Objetivo general del taller

El objetivo general de este taller es practicar la implementación de programas usando programación orientada a objetos en Java, aplicando conceptos de herencia.

Objetivos específicos del taller

Durante el desarrollo de este taller se buscará el desarrollo de las siguientes habilidades:

1. Ser capaz de leer diagramas UML y construir las estructuras correspondientes en Java.
2. Ser capaz de leer el JavaDoc de un conjunto de clases y utilizarlo para implementar sus métodos y para invocar métodos de clases existentes.
3. Practicar la implementación de algoritmos en Java.
4. Practicar la implementación de la persistencia a través de archivos JSON.

Instrucciones generales

1. Descargue de Bloque Neón el archivo `Taller3-Aerolineas_esqueleto.zip` y descomprímalo en una carpeta dentro de su computador a la que pueda llegar con facilidad.
2. Lea con cuidado este documento: en este documento encontrará información importante para completar el programa mientras que va entendiendo lo que está haciendo.
3. Construya las clases que aparecen en los diagramas UML y en la documentación (javadoc) que se encuentra dentro de la carpeta docs.
4. Implemente los métodos de todas las clases de la aplicación.

El taller debe realizarse de forma individual.

Descripción del taller: Aerolínea

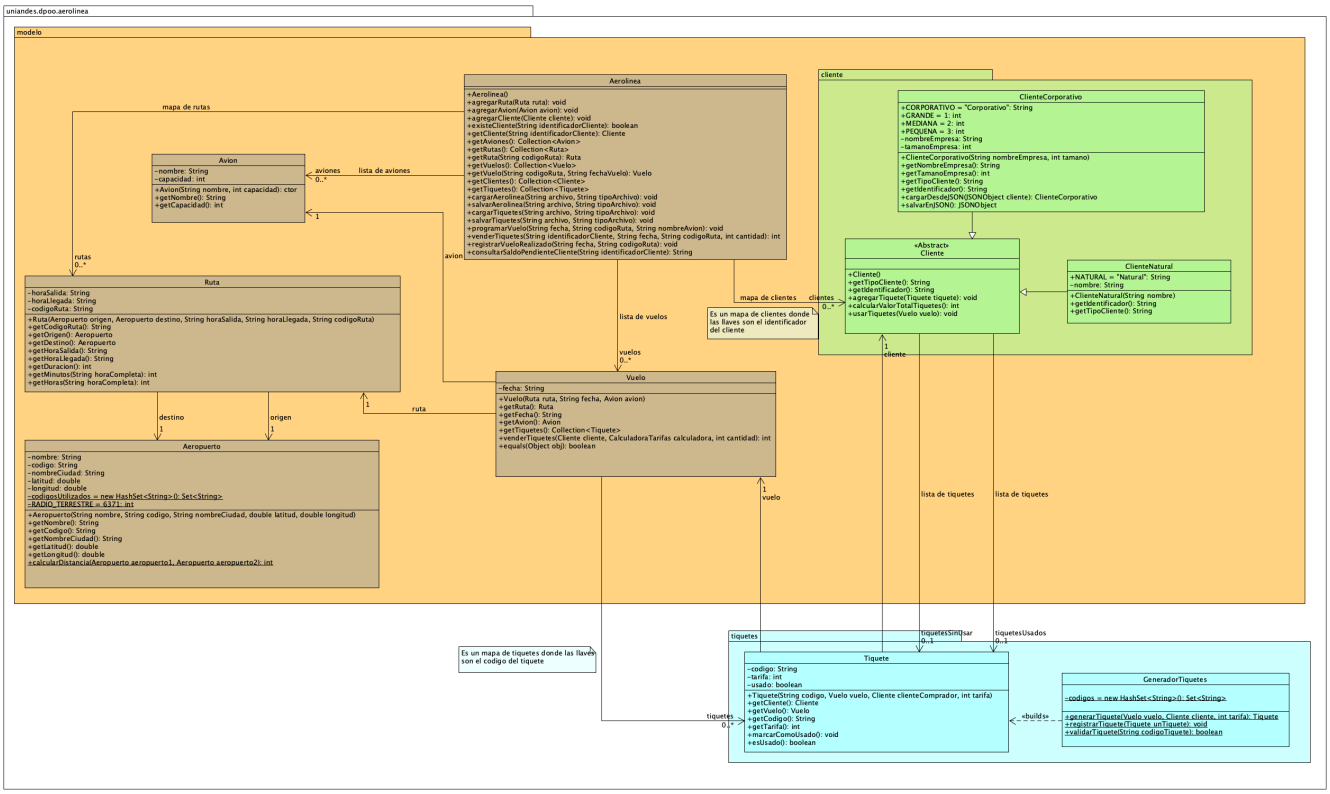
En este taller se va a construir un programa para manejar una parte de la información de una aerolínea. El taller incluye un esqueleto en el cual hay varias clases ya implementadas, pero al cual también le hacen falta varias clases. Las clases que ya están implementadas deberían servirle como guía para construir las que hacen falta.

Los elementos principales del caso de la aerolínea son los siguientes:

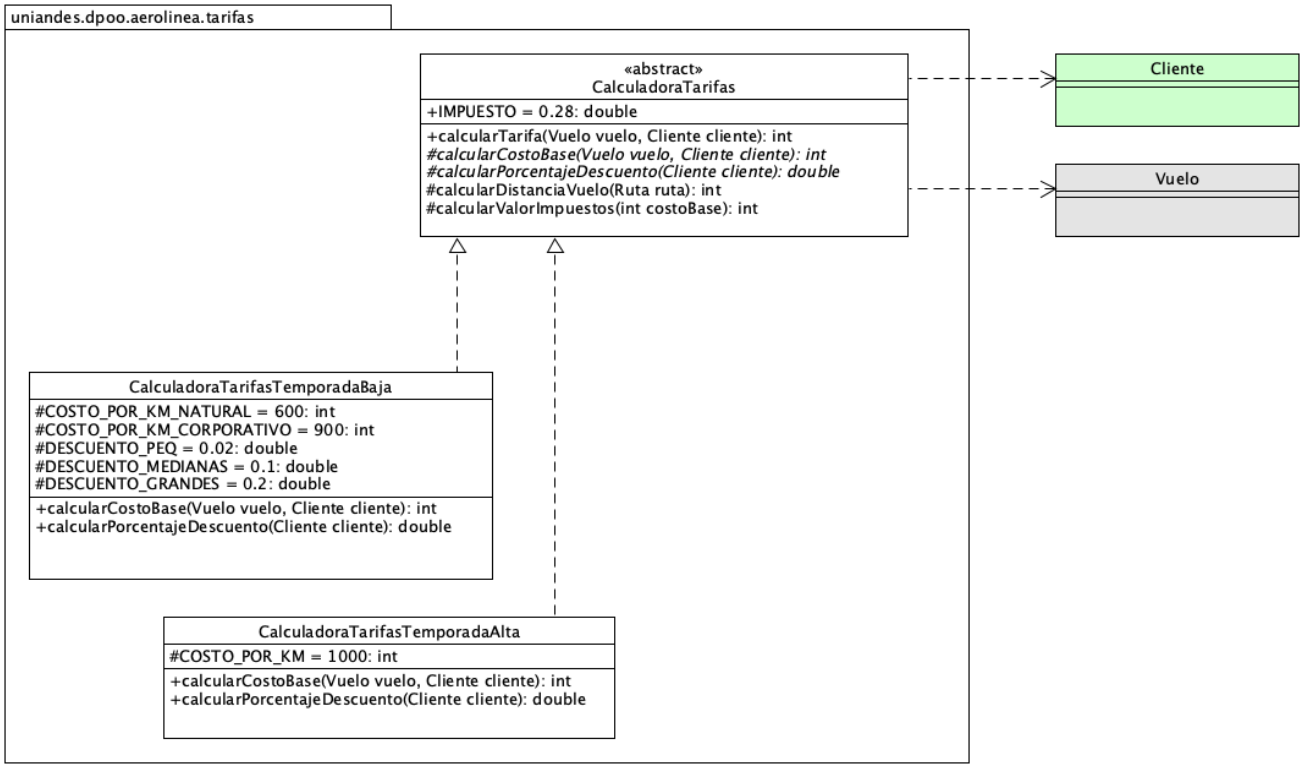
- Aeropuertos a los cuales puede llegar la aerolínea
- Rutas que la aerolínea tiene asignadas entre un aeropuerto origen y un aeropuerto destino, a una determinada hora del día.
- Vuelos que la aerolínea tiene programados para cumplir con una ruta en una cierta fecha.
- Aviones que son propiedad de la aerolínea y se asignan a los vuelos.
- Tiquetes que son comprados por los clientes para usarse en un determinado vuelo.

La aerolínea tiene dos tipos de clientes (Naturales y Corporativos), para los cuales se aplican diferentes tarifas en el momento de comprar un vuelo.

La siguiente imagen muestra las principales clases de la aplicación.

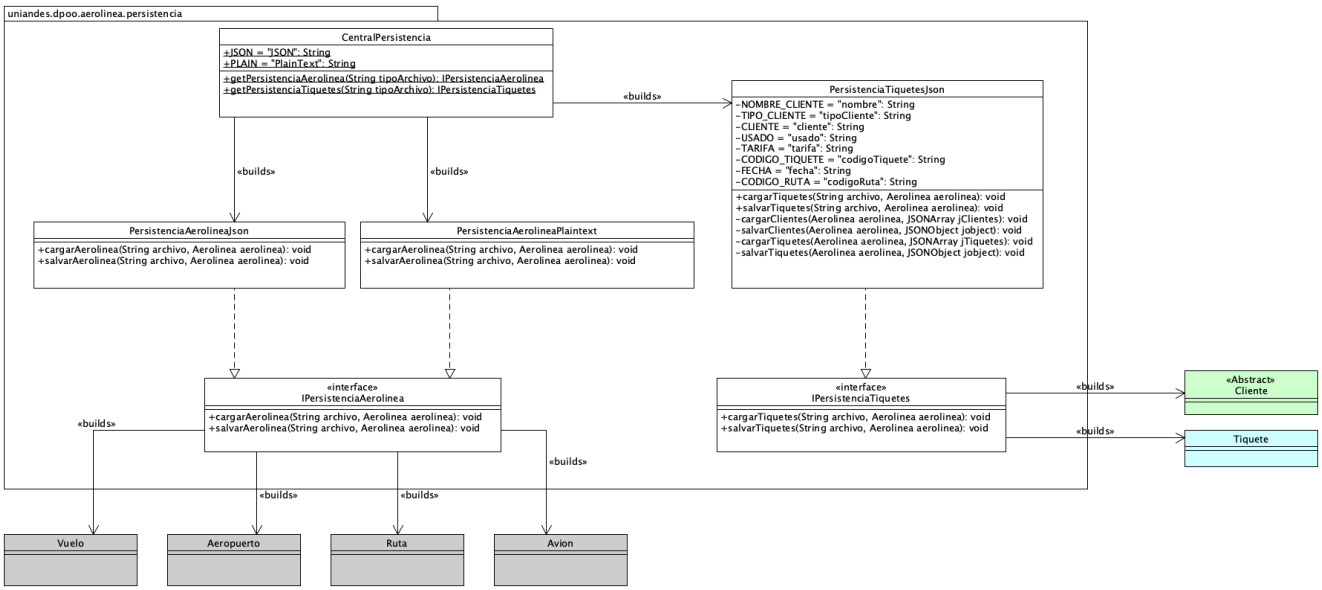


El diagrama anterior muestra la clase `CalculadoraTarifas`: esa es una clase abstracta encargada de calcular cuánto se le debe cobrar a un cliente particular para un vuelo particular. El proyecto incluye dos implementaciones concretas – una para temporada baja y otra para temporada alta - pero podría haber más si hubiera más reglas para calcular las tarifas.

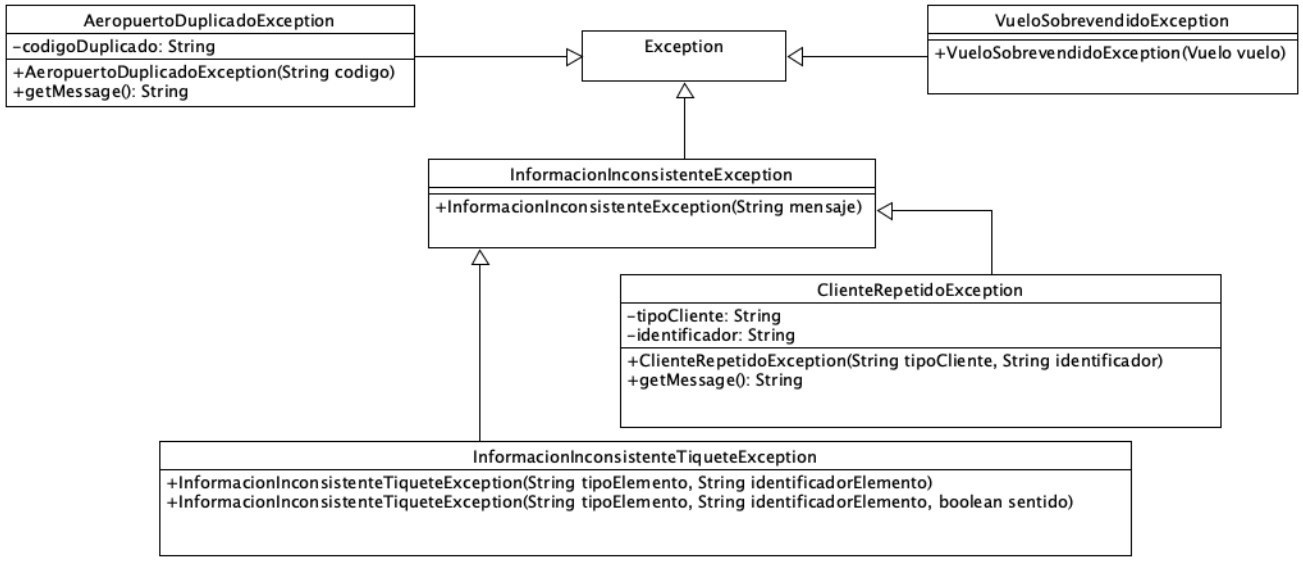


La persistencia de la aplicación está separada en dos partes. Por una parte, se encuentra la persistencia de los elementos más estables de la aerolínea (aviones, aeropuertos, rutas, vuelos) y por otra parte los elementos relacionados con los tiquetes y los clientes. Esta separación se materializa en dos interfaces: `IPersistenciaAerolinea` e `IPersistenciaTiquetes`.

Adicionalmente, la aplicación está preparada para realizar la persistencia sobre diferentes tipos de archivos. En este taller, sólo vamos a utilizar la persistencia sobre archivos **JSON** (la parte de los tiquetes ya está implementada). El siguiente diagrama muestra las clases relacionadas con la persistencia.



Finalmente, el proyecto incluye una jerarquía de excepciones que se utiliza para señalar de forma más o menos precisa cuales son los problemas que se presentan durante la ejecución.



Javadoc / API

Dentro de la carpeta `docs` del esqueleto encontrará una serie de archivos HTML que describen cada una de las clases de la aplicación, incluyendo sus métodos y atributos. Esto es similar al API de Java que seguramente usted ha consultado para saber cómo utilizar clases del JDK.

Estos archivos HTML fueron generados a partir de la documentación de las clases que se realizó usando el formato estandarizado Javadoc (son los comentarios que inician por `/**  … */` antes de cualquier elemento de Java).

Actividades

1. Después de leer este documento, utilice los diagramas UML y la información del Javadoc para construir el esqueleto completo de la aplicación.
2. Implemente todos los métodos de la aplicación. Dentro del esqueleto, algunos métodos están marcado con el comentario **TODO** para indicar que ahí se debe completar algo aún.
3. Puede usar la clase `ConsolaAplicacion` para implementar pruebas que le sirvan para verificar el funcionamiento de la aplicación.

Entrega

1. Cree un repositorio privado donde quedarán sus talleres y deje ahí todos los archivos relacionados con sus entregas.

2. Entregue a través de Bloque Neón el URL para el repositorio, en la actividad designada como “**Taller 3**”.