Practical 5

1)

```
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);
   }
}

package com.mycompany.p5q1;
public interface MyFirstInterface
{
   int x = 10; // Variable declaration

   void display(); // Abstract method declaration
}
package com.mycompany.p5q1;
public class P5Q1
{

   public static void main(String[] args)
   {

   }
}
```

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

**The value 'x' is declared as ,poblic static final, by default an interface, so whether you explicitly mention these keywords or not, the variable will be treated as a constant(final), static and accessible through the interface. There is no difference between declaring 'int x'; andpublic static final int x; within an interface

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

**Abstract methods are implicitly declaring a method interface. Therefore, Whether you include the 'abstract, keyword or not, the method will be treated as abstract.

2)

```
package com.mycompany.p5q2;

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

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();
   }
}

package com.mycompany.p5q2;
public interface Speaker
{
   void speak();
}
```

```
package com.mycompany.p5q2;
public class Priest implements Speaker
{
  @Override
  public void speak()
  {
     System.out.println("As a priest, I preach");
  }
}
```

```
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 ");
      }
}
```

3)

```
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);
   }
}
```

```
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
   }
}
```

```
package com.mycompany.p5q3;
public class P5Q3
 {
   public static void main(String[] args) {
      System.out.println("Hello World!");
   }
}
```

4)

```
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();
   }
}

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;
   }
}
```

```
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());
    }
}
```