TELUSKO

21-What is Inheritance

What is Inheritance?

Inheritance in Java is a mechanism by which one class (called a child or subclass) can inherit the properties and behaviours (fields and methods) of another class (called a parent or superclass). This is a fundamental concept in Object-Oriented Programming (OOP), allowing for the creation of new classes based on existing ones. It promotes reusability and can help reduce redundancy in your code.

By using inheritance, you can:

- Reuse existing code from parent classes.
- Add or modify functionalities in child classes.
- Maintain a clear and organized structure in your codebase.

Key Concepts in Inheritance

1. **Class**:
    - A blueprint from which individual objects are created. It defines a group of objects with common properties.
2. **Subclass/Child Class**:
    - A class that inherits the properties and methods from another class. It is also known as a derived class, extended class, or child class.
3. **Superclass/Parent Class**:
    - The class from which a subclass inherits properties and methods. It is also referred to as a base class or parent class.
4. **Reusability**:
    - The concept of reusing fields and methods of an existing class when creating a new class. This is one of the primary benefits of inheritance, allowing for more efficient and maintainable code.

Syntax of Java Inheritance

```
class SubclassName extends SuperclassName {
    // fields and methods
}
```

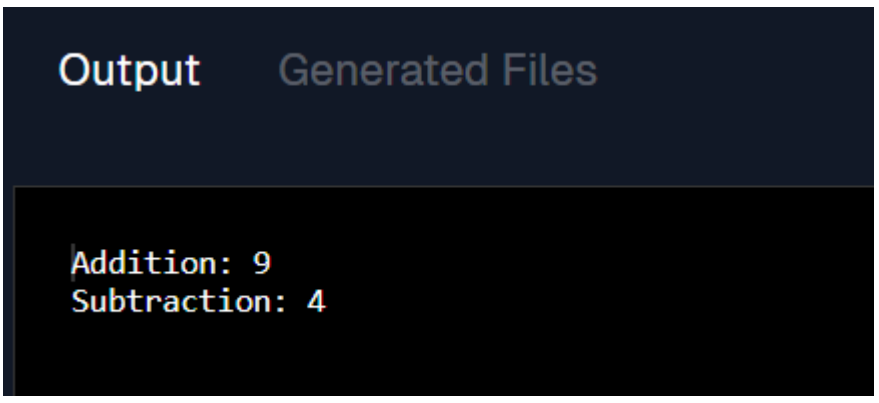
TELUSKO

Example: Simple Calculator

Let's start with a simple calculator class that performs basic arithmetic operations like addition and subtraction:

```
class Calc {
    public int add(int n1, int n2) {
        return n1 + n2;
    }

    public int sub(int n1, int n2) {
        return n1 - n2;
    }
}

public class Demo {
    public static void main(String[] args) {
        Calc obj = new Calc();
        int r1 = obj.add(4, 5);
        int r2 = obj.sub(7, 3);
        System.out.println("Addition: " + r1 + "\nSubtraction: " + r2);
    }
}
```

Output:



Explanation:

This example demonstrates a simple calculator that performs addition and subtraction. The Calc class contains two methods: add and sub. The Demo class creates an instance of Calc and uses it to perform the operations.

Extending the Calculator: Advanced Calculator

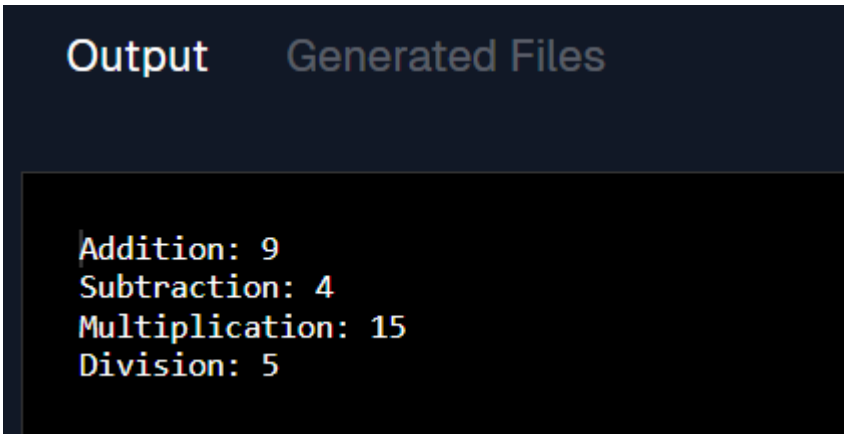
Suppose you need a calculator with advanced features like multiplication and division. You have two options:

1. Add more methods to the Calc class.
2. Create a new class (AdvCalc) that extends Calc and adds the extra features.

Option 2 is more feasible as it avoids code redundancy and keeps the code organized.

```
class Calc {
    public int add(int n1, int n2) {
        return n1 + n2;
    }

    public int sub(int n1, int n2) {
        return n1 - n2;
    }
}
class AdvCalc extends Calc {
    public int mul(int n1, int n2) {
        return n1 * n2;
    }

    public int div(int n1, int n2) {
        return n1 / n2;
    }
}

public class Demo {
    public static void main(String[] args) {
        AdvCalc obj = new AdvCalc();
        int r1 = obj.add(4, 5);  // Inherited method
        int r2 = obj.sub(7, 3);  // Inherited method
        int r3 = obj.mul(5, 3);  // New method
        int r4 = obj.div(15, 3); // New method

        System.out.println("Addition: " + r1 + "\nSubtraction: " + r2 +
                           "\nMultiplication: " + r3 + "\nDivision: " + r4);
    }
}

```

Output:



Explanation:

In this example, the AdvCalc class extends the Calc class. It inherits the add and sub methods and introduces two new methods: mul (for multiplication) and div (for division). This demonstrates the power of inheritance, where the AdvCalc class can reuse the existing methods from Calc while adding its own functionalities.