Payroll System explanation

1. Introduction

This program is a simple payroll management example built with Java. It demonstrates how object-oriented programming concepts such as abstraction, inheritance, and polymorphism can be applied to model employees and their salaries. The system can handle both full-time and part-time employees, calculate their pay, and manage them in a payroll list.

2. Class Design

Employee (Abstract Class)

The Employee class is the base for all types of employees. It stores common details like name and id. It also declares an abstract method **calculateSalary()** which forces subclasses to provide their own salary calculation logic. The toString() method is overridden to print employee details along with their salary.

FullTimeEmployee

This class extends Employee and represents workers with a fixed monthly salary. It has one extra field, monthlySalary, and its calculateSalary() method simply returns that value.

PartTimeEmployee

This class also extends Employee but models employees who are paid by the hour. It keeps track of hoursWorked and hourlyRate. The salary is calculated by multiplying these two values.

PayrollSystem

This class manages a list of employees. It provides methods to add new employees, remove them by ID, and display all employees currently in the system. Internally, it uses an ArrayList to store the employee objects.

Empayroll (Main Class)

The main method demonstrates how the system works:

- A PayrollSystem object is created.
- A full-time employee and a part-time employee are added.
- The list of employees is displayed.
- One employee is removed by ID.
- The remaining employees are displayed again.

3. Program Flow

1. Create the payroll system.
2. Add employees (both full-time and part-time).
3. Show the initial list of employees.
4. Remove one employee.
5. Show the updated list.

4. Example Output

```
Initial Employee Details:
Employee [name=John Doe, id=101, salary=5000.0]
Employee [name=Jane Smith, id=102, salary=450.0]

Removing Employee...

Remaining Employee Details:
Employee [name=Jane Smith, id=102, salary=450.0]
```

5. Key Takeaways

- Abstract classes define a common template but leave specific behavior (like salary calculation) to subclasses.
- Polymorphism allows different employee types to be treated uniformly while still using their own salary logic.
- Encapsulation is used by keeping fields private and exposing them through getters.
- Inheritance helps avoid code duplication by sharing common properties in the base class.
- Collections like ArrayList make it easy to manage groups of employees.