Java Practical 5

ID-28519

Exercise 01:

```
package com.mycompany.main;
public class InterfaceImplemented implements MyFirstinterface
{
  @Override
  public void display()
  {
     System.out.println("The value of x is: "+x);
  }
}
```

```
package com.mycompany.main;
public interface MyFirstinterface
{
   int x=5;
   void display();
}
```

```
package com.mycompany.main;
public class Main
{

   public static void main(String[] args)
   {
```

```
    InterfaceImplemented rt= new InterfaceImplemented();
    rt.display();
  }
}
```

//Question1 -> answer. No, because inside an interface by default are public static void.

//Question2 -> answer. No, because all the methods are inside an interface are abstract by the default.

//Question3 -> answer. No, if a variable is final, then it cannot change other than the value assigned in intialization.

Exercise 02:

```
package com.mycompany.exercise2;
public class Exercise2 {

  public static void main(String[] args) {
    Speaker sp1 = new Priest();
    sp1.speak("Bless you");
    Speaker sp2 = new Politician();
    sp2.speak("Vote me");
    Speaker sp3 = new Lecturer();
    sp3.speak("Now we are going to do a Java program");
  }
}
```

```
package com.mycompany.exercise2;
public class Politician implements Speaker{
  @Override
```

```
    public void speak(String phrase)
    {
        System.out.println(i + " Politician: " + phrase);
    }
}

package com.mycompany.exercise2;
public interface Speaker {
    int i=100;
    void speak(String line);
}

package com.mycompany.exercise2;
public class Lecturer implements Speaker {
   @Override
   public void speak(String phrase)
   {
      System.out.println(i + " Lecturer: " + phrase);
   }
}

package com.mycompany.exercise2;
public class Priest implements Speaker {
    @Override
    public void speak(String phrase)
    {
        System.out.println(i + " Priest: " + phrase);
```

```
    }
}
```

Exercise 03:

```
package com.mycompany.item;

/**
 *
 * @author User
 */
public class Item {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

```
package com.mycompany.item;
class Undergraduate extends Student
{

}
```

```
package com.mycompany.item;
final class Student
{
    final int marks = 100;
```

```
    final void display();
    //Output is Undergraduate class can not inherit from the student class and in student class it is missing method body or declare abstract.
    //Reason for this is we use the keyword called "final". Final keyword means if we do not want other classes to inherit from a class.
    //The other reason is that, the method body or the declaration of abstract is missing.
}
```

Exercise 04:

```
package com.mycompany.exercise5;
public class Exercise5
{

    public static void main(String[] args) {
        Circle c1 = new Circle(6);
        c1.calculateArea();
        c1.display();
        Rectangle r1 = new Rectangle(2,4);
        r1.calculateArea();
        r1.display();
    }
}
```

```
package com.mycompany.exercise5;
abstract public class Shape
{
   double area;
   abstract double calculateArea();
```

```
  public void display()
  {
     System.out.println(area);
  }
}

class Circle extends Shape
   {
      double r;
      public Circle(int r)
      {
         this.r = r;
      }
      double calculateArea()
      {
         final double p=3.1415;
         area=p*r*r;
         return area;
      }
   }

class Rectangle extends Shape
{
   double x;
   double y;
   public Rectangle(int x, int y)
   {
      this.x = x;
      this.y = y;
```

```
    }
    double calculateArea()
    {
        area=x*y;
        return area;
    }
}
```