Question:

To understand the concept of inheritance and construct programming constructs to analyze inheritance and understand and analyze how polymorphism can be appropriately used and to be actualized both at compile time and run time.

Problem Statement:

1. A class Shape having at most two dimensions. Define two subclasses of Shape.

i) Override a method area.

ii)Show compile time and run time polymorphism(Dynamic method dispatch).

iii)Use a final method for display.

iv) use super keyword.

Code:

```
// "static void main" must be defined in a public class.
class shape{
    //taking 2 dimensions
    int dim1;
    int dim2;
    //assign d1 to dim1 - circle
    shape(int d1) {
        dim1 = d1;
    }
    //assign dim1 and dim2 - rectangle
    shape(int d1,int d2) {
        dim1 = d1;
        dim2 = d2;
    }
    //overridden method area
    void area(){
        System.out.println(" shape is unknown. ");
    }
    //overloaded method area for static binding/compile time binding
    void area(int d1) {
         double ar = 3.14*d1*d1;
        System.out.println("Area of circle is:"+ar);
    }
```

```
    void area(int d1, int d2) {
        int a = 2*((d1 * d2) + (d2 * d1));
        System.out.println("Area of rectangle is:"+a);
    }
    //using final method for display
    final void display() {
        System.out.println("It is confirmed this is a shape.");
    }
}
class circle extends shape {
    //using super for using constructor of super class
    circle(int radius) {
        super(radius); //super keyword
    }
    //overridden method area
    void area() {
        double ar = 3.14*dim1*dim1;
        System.out.println("area of circle is:"+ar);
    }

}
class rectangle extends shape {
    rectangle(int d1, int d2) {
        super(d1,d2); //using super keyword
    }
    //overridden method area
    void area() {
        int a = 2*((dim1 * dim2) + (dim2 * dim1));
        System.out.println("Area of rectangle is:"+a);
    }
}
public class Day5 {
    public static void main(String[] args) {
        shape s = new shape(5);
        s.area(5); //compile time polymorphism
        s.area(2,3);
        shape obj = new circle(5);
        obj.area(); //runtime polymorphism
        obj = new rectangle(2,3);
```

```
        obj.area();
        obj.display();
    }
}

/*
super() => used to access the constructor of super class
super.methodname() => used to access the method of super class

*/
```

Output: