Practical 05: Inheritance & Abstract Classes 28436 Rayeezul Atheeb

Exercise 01:

Declare an interface called “MyFirstInterface”. Decalre integer type variable called “x”. Declare an abstract method called “display()”.

1. Try to declare the variable with/without public static final keywords. Is there any difference between these two approaches? Why?

```
public interface MyFirstInterface
{
    int x = 10;
}
```

2. Declare the abstract method with/without abstract keyword. Is there any difference between these two approaches? Why?

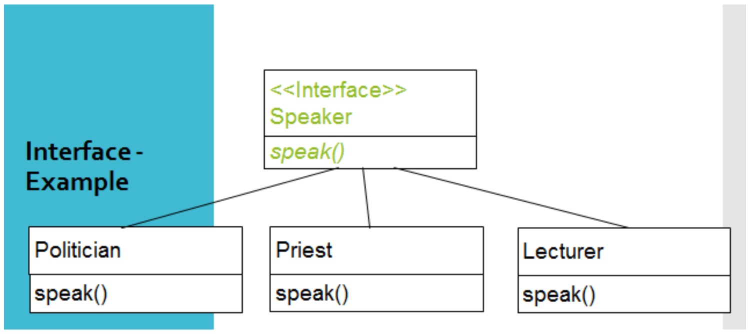
```
public interface MyFirstInterface
{
    void display();

}
```

3. Implement this into a class called “IntefaceImplemented” . Override all the abstract methods. Try to change the value of x inside this method and print the value of x. Is it possible for you to change x? why?

```
public class InterfaceImplemented implements MyFirstInterface
{
    @Override
    public void display()
    {
        x = 20;
        System.out.println(x);
    }
}
```

Exercise 02:

Develop a code base for the following scenario. Recall what we have done at the lecture…



Main

```
public class Practical5
{

    public static void main(String[] args)
    {
        Lecturer obj1=new Lecturer();
        obj1.speak();

        Politician obj2=new Politician();
        obj2.speak();

        Priest obj3=new Priest();
        obj3.speak();
    }
}
```

Politician

```
public class Politician implements Speaker
{
        @Override
        public void speak()
        {
            System.out.println("As a politician, I stand up for your rights ");
        }

}
```

Priest

```
public class Priest implements Speaker
{
   @Override
   public void speak()
   {
       System.out.println("As a priest, I preach");
   }
}
```

Speaker

```
public interface Speaker
{
    void speak();
   
}
```

Lecturer

```
public class Lecturer implements Speaker
{
    @Override
    public void speak()
    {
        System.out.println("As a lecturer, I conduct lectures");
    }
}
```

Exercise 03:

Try following code. What is the outcome? Why?

Class 01:

final class Student {

final int marks = 100;

final void display();

}

Class 02:

class Undergraduate extends Student{}

Student Class

```
final class Student
 {
    final int marks = 100;
    final void display();
}
```

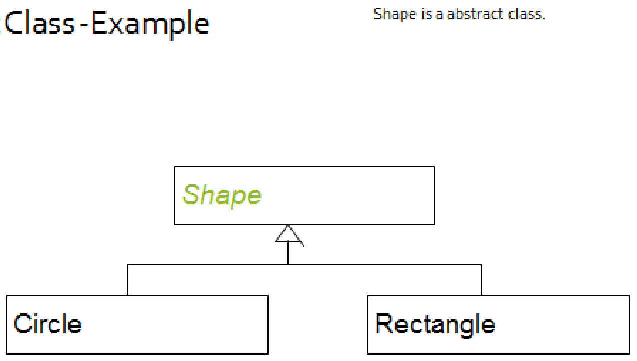
Undergraduate Class

```
class Undergraduate extends Student
{

}
```

Exercise 04:

Develop a code base for the following scenario. Shape class contains an abstract method called “calculateArea” and non-abstract method called “display”. Try to pass required values at the instantiation. Recall what we have done at the lecture…

Abstract Class - Example



Main

```
public class Practical5
{

    public static void main(String[] args)
    {
        Rectangle rectangle = new Rectangle("Rectangle", 10, 5);
        rectangle.display();

        Circle circle = new Circle("Circle", 5);
        circle.display();
    }
}
```

Rectangle

```
public class Rectangle extends Shape
{

    public Rectangle(String name, double length, double width)
   {
        super(name, length, width);
    }

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

Shape

```
abstract class Shape
 {
    private String name;
    private double length;
    private double width;

    public Shape(String name, double length, double width)
  {
        this.name = name;
        this.length = length;
        this.width = width;
    }

    public String getName()
  {
        return name;
    }

    public double getLength()
    {
        return length;
    }

    public double getWidth()
    {
        return width;
    }

    abstract double calculateArea();

    public void display() {
        System.out.println("The area of " + name + " is " + calculateArea());
    }
}
```

Circle

```
public class Circle extends Shape {

    public Circle(String name, double radius) {
        super(name, radius, radius);
    }

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