Project Name: Employee Payroll System Project in Java.

Introduction: This Project contains a comprehensive example of building a terminal-based Employee Payroll System using Object-Oriented Programming (OOP) principles.Through this project, i use how to design and implement abstract classes, concrete subclasses, and explore inheritance, encapsulation, abstraction, and polymorphism in Java.

Code Structure and OOP Concepts:

1. Employee Class:

- Attributes (name and id):

These represent the basic information about an employee. They are initialized through the constructor.

- Getter Methods (getName() and getId()):

Provide access to the private attributes.

- Abstract Method (calculateSalary()):

Requires subclasses (FullTimeEmployee and PartTimeEmployee) to provide their own implementation.

- toString() Method:

Generates a formatted String representation of the employee, including name, id, and calculated salary.

2. FullTimeEmployee Class (Subclass of Employee):

- Attribute (monthlySalary):

Represents the monthly salary for full-time employees.

- Constructor:

Initializes the attributes inherited from the Employee class and the specific attribute for full-time employees.

- Method Override (calculateSalary()):

Provides a concrete implementation of how the salary is calculated for full-time employees.

3. PartTimeEmployee Class (Subclass of Employee):

- Attributes (hoursWorked and hourlyRate):

Represent the hours worked and hourly rate for part-time employees.

- Constructor:

Initializes the attributes inherited from the Employee class and the specific attributes for part-time employees.

- Method Override (calculateSalary()):

Provides a concrete implementation of how the salary is calculated for part-time employees.

4. PayrollSystem Class:

- Attribute (employeeList):

A list to store instances of the Employee class.

- Constructor:

Initializes the employeeList.

- addEmployee(Employee employee) Method:

Adds an instance of the Employee class (or its subclasses) to the employeeList.

- removeEmployee(int id) Method:

Removes an employee from the employeeList based on the provided ID.

- displayEmployees() Method:

Prints the details of all employees in the employeeList.

5. Main Class:

- main(String[] args) Method:

Acts as the entry point of the program.
Instantiates the **PayrollSystem.**
Creates instances of **FullTimeEmployee and PartTimeEmployee.**
Adds employees to the **PayrollSystem.**
Displays the **employee details.**
Removes an **employee.**
Displays the **remaining employee details.**

How It All Works Together:

1. Object Instantiation:

- Instances of classes are created in the Main class.
- FullTimeEmployee and PartTimeEmployee instances are added to the PayrollSystem.

2.Method Invocation:

- calculateSalary() method is invoked when displaying employee details, and the appropriate implementation (full-time or part-time) is used.

3.Polymorphism:

- The PayrollSystem class works with instances of the Employee class, demonstrating polymorphism.
- The calculateSalary() method is polymorphic, with different behaviors in FullTimeEmployee and PartTimeEmployee instances.

4.Composition:

- The PayrollSystem class is composed of instances of the Employee class, demonstrating composition.

5.User Interaction:

- The main method simulates interactions by adding employees, displaying details, removing an employee, and displaying the updated details.

In conclusion,my Java code presents a robust implementation of an employee payroll system using Object-Oriented Programming (OOP) principles. The use of abstraction, inheritance, encapsulation, polymorphism, and composition contributes to a well-organized and modular code structure.

The Employee class serves as an abstract blueprint, defining common attributes and an abstract method for salary calculation. Subclasses, FullTimeEmployee and PartTimeEmployee, extend this base class, providing concrete implementations for calculating salaries based on their specific employment types.

The PayrollSystem class acts as a container, utilizing composition to manage a list of diverse employees. This allows for flexibility and scalability in handling different employee types within the system. The code demonstrates encapsulation by restricting access to attributes and promoting the use of getter methods.

The Main class serves as the entry point of the program, showcasing the instantiation of the payroll system, addition of employees, and interactions with the system such as displaying employee details and removing an employee.