|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
|  | **Course:** | **Object Oriented Programming** | **Course Code:** | **CS-217** |
| **Program:** | **BS Computer Science** | **Semester:** | **Spring 2024** |
| **Duration:** | **40 minutes** | **Total Marks:** | **20** |
| **Date:** | **-** | **Weight:** | **-** |
| **Section:** | **D** | **Page(s):** | **1** |
| **Exam:** | **Quiz 4 (a)** | **Roll No.** |  |

**Question:  
  
a)**A software company requires a payroll management system for its employees. It employs different types of employees who are all paid differently as described below. All types of employees must have a function that calculates their salaries, but an employee can only be one of three types: developer, project manager, or sales representative. All employees "must" belong to at least one of these categories. An unclassified employee "cannot be paid a salary".

1. All Employees have the following attributes in common: **name, id, baseSalary** and a **calculateSalary()** method
2. Implement three classes: Developer, ProjectManager, and SalesRepresentative, each inheriting from the Employee class with unique attributes:
   * For Developer: **programmingLanguage** (string), **yearsOfExperience** (int). They are paid a bonus based on their years of experience, $1000 for every year in addition to their base salary.
   * For ProjectManager: **numProjects** (int), **bonusRate** (double). They are paid a bonus based on the number of projects they manage in addition to their base salary. (bonus = projects x rate)
   * For SalesRepresentative: **totalSales** (double), **commissionRate** (double). They are paid a commission based on their total sales in addition to their base salary. (bonus = sales x rate)
3. Implement default and parameterized constructors, destructors, and a calculateSalary() method in each derived class to calculate the salary of the respective employee type based on the provided attributes.

**b)**Give output of the following main function:

int main() {

// Create employee objects

Employee\*\* employees = new Employee\*[3];

employees[0] = new Developer("John Doe", 100, 5000, "Python", 5);

employees[1] = new ProjectManager("Jane Smith", 200, 4000, 3, 1000.0);

employees[2] = new SalesRepresentative("Bob Johnson", 300, 3500, 10000.0, 0.5);

// Calculate and display salaries

for (int i = 0; i < 3; i++) {

cout << "Name: " << employees[i]->name << endl;

cout << "ID: " << employees[i]->id << endl;

cout << "Base Salary: $" << employees[i]->baseSalary << endl;

cout << "Total Salary: $" << employees[i]->calculateSalary() << endl;

cout << endl;

}

}

**SOLUTION:**

a)

// Employee base class

class Employee {

protected:

string name;

int id;

double baseSalary;

public:

Employee(string n, int i, double b) : name(n), id(i), baseSalary(b) {}

virtual double calculateSalary() { return baseSalary; }

};

// Developer derived class

class Developer : public Employee {

private:

string programmingLanguage;

int yearsOfExperience;

public:

Developer(string n, int i, double b, string pl, int ye)

: Employee(n, i, b), programmingLanguage(pl), yearsOfExperience(ye) {}

double calculateSalary() override {

double bonus = yearsOfExperience \* 1000; // Assume $1000 bonus per year of experience

return baseSalary + bonus;

}

};

// ProjectManager derived class

class ProjectManager : public Employee {

private:

int numProjects;

double bonusRate;

public:

ProjectManager(string n, int i, double b, int np, double br)

: Employee(n, i, b), numProjects(np), bonusRate(br) {}

double calculateSalary() override {

double bonus = numProjects \* bonusRate;

return baseSalary + bonus;

}

};

// SalesRepresentative derived class

class SalesRepresentative : public Employee {

private:

double totalSales;

double commissionRate;

public:

SalesRepresentative(string n, int i, double b, double ts, double cr)

: Employee(n, i, b), totalSales(ts), commissionRate(cr) {}

double calculateSalary() override {

double commission = totalSales \* commissionRate;

return baseSalary + commission;

}

};

**b)**

**Name: John Doe**

**ID: 100**

**Base Salary: $5000**

**Total Salary: $10000**

**Name: Jane Smith**

**ID: 200**

**Base Salary: $4000**

**Total Salary: $7000**

**Name: Bob Johnson**

**ID: 300**

**Base Salary: $3500**

**Total Salary: $8500**