DOCUMENTACIÓN TRABAJO PRÁCTICO TERMINAL PORTUARIA:

INTEGRANTES:

Lucio Jara
email: luciojara8@gmail.com

Ian Bechtoltd
email: cristopherbechtholdt@gmail.com

Gabriel Martinez
email: gabi.m@hotmail.com.ar

PATRONES DE DISEÑO UTILIZADOS:

PATRÓN STATE: BuqueState

Roles:

- State: BuqueState
- Context: Buque
- ConcreteClasses: Arrived, Inbound, Working, Departing, Outbound

PATRÓN STRATEGY: SeleccionadorCircuito

Roles:

- Context: TerminalPortuaria
- Strategy: SeleccionadorCircuito
- ConcreteStrategies: MenorPrecioRecorrido, MenorTiempoTotalRecorrido, MenorCantidadTerminales.

PATRÓN COMPOSITE: BusquedaMaritima

Roles:

- Component: BusquedaMaritima.
- Leaf: FechaCondicion, FechaLlegada, FechaSalida.
- Composite: And, Or.

PATRÓN OBSERVER:

Roles:

- Subject: Gps, GpsImpl
- Observer: Localizable
- Concrete Observer: Buque

DECISIONES DE DISEÑO:

INTERFACES IMPLEMENTADAS:

- Cliente: esta interfaz está diseñada para que sea implementada por el Consignee y por el Shipper, se tomó la decisión de que sea una interfaz y no una clase abstracta porque son clases que tienen cosas en común pero su implementación es distinta. Además de que está abierta al principio open-closed de SOLID.

- Factura: esta interfaz está diseñada para que sea implementada por FacturaSimple, se tomó esta decisión de que sea una interfaz para poder respetar el principio open-closed de SOLID en caso de que aparezca otro tipo de Factura.
- MailManager: Esta clase se inyecta en TerminalPortuaria para proveer la opción de enviar y recibir mails. Cumple la inversión de dependencias.

- BusquedaMaritima: Es la interfaz que permite evaluar una expresión compleja de viajes, para filtrarlos según el interés del usuario. Está hecho para implementar el patrón State.
- BuqueState: Es una interfaz que permite cambiar polimórficamente el estado del buque, para que accione coherentemente. Permite cambiarlo en tiempo de compilación.

CLASES ABSTRACTAS Y SUBCLASES IMPLEMENTADAS:

- Orden : Se decidió hacer la clase Orden como una clase abstracta porque posee comportamiento y atributos que sus subclases OrdenImportacion y OrdenExportacion tienen en común, solamente varía el tipo de atributo orden para cada una y el método *getFactura()*.

- Container: Se decidió hacer a la clase Container como una clase abstracta porque posee comportamientos y atributos que sus subclases Dry, Tanque y Reefer tienen en común, solamente la Reefer tiene una variación en el código.

- Servicio: Se decidió hacer a la clase Servicio como una clase abstracta posee comportamientos y atributos que sus subclases Lavado, Pesado , Excedente y Electricidad tienen en común, todas las subclases tienen en común el Container como atributo y el método abstracto *montoTotal(Orden)*.

- FacturaResponsableViaje: Se decidió hacer a la clase FacturaResponsableViaje como una subclase de FacturaSimple porque implementa su mismo protocolo modificando solamente el método getMontoTotalFacturado() al sumarle al monto el precio final por los tramos recorridos.

- FechaCondicion: Se eligió como clase abstracta de BusquedaMaritima para el caso “Leaf” ya que compartían la evaluación, pero diferían en qué fecha devolver.

- SeleccionadorCircuito: Como la lógica de búsqueda se repetía al seleccionar el mejor circuito, se delega a la clase abstracta, dejando la evaluación de la condición a las subclases específicas.

Detalles de diseño:

1- Utilizamos un poco el paradigma funcional en FechaCondición para dar más flexibilidad al usuario sobre qué condición utilizar para la BúsquedaMaritima. Esto se podría haber hecho con clases de operadores con tipos genéricos comparables. Por la reutilización de métodos útiles de la clase Date, decidimos dejarlo así.
2- Decidimos delegar la comprobación de la llegada del turno, tanto en la exportación y la importación, a la clase Turno. También decidimos dar un turno tanto a la importación como a la exportación por coherencia.
3- Delegamos la tenencia de las ordenes y turnos a las clases OrdenesExportacionManager y OrdenesImportacionManager para cumplir con el Single Responsability Principle