1er Ejercicio

Implementación de los métodos de patrones creacionales:

* Factory Method: El Factory Method es un patrón de diseño que permite delegar la creación de objetos a las subclases, manteniendo una interfaz común en la superclase.
  + Ventajas:
    - Evitas un acoplamiento fuerte entre el creador y los productos concretos.
    - Puedes mover el código de creación de producto a un lugar del programa, haciendo que el código sea más fácil de mantener.
* Singleton Method: Singleton es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia
  + Ventajas:
    - Puedes tener la certeza de que una clase tiene una única instancia.
    - Obtienes un punto de acceso global a dicha instancia.
    - El objeto Singleton solo se inicializa cuando se requiere por primera vez.

2do Proyecto

Implementación de los métodos de los patrones estructurales:

* Adapter: **Adapter** es un patrón de diseño estructural que permite la colaboración entre objetos con interfaces incompatibles.
  + Ventajas:
    - Puedes separar la interfaz o el código de conversión de datos de la lógica de negocio primaria del programa.
    - Puedes introducir nuevos tipos de adaptadores al programa sin descomponer el código cliente existente.
* Facade: Facade es un patrón de diseño estructural que proporciona una interfaz simplificada a una biblioteca, un framework o cualquier otro grupo complejo de clases.
  + Ventajas:
    - Puedes aislar tu código de la complejidad de un subsistema.

3er Ejercicio

Metodos utilizados en los patrones de comportamiento:

* Strategy: Strategy es un patrón de diseño de comportamiento que te permite definir una familia de algoritmos, colocar cada uno de ellos en una clase separada y hacer sus objetos intercambiables
  + Ventajas:
    - Puedes intercambiar algoritmos usados dentro de un objeto durante el tiempo de ejecución.
    - Puedes aislar los detalles de implementación de un algoritmo del código que lo utiliza.