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

```
public abstract class Animal {
  public abstract void Comer();
  public abstract void Volar();
}

public class Perro: Animal {
  public override void Comer() {
    // El perro come
  }

  public override void Volar() {
    throw new NotImplementedException();
  }
}
```

EJERCICIO 2

```
public class Documento {
  public string Contenido {
    get;
    set;
  }
}

public class Impresora {
  public void Imprimir(Documento documento) {
    Console.WriteLine(documento.Contenido);
  }

  public void Escanear(Documento documento) {
    // Código complejo para escaneo...
  }
}
```

EJERCICIO 3

```
public class BaseDeDatos {
  public void Guardar(Object objeto) {
    // Guarda el objeto en la base de datos
  }

  public void EnviarCorreo(string correo, string
      mensaje) {
    // Envía un correo electrónico
  }
}
```

EJERCICIO 4

```
public class Robot {
  public void Cocinar() {
    // Cocina algo
  }

  public void Limpiar() {
    // Limpia algo
  }

  public void RecargarBateria() {
    // Recarga la batería
  }
}
```

EJERCICIO 5

```
public class Cliente {
  public void CrearPedido() {
    // Crear un pedido
  }
}
```

EJERCICIO 6

```
public class Pato {
  public void Nadar() {
    // Nada
  }

  public void Graznar() {
    // Grazna
  }

  public void Volar() {
    // Vuela
  }
}

public class PatoDeGoma: Pato {
  public override void Volar() {
    throw new NotImplementedException();
  }
}
```

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
    }
}
```