2100030320

M. Bhavana

EPAM – LAB-4

Task1: To create classes Employee**,** SalesPerson**,** Manager **and** Company **with predefined functionality.**

Low level requires:

1. To create basic class **Employee** and declare following content:

- Three closed fields – text field **name** (employee last name), money fields – **salary** and **bonus**
- Public property **Name** for reading employee's last name
- Public property **Salary** for reading and recording salary field
- Constructor with parameters string **name** and money **salary** (last name and salary are set)
- Virtual method **SetBonus** that sets bonuses to salary, amount of which is delegated/conveyed as bonus
- Method ToPay that returns the value of summarized salary and bonus.

2. To create class **SalesPerson** as class **Employee** inheritor and declare within it:

- Closed integer field **percent** (percent of sales targets plan performance/execution)
- Constructor with parameters: **name** – employee last name, **salary**, **percent** – percent of plan performance, first two of which are passed to basic class constructor
- Redefine virtual method of parent class **SetBonus** in the following way: if the sales person completed the plan more than 100%, so his bonus is doubled (is multiplied by 2), and if more than 200% - bonus is tripled (is multiplied by 3)

3. To create class **Manager** as **Employee** class inheritor, and declare with it:

- Closed integer field **quantity** (number of clients, who were served by the manager during a month)
- Constructor with parameters string **name** – employee last name, **salary** and integer **clientAmount** – number of served clients, first two of which are passed to basic class constructor.
- Redefine virtual method of parent class **SetBonus** in the following way: if the manager served over 100 clients, his bonus is increased by 500, and if more than 150 clients – by 1000.

LAB4-TASK-1-EMP | Employee

```
// See https://aka.ms/new-console-template for more information
using System;
7 references
class Employee
{
    private string name;
    private decimal salary;
    private decimal bonus;
    3 references
    public string Name{
        get { return name; }
    }
    0 references
    public decimal Salary     {
        get { return salary; }
        set { salary = value; }
    }
    3 references
    public Employee(string name, decimal salary)  {
        this.name = name;
        this.salary = salary;
    }
    11 references
    public virtual void SetBonus(decimal bonus)   {
        this.bonus = bonus;
    }
    3 references
    public decimal ToPay()
    {
        return salary + bonus;
    }
}
3 references
```

What's New? | Program.cs

LAB4-TASK-1-EMP | Employee

```
class SalesPerson : Employee
{
    private int percent;
    1 reference
    public SalesPerson(string name, decimal salary, int percent)
        : base(name, salary)
    {
        this.percent = percent;
    }
    9 references
    public override void SetBonus(decimal bonus)     {
        if (percent > 200)
            base.SetBonus(bonus * 3);
        else if (percent > 100)
            base.SetBonus(bonus * 2);
        else
            base.SetBonus(bonus);
    }
}
3 references
class Manager : Employee
{
    private int quantity;
    1 reference
    public Manager(string name, decimal salary, int quantity)
        : base(name, salary)     {
        this.quantity = quantity;
    }
    9 references
    public override void SetBonus(decimal bonus)    {
        if (quantity > 150)
            base.SetBonus(bonus + 1000);
        else if (quantity > 100)
```

```
            base.SetBonus(bonus + 500);
        else
            base.SetBonus(bonus);
    }
}
0 references
class Company{
    0 references
    static void Main()   {
        Employee emp = new Employee("Bhavana", 50000);
        emp.SetBonus(1000);
        Console.WriteLine($"Employee: {emp.Name}, ToPay: {emp.ToPay()}");
        SalesPerson salesPerson = new SalesPerson("Tulasi", 60000, 120);
        salesPerson.SetBonus(1500);
        Console.WriteLine($"SalesPerson: {salesPerson.Name}, ToPay: {salesPerson.ToPay()}");
        Manager manager = new Manager("Aruna", 70000, 160);
        manager.SetBonus(2000);
        Console.WriteLine($"Manager: {manager.Name}, ToPay: {manager.ToPay()}");
    }
}
```

OUTPUT:

```
Microsoft Visual Studio Debu

Employee: Bhavana, ToPay: 51000
SalesPerson: Tulasi, ToPay: 63000
Manager: Aruna, ToPay: 73000

C:\Users\Bhavana\Documents\EPAM\LAB4-TASK-1-EMP
To automatically close the console when debuggi
le when debugging stops.
Press any key to close this window . . .
```

TASK-2:

1. Create class Company and declare within it:

- Closed field **employees** (staff) – an array of Employee type.
- Constructor that receives employee array of **Employee** type with arbitrary length
- Method **GiveEverybodyBonus** with money parameter **companyBonus** that sets the amount of basic bonus for each employee.
- Method **TotalToPay** that returns total amount of salary of all employees including awarded bonus
- Method **NameMaxSalary** that returns employee last name, who received maximum salary including bonus.

```
LAB-4-TASK-2                                                    Program
// See https://aka.ms/new-console-template for more information
using System;
7 references
class Employee{
    1 reference
    public string FirstName { get; set; }
    2 references
    public string LastName { get; set; }
    5 references
    public double Salary { get; set; }
    3 references
    public Employee(string firstName, string lastName, double salary)    {
        FirstName = firstName;
        LastName = lastName;
        Salary = salary;
    }
}
3 references
class Company{
    private Employee[] employees;
    1 reference
    public Company(Employee[] employees)   {
        this.employees = employees;
    }
    1 reference
    public void GiveEverybodyBonus(double companyBonus)     {
        foreach (var employee in employees)         {
            employee.Salary += companyBonus;
        }
    }
    1 reference
    public double TotalToPay()    {
        double totalSalary = 0;
        foreach (var employee in employees)         {
```

```
What's New?    Program.cs*
LAB-4-TASK-2                                                    Program                    Main()
            totalSalary += employee.Salary;         }
        return totalSalary;
    }
    1 reference
    public string NameMaxSalary()    {
        double maxSalary = 0;
        string maxSalaryEmployeeLastName = "";
        foreach (var employee in employees)        {
            if (employee.Salary > maxSalary)            {
                maxSalary = employee.Salary;
                maxSalaryEmployeeLastName = employee.LastName;           }       }
        return maxSalaryEmployeeLastName;
    }
}
0 references
class Program{
    0 references
    public  static void Main()    {
        Employee[] employees = {
            new Employee("Roshni", "Bala", 50000),
            new Employee("Mallampati", "Laskhmi", 60000),
            new Employee("Ram", "Ravi", 75000)
        };
        Company company = new Company(employees);
        company.GiveEverybodyBonus(1000);
        Console.WriteLine($"Total salary to pay: {company.TotalToPay()}");
        Console.WriteLine($"Last name of employee with maximum salary: {company.NameMaxSalary()}");
    }
}
```

Output:

```
Microsoft Visual Studio Debu

Total salary to pay: 188000
Last name of employee with maximum salary: Ravi

C:\Users\Bhavana\Documents\EPAM\LAB-4-TASK-2\bin\Debug\net8.0\LAB
To automatically close the console when debugging stops, enable T
le when debugging stops.
Press any key to close this window . . .
```