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?

Answers

```
public interface MyFirstInterface {
   int x = 10; // Same as 'public static final int x = 10;'
}
```

Why?

- In an interface, all variables are implicitly public, static, and final. Therefore, whether you explicitly declare the variable with public static final keywords or not, it will be treated the same. Both approaches will result in a public static final variable in the interface.

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

Answers

```
public interface MyFirstInterface {
   void display(); // Same as 'public abstract void display();'
}
```

Why?

- In an interface, all methods are implicitly abstract and public. Therefore, whether you explicitly declare the method with the abstract keyword or not, it will be treated the same. Both approaches will result in an abstract method in the interface.

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?

Answers

```
public class InterfaceImplemented implements MyFirstInterface {
   @Override
   public void display() {

      System.out.println("Value of x: " + x); // x is accessible in the
implementing class
   }

   public static void main(String[] args) {
      InterfaceImplemented obj = new InterfaceImplemented();
      obj.display(); // Output: Value of x: 10
   }
}
```

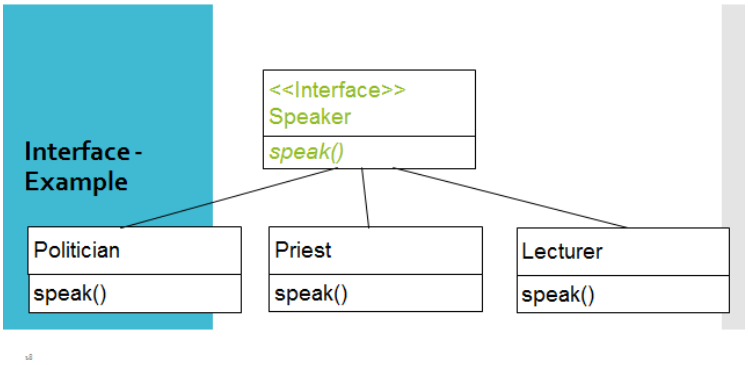
Why?

- in the InterfaceImplemented class, we implement the display() method, but we can't change the value of x inside this method. The reason is that all variables in an interface are implicitly final, which means they are constants and their values cannot be changed once set.

However, we can access the value of x in the implementing class and use it. In the display() method, we print the value of x, which will be 10, the default value defined in the MyFirstInterface

Exercise 02:

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



Answers

```
public abstract class Speaker {
   public abstract void speak();
}
public class Politician extends Speaker {
   @Override
   public void speak() {
      System.out.println("I am a politician, and I am here to speak to you about my policies.");
   }
}
```

```
public class Priest extends Speaker {
   @Override
```

```
    public void speak() {
        System.out.println("I am a priest, and I am here to speak to you about the importance of faith.");
    }
}

public class Lecturer extends Speaker {
    @Override
    public void speak() {
        System.out.println("I am a lecturer, and I am here to speak to you about the importance of education.");
    }

    public static void main(String[] args) {
        Politician politician = new Politician();
        Priest priest = new Priest();
        Lecturer lecturer = new Lecturer();

        politician.speak();
        priest.speak();
        lecturer.speak();
    }
```

}

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{}
```

Answers

The outcome of in this code compilation error. The reason for this is that the Student class is declared as final, which means that it cannot be inherited from. However, the Undergraduate class inherits from the Student class, which is not allowed.

```
final class Student {
   final int marks = 100;
   final void display() {
      System.out.println("Marks: " + marks);
   }
}
class Undergraduate extends Student() {
}
```

Correct code

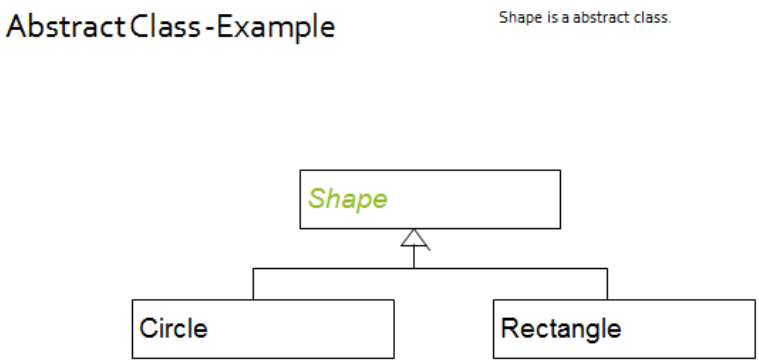
Why?

The final keyword prevents the Student class from being inherited from. This is because the final keyword is used to indicate that a class cannot be changed. If the Student class were not declared as final, then the Undergraduate class would be able to inherit from it.

However, the Student class is declared as final, so the Undergraduate class cannot inherit from it. This is why the code will not compile.

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…



Answers

```
abstract class Shape {

   abstract double calculateArea();


   void display() {

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

   }
```

```
}

class Circle extends Shape {
    private double radius;
    Circle(double radius) {
        this.radius = radius;
    }
    @Override
    double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Rectangle extends Shape {

    private double width;
    private double height;

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

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

public class Main {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        Rectangle rectangle = new Rectangle(10, 20);

        circle.display();
        System.out.println("The area of the circle is: " + circle.calculateArea());

        rectangle.display();
        System.out.println("The area of the rectangle is: " +
rectangle.calculateArea());
    }
}
```