TASK 7: Inheritance & Polymorphism – Employee Payroll System.

What is Inheritance in Java?

Inheritance is an Object-Oriented Programming (OOP) concept in which one class acquires the properties and behaviors of another class.
The class that is inherited from is called the parent (or superclass), and the class that inherits is called the child (or subclass).

In Java, inheritance is implemented using the extends keyword.

Why Inheritance is Needed:-

- Reuse existing code
- Reduce duplication
- Represent real-world relationships
- Support polymorphism
- Improve maintainability and scalability

Basic Syntax of Inheritance:-

```
class Parent {
   int x = 10;

   void show() {
      System.out.println("This is parent class");
   }
}

class Child extends Parent {
   void display() {
      System.out.println("This is child class");
   }
}
```

Types of Inheritance :-

1. Single Inheritance

One child inherits one parent.

```
class A {}

class B extends A {}
```

2. Multilevel Inheritance

A class inherits from a child class.

```
class A {}

class B extends A {}

class C extends B {}
```

3. Hierarchical Inheritance

Multiple classes inherit from one parent.

```
class A {}

class B extends A {}

class C extends A {}
```

4. Multiple Inheritance (Not Supported with Classes)

Java does not support multiple inheritance using classes to avoid ambiguity (Diamond Problem). However, it is supported using interfaces.

```
interface A {}

interface B {}

class C implements A, B {}
```

Method Overriding in Inheritance

Child class can provide its own implementation of a parent method.

```
class Animal {
    void sound() {
        System.out.println("Animal makes sound");
    }
}
```

```
class Dog extends Animal {
   @Override
   void sound() {
      System.out.println("Dog barks");
   }
}
```

This enables runtime polymorphism.

Advantages of Inheritance:-

- Code reusability
- Faster development
- Easy maintenance
- Logical class hierarchy
- Enables polymorphism

Disadvantages of Inheritance:-

- Tight coupling
- Changes in parent may affect child
- Increases complexity if overused

Polymorphism :-

Polymorphism is an Object-Oriented Programming (OOP) concept where one method or object can take many forms.

The word polymorphism comes from:

- Poly = many
- Morph = forms

In Java, polymorphism allows the same method name to perform different actions depending on the object.

Types of Polymorphism in Java

1. Compile-Time Polymorphism (Method Overloading)

2. Runtime Polymorphism (Method Overriding)

Compile-Time Polymorphism (Method Overloading):-

When multiple methods have the same name but different parameters, it is called method overloading.
The decision is made at compile time.

Example:-

```
class Calculator {


   int add(int a, int b) {

      return a + b;

   }


   int add(int a, int b, int c) {

      return a + b + c;

   }


   double add(double a, double b) {

      return a + b;

   }

}
```

Runtime Polymorphism :-

When a child class provides its own implementation of a method already defined in the parent class, it is called method overriding.
The method call is resolved at runtime, not compile time

Example:-

```
class Employee {

   double calculateSalary() {

      return 0;

   }

}
```

```
class FullTimeEmployee extends Employee {
   @Override
   double calculateSalary() {
      return 50000;
   }
}

class PartTimeEmployee extends Employee {
   @Override
   double calculateSalary() {
      return 20000;
   }
}
public class Main {
   public static void main(String[] args) {

      Employee e1 = new FullTimeEmployee();
      Employee e2 = new PartTimeEmployee();

      System.out.println(e1.calculateSalary());
      System.out.println(e2.calculateSalary());
   }
}
```

Output:-

```
50000.0
20000.0
```

Advantages of Polymorphism

- Code flexibility
- Loose coupling
- Easy scalability

- Improves readability
- Supports Open/Closed Principle

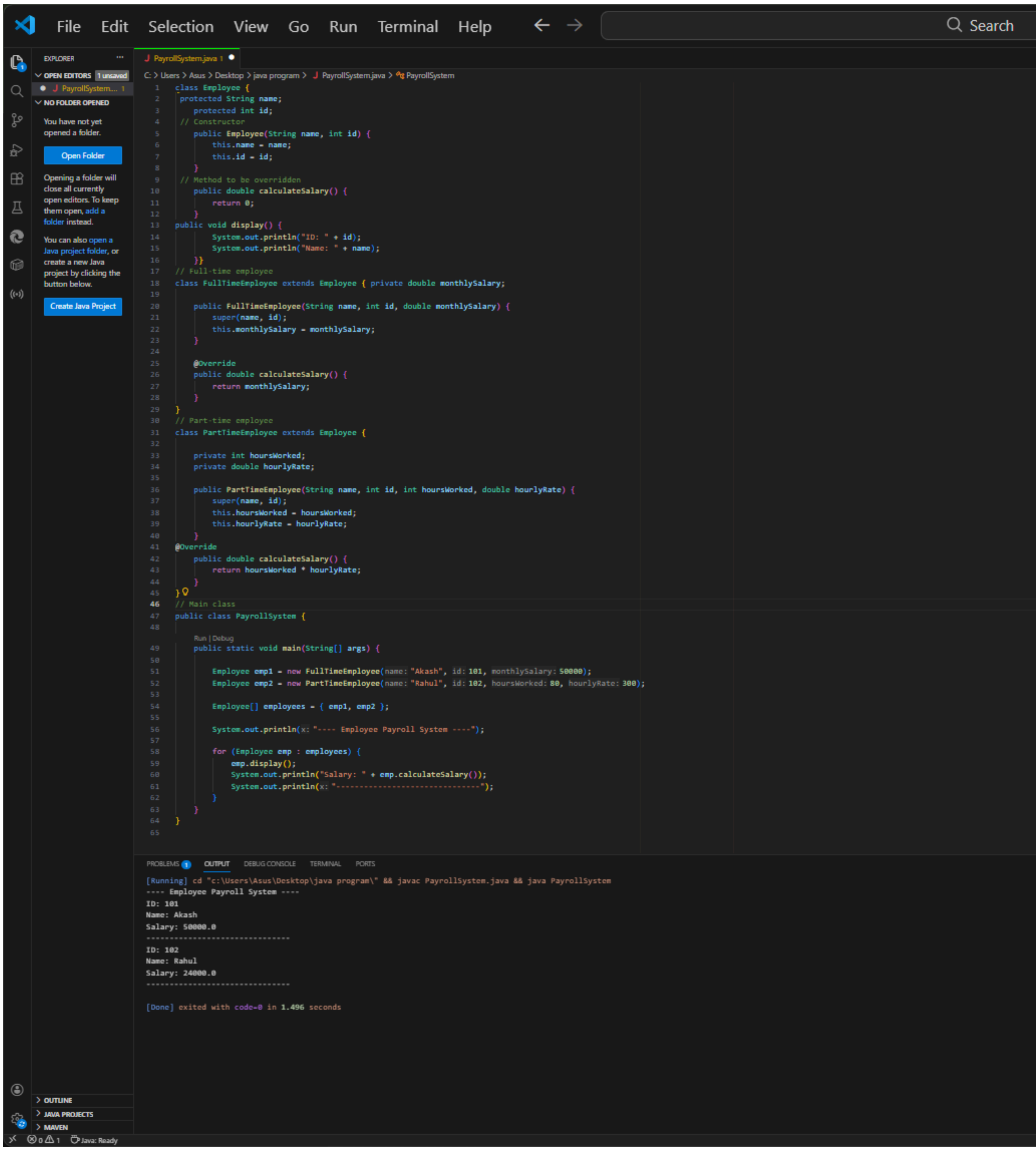
What is an Employee Payroll System?

An Employee Payroll System is a program that calculates and displays employee salaries based on their job type.
In Java, this is a perfect use case for Inheritance and Polymorphism, because different employees have different salary rules, but they all belong to the same category Employee.

Core Idea:-

- Create a base class Employee
- Create child classes like FullTimeEmployee and PartTimeEmployee
- Override the salary calculation method
- Use parent class reference to achieve runtime polymorphism.

Employee Payroll System:-



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can also open a Java project folder, or create a new Java project by clicking the button below.

```java
class Employee {
 protected String name;
    protected int id;
 // Constructor
    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }
 // Method to be overridden
    public double calculateSalary() {
        return 0;
    }
public void display() {
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
    }}
// Full-time employee
class FullTimeEmployee extends Employee { private double monthlySalary;

    public FullTimeEmployee(String name, int id, double monthlySalary) {
        super(name, id);
        this.monthlySalary = monthlySalary;
    }

    @Override
    public double calculateSalary() {
        return monthlySalary;
    }
}
// Part-time employee
class PartTimeEmployee extends Employee {

    private int hoursWorked;
    private double hourlyRate;

    public PartTimeEmployee(String name, int id, int hoursWorked, double hourlyRate) {
        super(name, id);
        this.hoursWorked = hoursWorked;
        this.hourlyRate = hourlyRate;
    }
@Override
    public double calculateSalary() {
        return hoursWorked * hourlyRate;
    }
}
// Main class
public class PayrollSystem {

    public static void main(String[] args) {

        Employee emp1 = new FullTimeEmployee(name: "Akash", id: 101, monthlySalary: 50000);
        Employee emp2 = new PartTimeEmployee(name: "Rahul", id: 102, hoursWorked: 80, hourlyRate: 300);

        Employee[] employees = { emp1, emp2 };

        System.out.println(x: "---- Employee Payroll System ----");

        for (Employee emp : employees) {
            emp.display();
            System.out.println("Salary: " + emp.calculateSalary());
            System.out.println(x: "------------------------------");
        }
    }
}
```

```
[Running] cd "c:\Users\Asus\Desktop\java program\" && javac PayrollSystem.java && java PayrollSystem
---- Employee Payroll System ----
ID: 101
Name: Akash
Salary: 50000.0
------------------------------
ID: 102
Name: Rahul
Salary: 24000.0
------------------------------

[Done] exited with code=0 in 1.496 seconds
```