Práctica 4

Equipo:equipo

Integrantes:

- Arcos Morales Ramón. No: 319541478
- Casarrubias Casarrubias Victor Manuel. No: 421003581
- López Asano Miguel Akira. No: 320219089

Para compilar la práctica con jdk 20:
javac src/*.java
cd src/
java Main
Principios de diseño de los patrones:

- El patrón Builder se caracteriza por construir objetos complejos a partir de más simples. Se encapsula la creación. Así cada componente se encuentra modularizado y un director permite organizar la construcción del objeto complejo. Una desventaja de este patrón es que necesita un mayor conocimiento del dominio del cliente que otros patrones.
- El patrón Factory se caracteriza por diferir la creación del objeto a subclases y así escoger cuál se usará para instanciar. Oculta toda la lógica de creación. Una desventaja es que si se usan clases pertenecientes a una estructura de herencia cambiante, puede ser complicado gestionar los cambios que se necesitan realizar a la fábrica.
- El patrón Abstract Factory por tener una interfaz que proporciona fábricas de objetos relacionados. Se tiene una fábrica quee crea fábricas. Cada una de esas fábricas sigue el patrón Factory. Si se tiene un objeto complejo con varios componentes con opciones distintas de cada uno, se tendría el mismo número de fábricas; lo que puede llevar a cierta dificultad en el mantenimiento, siendo así una desventaja de abstract factory.

Algunas consideraciones:

- Se decidió utilizar el patrón Builder, puesto que se tiene completo conocimiento del proceso de creación de las naves para esta práctica. Como builder se utiliza especialmente para objetos compuestos, se pensó que sería la opción más sencilla de implementar. Así, nos permitió crear las

naves en varios pasos y de manera variable, como lo indica las especificaciones. Finalmente encapsulamos la instancia de la nave y solo utilizamos nuevas instancias de los componentes.