**Report for the Employee Management System:**

**Problem the System Will Solve:**

**The Employee Management System is designed to:**

**1. \*\*Simplify Employee Record Management:\*\* Automates the process of adding, updating, viewing, and deleting employee records to eliminate manual handling and reduce the risk of duplication or data loss.**

**2. \*\*Enhance Operational Efficiency:\*\* Speeds up administrative tasks like performance tracking, salary calculations, and report generation.**

**3. \*\*Support Decision Making:\*\* Provides HR personnel with detailed insights and reports on employee performance and organizational data.**

**\*\*Purpose and Expected Functionality:\*\***

**\*\*Purpose of the System:\*\***

**The purpose of the Employee Management System is to provide a comprehensive, automated tool for HR departments to manage employee-related tasks. It eliminates repetitive manual work and provides reliable, efficient, and error-free record keeping.**

**\*\*Expected Functionality:\*\***

**1. \*\*Employee Management:\*\***

**- Add new employees with details such as ID, name, department, and position.**

**- View detailed employee records in a well-organized layout.**

**- Update or modify employee details like salary, designation, or department.**

**- Delete employee records when necessary.**

**2. \*\*Search and Filtering:\*\***

**- Quick search functionality by ID, name, or department.**

**3. \*\*Performance Tracking:\*\***

**- Maintain and update performance reviews to assist in employee evaluations.**

**4. \*\*Salary Management:\*\***

**- Automate salary calculations, including deductions and bonuses.**

**5. \*\*Role-Based Access:\*\***

**- Implement authorization to restrict access to sensitive operations.**

**6. \*\*Reporting:\*\***

**- Generate insightful reports, such as departmental headcounts and salary summaries.**

**7. \*\*User Interface:\*\***

**- Provide a responsive and intuitive interface for ease of use.**

**---**

**\*\*Key Components:\*\***

**\*\*Classes:\*\***

**1. \*\*Employee:\*\***

**- Represents individual employee details and operations.**

**2. \*\*PerformanceReview:\*\***

**- Manages performance-related data for employees.**

**3. \*\*SalaryManager:\*\***

**- Handles salary computations, including bonuses and deductions.**

**4. \*\*HRInterface:\*\***

**- Frontend interface for HR personnel to interact with the system.**

**5. \*\*DatabaseManager:\*\***

**- Responsible for storing, retrieving, and updating employee data.**

**6. \*\*ReportGenerator:\*\***

**- Generates various employee-related reports.**

**---**

**\*\*OOP Principles in Action:\*\***

**1. \*\*Encapsulation:\*\***

**- Each class encapsulates its data, ensuring internal consistency and preventing unauthorized access.**

**2. \*\*Inheritance:\*\***

**- Shared properties like employee ID and name are defined in a base class, extended by specialized classes for different roles.**

**3. \*\*Polymorphism:\*\***

**- Methods like `generateReport()` are overridden in `PerformanceReport` and `SalaryReport` classes to generate different types of reports.**

**4. \*\*Abstraction:\*\***

**- Abstract methods in a `BaseReport` class allow different report types to define their specific behavior.**

**---**

**\*\*UML Diagram:\*\***

**```plaintext**

**Employee**

**------------------**

**- id: int**

**- name: String**

**- department: String**

**- position: String**

**- salary: double**

**------------------**

**+ getDetails(): String**

**+ updateDetails(...): void**

**PerformanceReview**

**------------------**

**- employeeId: int**

**- reviewDate: Date**

**- comments: String**

**- score: int**

**------------------**

**+ addReview(...): void**

**+ getReview(...): String**

**SalaryManager**

**------------------**

**- employeeId: int**

**- baseSalary: double**

**- deductions: double**

**- bonuses: double**

**------------------**

**+ calculateSalary(): double**

**DatabaseManager**

**------------------**

**- employees: List<Employee>**

**------------------**

**+ addEmployee(...): void**

**+ getEmployee(...): Employee**

**+ updateEmployee(...): void**

**+ deleteEmployee(...): void**

**ReportGenerator**

**------------------**

**- reportType: String**

**------------------**

**+ generateReport(): String**

**```**

**Code:**

**import java.util.ArrayList;**

**import java.util.Scanner;**

**class Employee {**

**private int id;**

**private String name;**

**private String department;**

**private String position;**

**private double salary;**

**public Employee(int id, String name, String department, String position, double salary) {**

**this.id = id;**

**this.name = name;**

**this.department = department;**

**this.position = position;**

**this.salary = salary;**

**}**

**public int getId() {**

**return id;**

**}**

**public String getName() {**

**return name;**

**}**

**public String getDepartment() {**

**return department;**

**}**

**public String getPosition() {**

**return position;**

**}**

**public double getSalary() {**

**return salary;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public void setDepartment(String department) {**

**this.department = department;**

**}**

**public void setPosition(String position) {**

**this.position = position;**

**}**

**public void setSalary(double salary) {**

**this.salary = salary;**

**}**

**@Override**

**public String toString() {**

**return "ID: " + id + ", Name: " + name + ", Department: " + department +**

**", Position: " + position + ", Salary: " + salary;**

**}**

**}**

**public class EmployeeManagementSystem {**

**private static final ArrayList<Employee> employees = new ArrayList<>();**

**private static final Scanner scanner = new Scanner(System.in);**

**public static void main(String[] args) {**

**boolean running = true;**

**while (running) {**

**System.out.println("\n=== Employee Management System ===");**

**System.out.println("1. Add Employee");**

**System.out.println("2. View Employees");**

**System.out.println("3. Update Employee");**

**System.out.println("4. Delete Employee");**

**System.out.println("5. Search Employee");**

**System.out.println("6. Exit");**

**System.out.print("Choose an option: ");**

**int choice = scanner.nextInt();**

**scanner.nextLine(); // Consume the newline character**

**switch (choice) {**

**case 1 -> addEmployee();**

**case 2 -> viewEmployees();**

**case 3 -> updateEmployee();**

**case 4 -> deleteEmployee();**

**case 5 -> searchEmployee();**

**case 6 -> {**

**running = false;**

**System.out.println("Exiting the system. Goodbye!");**

**}**

**default -> System.out.println("Invalid choice! Please try again.");**

**}**

**}**

**}**

**private static void addEmployee() {**

**System.out.print("Enter Employee ID: ");**

**int id = scanner.nextInt();**

**scanner.nextLine(); // Consume the newline character**

**System.out.print("Enter Employee Name: ");**

**String name = scanner.nextLine();**

**System.out.print("Enter Department: ");**

**String department = scanner.nextLine();**

**System.out.print("Enter Position: ");**

**String position = scanner.nextLine();**

**System.out.print("Enter Salary: ");**

**double salary = scanner.nextDouble();**

**employees.add(new Employee(id, name, department, position, salary));**

**System.out.println("Employee added successfully!");**

**}**

**private static void viewEmployees() {**

**if (employees.isEmpty()) {**

**System.out.println("No employees found.");**

**} else {**

**System.out.println("\nEmployee Records:");**

**for (Employee employee : employees) {**

**System.out.println(employee);**

**}**

**}**

**}**

**private static void updateEmployee() {**

**System.out.print("Enter Employee ID to update: ");**

**int id = scanner.nextInt();**

**scanner.nextLine(); // Consume the newline character**

**for (Employee employee : employees) {**

**if (employee.getId() == id) {**

**System.out.print("Enter New Name (Leave blank to skip): ");**

**String name = scanner.nextLine();**

**if (!name.isEmpty()) {**

**employee.setName(name);**

**}**

**System.out.print("Enter New Department (Leave blank to skip): ");**

**String department = scanner.nextLine();**

**if (!department.isEmpty()) {**

**employee.setDepartment(department);**

**}**

**System.out.print("Enter New Position (Leave blank to skip): ");**

**String position = scanner.nextLine();**

**if (!position.isEmpty()) {**

**employee.setPosition(position);**

**}**

**System.out.print("Enter New Salary (-1 to skip): ");**

**double salary = scanner.nextDouble();**

**if (salary >= 0) {**

**employee.setSalary(salary);**

**}**

**System.out.println("Employee updated successfully!");**

**return;**

**}**

**}**

**System.out.println("Employee not found.");**

**}**

**private static void deleteEmployee() {**

**System.out.print("Enter Employee ID to delete: ");**

**int id = scanner.nextInt();**

**employees.removeIf(employee -> employee.getId() == id);**

**System.out.println("Employee deleted successfully!");**

**}**

**private static void searchEmployee() {**

**System.out.print("Enter Employee Name to search: ");**

**String name = scanner.nextLine();**

**System.out.println("Search Results:");**

**for (Employee employee : employees) {**

**if (employee.getName().equalsIgnoreCase(name)) {**

**System.out.println(employee);**

**}**

**}**

**}**

**}**

**\*\*Implementation Summary:\*\***

**- The Employee Management System is implemented in Java, leveraging Swing for the UI and Java Collections for backend data management.**

**- Core functionalities include CRUD operations for employee records, performance tracking, salary computation, and role-based access.**

**- Reports provide valuable insights, aiding in organizational decision-making.**

**---**

**\*\*Benefits of the System:\*\***

**1. \*\*Efficiency:\*\* Automates repetitive tasks and speeds up administrative processes.**

**2. \*\*Accuracy:\*\* Minimizes errors in employee data and salary calculations.**

**3. \*\*Scalability:\*\* Easily adaptable for organizations of different sizes.**

**4. \*\*User-Friendly:\*\* Simplifies HR operations through an intuitive interface.**

**The system is a complete solution for modern HR management, ensuring reliability, efficiency, and scalability.**