.Net Programming Lab-4

Name : Ch Bala Gowtham

Regd no: 2000032067

Sec : 13

IN-LAB:

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.

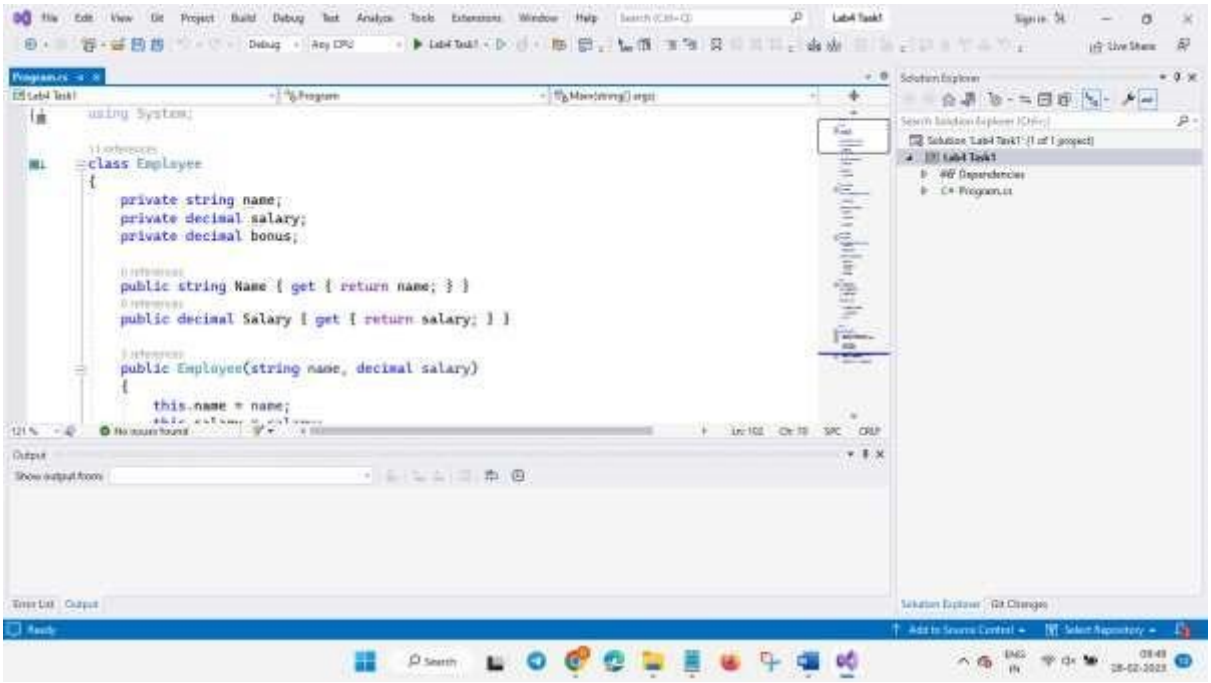
Solution:

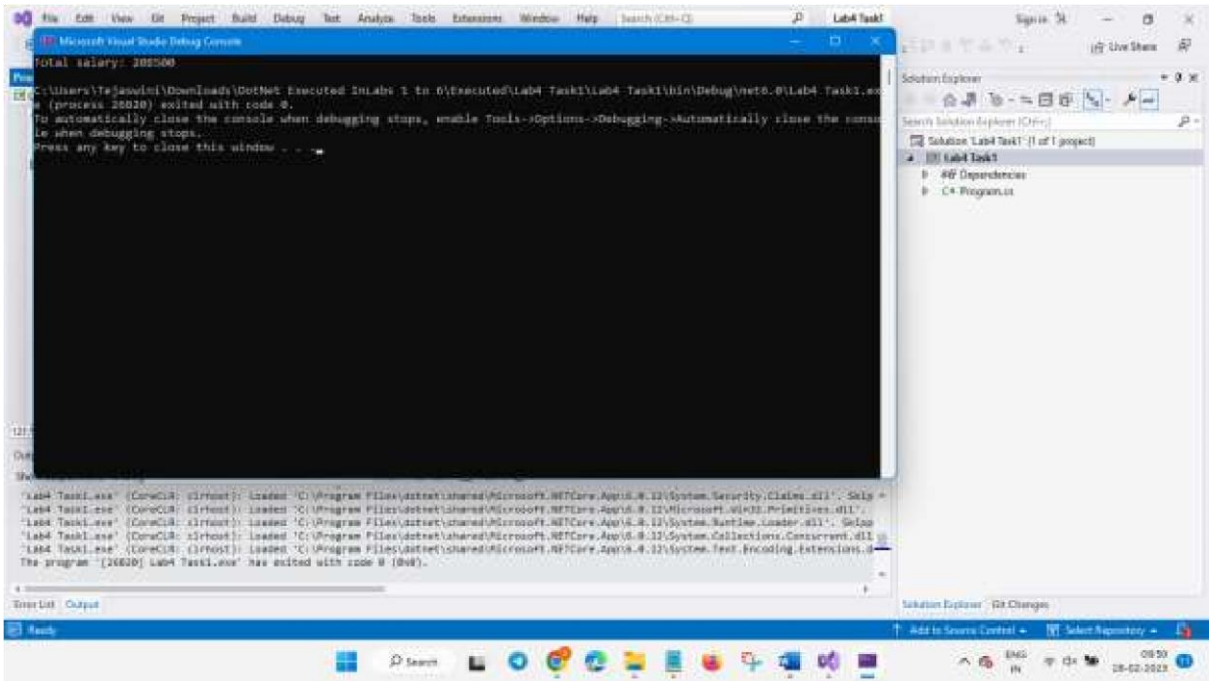
```
using System;

class Employee
{
   private string name;
private decimal salary;
private decimal bonus;

   public string Name { get { return name; } }
public decimal Salary { get { return salary; } }

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

   public virtual void SetBonus(decimal bonus)
   {
      this.bonus = bonus;
   }

   public decimal ToPay()
   {
```

```
        return salary + bonus;
    }
}

class SalesPerson : Employee
{
    private int percent;

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

    public override void SetBonus(decimal bonus)
    {
        if (percent > 200)
bonus *= 3;          else if
(percent > 100)
            bonus *= 2;

        base.SetBonus(bonus);
    }
}

class Manager : Employee
{
    private int quantity;

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

    public override void SetBonus(decimal bonus)
    {
```

```
        if (quantity > 150)
bonus += 1000;        else
if (quantity > 100)
bonus += 500;

        base.SetBonus(bonus);
    } } class
Company
{
    private Employee[] employees;

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

    public decimal TotalSalary()
    {
        decimal total = 0;
        foreach (Employee employee in employees)
        {
            total += employee.ToPay();
        }
        return total;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Employee employee1 = new Employee("Smith", 50000);
        SalesPerson salesPerson1 = new SalesPerson("Johnson", 60000, 150);
        Manager manager1 = new Manager("Williams", 70000, 120);

        employee1.SetBonus(2000);
salesPerson1.SetBonus(8000);
manager1.SetBonus(10000);
```

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

        Console.WriteLine("Total salary: " + company.TotalSalary());
    }
}
```





TASK 2: Advanced level requires:

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

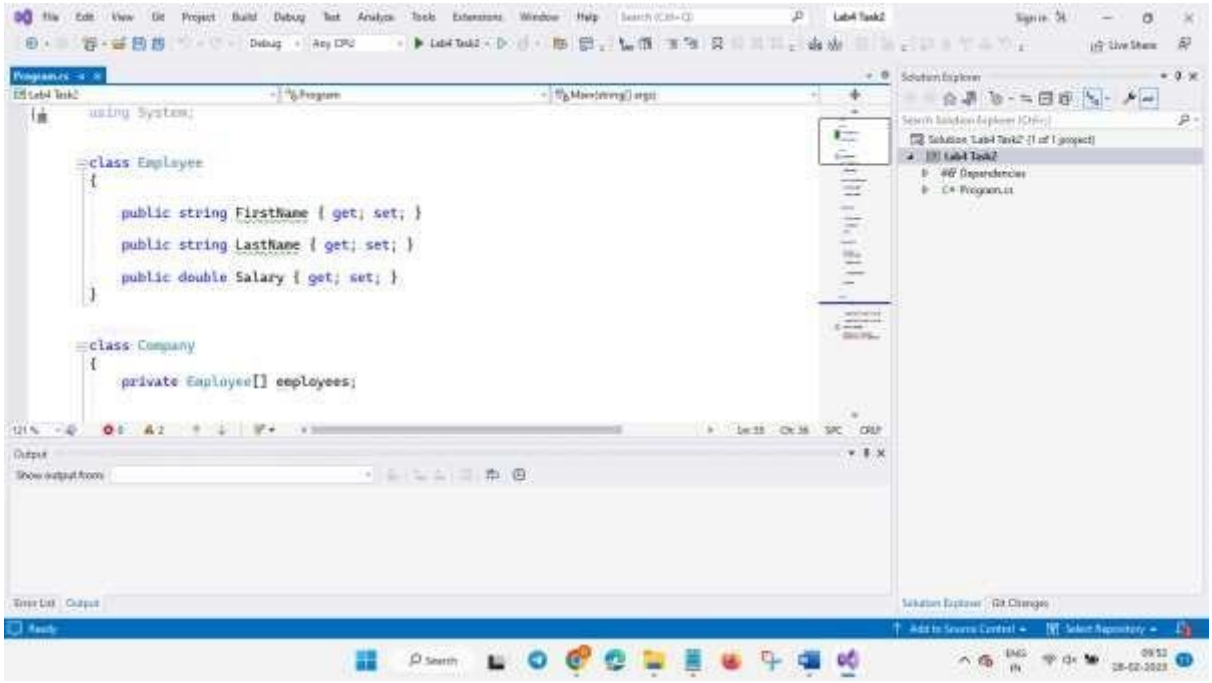
- Closed field **employees** (staff) – an array of Employee type.
- Constructor that receives employee array of **Employee** type with arbitrary length
- Method **GiveEverybodyBonus** with money parameter **companyBonus** that sets the amount of basic bonus for each employee.
- Method **TotalToPay**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

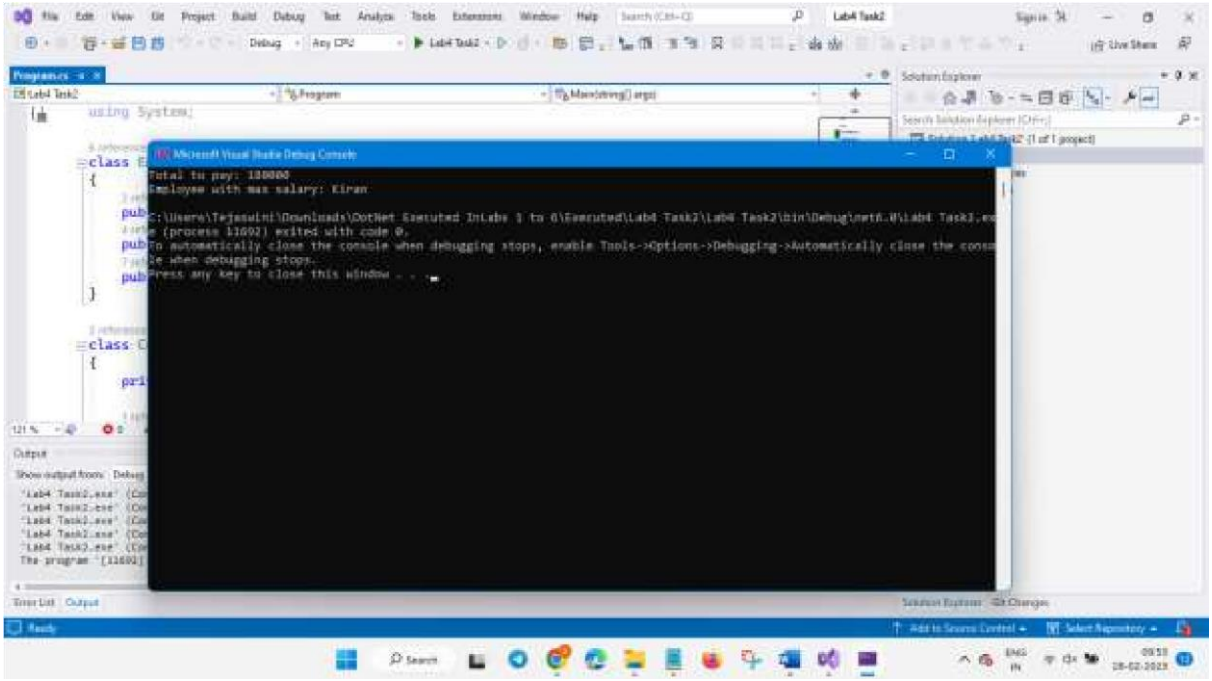
Solution:

```
using System;

class Employee
{
   public string FirstName { get; set; }
public string LastName { get; set; }
public double Salary { get; set; }
}

class Company
{
   private Employee[] employees;

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

   public void GiveEverybodyBonus(double companyBonus)
```

```
    {
        foreach (var employee in employees)
        {
            employee.Salary += companyBonus;
        }
    }

    public double TotalToPay()
    {
        double total = 0;
        foreach (var employee in employees)
        {
            total += employee.Salary;
        }
        return total;
    }

    public string NameMaxSalary()
    {
        double maxSalary = 0;        string
maxSalaryName = "";        foreach (var
employee in employees)
        {
            if (employee.Salary > maxSalary)
            {
                maxSalary = employee.Salary;
maxSalaryName = employee.LastName;
            }
        }
        return maxSalaryName;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Employee[] employees = {
```

```
        new Employee { FirstName = "Dinesh", LastName = "Kumar", Salary
=
50000 },
        new Employee { FirstName = "Vamsi", LastName = "Kiran", Salary =
60000 },
        new Employee { FirstName = "Vardhan", LastName = "Perla", Salary
=
55000 }
    };
    Company company = new Company(employees);
company.GiveEverybodyBonus(5000);
    Console.WriteLine("Total to pay: " + company.TotalToPay());
    Console.WriteLine("Employee with max salary: " +
company.NameMaxSalary());
  }
}
```





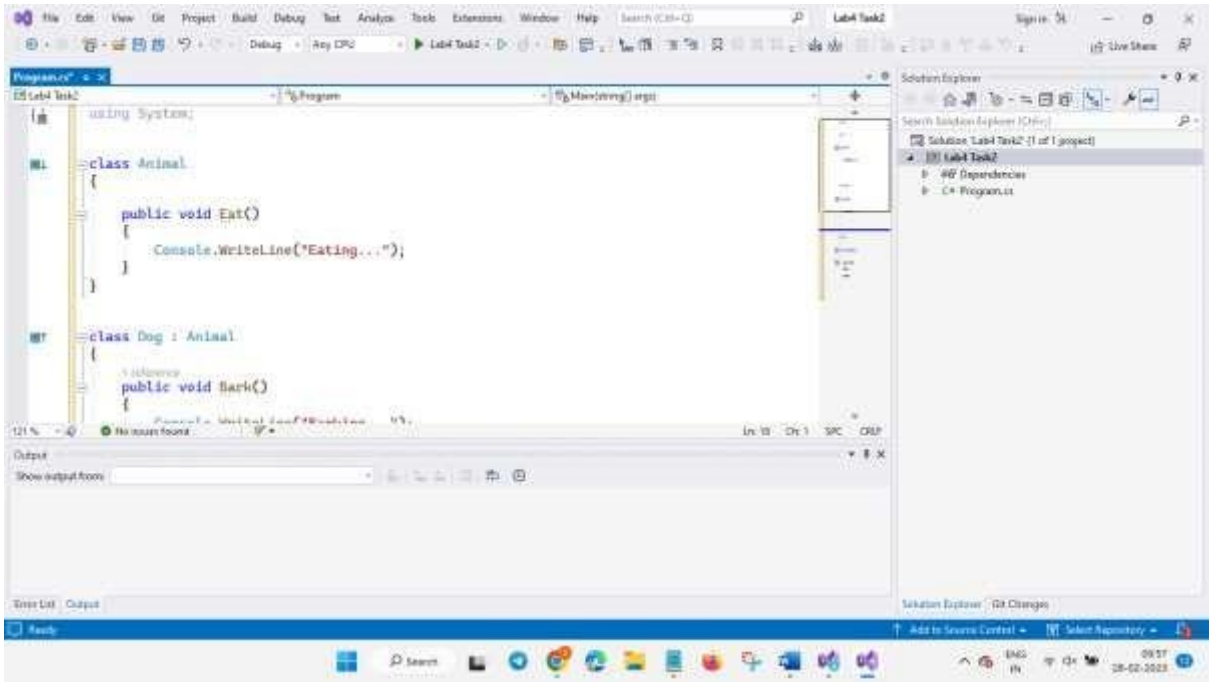
POST-LAB

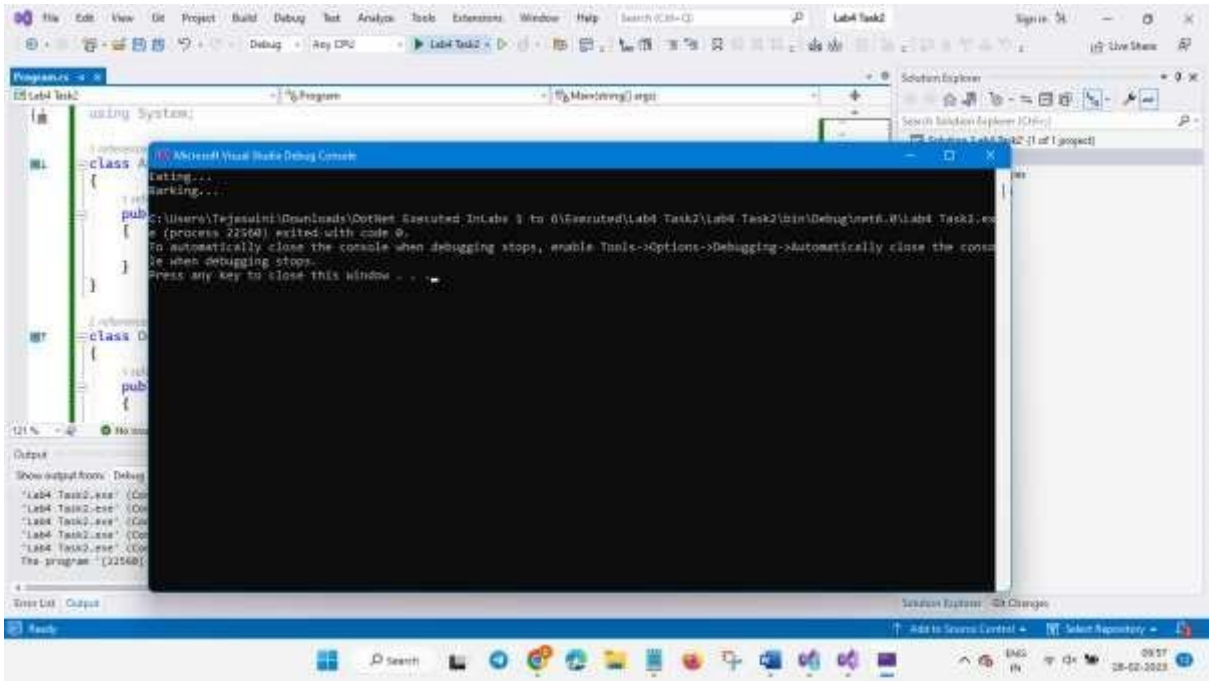
1. Implement a small Application with help of the Inheritance and Analyse the type of Inheritance used in the application. Justify Answer?

Solution:

```
using System;

class Animal {
public void Eat()
    {
        Console.WriteLine("Eating...");
    }
}

class Dog : Animal {
public void Bark()
    {
        Console.WriteLine("Barking...");
    }
}
```

```
class Program
{
   static void Main(string[] args)
   {
      Dog myDog = new Dog();
myDog.Eat();
      myDog.Bark();
   }
}
```




Microsoft Visual Studio Debug Console
Eating...
Barking...
Press any key to close this window . . .