Name : S.M.B Dissanayake
Student ID : 27357

Practical 05

<table>
<tr><td>1.</td><td>i. Variables in the interface are inherently public, static, and final. As a result, they do not need to be explicitly declared with these terms. Both ways will produce the same results. However, adding the keywords public static final to the variable definition will have no effect in this scenario.

ii. All methods in an interface are implicitly abstract. As a result, they do not need to be explicitly declared with the abstract keyword. Both ways will produce the same results. However, adding the abstract keyword explicitly to the method definition has no effect in this scenario.

iii. The variable x in the interface is implicitly final, which implies that once assigned, its value cannot be modified. Even though you can access x from the implementing class's display() method, any attempt to modify its value will result in a compilation error. As a result, changing the value of x within the implementing class is not possible.</td></tr>
<tr><td>2.</td><td>package com.mycompany.speakmain;

public interface Speaker {
  void speak();
}

public class Politician implements Speaker{

  @Override
  public void speak()
  {
  System.out.println("Politician speaking...");
  }
}

public class Priest implements Speaker{

  @Override
  public void speak()
  {
  System.out.println("Priest speaking...");
  }
}

public class Lecturer implements Speaker{</td></tr>
</table>

	static Lecturer lecturer;
	
	@Override
	public void speak()
	{
	System.out.println("Lecturer speaking...");
	}
	}
	
	package com.mycompany.speakmain;
	
	import static com.mycompany.speakmain.Lecturer.lecturer;
	
	public class SpeakMain {
	
	public static void main(String[] args) {
	Politician politician=new Politician();
	politician.speak();
	
	Priest priest=new Priest();
	priest.speak();
	
	Lecturer.lecturer=new Lecturer();
	lecturer.speak();
	}
	}

3.	The Student class in Class 01 is declared as final, which means it cannot be inherited by any other class. Furthermore, the display() method is marked as final, which means that no subclass can override it. Class 02 attempts to inherit from the Student class, but because Student is tagged as final, it cannot be extended. A compilation error will occur as a result of this. The final keyword is used to limit class inheritance or method and variable overrides. When a class or function is declared final, it cannot be subclassed or overwritten. Depending on our intended design, we may either remove the final keyword from the Student class or remove the inheritance attempt in the Undergraduate class to resolve the compilation fault.

4.	package com.mycompany.shapemain;
	
	abstract class Shape {
	abstract double calculateArea();
	
	void display()

```
    {
        System.out.println("Displaying shape...");
    }
}

public class Circle extends Shape{
    private double radius;

    public Circle(double radius)
    {
        this.radius=radius;
    }

    @Override
    public double calculateArea()
    {
        return Math.PI*radius*radius;
    }
}

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

    public Rectangle(double length,double width)
    {
        this.length=length;
        this.width=width;
    }

    @Override
    public double calculateArea()
    {
        return length*width;
    }
}

public class ShapeMain {

    public static void main(String[] args) {
        Circle circle=new Circle(5.0);
        System.out.println("Area of a circle: "+circle.calculateArea());
        circle.display();

        Rectangle rectangle=new Rectangle(3,4);
        System.out.println("Area of a rectangle: "+rectangle.calculateArea());
        circle.display();
    }
}
```