Exercise 01

public interface MyFirstInterface {

int x = 10;

void display();

}

public class InterfaceImplemented implements MyFirstInterface {

@Override

public void display() {

// Trying to change the value of 'x'

// This will result in a compilation error.

// x = 20;

System.out.println("Value of x: " + x);

}

}

1. // Both declarations are equivalent in an interface

int x = 10; // Implicitly public, static, and final

public static final int x = 10; // Same as above, but explicitly written

In an interface, all variables are implicitly public, static, and final. It means that you cannot declare a non-static instance variable in an interface, and all variables are constants (i.e., their values cannot be changed once set). Therefore, when declaring the variable x in the interface MyFirstInterface, it will always be implicitly public, static, and final, regardless of whether you explicitly write those keywords or not

2. // Both declarations are equivalent in an interface

void display(); // Implicitly public and abstract

public abstract void display(); // Same as above, but explicitly written

in an interface, all methods are implicitly public and abstract. Therefore, when declaring the abstract method display() in the interface MyFirstInterface, it is always implicitly public and abstract, regardless of whether you explicitly write those keywords or not

3. @Override

public void display() {

// This line will cause a compilation error.

// x = 20; // Cannot assign a value to 'x' because it is final.

System.out.println("Value of x: " + x);

}

In the InterfaceImplemented class, we override the display() method from the MyFirstInterface. However, since the variable x in the interface is implicitly final, its value cannot be changed after initialization.

Exercise 02

Speaker.java

public interface Speaker {

void speak();

}

Politician.java

public class Politician implements Speaker {

@Override

public void speak() {

System.out.println("Politician is speaking about politics.");

}

}

Priest.java

public class Priest implements Speaker {

@Override

public void speak() {

System.out.println("Priest is delivering a religious sermon.");

}

}

Lecturer.java

public class Lecturer implements Speaker {

@Override

public void speak() {

System.out.println("Lecturer is giving an academic lecture.");

}

}

TestSpeaker.java

public class TestSpeaker {

public static void main(String[] args) {

Speaker politician = new Politician();

Speaker priest = new Priest();

Speaker lecturer = new Lecturer();

System.out.println("Politician:");

politician.speak();

System.out.println("\nPriest:");

priest.speak();

System.out.println("\nLecturer:");

lecturer.speak();

}

}

Output

Politician:

Politician is speaking about politics.

Priest:

Priest is delivering a religious sermon.

Lecturer:

Lecturer is giving an academic lecture.

Exercise 03

Class 01

final class Student {

final int marks = 100;

final void display() {

System.out.println("Display method in Student class.");

}

}

Class 02

class Undergraduate extends Student {

// This is a subclass of Student

}

Since the Student class is marked as final, it cannot be extended or subclassed, and any attempt to do so will result in a compilation error

Exercise 04

AbstractClass.java

abstract class Shape {

// Abstract method - to be implemented by subclasses

abstract double calculateArea();

// Non-abstract method

void display() {

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

}

}

Circle.java

class Circle extends Shape {

private double radius;

// Constructor

Circle(double radius) {

this.radius = radius;

}

@Override

double calculateArea() {

return Math.PI \* radius \* radius;

}

}

Rectangle.java

class Rectangle extends Shape {

private double length;

private double width;

// Constructor

Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

@Override

double calculateArea() {

return length \* width;

}

}

TestShape.java

public class TestShape {

public static void main(String[] args) {

// Create a circle with radius 5

Circle circle = new Circle(5);

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

circle.display();

// Create a rectangle with length 4 and width 3

Rectangle rectangle = new Rectangle(4, 3);

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

rectangle.display();

}

}

Output

Area of Circle: 78.53981633974483

This is a shape.

Area of Rectangle: 12.0

This is a shape.