**OBJECT-ORIENTED PROGRAMMING**

|  |  |
| --- | --- |
| Graded Lab 3 | |
| **Topic** | Inheritance and polymorphism |
| **Objective** | To Evaluate a C++ program with a basic OOP structure |

**Problem Statement:**

Develop a C++ program for weekly payroll calculations in a company that employs various types of workers. The workforce includes Salaried employees with fixed weekly salaries, Commission employees earning a percentage of sales, and **Base\_Salary\_Plus\_Commission\_employees** receiving both a base salary and a sales percentage. The company aims to boost the earnings of **Base\_Salary\_Plus\_Commission\_employees** by 10% in the current pay period to provide a performance incentive.

Utilize abstract classes and polymorphism to create a flexible payroll system. The program should implement an abstract base class, 'Employee,' defining shared functionalities for all worker types. The employee hierarchy includes **'SalariedEmployee'** and **'CommissionEmployee,'** both directly derived from 'Employee.' Additionally, **' Base\_Salary\_Plus\_Commission\_employees** **'** extends **'CommissionEmployee.'** Essential attributes such as first name, last name, and social security number are private data members within the abstract base class 'Employee' to ensure consistency across all worker instances.

Detailed Class Descriptions:

1. **Employee (Abstract Base Class): (5)**
   * Private Member Variables:
     + **firstName** (string).
     + **lastName** (string)
     + **socialSecurityNumber** (string): to store the social security number of the employee.
   * Public Member Functions:
     + **Employee(const string& firstName, const string& lastName, const string& socialSecurityNumber)**
     + **virtual double calculateEarnings() const = 0;**
     + **void displayInfo() const;**
2. **SalariedEmployee (Derived from Employee): (5)**
   * Private Member Variables:
     + **weeklySalary** (double): to store the fixed weekly salary.
   * Public Member Functions:
     + **SalariedEmployee(const string& firstName, const string& lastName, const string& socialSecurityNumber, double weeklySalary)**: Constructor to initialize salaried employee details.
     + **double calculateEarnings() const;** Calculates weekly earnings based on the fixed salary.
3. **CommissionEmployee (Derived from Employee): (5)**
   * Private Member Variables:
     + **commissionRate** (double): to store the commission rate.
     + **sales** (double): to store the sales amount.
   * Public Member Functions:
     + **CommissionEmployee(const string& firstName, const string& lastName, const string& socialSecurityNumber, double commissionRate, double sales)**: Constructor to initialize commission employee details.
     + **double calculateEarnings() const ;**: Calculates weekly earnings based on the commission and sales.
4. **BasePlusCommissionEmployee (Derived from CommissionEmployee): (5)**
   * Private Member Variables:
     + **baseSalary** (double): to store the base salary.
   * Public Member Functions:
     + **BasePlusCommissionEmployee(const string& firstName, const string& lastName, const string& socialSecurityNumber, double commissionRate, double sales, double baseSalary)**: Constructor to initialize base salary plus commission employee details.
     + **double calculateEarnings() const ;** Calculates weekly earnings based on the commission, sales, and a 10% boost to the base salary.

**Your program should work for the main program given on the next page without any change. Include necessary libraries and header files.**

**Here is the sample output: (5)**

Name: John Doe

SSN: 123-45-6789

Weekly Earnings: $1000

Name: Jane Smith

SSN: 987-65-4321

Weekly Earnings: $250

Name: Bob Johnson

SSN: 567-89-0123

Weekly Earnings: $1000.1

**int main()** {

// Sample usage of the employee hierarchy

**SalariedEmployee salariedEmp("John", "Doe", "123-45-6789", 1000.0);**

**CommissionEmployee commissionEmp("Jane", "Smith", "987-65-4321", 0.05, 5000.0);**

**BasePlusCommissionEmployee basePlusCommissionEmp("Bob", "Johnson", "567-89-0123", 0.07, 8000.0, 1200.0);**

// Display information and calculate earnings for each employee

**salariedEmp.displayInfo();**

cout << "Weekly Earnings: $" << salariedEmp.calculateEarnings() << "\n\n";

**commissionEmp.displayInfo();**

cout << "Weekly Earnings: $" << commissionEmp.calculateEarnings() << "\n\n";

**basePlusCommissionEmp.displayInfo();**

cout << "Weekly Earnings: $" << basePlusCommissionEmp.calculateEarnings() << "\n";

return 0;

}