UNIDAD TEMÁTICA 5 – Patrones de diseño– Trabajo de Aplicación 3

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

```
public class Sandwich
{
    public string Bread { get; set; }
    public string Cheese { get; set; }
    public string Meat { get; set; }
    public string Vegetables { get; set; }
    public string Condiments { get; set; }

    public Sandwich(string bread, string cheese, string meat,
      string vegetables, string condiments)
    {
        Bread = bread;
        Cheese = cheese;
        Meat = meat;
        Vegetables = vegetables;
        Condiments = condiments;
    }

    public override string ToString()
    {
        return $"Sandwich with {Bread} bread, {Cheese} cheese,
            {Meat} meat, {Vegetables} vegetables, and {Condiments}
condiments.";
    }
}

class Program
{
    static void Main(string[] args)
    {
        Sandwich hamSandwich = new Sandwich("White", "Swiss", "Ham",
            "Lettuce, Tomato", "Mayo, Mustard");
        Sandwich turkeySandwich = new Sandwich("Wheat", "Cheddar",
            "Turkey", null, "Mayo");

        Console.WriteLine(hamSandwich);
        Console.WriteLine(turkeySandwich);
    }
}
```

EJERCICIO 2

```
public abstract class GameUnit
{
    public int Health { get; set; }
    public int Attack { get; set; }
    public int Defense { get; set; }

    // Simula la carga de recursos costosos como modelos 3D, texturas, etc.
    public virtual void LoadResources()
    {
        Console.WriteLine("Loading resources...");
    }
}

public class Archer : GameUnit
{
    public Archer()
    {
        LoadResources();
        Health = 100;
        Attack = 15;
        Defense = 5;
    }
}

public class Knight : GameUnit
{
    public Knight()
    {
        LoadResources();
        Health = 200;
        Attack = 20;
        Defense = 10;
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Creating original Archer...");
        Archer originalArcher = new Archer();

        Console.WriteLine("Copying Archers manually...");
        Archer copiedArcher1 = new Archer
        {
            Health = originalArcher.Health,
            Attack = originalArcher.Attack,
            Defense = originalArcher.Defense
        };

        Archer copiedArcher2 = new Archer
        {
            Health = originalArcher.Health,
            Attack = originalArcher.Attack,
            Defense = originalArcher.Defense
        };

        Console.WriteLine("Creating original Knight...");
        Knight originalKnight = new Knight();

        Console.WriteLine("Copying Knights manually...");
        Knight copiedKnight1 = new Knight
        {
            Health = originalKnight.Health,
            Attack = originalKnight.Attack,
            Defense = originalKnight.Defense
        };

        Knight copiedKnight2 = new Knight
        {
            Health = originalKnight.Health,
            Attack = originalKnight.Attack,
            Defense = originalKnight.Defense
        };
    }
}
```

EJERCICIO 3

```
public class MessagingApp
{
    public void SendMessage(string serviceType, string message)
    {
        if (serviceType == "SMS")
        {
            Console.WriteLine($"Sending SMS message: {message}");
            // Lógica para enviar SMS...
        }
        else if (serviceType == "Email")
        {
            Console.WriteLine($"Sending Email: {message}");
            // Lógica para enviar Email...
        }
        else if (serviceType == "Facebook")
        {
            Console.WriteLine($"Sending Facebook Message: {message}");
            // Lógica para enviar mensaje de Facebook...
        }
    }
}
```

EJERCICIO 4

```
public class Book
{
    public string Title { get; set; }
    public string Author { get; set; }
    public List<string> BorrowedStudents { get; set; }

    public Book()
    {
        // Simular la carga de recursos.
        Console.WriteLine("Acquiring a new book...");
        BorrowedStudents = new List<string>();
    }

    public void BorrowBook(string studentName)
    {
        BorrowedStudents.Add(studentName);
    }

    public void PrintBorrowedStudents()
    {
        Console.WriteLine($"Book: {Title}, Borrowed by: {string.Join(", ",
BorrowedStudents)}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Adquirir el libro original.
        Book originalBook = new Book
        {
            Title = "Harry Potter",
            Author = "J.K. Rowling"
        };
```

```
        // Prestar el libro original a un estudiante.
        originalBook.BorrowBook("Alice");

        // Adquirir una copia adicional del mismo libro manualmente.
        Book additionalCopy = new Book
        {
            Title = originalBook.Title,
            Author = originalBook.Author,
            BorrowedStudents = new List<string>() // Inicializar la lista
vacía.
        };

        // Prestar la copia adicional a otro estudiante.
        additionalCopy.BorrowBook("Bob");

        // Imprimir los estudiantes a los que se les prestó cada copia del
libro.
        originalBook.PrintBorrowedStudents();
        additionalCopy.PrintBorrowedStudents();
    }
}
```

EJERCICIO 5

```
public class TravelPlan
{
    public TravelPlan(string flight, string hotel, string carRental,
string[] activities, string[] restaurantReservations, ...)
    {
        // Constructor con muchos parámetros, algunos de los cuales pueden
ser opcionales (nulos o valores predeterminados).
    }

    // Propiedades y métodos...
}

// Ejemplo de uso:

TravelPlan plan = new TravelPlan("Flight1", "Hotel1", null, new string[]
{"Tour1", "Tour2"}, null, ...);
```

EJERCICIO 6

```
class Program
{
    static void Main()
    {
        // Crear un servicio
        SomeService service = new SomeService();

        // Realizar una tarea que requiere configuración
        service.PerformTask();

        // Otro ejemplo: acceder a la configuración desde otra parte de la
aplicación
        string apiEndpoint =
ConfigurationManager.Instance.GetConfiguration("apiEndpoint");
        Console.WriteLine($"API Endpoint: {apiEndpoint}");
    }

}
```