Practical 05: Inheritance & Abstract Classes

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

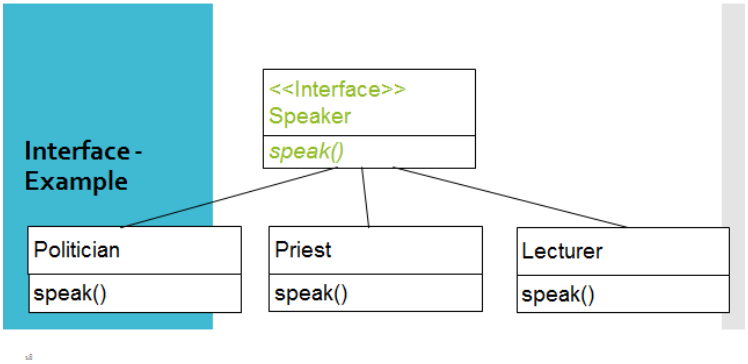
```
package com.mycompany.p5q1;

public class P5Q1

{

   public static void main(String[] args)

   {

   }

}

// InterfaceImplemented class

package com.mycompany.p5q1;

public class InterfaceImplemented implements MyFirstInterface

{

   @Override

   public void display()

   {

      x = 20; // Error: Cannot assign a value to a final variable x

      System.out.println(x);

   }

}
```

Practical 05: Inheritance & Abstract Classes

```
// MyFirstInterface class

package com.mycompany.p5q1;

public interface MyFirstInterface

{

   int x = 10; // Variable declaration

   void display(); // Abstract method declaration

}
```

Exercise 02:

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




```
package com.mycompany.p5q2;

public class P5Q2 {

public static void main(String[] args)

   {

 Lecturer obj1=new Lecturer();

  obj1.speak();

Politician obj2=new Politician();

  obj2.speak();
```

```
 Priest obj3=new Priest();
  obj3.speak();}
}
}
// Priest class
package com.mycompany.p5q2;
public class Priest implements Speaker
{
  @Override
  public void speak()
  {
     System.out.println("As a priest, I preach");
  }
}
// Politician class
package com.mycompany.p5q2;
public class Politician implements Speaker
{    @Override
      public void speak()
      {
         System.out.println("As a politician, I stand up for your rights ");
      }

}
```

```
// Lecturer class
package com.mycompany.p5q2;
public class Lecturer implements Speaker
{
   @Override
   public void speak()
   {
      System.out.println("As a lecturer, I conduct lectures");
   }
}
// Speaker class
package com.mycompany.p5q2;
public interface Speaker
{
   void speak();

}
```

Exercise 03:

Try following code. What is the outcome? Why?

Class 01:

```
final class Student {
```

Class 02:

```
class Undergraduate extends Student{}
```

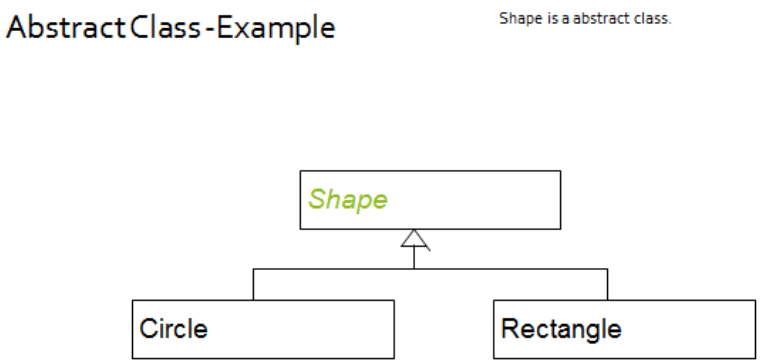
```
        final int marks = 100;
        final void display();
}
```

```
package com.mycompany.p5q3;
public class P5Q3 {
   public static void main(String[] args) {
      System.out.println("Hello World!");
   }
}
// Student class
package com.mycompany.p5q3;
public class Student
{
   final int marks =100;
   //marks=90
   //final value cannot be re assigned
   final void display()
   {
      System.out.println(marks);
   }
}
```

```
// Undergraduate class

package com.mycompany.p5q3;

public class Undergraduate extends Student //final class can never be a sub class

{

   @Override

   void display()

   {

     //A final method cannot be overriding

   }


}
```

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…



```
// P5Q4 class
package com.mycompany.p5q4;
import java.lang.Math;
public class P5Q4
{
   public static void main(String[] args)
   {
      Rectangle rectangle = new Rectangle("Rectangle", 10, 5);
      rectangle.display();
      Circle circle = new Circle("Circle", 5);
      circle.display();
   }
}
// Circle class
package com.mycompany.p5q4;
import java.lang.Math;
public class Circle extends Shape {
   public Circle(String name, double radius) {
      super(name, radius, radius);
   }
   @Override
   double calculateArea() {
      return Math.PI * radius * radius;
   }
}
```

```
// Shape class
package com.mycompany.p5q4;
import java.lang.Math;
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());
   }
```

```
}
// Rectangle class
package com.mycompany.p5q4;
import java.lang.Math;
public class Rectangle extends Shape {
   public Rectangle(String name, double length, double width) {
      super(name, length, width);
   }
   @Override
   double calculateArea() {
      return length * width;
   }
}
```