INFORME OBJETOS 2

Integrantes

- Bracco Bautista - bautista.bracco@gmail.com
- Laime Matias - matulai1221@gmail.com

Diseño

Para el desarrollo del diseño en general del proyecto se realizó en conjunto un diagrama de clases(UML) en el cual se definió y discutió acerca de las clases, interfaces y posibles usos de patrones de diseño que serían necesarios para el desarrollo del proyecto.
Si bien el diseño inicial tuvo pequeñas modificaciones a medida que se fue implementando el proyecto, la base del mismo no se modificó radicalmente.

Detalles de implementación

Para el caso en el cuál la terminal debe avisar al consignee o shipper lo simulamos realizando una impresión en consola.
Para realizar el filtro creamos una nueva clase llamada RutaMaritima, la cual la pasamos como parametro al metodo que realiza el filtrado junto al filtro

Problemas

Nos enfrentamos a varios problemas en este trabajo, principalmente ocasionados por la inexperiencia para desarrollar un trabajo de esta magnitud en grupo.
Iniciamos nuestro proyecto enfrentándonos a uno de los mayores desafíos: la adopción del sistema de control de versiones distribuido Git y el desconocimiento del funcionamiento de la plataforma github. Este obstáculo surgió debido a la falta de experiencia de dos miembros del equipo con el uso de este sistema, resultando en un considerable retraso al iniciar el desarrollo del programa.
Otro problema relacionado al primero es el inadecuado flujo de trabajo empleado por los integrantes, como la falta de criterio para crear nuevas ramas, definirles un nombre apropiado y realizar un pull diario para tratar de manipular siempre la versión más actualizada posible de la rama.
Otro desafío al tratar de asimilar el principio de inversión de dependencia para organizar nuestras clases y módulos, asegurándonos de que estas se relacionan exclusivamente con abstracciones e interfaces, y no dependieran directamente de implementaciones concretas. Implementar este enfoque resultó inicialmente complicado porque solo lo aplicamos en la práctica de mocking. Este proceso nos llevó tiempo y esfuerzo para comprender completamente cómo integrar este principio de diseño en nuestro código de manera coherente.

Patrones

Strategy:

Contexto: **Terminal**

Estrategia: **MejorCircuitoStrategy**

Cliente: **TerminalInterface**

Estrategias concretas:

- **MenorCantidadDeTerminalesStrategy**
- **MenorPrecioStrategy**
- **MenorTiempoStrategy**

State

Contexto: **Buque**

Estado: **StateBuque**

Subclases de estado concreto:

- **Outbound**
- **Inbound**
- **Arrived**
- **Departing**
- **Working**

Composite

Cliente: **Terminal**

Componente: **Filtro**

Hoja: **FiltroFecha, FiltroPuerto**

Compuesto: **OperadorLógico**