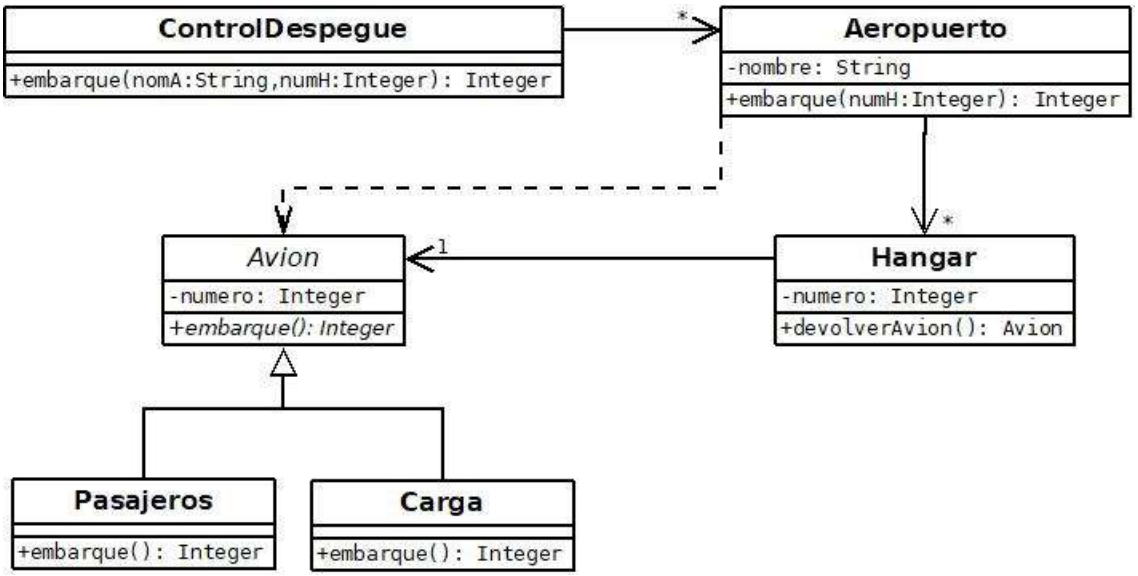
Práctico de Programación 4 – clase 9

Práctico 5, Ejercicio 2:

La confección del Diagrama de Clases de Diseño (DCD) a partir del Modelo de Dominio y de los Diagramas de Comunicación es una tarea conceptualmente sencilla y bastante sistemática. Sin embargo, se debe realizar de forma cuidadosa para no omitir ningún detalle. Ver las diapositivas de Teórico *14 - Diseño: Estructura*.

Para este caso el DCD resultante es el siguiente:



La **operación**:

- ControlDespegue::embarque surge del punto de entrada de la operación.
- Aeropuerto::embarque surge del mensaje 2.
- Hangar::devolverAvion surge del mensaje 2.2.
- Avion::embarque surge del mensaje 2.3. La operación no tiene método en la clase Avion, pero sí tiene en sus derivadas Pasajeros y Carga.

La **navegabilidad**:

- De ControlDespegue hacia Aeropuerto surge del mensaje 1, el cual sugiere que la colección de aeropuertos es administrada por el controlador. Por ser una colección, la multiplicidad es *.
- De Aeropuerto hacia Hangar es similar al punto anterior, en particular por el mensaje 2.1.
- De Hangar hacia Avion surge del mensaje 2.2. En particular, el tipo asociativo Hangar (entre Aeropuerto y Avion) se diseña de forma tal que el aeropuerto tiene visibilidad sobre sus hangares y cada hangar tiene visibilidad sobre su único avión (multiplicidad 1 en el DCD).

La **dependencia** de Aeropuerto con Avion es porque el Hangar le devuelve su Avion (2.2), al cual luego el Aeropuerto le envía el mensaje embarque (2.3). Notar que es una visibilidad temporal (no permanente).

Patrones de diseño

En esta clase veremos dos patrones de diseño que no se ven en el Teórico: Template Method y Strategy.

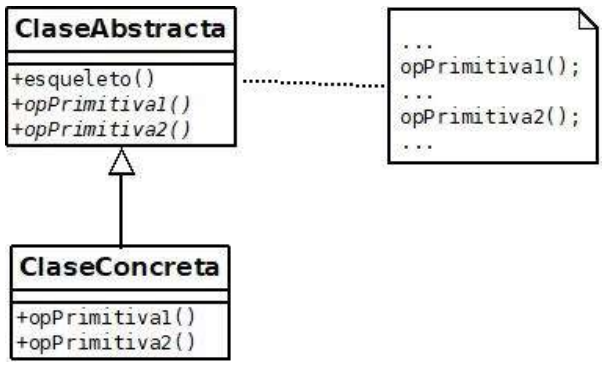
Template Method

Propósito (problema al que aplica): Definir el esqueleto de un algoritmo en una operación, delegando algunos pasos a subclases. Permitir a las subclases redefinir ciertos pasos de un algoritmo sin cambiar su estructura.

Participantes:

- ClaseAbstracta: Define operaciones primitivas abstractas que son implementadas por subclases concretas, que representan pasos de un algoritmo. Implementa un esqueleto de algoritmo que invoca a las operaciones primitivas, así como otras operaciones definidas en ClaseAbstracta o en otros objetos.
- ClaseConcreta: Implementa las operaciones primitivas para ejecutar los pasos del algoritmo que son específicos a la subclase.

Estructura y comportamiento:



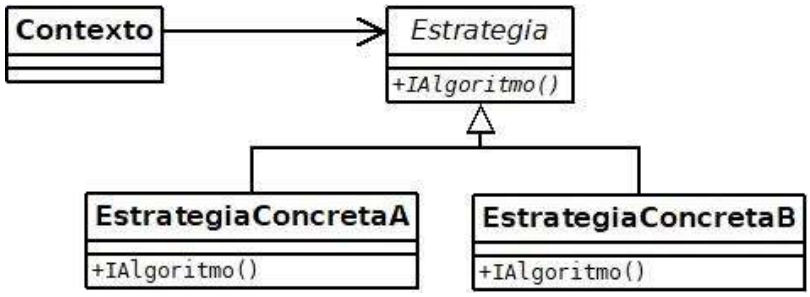
Strategy

Propósito (problema al que aplica): Definir una familia de algoritmos, encapsular cada uno y hacerlos intercambiables. Permitir que el algoritmo varíe independientemente del cliente que lo use.

Participantes:

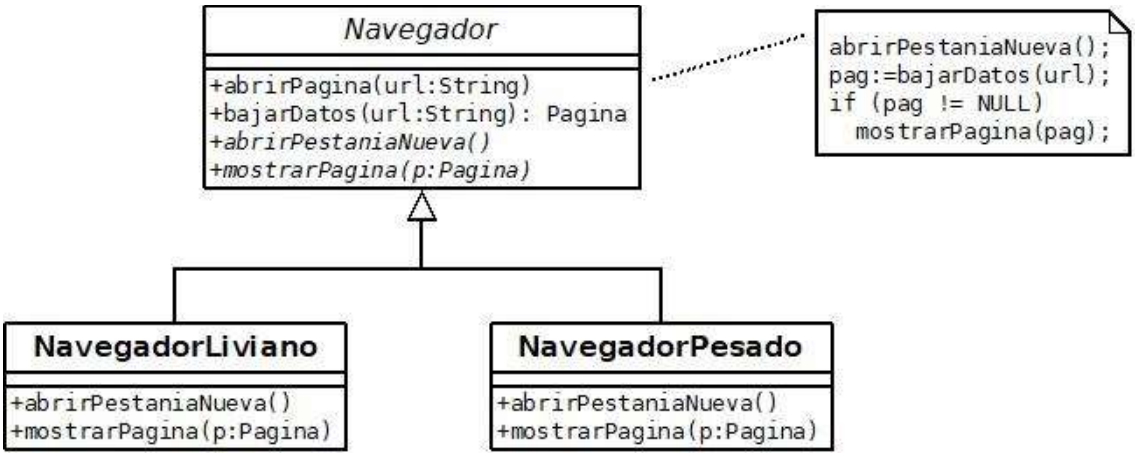
- Estrategia: Declara una interface común a todos los algoritmos soportados. Contexto usa esa interface para invocar el algoritmo definido por una EstrategiaConcreta.
- EstrategiaConcreta: Implementa el algoritmo usando la interface Estrategia.
- Contexto: Mantiene una referencia a un objeto que implementa la interface Estrategia.

Estructura y comportamiento:



Práctico 5, Ejercicio 7:

Parte 1



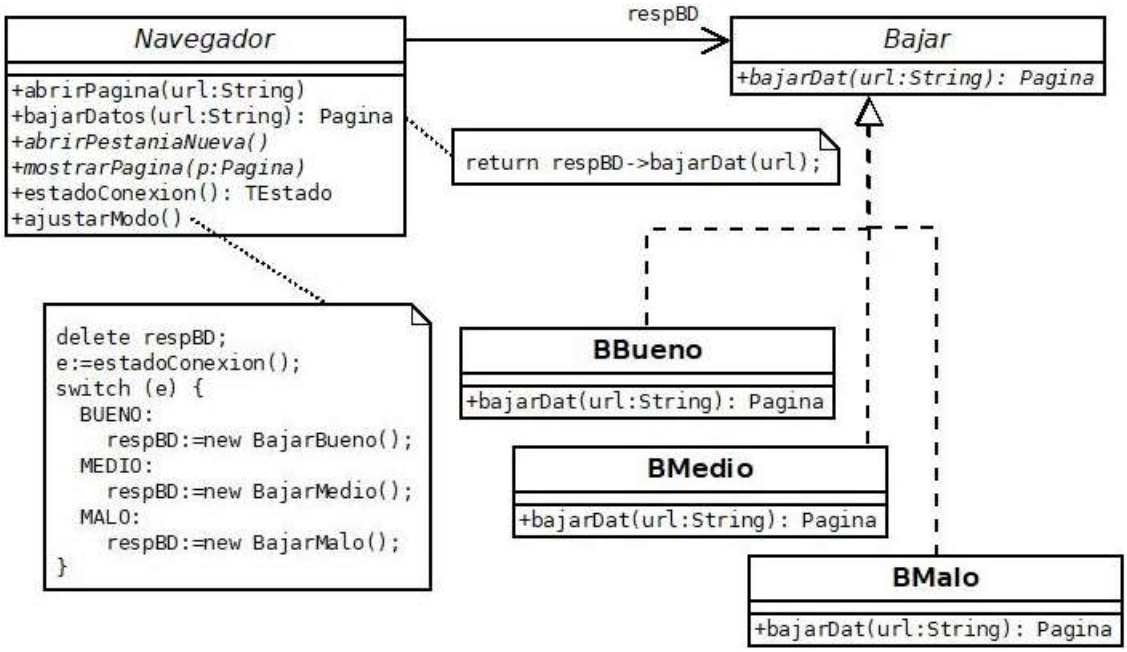
Se utiliza el patrón Template Method con los siguientes roles:

- Navegador es la ClaseAbstracta. La operación abrirPagina es el esqueleto y las operaciones abrirPestania y mostrarPagina son primitivas, a ser implementadas en la ClaseConcreta.
- NavagadorLiviano y NavegadorPesado son dos clases que representan cada una a ClaseConcreta. Dan métodos a las operaciones primitivas.

Parte 2

Respecto a la solución de la parte 1:

- Se agregan las operaciones ajustarModo y estadoConexion en la clase Navegador.
- Se modifica el comportamiento de la operación bajarDatos de la clase Navegador.
- Se agrega la interface Bajar y sus realizaciones BBueno, BMedio y BMalo.



Se utiliza el patrón Strategy con los siguientes roles:

- Navagador es el Contexto.
- Bajar es la Estrategia, donde la operación bajarDat cumple el rol de IAlgoritmo.
- BBueno, BMedio y BMalo son estrategias concretas, que dan diferentes métodos a la operación bajarDat.

Observaciones:

- La operación Navegador::bajarDatos delega la responsabilidad de bajar los datos a la interface Bajar, a través de la referencia respBD.
- La operación Navegador::ajustarModo destruye al objeto referenciado por respBD y crea uno nuevo dependiendo del estado de la conexión.