**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?**

public interface MyFirstInterface {

int x = 10;

// With or without public static final keywords

void display();

}

In Java interfaces, all variables are implicitly public, static, and final. Therefore, declaring the variable "x" with or without the **public static final** keywords will be the same. Both approaches make "x" a public static final constant variable. Any class implementing this interface will have access to this variable as a constant, and its value cannot be changed within the implementing class.

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

public interface MyFirstInterface {

void display();

// With or without abstract keyword

}

In Java interfaces, all methods are implicitly abstract and public, so explicitly adding the abstract keyword is optional. Both approaches are equivalent, and the method "display()" will be treated as an abstract method in either case. Any class implementing this interface will need to provide an implementation for the "display()" method.

1. **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?**

public class InterfaceImplemented implements MyFirstInterface {

int x = 20; // This is a new variable in the class, not related to the interface's "x"

public void display() {

x = 30; // Changing the value of the class-level "x" variable, not the interface-level "x"

System.out.println("Value of x inside the implemented class: " + x);

}

public static void main(String[] args) {

InterfaceImplemented obj = new InterfaceImplemented();

obj.display();

}

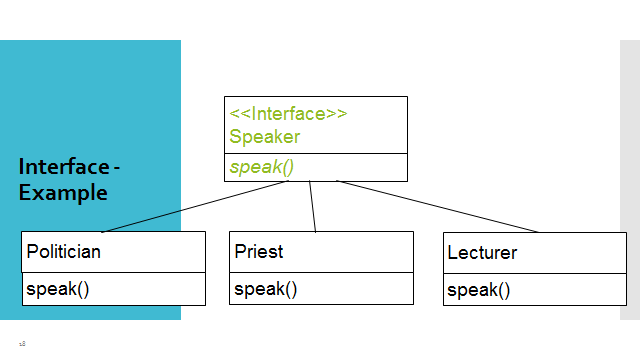
}

In the class "InterfaceImplemented," the variable "x" declared inside the class is different from the variable "x" declared in the interface "MyFirstInterface." When you implement an interface in a class, any variables declared inside the class are independent of the variables in the interface, even if they share the same name.

So, the value of "x" inside the "display()" method refers to the class-level "x," which is set to 30, and it is indeed possible to change the value of this class-level "x" variable. However, changing this value does not affect the interface-level "x" constant, which remains 10.

**Exercise 02:**

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



**Exercise 03:**

Try following code. What is the outcome? Why?

Class 01: Class 02:

final class Student { class Undergraduate extends Student{}

final int marks = 100;

final void display();

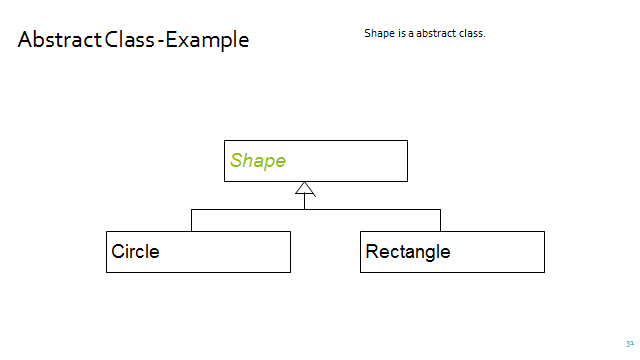
}

Apologies, but I made a mistake in my previous response. The provided code will not compile due to the incorrect declaration of the **final** method without an implementation in the **Student** class.

Since the code cannot be compiled, there will be no output to display. The compilation errors in Class 01 will prevent the program from executing and producing any output.

**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 Shape {

public abstract double calculateArea();

public void display() {

System.out.println("Area: " + calculateArea());

}

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

class Rectangle extends Shape {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public double calculateArea() {

return length \* width;

}

}

public class Main {

public static void main(String[] args) {

// Instantiate a Circle and calculate/display its area

double circleRadius = 5.0;

Circle circle = new Circle(circleRadius);

System.out.println("Circle Area:");

circle.display();

// Instantiate a Rectangle and calculate/display its area

double rectangleLength = 10.0;

double rectangleWidth = 8.0;

Rectangle rectangle = new Rectangle(rectangleLength, rectangleWidth);

System.out.println("Rectangle Area:");

rectangle.display();

}

}