Informe del ejercicio 2 de la práctica de diseño

Principios de diseño usados

1. Principio de Responsabilidad Única (SRP):
Este principio establece que una clase debe tener solo una razón para cambiar (una única responsabilidad) .En el codigo la clase Fleet tiene la responsabilidad de representar una flota y manejar operaciones relacionadas con la flota, como recibir daño y mostrar información.

2. Principio de Abierto/Cerrado (OCP):
Este principio establece que una clase debe ser abierta para extensión pero cerrada para modificación. Se logra mediante la extensión de la funcionalidad sin modificar el código existente. En el código la introducción de nuevos tipos de nodos en el mapa (como AirRaid o Maelstrom) permite extender el comportamiento sin modificar las clases existentes.

3. Principio de Sustitución de Liskov (LSP):
Este principio establece que los objetos de una clase base deben poder ser sustituidos por objetos de sus clases derivadas sin afectar la corrección del programa. En el código las clases derivadas de MapNode (cómo End, Battle, etc...) pueden ser utilizadas en lugar de MapNode en el código sin afectar su funcionamiento.

4. Principio de Segregación de Interfaces (ISP):
Este principio establece que una clase no debe verse obligada a implementar interfaces que no utiliza. En el código aunque no se utilizan interfaces, el diseño no presenta una violación directa de este principio.

5. Principio de Inversión de Dependencias (DIP):
Este principio establece que las clases de alto nivel no deben depender de clases de bajo nivel, sino de abstracciones. Además, las abstracciones no deben depender de detalles, sino de otras abstracciones. En el código las clases de nivel superior dependen de clases abstractas en lugar de depender directamente de implementaciones concretas.

Patrón de diseño

El patrón Composición se utiliza para tratar tanto a los objetos individuales como las composiciones de objetos de manera uniforme. En nuestro caso, los nodos del mapa se manejan a través de la clase abstracta “MapNode”, lo cual nos permite construir estructuras jerárquicas de nodos.

Diagrama de clases

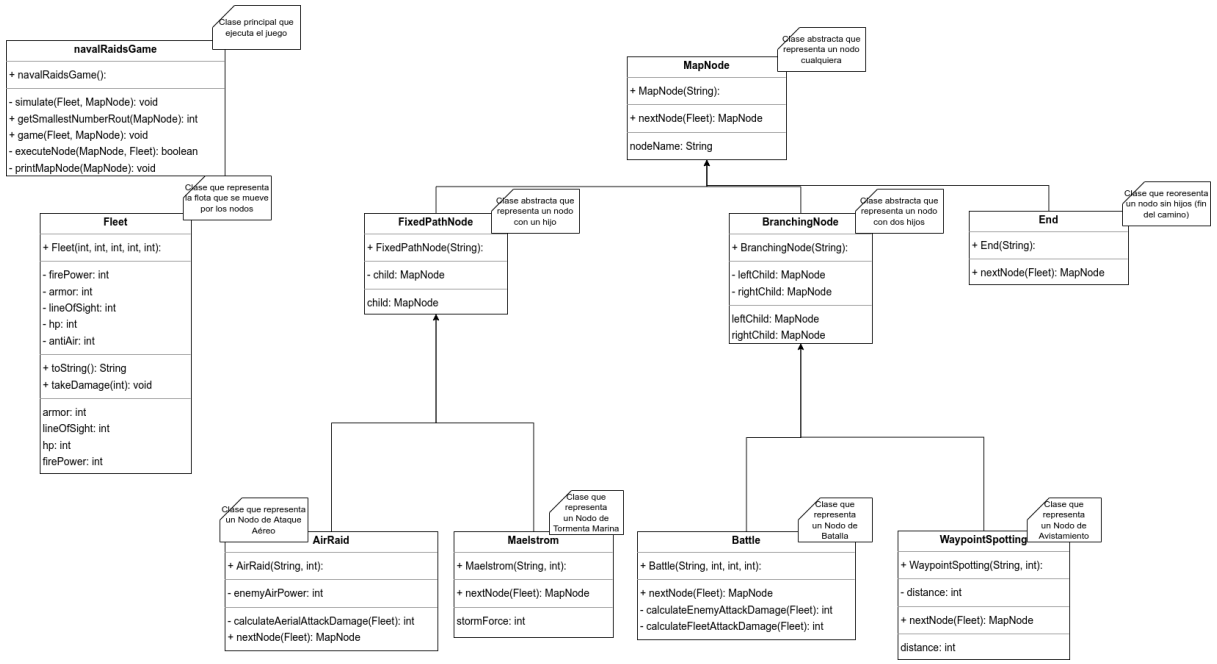
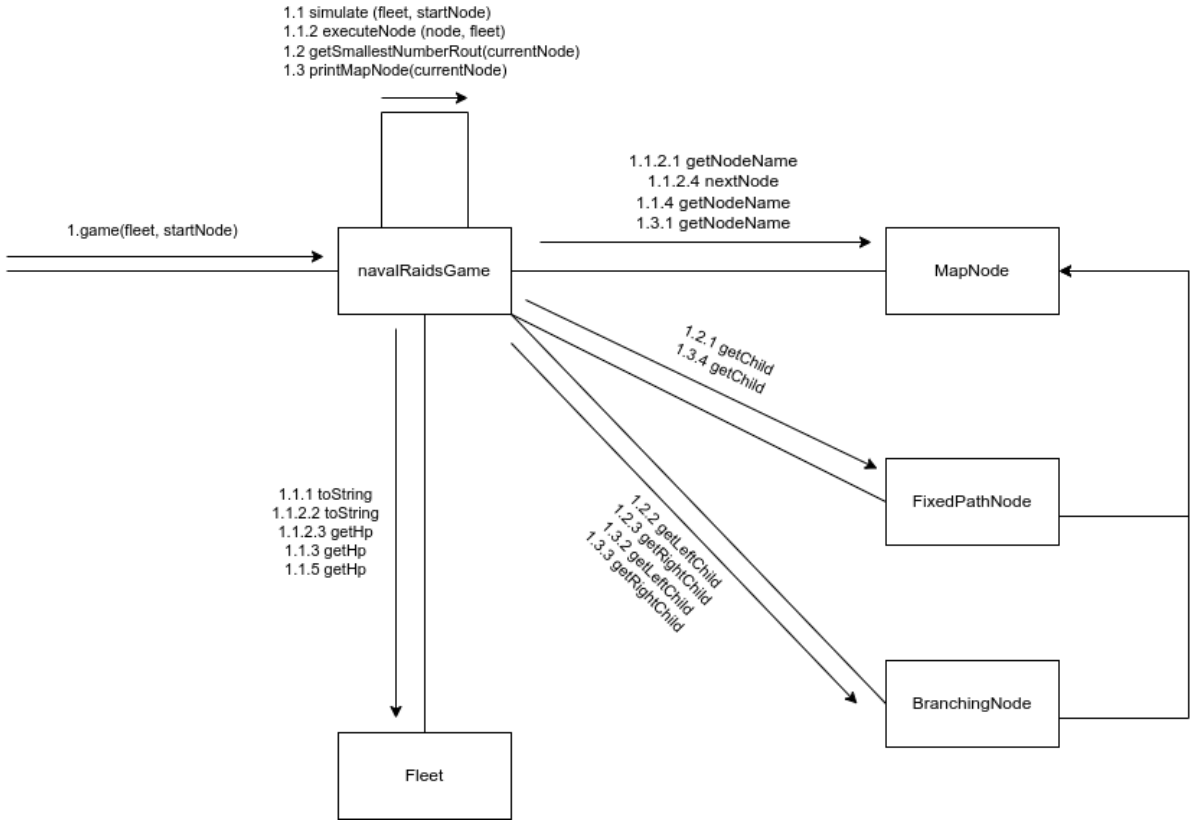


Diagrama dinámico



Este es un diagrama de comunicación y es el más adecuado para este problema porque es especialmente efectivo para representar la secuencia de mensajes e interacciones entre objetos a lo largo del tiempo.