**Payroll System Design and Implementation Report**

**1. Introduction**

The Payroll System is designed to streamline employee management, payroll calculations, and salary processing. The system applies object-oriented programming principles such as **inheritance**, **interfaces**, **polymorphism**, and **generics**. Additionally, it incorporates modern programming techniques like **lambda expressions** and **collections** to enhance efficiency and readability.

**2. Use Case Diagram**

The Use Case Diagram highlights the system's core functionalities and the interactions between users and the system.

**Actors:**

1. **Admin**:
   * Adds, updates, and removes employee records.
   * Processes payroll for all employees.
2. **Employee**:
   * Views payslips.
   * Views personal details.

**Use Cases:**

* **Admin**:
  + Add Employee
  + Update Employee
  + Remove Employee
  + Process Payroll
* **Employee**:
  + View Payslip
  + View Personal Details

**Diagram Description:**

The Admin is responsible for managing employees and payroll operations, while the Employee interacts with the system to view personal information and payslips. This design ensures **separation of duties** and clearly defines the system's boundaries.

**3. Class Diagram**

The Class Diagram illustrates the structure of the system, showing the relationships between classes and interfaces.

**Classes and Relationships:**

1. **User (Interface)**:
   * Defines operations for managing employees and payroll.
   * Implemented by the Admin class.
2. **Employee (Abstract Class)**:
   * Represents shared attributes: id, name, email.
   * Includes the abstract method calculatePay() to be implemented by subclasses.
3. **Subclasses**:
   * **HourlyEmployee**: Calculates pay based on hourlyRate and hoursWorked.
   * **SalariedEmployee**: Calculates pay using a fixed monthlySalary.
4. **Admin**:
   * Implements the User interface to manage employees and payroll.

**Diagram Summary:**

The system makes use of the following:

* **Inheritance**: HourlyEmployee and SalariedEmployee inherit from the Employee class.
* **Interfaces**: The User interface is implemented by the Admin class.
* **Polymorphism**: The calculatePay() method is overridden in subclasses to provide specific salary calculations.
* **Generics**: Collections like Map and List are used for efficient data management.

**4. Code Implementation**

**Key Features:**

1. **Inheritance**:  
   The Employee class is abstract and extended by HourlyEmployee and SalariedEmployee.

public abstract class Employee {

private String id, name, email;

public abstract double calculatePay();

}

public class HourlyEmployee extends Employee {

private double hourlyRate; private int hoursWorked;

@Override public double calculatePay() { return hourlyRate \* hoursWorked; }

}

1. **Interfaces**:  
   The User interface defines essential methods for employee and payroll management.

public interface User {

void addEmployee(Map<String, Employee> employees, Employee emp);

void processPayroll(List<Employee> employees);

}

1. **Polymorphism**:  
   The calculatePay() method is overridden in subclasses to calculate salaries differently.
2. **Generics and Collections**:  
   The Admin class uses Map and List collections to manage employees efficiently.

Map<String, Employee> employeeMap = new HashMap<>();

List<Employee> employeeList = new ArrayList<>();

1. **Lambda Expressions**:  
   Lambda expressions simplify payroll processing by enabling concise iteration.

employees.forEach(e -> System.out.println(e.getName() + " Salary: " + e.calculatePay()));

**5. Conclusion**

The Payroll System successfully applies core object-oriented programming principles:

* **Inheritance** and **Polymorphism** provide flexibility and extensibility.
* **Interfaces** enhance system consistency and modularity.
* **Generics** and **Collections** enable efficient data storage and retrieval.
* **Lambda Expressions** make code more concise, readable, and efficient.

This design ensures that the system is **maintainable**, **scalable**, and **efficient**, making it a robust solution for payroll operations.