Employee Management System

Key Classes and Their Responsibilities

1.PayrollPage Class

- Responsibility: Handles the user interface for managing payroll details and updates. It collects user input, validates it, and updates the employee data in the database.
- Important Methods:
  - actionPerformed(): Handles button clicks to save or validate the data.
  - isValidInput(): Validates user input for correctness.
  - clearFields(): Clears input fields after saving.

2.PayrollFactory Class

- Responsibility: Creates and returns the correct payroll calculation strategy based on the employee type (e.g., full-time or part-time).
- Important Method:
  - getPayrollStrategy(String employeeType): Returns the appropriate payroll strategy.

3.PayrollStrategy Interface

- Responsibility: Defines a contract for different payroll calculation strategies.
- Important Method:
  - calculateSalary(double baseSalary): Calculates salary based on the strategy.

4.PermanentStrategy Class

- Responsibility: Implements salary calculation for permanent employees.
- Important Methods:
  - calculateSalary(double baseSalary): Calculates salary with bonus and deductions for permanent employees.

5.ContractStrategy Class

- Responsibility: Implements salary calculation for contract employees.
- Important Methods:
  - calculateSalary(double baseSalary): Calculates salary with bonus and deductions for contract employees.

6.Employee Class

- Responsibility: Represents an employee with personal details like name, salary, phone, etc.
- Important Methods:
  - clone(): Creates a copy of the employee object.
  - toString(): Provides a string representation of the employee.

7.RealEmployeeData Class

- Responsibility: Handles the real database operations for updating employee data.
- Important Method:
  - updateEmployee(String empId, Employee updatedEmployee): Updates employee details in the database.

8.ProxyEmployeeData Class

- Responsibility: Acts as a proxy to the RealEmployeeData class. It can add extra logic like validation before delegating the update operation.
- Important Method:
  - updateEmployee(String empId, Employee updatedEmployee): Forwards the update request to RealEmployeeData.

9.UpdateEmployee Class

- Responsibility: Provides the GUI for updating employee details. Fetches existing employee details from the database and allows modification.
- Important Methods:
  - fetchEmployeeDetails(): Fetches and displays employee details.
  - actionPerformed(): Handles the "Update" and "Back" button clicks.

Design Patterns Used

01 Factory Pattern

02 Singletone Pattern

03 Prototype Pattern

04 Proxy Pattern

05 Builder Pattern

The Details of patterns

1.Factory Pattern

- Purpose: Used in the PayrollFactory to create different payroll strategies based on employee types (full-time or part-time).
- Benefit: Centralizes the object creation logic and provides flexibility for future extensions.

2.Singletone Pattern

- Purpose: The Singleton Pattern ensures that a class has only one instance and provides a global point of access to it. It is used for managing shared resources, like database connections or logging services.
- Benefit: Prevents multiple instances of a class that could result in resource conflicts or redundant operations (e.g., multiple database connections).

1.Builder Pattern

Purpose:

- The Builder Pattern simplifies creating complex objects by separating the construction process.
- Benefit:
- It improves code readability and flexibility, preventing errors from complex constructors.

4.Prototype Pattern

- Purpose: Used in the Employee class to clone employee objects.
- Benefit: Enables easy copying of employee objects without needing to recreate them from scratch.

5.Proxy Pattern

- Purpose: Used in the ProxyEmployeeData class to add validation or other logic before delegating the operation to RealEmployeeData.
- Benefit: Allows additional processing or control over the real data update logic.

Thank You