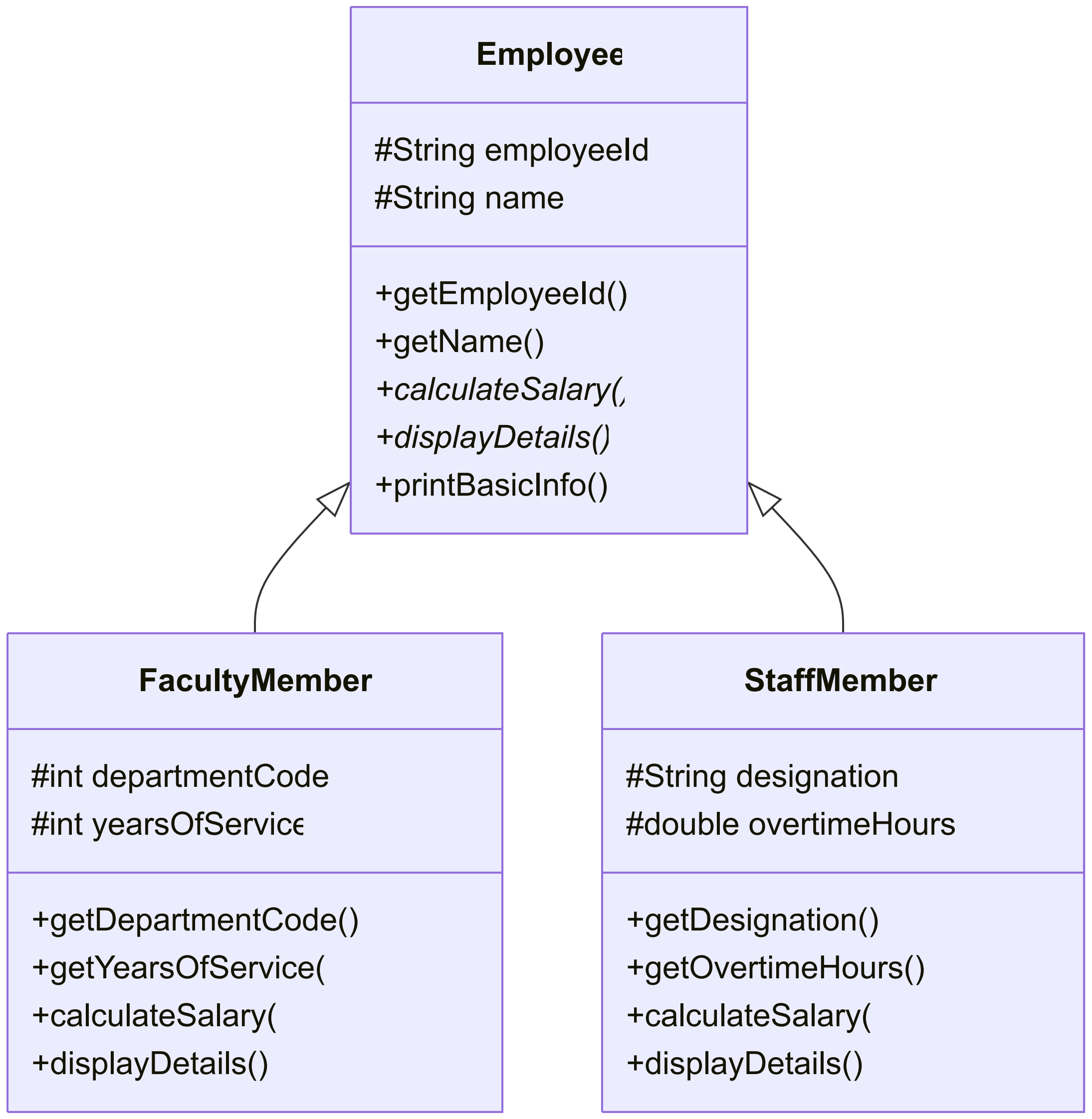
Object-Oriented Programming (OOP) Implementation Question

Scenario:
You are tasked with designing a **University Employee Management System** using Java. The system should model different types of university employees (e.g., faculty and staff) while demonstrating the following OOP concepts:

1. **Encapsulation** – Protect employee data using private fields with appropriate getters.
2. **Inheritance** – Create a base class (`Employee`) and derived classes (`FacultyMember`, `StaffMember`).
3. **Polymorphism** – Implement method overriding for salary calculation and employee details display.
4. **Abstraction** – Define abstract methods in the base class that must be implemented by subclasses.

Class Structure Overview



In the diagram above:

- **Arrows pointing upward (↑) indicate inheritance relationships, where FacultyMember and StaffMember inherit from Employee**
- **Methods marked with asterisks (*) are abstract methods that must be implemented by subclasses**
- **Members prefixed with # are private (encapsulated)**
- **+ indicates public methods**

Requirements:

1. Abstract Base Class (`Employee`)

- Should have **private** fields:
  - `employeeId` (String)
  - `name` (String)

- A **constructor** to initialize these fields.
- **Abstract methods**:
  - `calculateSalary()` → returns a `double`
  - `displayDetails()` → prints employee details
- A **protected helper method** (`printBasicInfo()`) to display `employeeId` and `name` .

2. Derived Class (`FacultyMember`)

- **Extends** `Employee` .
- Additional **private** fields:
  - `departmentCode` (int)
  - `yearsOfService` (int)
- **Constructor** to initialize all fields (including parent class fields).
- **Override** `calculateSalary()` :
  - Base salary = **50,000**
  - Experience bonus = **1,000 per year of service**
- **Override** `displayDetails()` :
  - Calls `printBasicInfo()` from parent class.
  - Displays `departmentCode` and `yearsOfService` .

3. Derived Class (`StaffMember`)

- **Extends** `Employee` .
- Additional **private** fields:
  - `designation` (String)
  - `overtimeHours` (double)
- **Constructor** to initialize all fields (including parent class fields).
- **Override** `calculateSalary()` :
  - Base salary = **40,000**
  - Overtime pay = **25 per hour**
- **Override** `displayDetails()` :
  - Calls `printBasicInfo()` from parent class.
  - Displays `designation` and `overtimeHours` .

4. Main Class (`UniversityManagementSystem`)

- Creates objects of `FacultyMember` and `StaffMember` .
- (Can Skip this step , create a single object)Stores them in an **array of `Employee` type** (demonstrating polymorphism).

- Loops through the array and calls `displayDetails()` and `calculateSalary()` for each employee.

Expected Output:

```
Employee Details:
ID: F001
Name: John Smith
Department Code: 101
Years of Service: 5
Monthly Salary: $55000.00

Employee Details:
ID: S001
Name: Jane Doe
Designation: Administrator
Overtime Hours: 20.0
Monthly Salary: $40500.00
```

Your Task:

Write the complete Java code for the above system, ensuring proper implementation of **encapsulation, inheritance, polymorphism, and abstraction**.
