Name: Gondrala Mani Sai
Id num: 2100031545

Lab4: Inheritance (Low and Advanced)

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.

TASK 2: Advanced level requires:

1. To fully complete Low level tasks.
2. Create class Company and declare within it:

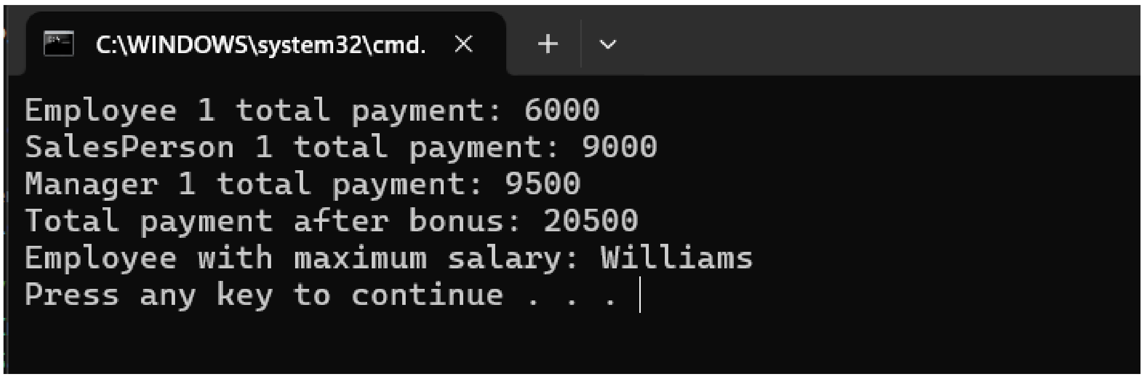
- Closed field **employees** (staff) – an array of Employee type.
- Constructor that receives employee array of **Employee** type with arbitrary length
- Method **GiveEverybodyBonus** with money parameter **companyBonus** that sets the amount of basic bonus for each employee.
- Method **TotalToPay** 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.

```
using System;
using System.Linq;

namespace CompanyManagement
{
    // Task 1: Basic class Employee
    public class Employee
    {
        private string name; // Employee last name
        private decimal salary; // Salary
        private decimal bonus; // Bonus

        public string Name => name; // Property for reading employee's last name
        public decimal Salary => salary; // Property for reading salary

        public Employee(string name, decimal salary)
        {
            this.name = name;
            this.salary = salary;
        }

        // Virtual method to set bonus
        public virtual void SetBonus(decimal bonus)
        {
            this.bonus = bonus;
        }

        // Method to calculate total pay
        public decimal ToPay()
        {
            return salary + bonus;
        }
    }

    // Task 1: SalesPerson class inheriting from Employee
```

```
public class SalesPerson : Employee
{
    private int percent; // Percent of sales targets plan performance

    public SalesPerson(string name, decimal salary, int percent) : base(name, salary)
    {
        this.percent = percent;
    }

    // Redefine SetBonus method
    public override void SetBonus(decimal bonus)
    {
        if (percent > 200)
            base.SetBonus(bonus * 3); // Triple bonus if plan exceeded by 200%
        else if (percent > 100)
            base.SetBonus(bonus * 2); // Double bonus if plan exceeded by 100%
        else
            base.SetBonus(bonus);
    }
}

// Task 1: Manager class inheriting from Employee
public class Manager : Employee
{
    private int quantity; // Number of clients served

    public Manager(string name, decimal salary, int quantity) : base(name, salary)
    {
        this.quantity = quantity;
    }

    // Redefine SetBonus method
    public override void SetBonus(decimal bonus)
    {
        if (quantity > 150)
            base.SetBonus(bonus + 1000); // Add 1000 bonus if served more than 150 clients
        else if (quantity > 100)
            base.SetBonus(bonus + 500); // Add 500 bonus if served more than 100 clients
        else
            base.SetBonus(bonus);
    }
}

// Task 2: Company class
public class Company
{
    private Employee[] employees; // Array of employees

    public Company(Employee[] employees)
    {
        this.employees = employees;
    }

    // Method to give bonus to all employees
    public void GiveEverybodyBonus(decimal companyBonus)
```

```
        {
            foreach (var employee in employees)
            {
                employee.SetBonus(companyBonus);
            }
        }

        // Method to calculate total payment
        public decimal TotalToPay()
        {
            decimal total = 0;
            foreach (var employee in employees)
            {
                total += employee.ToPay();
            }
            return total;
        }

        // Method to get name of employee with maximum salary
        public string NameMaxSalary()
        {
            var employee = employees.OrderByDescending(e => e.ToPay()).FirstOrDefault();
            return employee != null ? employee.Name : "";
        }
    }

    // Program class
    class Program
    {
        static void Main(string[] args)
        {
            // Creating employees
            Employee employee1 = new Employee("Smith", 5000);
            SalesPerson salesPerson1 = new SalesPerson("Johnson", 6000, 150);
            Manager manager1 = new Manager("Williams", 7000, 120);

            // Setting bonuses
            employee1.SetBonus(1000);
            salesPerson1.SetBonus(1500);
            manager1.SetBonus(2000);

            // Printing total payments
            Console.WriteLine($"Employee 1 total payment: {employee1.ToPay()}");
            Console.WriteLine($"SalesPerson 1 total payment: {salesPerson1.ToPay()}");
            Console.WriteLine($"Manager 1 total payment: {manager1.ToPay()}");

            // Creating company
            Company company = new Company(new Employee[] { employee1, salesPerson1, manager1
});

            // Giving bonus to everybody in the company
            company.GiveEverybodyBonus(500);

            // Printing total payments after bonus
            Console.WriteLine($"Total payment after bonus: {company.TotalToPay()}");
            // Printing name of employee with maximum salary
            Console.WriteLine($"Employee with maximum salary: {company.NameMaxSalary()}");
        }
    }
}
```

Output:



```
Employee 1 total payment: 6000
SalesPerson 1 total payment: 9000
Manager 1 total payment: 9500
Total payment after bonus: 20500
Employee with maximum salary: Williams
Press any key to continue . . .
```