UNIDAD TEMÁTICA 5 – Patrones de diseño– Trabajo de Aplicación 5

Para cada uno de los siguientes ejercicios, en equipo:

1- Determine que patrón puede resolver el problema de una forma más eficiente.
2- Agregue las clases, interfaces, métodos que considere necesarios para remediar la situación.

EJERCICIO 1

Este código es bastante básico y no es escalable. Por ejemplo, si queremos notificar a más estudiantes o si queremos que los estudiantes se suscriban a las notificaciones de diferentes exámenes, este diseño no sería adecuado.

```
class Program
{
    static void Main()
    {
        var exam = new Exam("Matemáticas");
        var student1 = new Student("Alice");
        var student2 = new Student("Bob");

        exam.NotifyStudents(student1, student2);
    }
}

class Exam
{
    public string Subject { get; }

    public Exam(string subject)
    {
        Subject = subject;
    }

    public void NotifyStudents(params Student[] students)
    {
        foreach (var student in students)
        {
            Console.WriteLine($"{student.Name}, hay un nuevo examen de
{Subject}!");
        }
    }
}

class Student
{
    public string Name { get; }

    public Student(string name)
    {
        Name = name;
    }
}
```

EJERCICIO 2

```
class Program
{
    static void Main()
    {
        var gameCharacter = new GameCharacter
        {
            Name = "John",
            Health = 100,
            Mana = 50
        };

        Console.WriteLine("Estado inicial:");
        gameCharacter.DisplayStatus();

        Console.WriteLine("\nGuardando estado...");
        var savedState = gameCharacter;

        Console.WriteLine("\nCambiando estados...");
        gameCharacter.Health -= 30;
        gameCharacter.Mana += 20;
        gameCharacter.DisplayStatus();

        Console.WriteLine("\nRestaurando estado...");
        gameCharacter = savedState;
        gameCharacter.DisplayStatus();
    }
}

class GameCharacter
{
    public string Name { get; set; }
    public int Health { get; set; }
    public int Mana { get; set; }

    public void DisplayStatus()
    {
        Console.WriteLine($"{Name} tiene {Health} de salud y {Mana} de
mana.");
    }
}
```

EJERCICIO 3

```
class Program
{
    static void Main()
    {
        var alice = new User("Alice");
        var bob = new User("Bob");

        alice.SendMessage("Hola Bob!", bob);
        bob.SendMessage("Hola Alice!", alice);
    }
}

class User
{
    public string Name { get; }

    public User(string name)
    {
        Name = name;
    }

    public void SendMessage(string message, User recipient)
    {
        Console.WriteLine($"{Name} to {recipient.Name}: {message}");
    }
}
```

EJERCICIO 4

```
class Program
{
    static void Main()
    {
        var television = new Television();

        string input = "";
        while (input != "exit")
        {
            Console.WriteLine("Escribe 'on' para encender, 'off' para
apagar, 'volumeup' para subir volumen, 'volumedown' para bajar volumen,
'exit' para salir.");
            input = Console.ReadLine();

            switch (input)
            {
                case "on":
                    television.TurnOn();
                    break;
                case "off":
                    television.TurnOff();
                    break;
                case "volumeup":
                    television.VolumeUp();
                    break;
                case "volumedown":
                    television.VolumeDown();
                    break;
            }
        }
```

```
    }
}

class Television
{
    private bool isOn = false;
    private int volume = 10;

    public void TurnOn()
    {
        isOn = true;
        Console.WriteLine("Televisión encendida.");
    }

    public void TurnOff()
    {
        isOn = false;
        Console.WriteLine("Televisión apagada.");
    }

    public void VolumeUp()
    {
        if (isOn)
        {
            volume++;
            Console.WriteLine($"Volumen: {volume}");
        }
    }

    public void VolumeDown()
    {
        if (isOn)
        {
            volume--;
            Console.WriteLine($"Volumen: {volume}");
        }
    }
}
```

EJERCICIO 5

```
class Program
{
    static void Main()
    {
        Animal[] animals = { new Lion(), new Monkey(), new Elephant() };

        foreach (var animal in animals)
        {
            animal.Feed();
            // Nota: Con el tiempo, aquí tendrás que agregar más
operaciones,
            // lo que hará que el código sea menos mantenible.
        }
    }
}

abstract class Animal
{
    public abstract void Feed();
}

class Lion : Animal
{
    public override void Feed()
    {
        Console.WriteLine("El león está siendo alimentado con carne.");
    }
}

class Monkey : Animal
{
    public override void Feed()
    {
        Console.WriteLine("El mono está siendo alimentado con bananas.");
    }
}

class Elephant : Animal
{
    public override void Feed()
    {
        Console.WriteLine("El elefante está siendo alimentado con pastito
.");
    }
}
```

EJERCICIO 6

Problema: agregar una luz amarillo intermitente entre el amarillo y el rojo.

```
class Program
{
    static void Main()
    {
        var trafficLight = new TrafficLight();

        for (int i = 0; i < 5; i++)
        {
            trafficLight.ChangeLight();
            Thread.Sleep(1000); // Wait 1 second
        }
    }
}

class TrafficLight
{
    enum Light { Red, Yellow, Green }
    private Light currentLight;

    public TrafficLight()
    {
        currentLight = Light.Red;
        Console.WriteLine("Luz inicial es Roja.");
    }

    public void ChangeLight()
    {
        switch (currentLight)
        {
            case Light.Red:
                currentLight = Light.Green;
                Console.WriteLine("Cambio a Verde.");
                break;
            case Light.Green:
                currentLight = Light.Yellow;
                Console.WriteLine("Cambio a Amarillo.");
                break;
            case Light.Yellow:
                currentLight = Light.Red;
                Console.WriteLine("Cambio a Rojo.");
                break;
        }
    }
}
```

EJERCICIO 7

un sistema de cálculo de envío para un servicio de e-commerce. Inicialmente, el sistema podría estar usando condicionales para determinar qué algoritmo de cálculo de envío usar. Este enfoque puede volverse difícil de mantener y no es muy flexible si se desean agregar más algoritmos de envío.

```
class Program
{
    static void Main()
    {
        var shippingCalculator = new ShippingCalculator();

        Console.WriteLine("Costo de envío con UPS: " +
shippingCalculator.CalculateShippingCost("UPS", 5));
        Console.WriteLine("Costo de envío con FedEx: " +
shippingCalculator.CalculateShippingCost("FedEx", 5));
        Console.WriteLine("Costo de envío con DAC: " +
shippingCalculator.CalculateShippingCost("DAC", 5));

    }
}

class ShippingCalculator
{
    public double CalculateShippingCost(string courier, double weight)
    {
        switch (courier)
        {
            case "UPS":
                return weight * 0.75;
            case "FedEx":
                return weight * 0.85;
            case "DAC":
                return weight * 0.65;
            default:
                throw new Exception("Courier no soportado.");
        }
    }
}
```

EJERCICIO 8

```
class Program
{
    static void Main()
    {
        var emailService = new EmailService();
        emailService.SendEmail("john.doe@example.com", "Nueva promoción",
"¡Revisa nuestra nueva promoción!");
        emailService.SendNewsletter("john.doe@example.com", "Newsletter de
Junio", "Aquí está nuestro newsletter de Junio.");
    }
}

class EmailService
{
    public void SendEmail(string recipient, string subject, string message)
    {
        Console.WriteLine($"Enviando correo a {recipient} con el asunto
'{subject}': {message}");
        // Agregar código para enviar correo
    }

    public void SendNewsletter(string recipient, string subject, string
message)
    {
        Console.WriteLine($"Enviando newsletter a {recipient} con el asunto
'{subject}': {message}");
        // Agregar código para enviar newsletter
    }
}
```

EJERCICIO 9

En un sistema de soporte técnico donde las consultas de los clientes se pueden manejar en diferentes niveles, como soporte de nivel 1, nivel 2, y nivel 3. Inicialmente, el sistema podría estar usando condicionales para determinar qué nivel de soporte debe manejar una consulta. Este enfoque puede volverse difícil de mantener y poco flexible si se desean agregar más niveles de soporte o cambiar las condiciones para el manejo de consultas.

```
class Program
{
    static void Main()
    {
        SupportSystem supportSystem = new SupportSystem();
        supportSystem.HandleSupportRequest(1, "No puedo iniciar sesión.");
        supportSystem.HandleSupportRequest(2, "Mi cuenta ha sido
bloqueada.");
        supportSystem.HandleSupportRequest(3, "Necesito recuperar datos
borrados.");
    }
}

class SupportSystem
{
    public void HandleSupportRequest(int level, string message)
```

```
    {
        if (level == 1)
        {
            Console.WriteLine("Soporte de Nivel 1: Manejando consulta - " +
message);
        }
        else if (level == 2)
        {
            Console.WriteLine("Soporte de Nivel 2: Manejando consulta - " +
message);
        }
        else if (level == 3)
        {
            Console.WriteLine("Soporte de Nivel 3: Manejando consulta - " +
message);
        }
        else
        {
            Console.WriteLine("Consulta no soportada.");
        }
    }
}
```

EJERCICIO 10

```
class Program
{
    static void Main()
    {
        GreetingSystem greetingSystem = new GreetingSystem();

        greetingSystem.Greet("USA", "John");
        greetingSystem.Greet("Spain", "Juan");
        greetingSystem.Greet("Japan", "Yuki");
    }
}

class GreetingSystem
{
    public void Greet(string nationality, string name)
    {
        if (nationality == "USA")
        {
            Console.WriteLine($"Hello, {name}!");
        }
        else if (nationality == "Spain")
        {
            Console.WriteLine($"¡Hola, {name}!");
        }
        else if (nationality == "Japan")
        {
            Console.WriteLine($"こんにちは, {name}!");
        }
        else
        {
            Console.WriteLine("Nationality not supported.");
        }
    }
}
```