Informe sobre Diseño y Patrones de Software

Introducción

Este informe detalla el diseño y los patrones de software implementados en un sistema de gestión de flota de barcos. El sistema permite realizar operaciones como la administración de barcos, cambios en sus estados, gestión de reparaciones y ejecución de misiones. El documento describe cómo se aplicaron los principios de diseño SOLID y, en particular, el Patrón State, proporcionando una visión tanto de la arquitectura del sistema como de su comportamiento dinámico.

Principios de Diseño Usados (SOLID)

1. Principio de Responsabilidad Única (SRP): Cada clase debe tener una única responsabilidad y, por lo tanto, una sola razón para cambiar. Ejemplo: La clase Flota se encarga únicamente de administrar los barcos, sin incluir lógica sobre los estados de los barcos.

2. Principio de Abierto/Cerrado (OCP): El código debe estar abierto a extensiones, pero cerrado a modificaciones. Ejemplo: La interfaz EstadoBarco permite agregar nuevos estados sin modificar las clases existentes.

3. Principio de Sustitución de Liskov (LSP): Las subclases deben poder sustituir a sus clases base sin alterar el comportamiento del programa. Ejemplo: Todas las subclases de Barco (Ultraligero, Ligero, etc.) pueden usarse donde se espera un Barco.

4. Las interfaces son específicas para cada cliente, evitando que las clases implementen métodos innecesarios. La interfaz EstadoBarco define solo los métodos necesarios para gestionar transiciones de estados (danar, reparar, hundir, etc.), asegurando simplicidad y claridad.

5. Principio de Inversión de Dependencia (DIP): Los módulos de alto nivel no deben depender de módulos de bajo nivel; ambos deben depender de abstracciones. La clase Flota interactúa con objetos abstractos como Barco y EstadoBarco, evitando dependencias directas con implementaciones específicas.

Patrón de Diseño Usado: State

El Patrón State permite que un objeto cambie su comportamiento de manera dinámica según su estado interno. Cada estado se representa mediante una clase separada que implementa una interfaz común. Esto elimina la necesidad de condicionales extensos y facilita la extensión del sistema.
En este sistema, los barcos pueden estar en diferentes estados (Operativo, EnEjercicio, Dañado, etc.). El Patrón State permite que el comportamiento de un barco cambie según su

estado sin añadir lógica compleja en la clase `Barco`. Además, facilita la incorporación de nuevos estados en el futuro.

Conclusión

Este sistema aplica principios de diseño SOLID y el Patrón State para garantizar un diseño modular, extensible y fácil de mantener. El uso del Patrón State permite manejar de manera dinámica los estados de los barcos, mientras que los principios SOLID aseguran una estructura coherente y flexible.

Diagrama de Clases

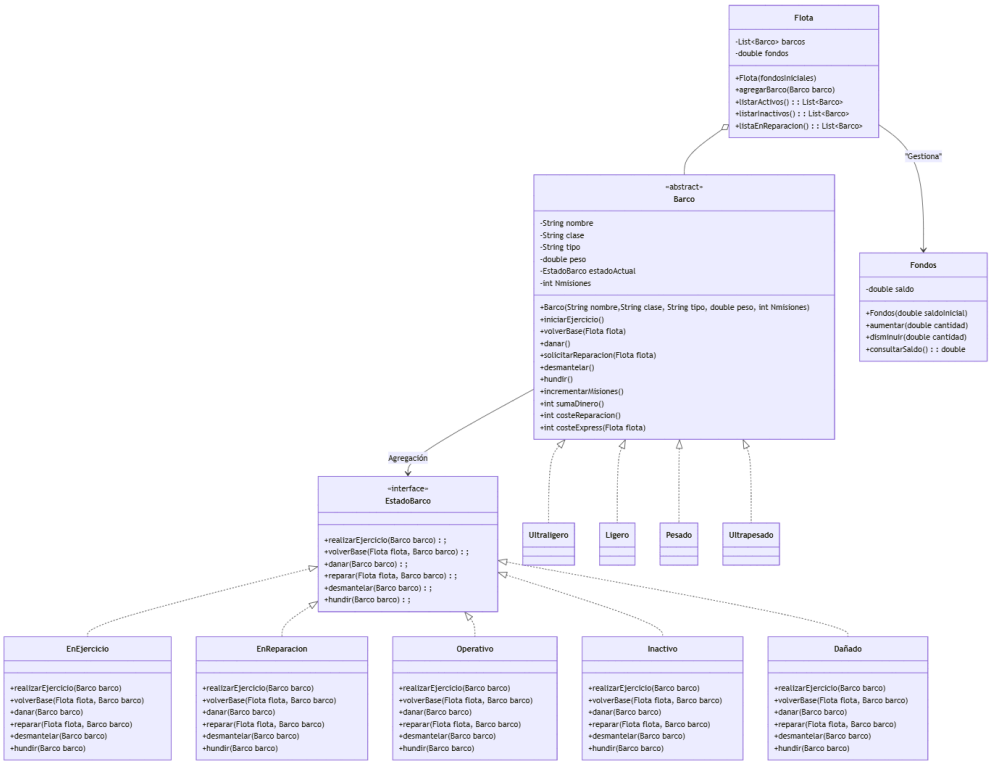

Diagrama de Estados