-Deven Prakash Paramaj

1BM19IS048

4th sem, 4A

Java lab test

2a) Identify the type of inheritance in the given diagram. Create a class A with two integer member

variables that are private, two float variables that are protected and two integer variables that are public.

Let class B inherit class A and class C and Class D are inherited from class B. Write appropriate

methods to illustrate the following

i) Usage of super keyword

ii) Function overriding

iii) Default constructors

iv) Parameterized constructors

v) How to we make a method not to be over ridden and a class not be inherited further

```
class A
{
    private int pri1, pri2;
    public int pub1, pub2;
    protected float pro1, pro2;

    A()
    {
        pri1 = pri2 = pub1 = pub2 = 0;
        pro1 = pro2 = 0;
    }

    A(int i1, int i2, int i3, int i4, float f1, float f2)
    {
        pri1 = i1;
```

```
        pri2 = i2;
        pub1 = i3;
        pub2 = i4;
        pro1 = f1;
        pro2 = f2;
    }

    void display()
    {
        System.out.println("Class A");
        System.out.println("Private Data : " + pri1 + ", " + pri2);
        System.out.println("Public Data : " + pub1 + ", " + pub2);
        System.out.println("Protected Data : " + pro1 + ", " + pro2);
    }
}

class B extends A
{
    B() { super(); }

    B(int i1, int i2, float f1, float f2)
    {
        super(0, 0, i1, i2, f1, f2);
    }

    void display()
    {
        System.out.println("Class B");
        System.out.println("Private Data is not Inherited");
        System.out.println("Public Data : " + pub1 + ", " + pub2);
        System.out.println("Protected Data : " + pro1 + ", " + pro2);
    }
}


final class C extends B
{
    C() { super(); }

    C(int i1, int i2, float f1, float f2)
    {
        super(i1, i2, f1, f2);
    }

    final void display()
    {
        System.out.println("Class C");
        System.out.println("Private Data is not Inherited");
        System.out.println("Public Data : " + pub1 + ", " + pub2);
```

```
        System.out.println("Protected Data : " + pro1 + ", " + pro2);
    }
}

class D extends B
{
    D() { super(); }

    D(int i1, int i2, float f1, float f2)
    {
        super(i1, i2, f1, f2);
    }

    void display()
    {
        System.out.println("Class D");
        System.out.println("Private Data is not Inherited");
        System.out.println("Public Data : " + pub1 + ", " + pub2);
        System.out.println("Protected Data : " + pro1 + ", " + pro2);
    }
}

class Inheritance
{

    public static void main(String[] args)
    {
        A a = new A(1, 2, 3, 4, 5.5f, 6.6f);
        a.display();
        A b = new B(7, 8, 9.9f, 10.1f);
        b.display();
        A c = new C(11, 12, 13.3f, 14.4f);
        c.display();
        A d = new D(15, 16, 17.7f, 18.8f);
        d.display();
    }
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Engineering college\4th sem\Java\lab\labTest> javac .\Inheritance.java
PS E:\Engineering college\4th sem\Java\lab\labTest> java Inheritance
Class A
Private Data : 1, 2
Public Data : 3, 4
Protected Data : 5.5, 6.6
Class B
Private Data is not Inherited
Public Data : 7, 8
Protected Data : 9.9, 10.1
Class C
Private Data is not Inherited
Public Data : 11, 12
Protected Data : 13.3, 14.4
Class D
Private Data is not Inherited
Public Data : 15, 16
Protected Data : 17.7, 18.8
PS E:\Engineering college\4th sem\Java\lab\labTest>
```

MODIFICATION

Student, College, Marks, - interface, abstract, final

```java
import java.util.LinkedList;

abstract class Student
{
    String name;
    String usn;
    int[] marks;

    Student()
    {
        marks = new int[3];
    }

    String getName()
    {
        return name;
    }
```

```
    String getUsn()
    {
        return usn;
    }

    int[] getMarks()
    {
        return marks;
    }

    public String toString() {

        String s = "Name : " + name + " | USN : " + usn;
        return s;
    }
}


interface College
{
    String getCollegeName();
    String getCollegeCode();
    boolean isCollegeStudent(Student s);
    void addStudent(Student s);
}

final class BmsceStudent extends Student
{
    BmsceStudent(String name, String usn, int[] marks)
    {
        this.name = name;
        this.usn = "1BM19IS" + usn;
        this.marks = marks;
    }

}

final class BmsITStudent extends Student
{
    BmsITStudent(String name, String usn, int[] marks)
    {
        this.name = name;
        this.usn = "1BY19IS" + usn;
        this.marks = marks;
    }

}

final class Bmsce implements College
```

```
{
    final String collegeName;
    final String collegeCode;
    LinkedList<Student> students;

    Bmsce()
    {
        collegeCode = "BM";
        collegeName = "Bmsce";
        students = new LinkedList<>();
    }

    public String getCollegeName() {
        return collegeName;
    }

    public String getCollegeCode() {
        return null;
    }

    public boolean isCollegeStudent(Student s) {

        if(s.usn.contains(collegeCode))
            return true;
        else
            return false;
    }

    public void addStudent(Student s)
    {
        if(isCollegeStudent(s) == false)
            return;
        else
            students.add(s);
    }

    public String toString()
    {
        String s = "College Name : " + collegeName + "\nCollegeCode : " +
collegeCode + "\nStudents : ";
        s += students.toString();
        return s;
    }
}

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

```java
    {
        int[] marks1 = {9, 9, 9};
        BmsceStudent student1 = new BmsceStudent("deven", "048", marks1);
        int[] marks2 = {8, 9, 8};
        BmsITStudent student2 = new BmsITStudent("dev2", "052", marks2);
        int[] marks3 = {7, 7, 7};
        BmsceStudent student3 = new BmsceStudent("dev3", "042", marks3);

        Bmsce bmsce = new Bmsce();

        bmsce.addStudent(student1);
        bmsce.addStudent(student2);
        bmsce.addStudent(student3);

        System.out.println(bmsce);

    }
```

Output: