Name: B.S.G.Senani

Student ID: 28189

OOP Using Java – Practical 5

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?

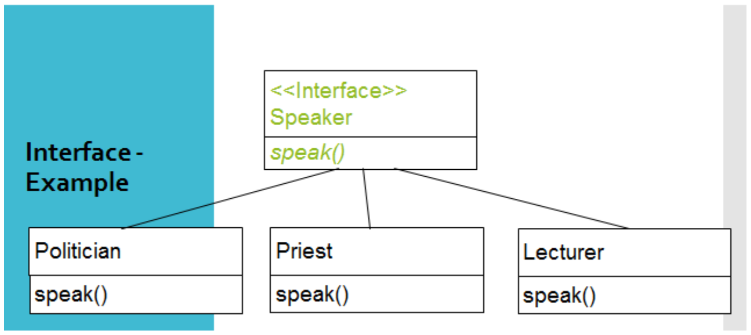
Declare an interface called "MyFirstInterface" with an integer type variable "x" and an abstract method called "display()".

```
// MyFirstInterface declaration

public interface MyFirstInterface {

   int x = 10; // Declaring a variable in an interface (implicitly public, static, and final)



   void display(); // Abstract method in the interface (implicitly public and abstract)

}
```

```
// InterfaceImplemented class implementing MyFirstInterface
public class InterfaceImplemented implements MyFirstInterface {
   // Overriding the abstract method from the interface
   @Override
   public void display() {
      // Trying to change the value of 'x'
      // Uncommenting the line below will cause a compilation error
      // x = 20;

      // Printing the value of 'x'
      System.out.println("Value of x inside the overridden method: " + x);
   }

   public static void main(String[] args) {
      InterfaceImplemented obj = new InterfaceImplemented();
      obj.display();
   }
}
```

Exercise 02:

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



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


class Politician implements Speaker {
  @Override
  public void speak() {
    System.out.println("I am a politician and I am speaking.");
  }
}


class Priest implements Speaker {
  @Override
  public void speak() {
    System.out.println("I am a priest and I am speaking.");
  }
}
```

```
class Lecturer implements Speaker {
  @Override
  public void speak() {
    System.out.println("I am a lecturer and I am speaking.");
  }
}


public class Main {
  public static void main(String[] args) {
    Speaker politician = new Politician();
    politician.speak();


    Speaker priest = new Priest();
    priest.speak();


    Speaker lecturer = new Lecturer();
    lecturer.speak();
  }
}
```

Output:

```
I am a politician and I am speaking.
I am a priest and I am speaking.
I am a lecturer and I am speaking.
```

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{}
```

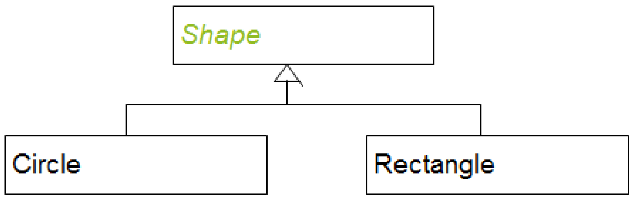
```
final class Student {

  final int marks = 100;

  final void display() {

    System.out.println("The marks of the student are: " + marks);

  }

}


class Undergraduate extends Student {

  // This method does not modify the marks or display() methods.

  public void getGrade() {

    System.out.println("The student's grade is: " + (marks / 100));

  }

}
```

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

Shape is a abstract class.



```
abstract class Shape {

  abstract void calculateArea();

  void display() {

    System.out.println("This is a shape.");

  }

}


class Circle extends Shape {

  private double radius;


  public Circle(double radius) {

    this.radius = radius;

  }
```

```
@Override
  void calculateArea() {
    double area = Math.PI * radius * radius;
    System.out.println("The area of the circle is: " + area);
  }
}


class Rectangle extends Shape {
  private double width;
  private double height;


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

```
@Override
  void calculateArea() {
    double area = width * height;
    System.out.println("The area of the rectangle is: " + area);
  }
}


public class Main {
  public static void main(String[] args) {
    Circle circle = new Circle(5);
    circle.calculateArea();


    Rectangle rectangle = new Rectangle(10, 20);
    rectangle.calculateArea();
  }
}
```