TRABAJO PRÁCTICO FINAL

Programación Orientada a Objetos II

Integrantes:

● Lucas Paolo (Email: lucaspaolo002@gmail.com).

● Ivan Rosendo (Email: ivanrosendo1102@gmail.com).

● Mauricio Velazquez (Email: mauriciovelazquez211@gmail.com).

Decisiones de diseño:

- En el método exportar de la TerminalGestionada decidimos siempre darle al cliente el viaje de la naviera con la fecha de salida más temprana.
- Incorporamos la clase Reloj que devuelve la hora actual para poder verificar si el Shipper y el Consignee llegan en horario o no, en los tests la hora del reloj está mockeada para nosotros poder hacer las pruebas con los diferentes horarios.
- Suponemos que todas las importaciones llegan a las 11:00 AM para que las verificaciones sean más sencillas de hacer, ya que así el Consignee tiene hasta las 11:00 AM del otro día para buscar su carga.
- Asimismo los turnos de los Shippers siempre se asignan a las 9:00 AM del día que el buque va a salir del puerto a través de la clase Turno.

Patrones utilizados:

Patrón Composite:

Aplicamos este patrón porque para los filtros se debian permitir combinaciones de estos mismos mediante los operadores AND y OR.

Los roles entonces son:

- La interfaz **Filtro** con el rol de **Componente**.
- Las clases **FiltroFechaDeLlegada**, **FiltroFechaDeSalida**, **FiltroPuertoDestino** con el rol de **Hojas** que implementan la interfaz Filtro.
- Las clases **FiltroAnd** y **FiltroOr** ambos con el rol de **Compuesto** que implementan también la interfaz Filtro.

Patrón State:

Aplicamos este patrón porque la clase Buque posee 5 fases que se relacionan entre sí, siendo Outbound la fase inicial.

Los roles entonces son:

- La clase **Buque** con el rol de **Contexto**.
- La interfaz **FaseDeBuque** con el rol de **State**.
- Las clases **FaseDeBuqueOutbound**, **FaseDeBuqueInbound**, **FaseDeBuqueArrived**, **FaseDeBuqueWorking** y **FaseDeBuqueDeparting** con el rol de **Estados Concretos** que implementan la interfaz FaseDeBuque.

Patrón Strategy:

Aplicamos este patrón porque la TerminalGestionada debía ser capaz de devolver el mejor circuito en base a criterios y de forma flexible. Para ello creamos la interfaz EstrategiaMejorCircuito.

Los roles entonces son:

- La **TerminalGestionada** con el rol de **Contexto**.
- La interfaz **EstrategiaMejorCircuito** con el rol de **Strategy**.
- Las clases **EstrategiaMenorCantidadDeTramos**, **EstrategiaMenorCosto** y **EstrategiaMenorTiempo** con el rol de **Estrategias Concretas** que implementan la interfaz EstrategiaMejorCircuito.

Como cada estrategia es independiente una de otra, se puede agregar en cualquier momento otro criterio.