

UNIDAD TEMÁTICA 5 – Patrones de diseño– Trabajo de Aplicación 1

Para cada uno de los siguientes ejercicios, en equipo:

- 1- Determine que principio SOLID se está violando, agregando una justificación.
- 2- Agregue las clases, interfaces, métodos que considere necesarios para remediar la situación.

EJERCICIO 1

```
1 public abstract class Animal {
2     public abstract void Comer();
3     public abstract void Volar();
4 }
5
6 public class Perro: Animal {
7     public override void Comer() {
8         // El perro come
9     }
10
11     public override void Volar() {
12         throw new NotImplementedException();
13     }
14 }
```

EJERCICIO 2

```
1 public class Documento {
2     public string Contenido {
3         get;
4         set;
5     }
6 }
7
8 public class Impresora {
9     public void Imprimir(Documento documento) {
10         Console.WriteLine(documento.Contenido);
11     }
12
13     public void Escanear(Documento documento) {
14         // Código complejo para escaneo...
15     }
16 }
```

EJERCICIO 3

```
1 public class BaseDeDatos {  
2     public void Guardar(Object objeto) {  
3         // Guarda el objeto en la base de datos  
4     }  
5  
6     public void EnviarCorreo(string correo, string  
7         mensaje) {  
8         // Envía un correo electrónico  
9     }  
}
```

EJERCICIO 4

```
1 public class Robot {  
2     public void Cocinar() {  
3         // Cocina algo  
4     }  
5  
6     public void Limpiar() {  
7         // Limpia algo  
8     }  
9  
10    public void RecargarBateria() {  
11        // Recarga la batería  
12    }  
13 }
```

EJERCICIO 5

```
1 public class Cliente {  
2     public void CrearPedido() {  
3         // Crear un pedido  
4     }  
5 }
```

EJERCICIO 6

```
1 public class Pato {
2     public void Nadar() {
3         // Nada
4     }
5
6     public void Graznar() {
7         // Grazna
8     }
9
10    public void Volar() {
11        // Vuela
12    }
13 }
14
15 public class PatoDeGoma: Pato {
16     public override void Volar() {
17         throw new NotImplementedException();
18     }
19 }
```

EJERCICIO 7

```
public interface IDatabase {
    void Connect();
    void Disconnect();
    void WriteData();
}

public class Database: IDatabase {
    public void Connect() {
        // logic for connecting
    }
    public void Disconnect() {
        // logic for disconnecting
    }
    public void WriteData() {
        // logic for writing data
    }
}

public class ReadDatabase: IDatabase {
    public void Connect() {
        // logic for connecting
    }
    public void Disconnect() {
        // logic for disconnecting
    }
    public void WriteData() {
        throw new NotImplementedException();
    }
}
```

EJERCICIO 8

```
public class FileSaver {
    public void SaveToFile(string fileName, Document doc) {
        if (string.IsNullOrEmpty(fileName))
            throw new ArgumentException();
        // logic for saving the document
    }
}

public class AutoSave : FileSaver {
    public void Save(Document doc) {
        SaveToFile("", doc);
    }
}
```

EJERCICIO 9

```
public class User {
    public bool IsAdmin { get; set; }
    public bool CanEditPost(Post post) {
        return IsAdmin || post.Author == this;
    }
}

public class Post {
    public User Author { get; set; }
}
```

EJERCICIO 10

```
public class MusicPlayer
{
    public void PlayMp3(string fileName)
    {
        // Lógica para reproducir archivos MP3
    }

    public void PlayWav(string fileName)
    {
        // Lógica para reproducir archivos WAV
    }

    public void PlayFlac(string fileName)
    {
        // Lógica para reproducir archivos FLAC
    }
}
```