**Employee Payroll Calculator**

**📘 Scenario**

A company’s payroll module must calculate monthly pay for different categories of employees (salaried, hourly‐wage, and commission‐based). While all employees have a common set of data (ID, name, department), each category requires its own “computePay” logic. Use an abstract base class to enforce that each subclass implements its own pay computation.

**✅ Requirements**

1. **Abstract Base Class: Employee**
   * **Fields (protected/private):**
     + int empId
     + String empName
     + String department
   * **Constructor:**
     + Initialize empId, empName, and department.
   * **Abstract Method:**
     + double computePay() – returns the net pay for that employee for the current month.
   * **Concrete Method:**
     + void displayEmployeeDetails()
       - Prints empId, empName, department, and the result of computePay() (formatted to two decimals).
2. **Derived Class: SalariedEmployee**
   * **Additional Field:**
     + double monthlySalary
   * **Constructor:**
     + Calls super(empId, empName, department) and initializes monthlySalary.
   * **Implement (override) Abstract Method:**
     + computePay() – simply returns monthlySalary.
3. **Derived Class: HourlyEmployee**
   * **Additional Fields:**
     + double hoursWorked
     + double hourlyRate
   * **Constructor:**
     + Calls super(empId, empName, department) and initializes hoursWorked, hourlyRate.
   * **Implement (override) Abstract Method:**
     + computePay() – returns (hoursWorked × hourlyRate). **Bonus:** If hoursWorked > 160, include overtime pay at 1.5× rate for extra hours.
4. **Derived Class: CommissionEmployee**
   * **Additional Fields:**
     + double salesAmount
     + double commissionRate – for example, 0.05 for 5%.
   * **Constructor:**
     + Calls super(empId, empName, department) and initializes salesAmount, commissionRate.
   * **Implement (override) Abstract Method:**
     + computePay() – returns salesAmount × commissionRate.
5. **Test Class: PayrollSystem**
   * In main():
     + Create at least one instance of each employee type (SalariedEmployee, HourlyEmployee, CommissionEmployee) with sample data.
     + Store them all in a List<Employee>.
     + Iterate through the list, calling displayEmployeeDetails() on each object to show polymorphic pay computation.