**Practical 05**

Exercise 01:

1. With “public” key word

public interface MyFirstinterface{

    public int x;

 //abstract method

 void display();

}

Without “public” keyword

public interface MyFirstinterface{

     int x;

 //abstract method

 void display();

}

Public keyword is optional, there is no difference between these two approaches in this case.

1. With “abstract” keyword

public interface MyFirstinterface{

 //abstract method

 abstract void display();

}

Without “abstract” keyword

public interface MyFirstinterface{

 //abstract method

  void display();

It is not required to use the abstract keyword when declaring an abstract method within an interface. The method is abstract by default if the abstract keyword is not present. As a result, there is no difference between defining an abstract method within an interface with or without the abstract keyword.

1. In the "InterfaceImplemented" class, we overrode the "display()" function from the "MyFirstInterface" interface. However, attempting to modify the value of the variable "x" within the "display()" method will result in a compilation error.

This issue occurs because interface variables are implicitly treated as public static final, which means they are constants whose values cannot be changed. Any attempt to change the value of "x" after it is declared in the interface will result in a compilation fault.

Exercise 02:

public interface speaker{

    public void speak();

}

public class politician implements speaker {

    public void speak(){

        System.out.println("Talks about politics");

    }

}

public class priest implements speaker {

     public void speak(){

        System.out.println("Talks about religious");

     }

}

public class lecturer implements speaker{

    public void speak(){

        System.out.println("Talks about module");

    }

}

Exercise 03:

Compilation error,

Why,

* The class is declared as final, which means it cannot be subclassed.
* The class "Undergraduate" tries to extend the class "Student," but because the "Student" class is designated final, it cannot be subclassed. A compilation error will occur as a result of this.

Exercise 04:

abstract class Shape {

abstract double calculateArea();

void display() {

System.out.println("This is a shape.");

}

}

class Rectangle extends Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

@Override

double calculateArea() {

return length \* width;

}

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

double calculateArea() {

return Math.PI \* radius \* radius;

}

}

public class Main {

public static void main(String[] args) {

Rectangle rectangle = new Rectangle(10, 12);

System.out.println("Rectangle Area: " + rectangle.calculateArea());

rectangle.display();

Circle circle = new Circle(4.2);

System.out.println("Circle Area: " + circle.calculateArea());

circle.display();

}

}