```
using System.Xml.Linq;
using static System.Net.Mime.MediaTypeNames;

namespace backendDay4
{
    6 references
    public class Employee
    {
        private string empName;
        private int empAge;
        private string empDepartment;
        private double empSalary;
        private int empYearHired;

        1 reference
        public string EmpName
        {
            get { return empName; }
            set {
                if (!string.IsNullOrWhiteSpace(value))
                {
                    empName = value;
                }else
                {
                    throw new ArgumentException("Invalid name. It cannot be empty");
                }
            }
        }

        1 reference
        public int EmpAge
        {
            get { return empAge; }
            set
            {
                if (value < 18 || value > 70)
                {
                    throw new ArgumentException("We do not have employees under 18 or over 70");
                }
                else
                {
                    empAge = value;
                }
            }
        }
        1 reference
        public string EmpDepartment
        {
            get { return empDepartment; }
            set
            {
                if (!string.IsNullOrWhiteSpace(value))
                {
                    empDepartment = value;
                }
                else
                {
                    throw new ArgumentException("Invalid department. It cannot be empty");
                }
            }
        }

        2 references
        public double EmpSalary
        {
            get { return empSalary; }
            set
            {
                if(value >= 15000)
                {
                    empSalary = value;
                }
                else
                {
                    throw new ArgumentException("Salary cannot be less than 15,000");
                }
            }
        }

```

```
1 reference
public int EmpYearHired
{
    get { return empYearHired; }
    set {
        int currentYear = DateTime.Now.Year;
        if (value >= 1990 && value <= currentYear)
        {
            empYearHired = value;
        }
        else
        {
            throw new ArgumentException("Invalid hire year.");
        }

    }
}

1 reference
public int EmpTotalYears
{
    get { return DateTime.Now.Year - empYearHired; }
}

1 reference
public void DisplayInfo()
{
    Console.WriteLine("Name\tAge\tDepartment\tSalary\tYear Hired\tTotal Years Working");
    Console.WriteLine($"{empName}\t{empAge}\t{empDepartment}\t\t{empSalary}\t{empYearHired}\t\t{EmpTotalYears}");
}
}
```

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Employee emp1 = new Employee();
        Employee emp2 = new Employee();

        Employee[] employees = { emp1, emp2 };

        for(int i = 0; i < 2; i++)
        {
            string inputName;
            do
            {
                Console.Write("Please enter the employee #" + (i + 1) + " name: ");
                inputName = Console.ReadLine();
            } while (string.IsNullOrWhiteSpace(inputName));
            employees[i].EmpName = inputName;

            int age;
            do
            {
                Console.Write("Please enter the employee #" + (i + 1) + " age (18 to 70): ");
                string input = Console.ReadLine();

                if (int.TryParse(input, out age) && age >= 18 && age <= 70)
                {
                    employees[i].EmpAge = age;
                    break;
                }
                else
                {
                    Console.WriteLine("Invalid age. Please enter a number between 18 and 70.");
                }
            } while (true);
```

```
            string dept;
            do
            {
                Console.Write("Please enter the employee #" + (i + 1) + " department: ");
                dept = Console.ReadLine();
            } while (string.IsNullOrWhiteSpace(dept));
            employees[i].EmpDepartment = dept;

            double salary;
            do
            {
                Console.Write("Please enter the employee #" + (i + 1) + " salary: ");
                string input = Console.ReadLine();

                if (double.TryParse(input, out salary) && salary >= 15000)
                {
                    employees[i].EmpSalary = salary;
                    break;
                }
                else
                {
                    Console.WriteLine("Invalid amout!");
                }
            } while (true);

            int years;
            do
            {
                Console.Write("Please enter the employee #" + (i + 1) + " hire year: ");
                string input = Console.ReadLine();
                int currentYear = DateTime.Now.Year;

                if (int.TryParse(input, out years) && years >= 1990 && years <= currentYear)
                {
                    employees[i].EmpYearHired = years;
                    break;
                }
                else
                {
                    Console.WriteLine("Invalid year. Please enter a year hired between 1990 and current year");
                }
            } while (true);
        }
```

```
            Console.WriteLine("----------\t Employee Information System \t---------- ");
            foreach(Employee emp in employees) {

                emp.DisplayInfo();
                Console.WriteLine();
            }
            //code pang update salary ni employee 1
            Console.WriteLine("\nUpdating salary for Employee #1");
            double newSalary;
            do
            {
                Console.Write("Enter new salary: ");
                string input = Console.ReadLine();
                if (double.TryParse(input, out newSalary) && newSalary >= 15000)
                {
                    emp1.EmpSalary = newSalary;
                    break;
                }
                else
                {
                    Console.WriteLine("Invalid amount. Must be at least 15,000.");
                }
            } while (true);

            Console.WriteLine("\nUpdated Employee #1 Info:");
            emp1.DisplayInfo();
        }
    }
}
```

CODE OUTPUT

```
Please enter the employee #1 name: john
Please enter the employee #1 age (18 to 70): 20
Please enter the employee #1 department: IT
Please enter the employee #1 salary: 22000
Please enter the employee #1 hire year: 2020
Please enter the employee #2 name: Rich
Please enter the employee #2 age (18 to 70): 21
Please enter the employee #2 department: Data
Please enter the employee #2 salary: 23000
Please enter the employee #2 hire year: 2022
----------       Employee Information System    ----------
Name    Age     Department      Salary  Year Hired      Total Years Working
john    20      IT              22000   2020            5

Name    Age     Department      Salary  Year Hired      Total Years Working
Rich    21      Data            23000   2022            3


Updating salary for Employee #1...
Enter new salary: 25000

Updated Employee #1 Info:
Name    Age     Department      Salary  Year Hired      Total Years Working
john    20      IT              25000   2020            5

D:\NICO\WPH\backendDay4\bin\Debug\net8.0\backendDay4.exe (process 11152) exited with code 0 (0x0).
Press any key to close this window . . .
```

ERROR HANDLING

EMPTY INPUT

```
D:\NICO\WPH\backendDay4\bin\Debug\net8.0\backendDay4.exe
Please enter the employee #1 name:
Please enter the employee #1 name:
Please enter the employee #1 name:
```

INVALID AGE

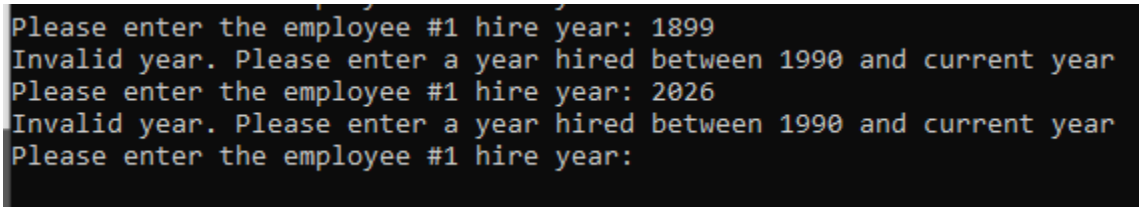
```
Please enter the employee #1 age (18 to 70): 17
Invalid age. Please enter a number between 18 and 70.
Please enter the employee #1 age (18 to 70): -2
Invalid age. Please enter a number between 18 and 70.
Please enter the employee #1 age (18 to 70): 71
Invalid age. Please enter a number between 18 and 70.
Please enter the employee #1 age (18 to 70):
```

INVALID AMOUNT – since I put a condition that the starting salary is >= 15000

```
Please enter the employee #1 salary: 12000
Invalid amout!
Please enter the employee #1 salary:
```

```
Updating salary for Employee #1
Enter new salary: 1200
Invalid amount. Must be at least 15,000.
Enter new salary:
```

INVALID YEAR- I put a restriction of year when the company started hiring.



CODE EXPLANATION

In this program, I created a basic employee information system using object-oriented programming in C#. Encapsulation is applied by declaring the employee fields as private and exposing them only through public properties, which keeps the internal data safe and only modifiable in controlled ways. These properties help control access by including if-else statements that validate the data before allowing assignment—for example, rejecting empty names or salaries below 15,000. The program prevents incorrect or missing data by using loops and validation logic in the Main method before assigning values to the properties, ensuring only valid input reaches the object. The number of years worked is automatically calculated using a read-only property (EmpTotalYears) that subtracts the hire year from the current year. A method named DisplayInfo() is used to neatly format and print out all the employee details, which helps keep the code organized and reusable. The program also meets the final requirement by properly updating the salary of Employee #1 using the EmpSalary property, ensuring that the updated amount is still validated. While working on this activity, I learned how powerful property validation can be in preventing bad data and how combining control flow with object-oriented programming leads to more reliable and maintainable applications.