Practical 05-Java Lab

Student name-G.O Wickramaratne

ID- 28039

1) i.

```
public interface MyFirstInterface {
   int x = 10; // With public static final keywords

   void display(); // With abstract keyword
}
```

There is no difference with or without public static final keywords since it is already set as default.

ii.

```
public class InterfaceImplemented implements MyFirstInterface {
   @Override
   public void display() {
      // Implementation goes here
      x = 20; // Changing the value of x
      System.out.println("The value of x is: " + x);
   }
}
```

There is no difference with or without the abstract keyword since the abstract keyword has already been set as default.

iii. **The variable of x is declared as a constant (with public static final keywords) there the value of x cannot be changed this will result in a compilation error when trying to modify.**

2)

```
//defines interface
public interface Speaker {

   void speak();
}

// Implement the Politician class

public class Politician implements Speaker{

   @Override
   public void speak() {
      System.out.println("Politician :Vote me!");
   }
}
// Implement the Lecturer class
public class Lecturer implements Speaker
  {
   @Override
   public void speak() {
      System.out.println(" Lecturer :The lesson today will be about oop is java");
```

	}
	}
	
	public class Priest implements Speaker
	// Implement the Priest class
	{
	@Override
	public void speak() {
	System.out.println("Priest : Bless you!");
	}
	}

3)	• The **display()** method in the **Student** class is declared as **final** but does not have an implementation. A **final** method cannot be overridden by subclasses, so it should either have a method body within the **Student** class or be declared as an abstract method, allowing subclasses to provide their own implementation. • The **Undergraduate** class is trying to extend the **Student** class, but the **Student** class itself is marked as **final**. Subclasses cannot extend final classes, so you can't extend Student with Undergraduate.

4)	abstract class Shape {
	
	abstract double calculateArea();
	
	void display() {
	
	System.out.println("Displaying shape information.");
	
	}
	
	}
	
	public class Circle extends Shape {
	
	private double radius;
	
	Circle(double radius) {

```
        this.radius = radius;
    }
    double calculateArea() {
        return 3.1459f * radius * radius;
    }
}

public class Rectangle extends Shape{
    private double length;
    private double width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    double calculateArea() {
        return length * width;
    }
}

public class Testshapes {

    public static void main(String[] args) {
        Circle circle = new Circle(5.0);
```

```
        Rectangle rectangle = new Rectangle(4.0, 6.0);



        circle.display();

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



        rectangle.display();

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

    }

}
```