Practical 06: 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?

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

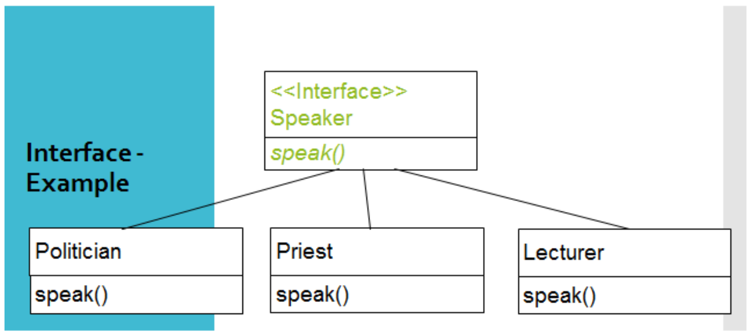
```
public class InterfaceImplemented implements MyFirstInterface {
   @Override
   public void display() {

      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();
    }
}
```