Trabajo Práctico Final
Programación con Objetos II
2do cuatrimestre 2023

Integrantes:

- Lorenzo Ferraces (lorenzoferraces@gmail.com)
- Matías Netto (matiasnetto03@gmail.com)
- Santino Cargnelutti (scargnelutti18@gmail.com)

Decisiones de diseño:

Como punto de partida para la realización de este trabajo creamos un diagrama UML que modificamos colaborativamente, con el objetivo de identificar las clases que formarían parte del trabajo. Definimos sus atributos y parte de sus mensajes. No intentamos abarcarlo todo en un principio porque aún teníamos dudas que fuimos resolviendo con las semanas.

A continuación, detallamos las clases que destacan por el patrón de diseño al que las incorporamos.

Patrón Composite para la búsqueda de rutas marítimas

La TerminalGestionada debía ser capaz de realizar búsquedas sobre las rutas marítimas, ofreciendo la posibilidad de aplicar los siguientes filtros:

- Puerto destino.
- Fecha de salida (desde la terminal gestionada).
- Fecha de llegada (a puerto destino).

Se debían permitir combinaciones de los filtros mediante los operadores AND y OR, por lo que aplicamos el patrón Composite. Definimos clases para los filtros y los operadores, y permitimos a estos últimos operar recursivamente. Los roles quedaron de la siguiente manera:

- IFiltrable: Es la interfaz que actúa como Componente.
- FiltroLlegaEn, FiltroPuertoDestino, FiltroSaleEn: Son clases concretas que implementan IFiltrable. Actúan como Hojas.
- FiltroAnd, FiltroOr: Son clases concretas que implementan IFiltrable. Actúan como Compuesto.

Patrón State para las fases del buque

La clase Buque atraviesa 5 fases desde que es instanciada, a saber: Outbound, Inbound, Arrived, Working y Departing. Estas fases son lineales, es decir, una le sigue a la otra. Debido a que el buque es una entidad ajena a la terminal, se asume que todos los buques tienen como fase inicial Outbound.

El Buque tiene diferentes comportamientos tales como avisar a la terminal sobre su arribo, avisar a la terminal sobre su partida, recibir la orden para realizar los trabajos de carga y descarga, entre otros. Para modelarlo, utilizamos el patrón State, por lo que creamos una interfaz FaseDeBuque, que es implementada por 5 clases concretas. Cada una de estas clases corresponden a las fases previamente mencionadas, e implementan los métodos que les corresponda realizar, dejando los demás vacíos. De esta forma, el buque reenvía a su fase actual los mensajes que dependan de ella.

- Buque: Actúa como contexto.
- FaseDeBuque: Es la interfaz que actúa como State..
- FaseDeBuqueOutbound, FaseDeBuqueInbound, FaseDeBuqueArrived, FaseDeBuqueWorking, FaseDeBuqueDeparting: Son los estados concretos, que implementan la interfaz.

Patrón Strategy para los criterios de mejor circuito

La TerminalGestionada es capaz de devolver el mejor circuito, según el criterio actual que tenga. Estos pueden ser:
a. Menor tiempo total de recorrido entre origen y destino.
b. Menor precio total de recorrido entre origen y destino.
c. Menor cantidad de terminales intermedias entre origen y destino.

Debido a que la terminal debía ser capaz de cambiar su criterio de forma flexible, aplicamos el patrón Strategy. De esta forma, para cada criterio creamos una clase concreta, que hereda de la clase abstracta CriterioCircuito. Las 3 implementan el método buscar(List<CircuitoMarítimo>) a su manera. Entonces, cuando se le pide el mejor circuito a la terminal, ésta le envía a su criterio actual el mensaje asociado al método previamente mencionado.

- TerminalGestionada: Es el contexto.
- CriterioCircuito: Es la clase abstracta que actúa como Strategy.
- CriterioCircuitoMenorTiempo, CriterioCircuitoMenorPrecio, CriterioCircuitoMenorNroDeTramos: Son las estrategias concretas que heredan de CriterioCircuito.

Ya que cada criterio es independiente del otro, si la terminal desea agregar uno nuevo, simplemente debe crear una clase que encapsule su algoritmo y extienda a CriterioCircuito.