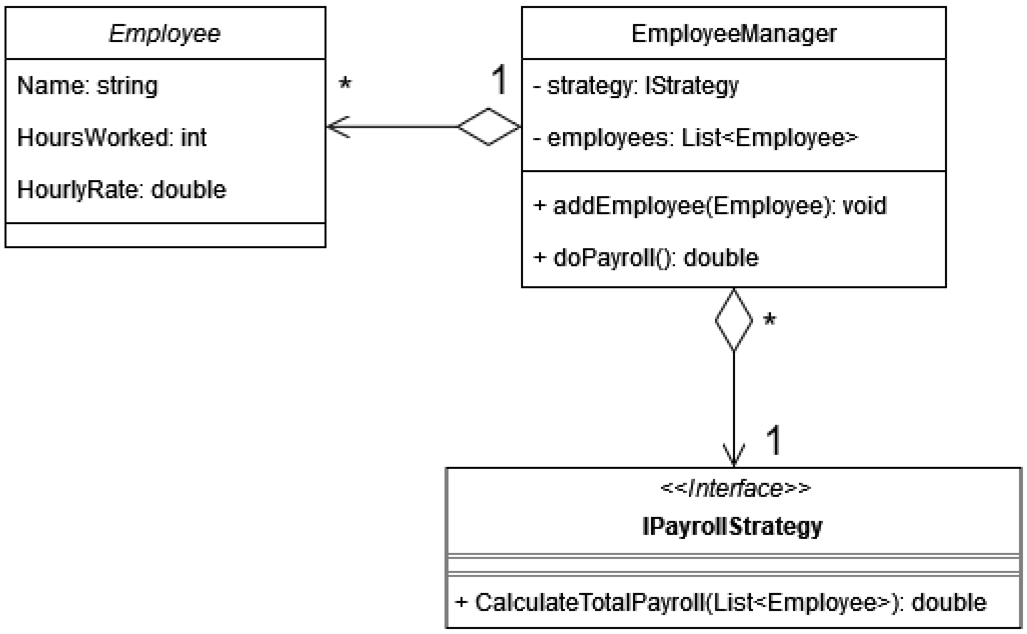
Practica parcial

Parte 1

```
class Program
{
    static void Main(string[] args)
    {
        var employeeManager = new EmployeeManager();
        employeeManager.AddEmployee(new Employee { Name = "Lala",
HoursWorked = 40, HourlyRate = 25 });
        employeeManager.AddEmployee(new Employee { Name = "Pepe",
HoursWorked = 50, HourlyRate = 20 });
        Console.WriteLine("Total Payroll: $" +
employeeManager.CalculateTotalPayroll());

        // Comentario: 2023-07-05; Para qué está esto? No sé,
        // pero no lo saco por si algo se rompe...
        employeeManager.OldPayrollSystem();
        employeeManager.OtherPayrollCalculation();

        employeeManager.AddEmployee(new Employee { Name = "Boss",
HoursWorked = 5, HourlyRate = 200 });
    }
}

public class Employee
{
    public string Name { get; set; }
    public int HoursWorked { get; set; }
    public double HourlyRate { get; set; }
}

public class EmployeeManager
{
    private List<Employee> employees = new List<Employee>();
    private List<int> oldPayrollSystemData = new List<int>(); // ¿Qué es
esto?

    public void AddEmployee(Employee employee)
    {
        employees.Add(employee);
    }
    public double CalculateTotalPayroll()
    {
        double total = 0;
        foreach (var employee in employees)
        {
            total += employee.HoursWorked * employee.HourlyRate;
        }
        return total;
    }

    // Comentario: 2023-07-05;
    // Esto parece ser parte de un viejo sistema de nómina que ya no se
usa.
    // Pero está aquí, y no estamos seguros de si es seguro eliminarlo.
    public void OldPayrollSystem()
    {
        ...
        Console.WriteLine("Old payroll system processed.");
    }

    // Comentario: 2023-07-05;
    // Esto parece ser parte de un viejo sistema de nómina que ya no se
usa.
    // Pero está aquí, y no estamos seguros de si es seguro eliminarlo.
    public void OtherPayrollCalculation()
    {
        ...
        Console.WriteLine("Old payroll system processed.");
    }
}
```

Strategy



```
public interface IPayrollStrategy
{
      public double CalculateTotalPayroll(List<Employee> employees)
}

public class NormalPayrollStrategy: IPayrollStrategy
{
      public double CalculateTotalPayroll(List<Employee> employees)
      {
            double total = 0;
            foreach (Employee employee in employees)
           {
           total += employee.HoursWorked * employee.HourlyRate;
           }
           Console.WriteLine("Normal payroll strategy processed.");
           return total;
      }
}

public class OldPayrollStrategy: IPayrollStrategy
{
     public double CalculateTotalPayroll(List<Employee> employees)
     {
           ...
           Console.WriteLine("Old payroll strategy processed.");
           return oldPayroll;
     }
}

public class OtherPayrollStrategy: IPayrollStrategy
{
     public double CalculateTotalPayroll(List<Employee> employees)
     {
           ...
           Console.WriteLine("Other payroll strategy processed.");
           return otherPayroll;
     }
}

public class EmployeeManager
{
     private IPayrollStrategy payrollStrategy;
     private List<Employee> employees = new List<Employee>();

     public void setPayrollStrategy (IPayrollStrategy  strategy)
     {
           payrollStrategy = strategy;
     }

     public double doPayroll()
     {
           if(payrollStrategy != null)
           {
                return payrollStrategy.CalculateTotalPayroll();
           }
           Console.WriteLine("No payroll strategy selected.");
           return null;
     }

     public void AddEmployee(Employee employee)
     {
           employees.Add(employee);
     }
}

public class Employee
{
     public string Name { get; set; }
     public int HoursWorked { get; set; }
     public double HourlyRate { get; set; }
}

class Program
```

```
{
     static void Main(string[] args)
     {
           var employeeManager = new EmployeeManager();
           employeeManager.AddEmployee(new Employee { Name = "Lala",
HoursWorked = 40, HourlyRate = 25 });
           employeeManager.AddEmployee(new Employee { Name = "Pepe",
HoursWorked = 50, HourlyRate = 20 });

           employeeManager.setPayrollStrategy(new
NormalPayrollStrategy());
           Console.WriteLine("Total Payroll: $" +
employeeManager.doPayroll());
           employeeManager.setPayrollStrategy(new OldPayrollStrategy());
           Console.WriteLine("Old Payroll: $" +
employeeManager.doPayroll());
           employeeManager.setPayrollStrategy(new
OtherPayrollStrategy());
           Console.WriteLine("Other Payroll: $" +
employeeManager.doPayroll());
     }
}
```

Parte 2

```
// UserProfile.cs
public class UserProfile
{
    public string Name { get; set; }
    public int Age { get; set; }

    public UserProfile(string name, int age)
    {
        Name = name;
        Age = age;
    }

    public void PrintProfile()
    {
        Console.WriteLine($"Name: {Name}, Age: {Age}");
    }
}

// Program.cs
```

```
class Program
{
    static void Main()
    {
        UserProfile profile = new UserProfile("Alice", 25);
        Console.WriteLine("Original Profile:");
        profile.PrintProfile();

        Console.WriteLine("\nUpdated Profile:");
        profile.Name = "Bob";
        profile.Age = 30;
        profile.PrintProfile();

        Console.WriteLine("\nUPS!!!:");
        profile.Name = "Alice";
        profile.Age = 25;
        profile.PrintProfile();
    }
}
```

Builder

```
public interface IBuilder
{
        public void reset()
        public void setName(string name)
        public void setAge(string age)
        public void getUserProfile()
}

public class UserProfileBuilder
{
        private UserProfile userProfile;

        public UserProfileBuilder()
        {
                userProfile = new UserProfile();
        }

        public UserProfile getUserProfile()
        {
                return userProfile;
        }

        public void reset()
        {
                userProfile = new UserProfile();
        }

        public void setName(string name)
        {
                userProfile.Name = name;
                return this;
        }

        public void setAge(string age)
        {
                userProfile.Age = age;
                return this;
        }
}

public class UserProfile
{
    public string Name { get; set; }
    public int Age { get; set; }

    public UserProfile(string name, int age)
    {
        Name = name;
        Age = age;
    }

    public void PrintProfile()
    {
        Console.WriteLine($"Name: {Name}, Age: {Age}");
    }
}

class Program
{
    static void Main()
    {
        UserProfileBuilder profileBuilder = new
UserProfileBuilder().setName("Alice").setAge(25)

        UserProfile profile = profileBuilder.getUserProfile()
        Console.WriteLine("Original Profile:");
        profile.PrintProfile();

        Console.WriteLine("\nUpdated Profile:");
        profileBuilder = profileBuilder.setName("Bob").setAge(30)
        profile = profileBuilder.getUserProfile()
        profile.PrintProfile();
        Console.WriteLine("\nUPS!!!:");
        profileBuilder = profileBuilder.setName("Alice").setAge(25)
        profile = profileBuilder.getUserProfile()
        profile.PrintProfile();
    }
}

public class Memento
{
      private string Name { get; set; }
      private int Age { get; set; }

      public Memento(name: string, age: string)
      {
            Name = name;
            Age = age;
      }

      public getState()
      {
            return Name, Age;
      }
}

public class Caretaker
{
      public Memento[] history

      public undo()
      {
            return history.pop();
      }
}
```

Ejercicio 3

decorator
observer