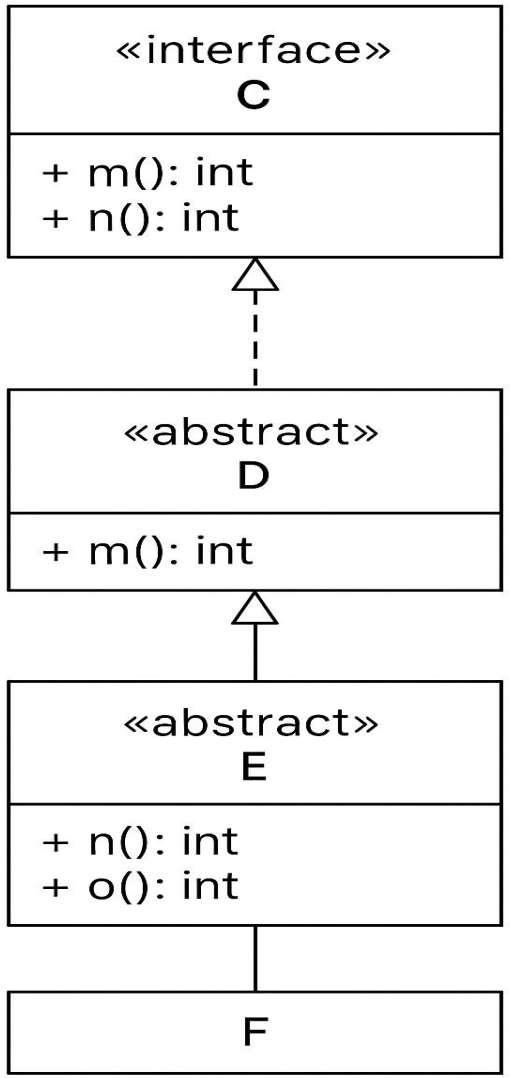
Write the output of following program along with reasoning (2).

```
public class Main {
    public static void main(String[] args) {


            int[] xarray = { 4, 2, 1, 3, 5 };
            int y = xarray[1] + 1;

            System.out.print(y + ", ");
            xarray[4] = 1;
            System.out.print(xarray[xarray[4]] + ",");
            y--;
            System.out.print(xarray[y]);


    }
}
```

UNIVERSITY OF KARACHI

Verify UML for Interface problem hierarchy week 14 and resolve errors (3)

```
public class Problem3Fixed {
    public static void main(String[] args) {
        // 1) Instantiate F (concrete)
        F f = new F();
        System.out.println("f.m() = " + f.m());
        System.out.println("f.n() = " + f.n());
        System.out.println("f.o() = " + f.o());
        // 2) Use E reference to E object
        E e = new E();
        System.out.println("e.m() = " + e.m());
        System.out.println("e.n() = " + e.n());
        System.out.println("e.o() = " + e.o());

        // 3) Use interface type C to refer to F
        C c = new F();
        System.out.println("c.m() = " + c.m());
        System.out.println("c.n() = " + c.n());

        // To call o(), you must ???:
        System.out.println("c.o() = " + c.o());
    }
}
```



UNIVERSITY OF KARACHI

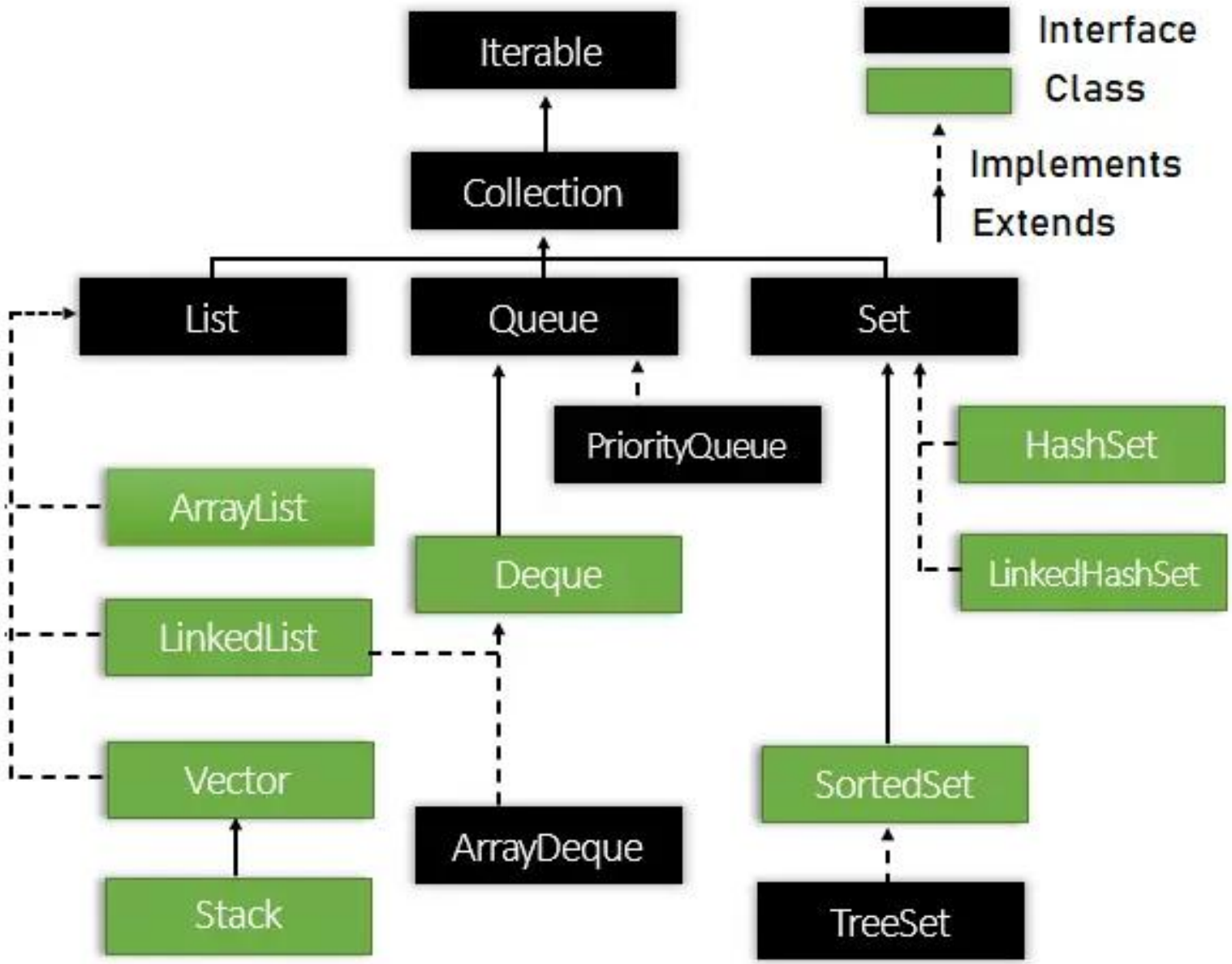
Draw UML and write the output from main on upcoming slide (4) !

```
class Gamer {
    String username = "Anonymous";

    void play() {
        System.out.println(username + " is playing a casual game.");
    }
}
class ProGamer extends Gamer {
    int rank = 1;

    @Override
    void play() {
        System.out.println(username + " is streaming a ranked match at rank " + rank + "!");
    }


    void stream() {
        System.out.println(username + " is live on Twitch with rank " + rank + "!");
    }
}
```

Main.java for dynamic dispatch and slicing concept (4)

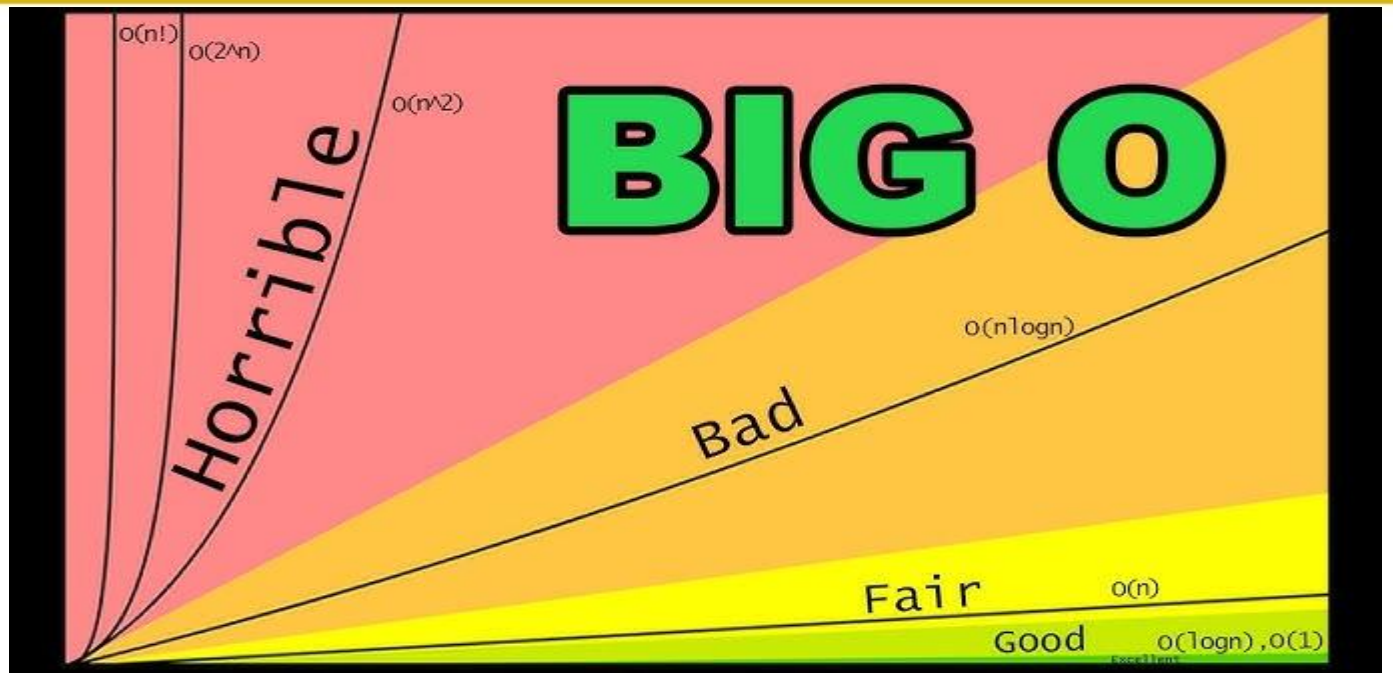
```
public class Game {
    public static void main(String[] args) {
        ProGamer pro = new ProGamer();
        pro.username = "LunaByte";
        pro.rank = 12;


        Gamer g = pro; // Upcasting – no slicing in Java!


        g.play(); // Calls overridden method in ProGamer (dynamic dispatch)
        // g.stream(); // Not accessible – reference type is Gamer


        // Downcast back to ProGamer
        ProGamer realPro = (ProGamer) g;
        realPro.stream(); // Works fine – full object still intact
    }
}
```

List ≈ Sets (Same Interface) but Maps are different



UNIVERSITY OF KARACHI

Judging Criteria (1) insertion, Removal , Retrieval



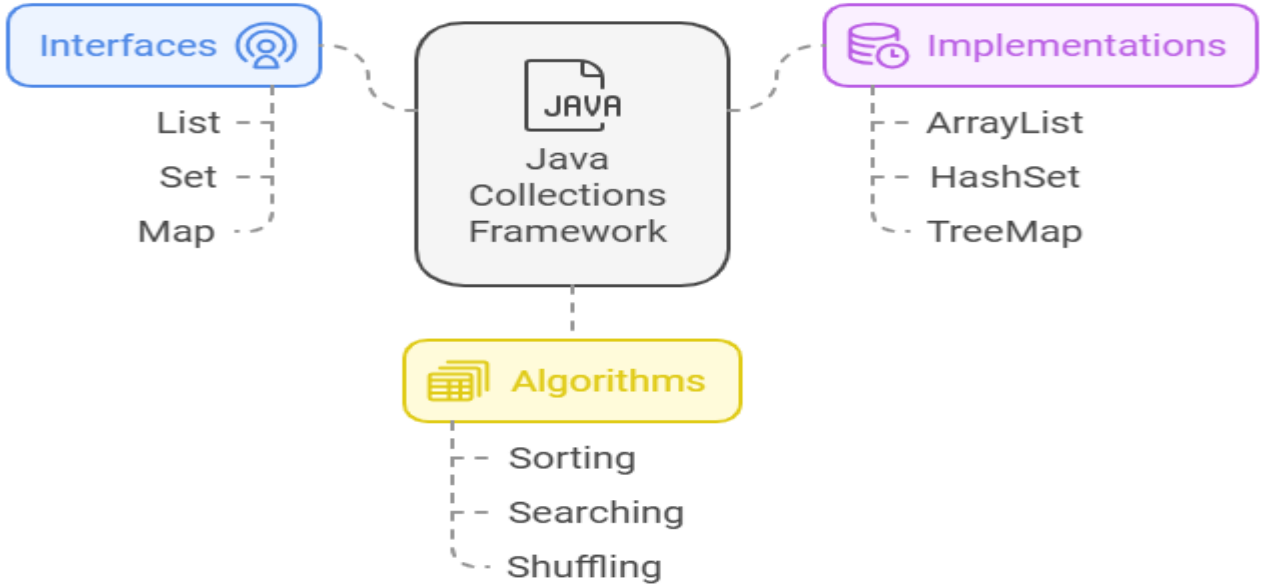
- ✓ **HashSet** = instant access by hash → lightning-fast ,unordered (sibling HashMap-dict)
- ✓ **LinkedHashSet** = HashSet + remembering order (sticky note trail on top) → a bit slower
- ✓ **TreeSet** = keeps everything sorted → rearrange every time -> must "climb" the tree → slower (log n)

UNIVERSITY OF KARACHI

Arrays.asList vs. String[ ] Output ? Which one to prefer and Why ?

```
import java.util.*;


public class JCF {
    public static void main(String[] args) {


        String[] arr = {"Zara", "Ali", "Bilal", "Nabeel"};
        List<String> names = Arrays.asList(arr);


        // Now you can do this:
        System.out.println(names.contains("Ali"));
        System.out.println(names.indexOf("Bilal"));
        Collections.sort(arr);
        Collections.sort(names);
        System.out.println(names);
```

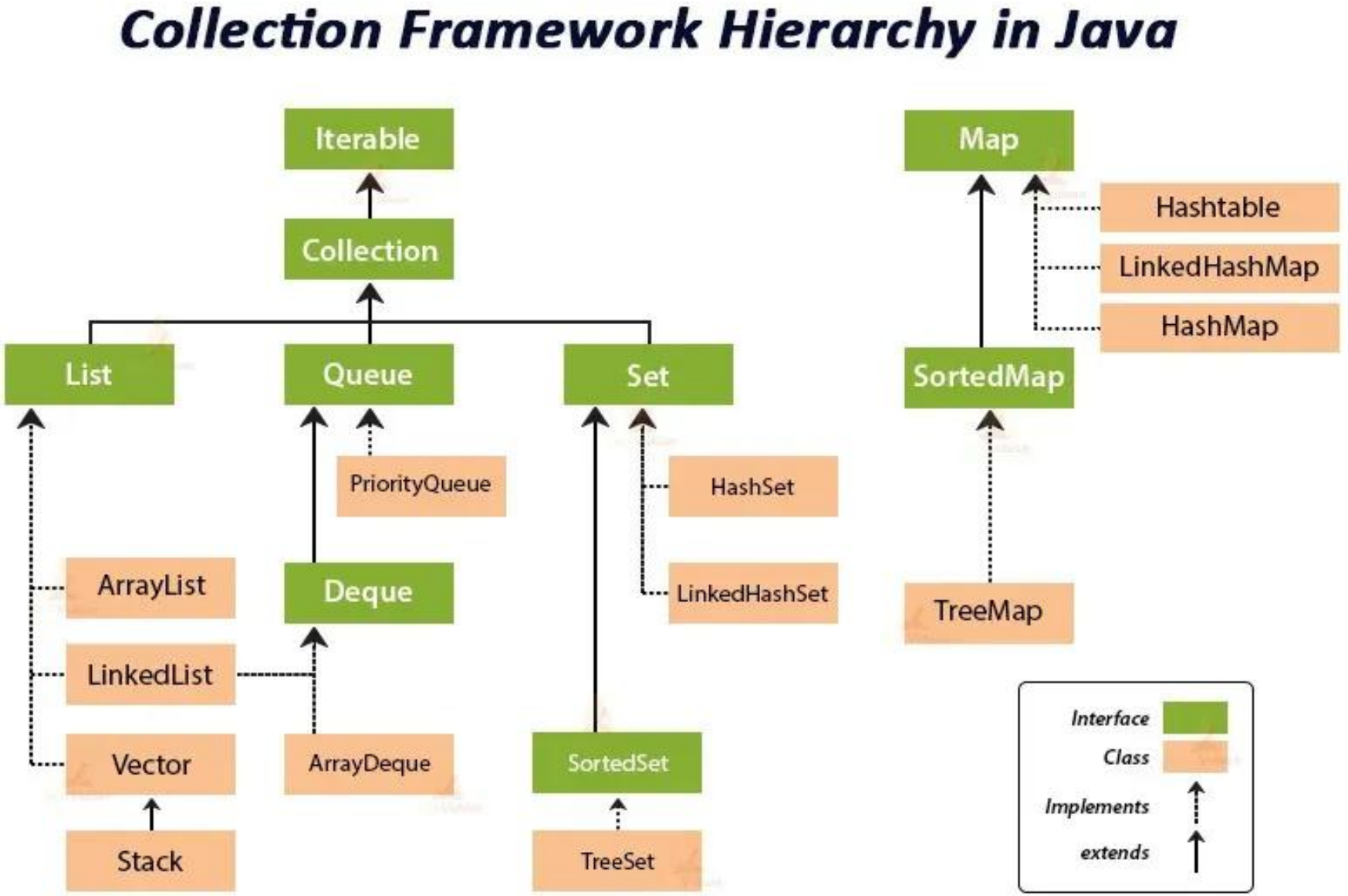
Judging Criteria (2) How they compare elements?



- ✓ **HashSet** = Uses equals() and hashCode() and removes possible duplicate elements
- ✓ **LinkedHashSet** = Also uses equals() and hashCode()
- ✓ **TreeSet** = compare() or compareTo()

UNIVERSITY OF KARACHI

Performance : HS > LHS > TS



UNIVERSITY OF KARACHI

Let's explore the sorting nature of HS, LHS, TS and HM? Output ?

```
import java.util.*;
public class JCF {
    public static void main(String[] args) {

        Set<String> hashSet = new HashSet<>();
        Set<String> linkedHashSet = new LinkedHashSet<>();
        Set<String> treeSet = new TreeSet<>();
        Map<String, Integer> hashMap = new HashMap<>();

        List<String> names = Arrays.asList("Zara", "Ali", "Bilal", "Nabeel");
        for (String name : names) {
            hashSet.add(name);
            linkedHashSet.add(name);
            treeSet.add(name);
            hashMap.put(name, name.length());
        }

        System.out.println("HashSet:         " + hashSet);
        System.out.println("LinkedHashSet:   " + linkedHashSet);
        System.out.println("TreeSet:         " + treeSet);
        System.out.println("HashMap:         " + hashMap);
    }}
```

UNIVERSITY OF KARACHI

HS, LHS, TS and HM: Output

```
[Running] cd "d:\Week 15-JAVA\MCQs\" && javac
JCF.java && java JCF
HashSet:          [Ben, Nina, Zara, Ali]
LinkedHashSet:    [Zara, Ali, Ben, Nina]
TreeSet:          [Ali, Ben, Nina, Zara]
HashMap:          {Ben=3, Nina=4, Zara=4, Ali=3}
```

3

Judging Criteria (3) Null element Mystery? Output ?

```
import java.util.*;
public class NullInSetsDemo {
    public static void main(String[] args) {
        // HashSet allows one null
        Set<String> hashSet = new HashSet<>();
        hashSet.add(null);
        hashSet.add("Ali");
        hashSet.add(null);
        System.out.println("HashSet: " + hashSet);
        // LinkedHashSet allows one null and keeps order
        Set<String> linkedHashSet = new LinkedHashSet<>();
        linkedHashSet.add("Ali");
        linkedHashSet.add(null);
        linkedHashSet.add("Bilal");
        System.out.println("LinkedHashSet: " + linkedHashSet);
        // TreeSet does NOT allow null
        Set<String> treeSet = new TreeSet<>();
        treeSet.add("Ali");
        try {
            treeSet.add(null); // will throw NullPointerException
        } catch (NullPointerException e) {
            System.out.println("TreeSet: cannot add null (throws exception)");
        }    } }
```

UNIVERSITY OF KARACHI

Null element Mystery? Output

```
[Running] cd "d:\Week 15-JAVA\MCQs\" && javac
NullInSetsDemo.java && java NullInSetsDemo
HashSet: [null, Ali]
LinkedHashSet: [Ali, null, Bilal]
TreeSet: cannot add null (throws exception)
```

UNIVERSITY OF KARACHI

4 Judging Criteria (4) Memory Occupation (HS < LHS < TS) Output ?

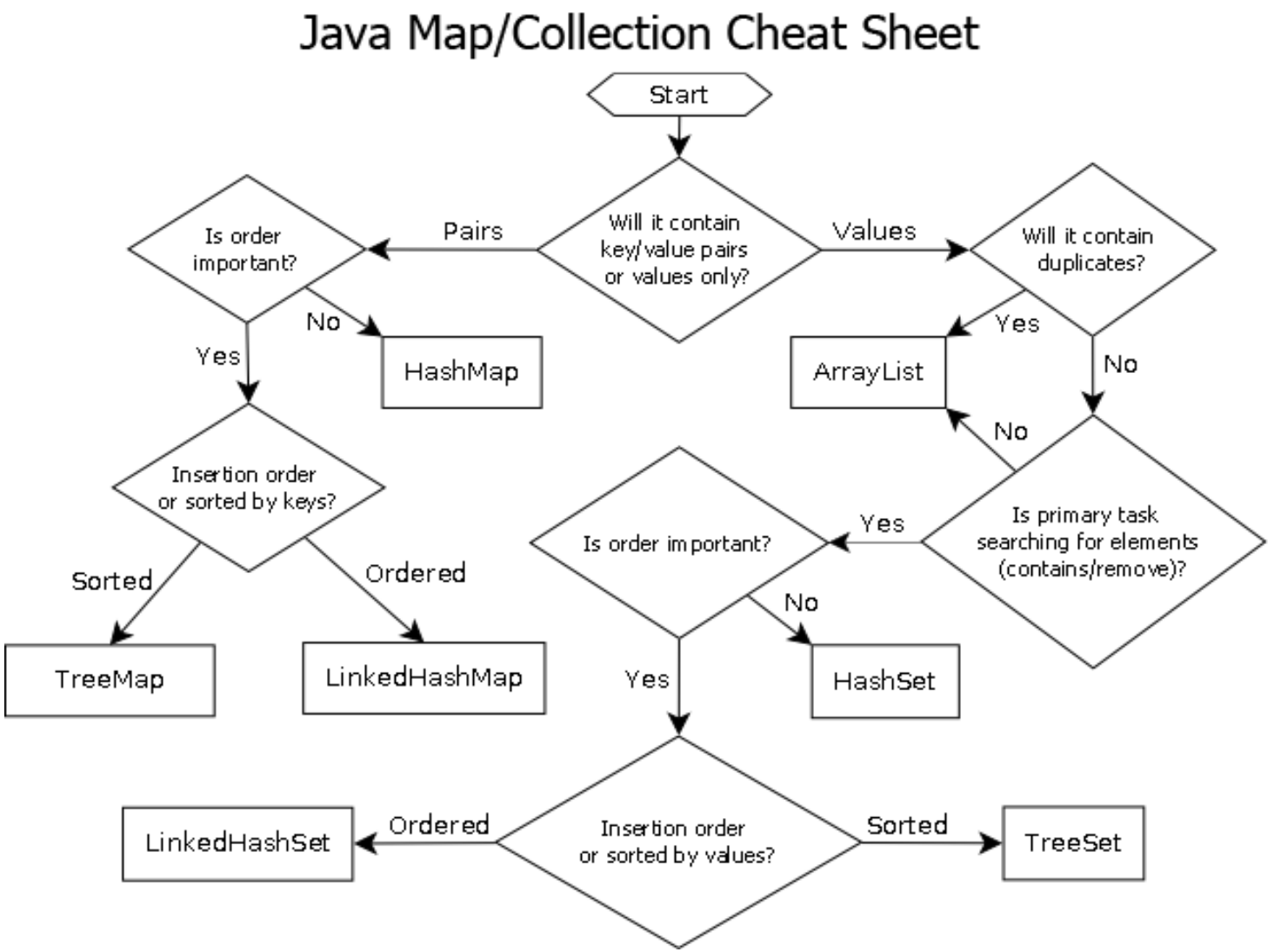
```
import java.util.*;
public class MemoryDemo {
    public static void main(String[] args) {
        measureSetMemory("HashSet", new HashSet<>());
        measureSetMemory("LinkedHashSet", new LinkedHashSet<>());
        measureSetMemory("TreeSet", new TreeSet<>());
    }
    static void measureSetMemory(String name, Set<String> set) {
        Runtime rt = Runtime.getRuntime();
        rt.gc(); // encourage garbage collection
        long before = rt.totalMemory() - rt.freeMemory();

        for (int i = 0; i < 100000; i++) {
            set.add("Item" + i);
        }

        long after = rt.totalMemory() - rt.freeMemory();
        System.out.printf("%-15s memory used ≈ %,d bytes%n", name, (after - before));
    }
}
```

UNIVERSITY OF KARACHI

Memory Occupation Output ?

```
[Running] cd "d:\Week 15-JAVA\MCQs\" && javac
MemoryDemo.java && java MemoryDemo
HashSet          memory used ? 12,546,504 bytes
LinkedHashSet    memory used ? 12,114,208 bytes
TreeSet          memory used ? 9,763,664 bytes
```

When to use? Decision ?



UNIVERSITY OF KARACHI

Let's go to Dinning !

```
public class kitchen {
    public static void main(String[] args) {
        PlaceSetting x = new PlaceSetting(9);
    } }
```

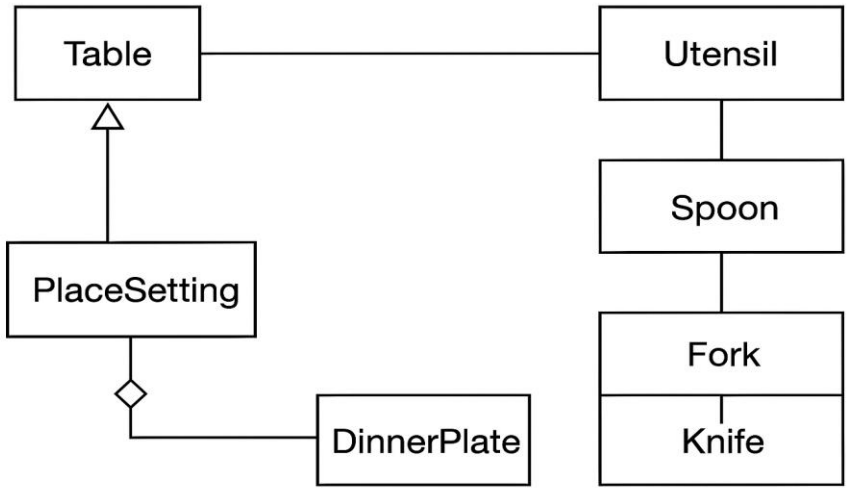
```
// ------------------------------------
// Combining inheritance + composition
// ------------------------------------
class PlaceSetting extends Table {
    private Spoon sp;
    private Fork frk;
    private Knife kn;
    private DinnerPlate dp;

    PlaceSetting(int i) {
        super(i + 1);
        sp = new Spoon(i + 2);
        frk = new Fork(i + 3);
        kn = new Knife(i + 4);
        dp = new DinnerPlate(i + 5);
        System.out.println("PlaceSetting constructor");
    }}
```

```
class Table {

    Table(int i) {

        System.out.println("Table constructor");

    }

}
```

UNIVERSITY OF KARACHI

Verify UML for Dining problem

```
class Plate {
    Plate(int i) {
        System.out.println("Plate constructor");
    }
}
class DinnerPlate extends Plate {
    DinnerPlate(int i) {
        super(i);
        System.out.println("DinnerPlate constructor");
    } }
```



UNIVERSITY OF KARACHI

Predict the output!

```
class Utensil {
    Utensil(int i) {
        System.out.println("Utensil constructor");
    }
}
class Spoon extends Utensil {
    Spoon(int i) {
        super(i);
        System.out.println("Spoon constructor");
    }}
```

```
class Fork extends Utensil {
    Fork(int i) {
        super(i);
        System.out.println("Fork constructor");
    }
}
class Knife extends Utensil {
    Knife(int i) {
        super(i);
        System.out.println("Knife constructor");
    }
}
```

UNIVERSITY OF
KARACHI

Answers

Explanation/Reasoning

- ✓ `ArrayList` without `<Point>` is a *raw type* — it stores `Object` references.
- ✓ Accessing `.x` on an `Object` is invalid.
- ✓ Using `ArrayList<Point>` ensures compile-time type safety and allows direct field access.

UNIVERSITY OF KARACHI

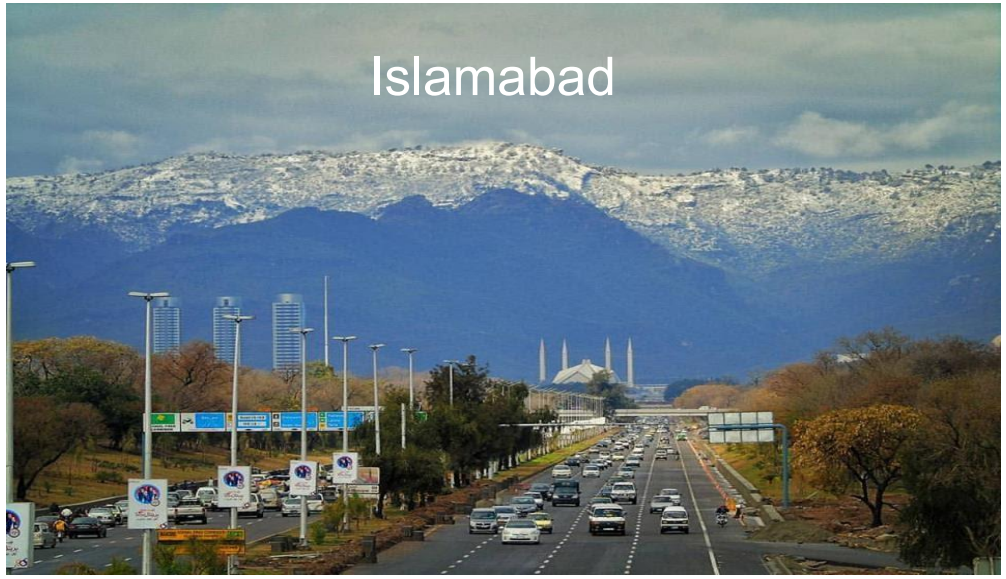
Step	Statement	xarray Values	y Value	Printed Output	Explanation
1	int[] xarray = {4, 2, 1, 3, 5};	{4, 2, 1, 3, 5}	—	—	Array initialized
2	int y = xarray[1] + 1;	{4, 2, 1, 3, 5}	3	—	xarray[1]=2 → y=3
3	System.out.print(y + ", ");	{4, 2, 1, 3, 5}	3	3,	Prints current value of y
4	xarray[4] = 1;	{4, 2, 1, 3, 1}	3	—	Replaces last element 5 → 1
5	System.out.print(xarray[xarray[4]] + ",");	{4, 2, 1, 3, 1}	3	2,	xarray[4]=1 → print xarray[1]=2
6	y--;	{4, 2, 1, 3, 1}	2	—	Decrement y by 1
7	System.out.print(xarray[y]);	{4, 2, 1, 3, 1}	2	1	xarray[2]=1

Department of Compute Science (UBIT Building), Karachi, Pakistan.

1200 Acres (5.2 Km sq.)
53 Departments
19 Institutes
25000 Students

UNIVERSITY OF KARACHI

Pakistan Zindabad, Pakistan Paindabad



UNIVERSITY OF
KARACHI