UNIVERSIDADE DA CORUÑA

Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.

GESTIÓN DE BILLETES

Principios:

Los principios de diseño que se han utilizado para la realización del primer ejercicio de la práctica son:

- **Principio de responsabilidad única:**

  Este principio nos indica que cada objeto debe de tener una responsabilidad única que está encapsulada en la clase. Por ejemplo, podemos observar su uso en las clases "Or" y "And" , ya que su única finalidad es filtrar la lista dependiendo del criterio que se utilice.

- **Principio abierto-cerrado:**

  Este principio defiende que las entidades software deberían ser abiertas para permitir su extensión, pero cerradas frente a la modificación. En nuestro programa se puede observar este principio, ya que, en caso de querer añadir otro filtro de billetes, solamente tendríamos que añadir una subclase al sistema (sin modificar el código preexistente).

- **Principio de la inversión de la dependencia:**

  Este principio se basa en depender de interfaces o clases abstractas en vez de depender de clases y funciones concretas. La utilización de este principio se puede observar en nuestro programa, ya que se depende de la interfaz "ORyAND" para escoger el filtro de billetes que se requiera. Además, también podemos observarlo con el interfaz "Comparaciones", que nos indica cómo filtrar dependiendo del campo del billete que se indique (origen, destino, fecha y precio).

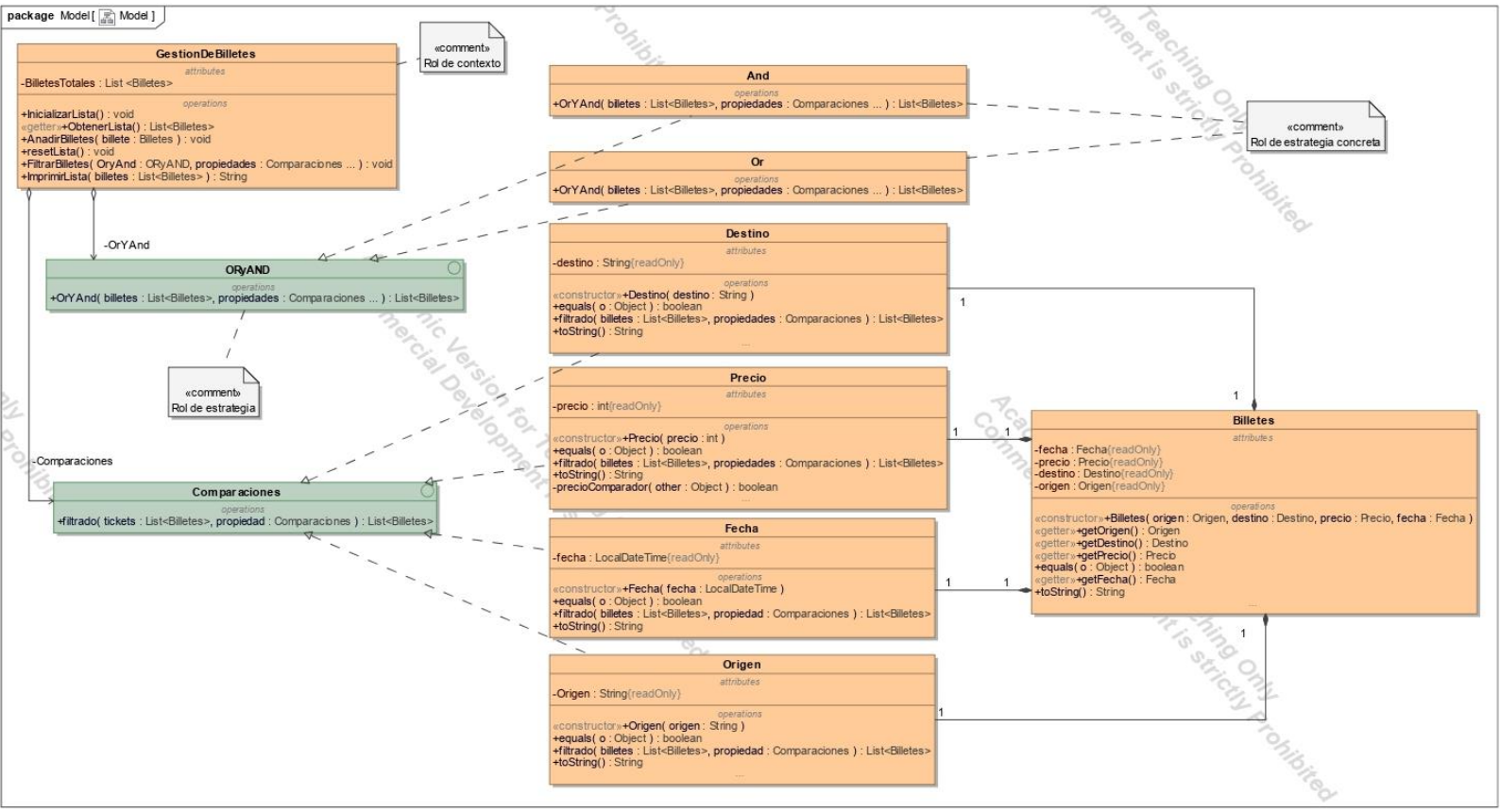
Patrones:

Durante este ejercicio, hemos elegido utilizar un **patrón inmutable** y un **patrón estrategia** para plantear la solución al problema planteado. El patrón inmutable se encarga de diseñar clases en las cuales toda la información contenida en cada instancia es proporcionada en el momento de la creación y no puede modificarse durante el tiempo de vida del objeto. Este patrón puede ser fácilmente reconocido en la clase "Billetes", ya que no incluye métodos que modifiquen su estado, no puede ser extendida, todos sus atributos son finales y privados y ninguno de sus atributos son mutables. Se ha elegido este patrón ya que los billetes de autobús no pueden ser modificados una vez se hayan creado.

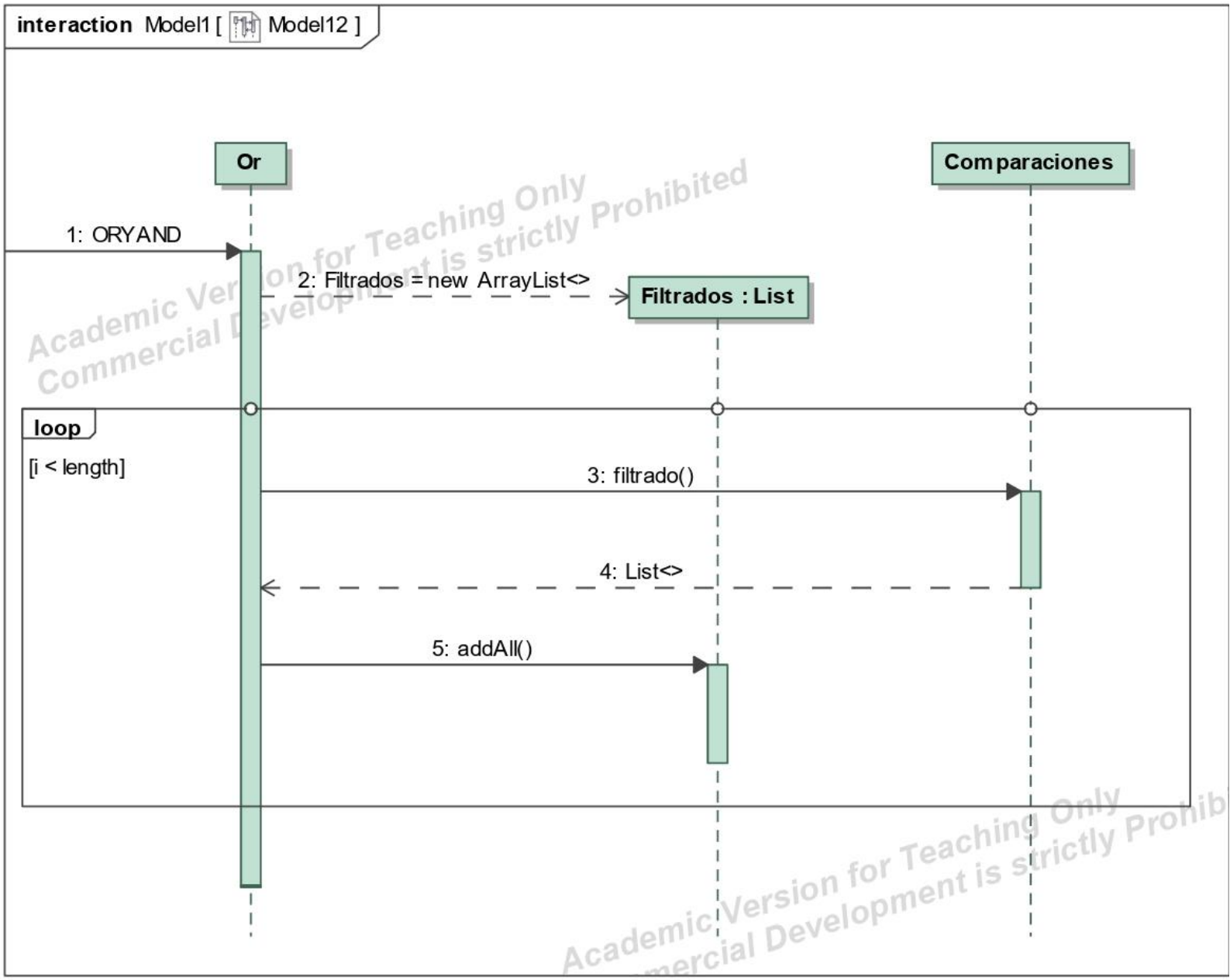
Además, como mencionado anteriormente, utilizamos un patrón estrategia. Este patrón se basa en una familia de algoritmos que se encapsulan y de esta forma se pueden hacer intercambiables mediante una interfaz. En nuestro caso hemos diseñado

dos algoritmos de filtrado de billetes (Clases “Or” y “And”) que son elegidos a través de una interfaz. El patrón estrategia está formado por una estrategia, que sería la interfaz común a los algoritmos soportados, un contexto, que delega en el interfaz el cálculo del algoritmo, y estrategias concretas, que implementan el algoritmo utilizando el interfaz definido. En nuestro caso, la clase “GestionDeBilletes” juega el rol de contexto, la clase “ORyAND” juega el rol de estrategia y las clases “Or” y “And” juegan el rol de estrategias concretas. Se eligió este patrón porque, en caso de querer añadir un nuevo filtro, este será muy fácil de añadir, ya que solamente tendremos que añadir una nueva subclase al sistema.

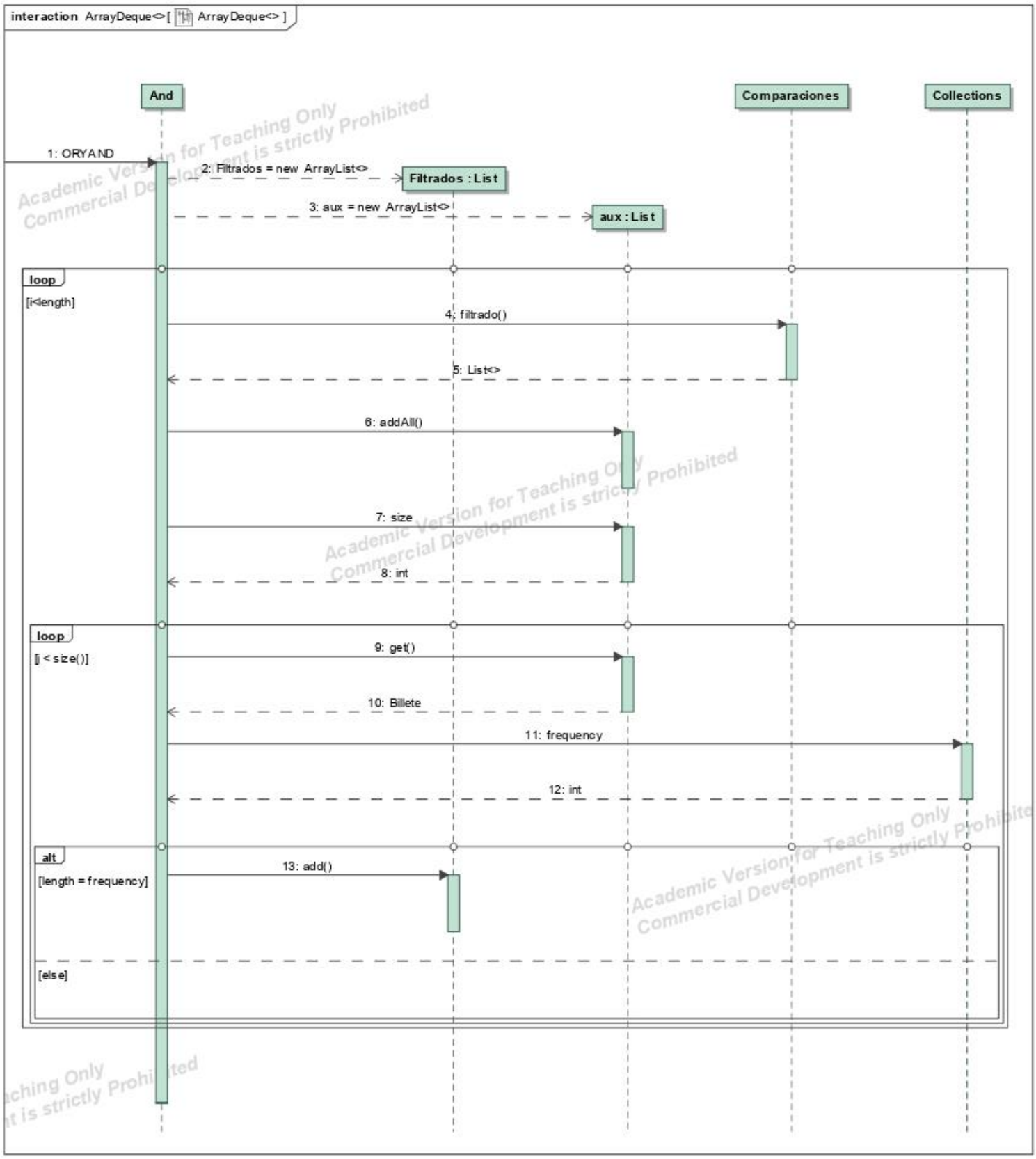
A continuación, dejamos los diagramas necesarios para la comprensión del código:



Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.



Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.



Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.

PLANIFICADOR DE TAREAS

Principios:

Los principios de diseño que se han utilizado para la realización de esta segunda parte de la práctica son:

- **Principio de responsabilidad única:**
  Este principio nos indica que cada objeto debe de tener una responsabilidad única que está encapsulada en la clase. Podemos observar su uso en las clases que sirven como algoritmos de ordenación (DependenciaDebil, DependenciaFuerte y OrdenJerárquico), ya que su única finalidad es ordenar el grafo que se le está asignando.

- **Principio de la inversión de la dependencia:**
  Este principio se basa en depender de interfaces o clases abstractas en vez de depender de clases y funciones concretas. La utilización de este principio se puede observar en nuestro programa, ya que se depende de la interfaz "Dependencias" para escoger el algoritmo de ordenación que se requiera.

- **Principio abierto-cerrado:**
  Este principio defiende que las entidades software deberían ser abiertas para permitir su extensión, pero cerradas frente a la modificación. En nuestro programa se puede observar este principio, ya que es muy sencillo añadir un nuevo tipo de ordenación, ya que solamente es necesario añadir una subclase al sistema (sin modificar el código preexistente).

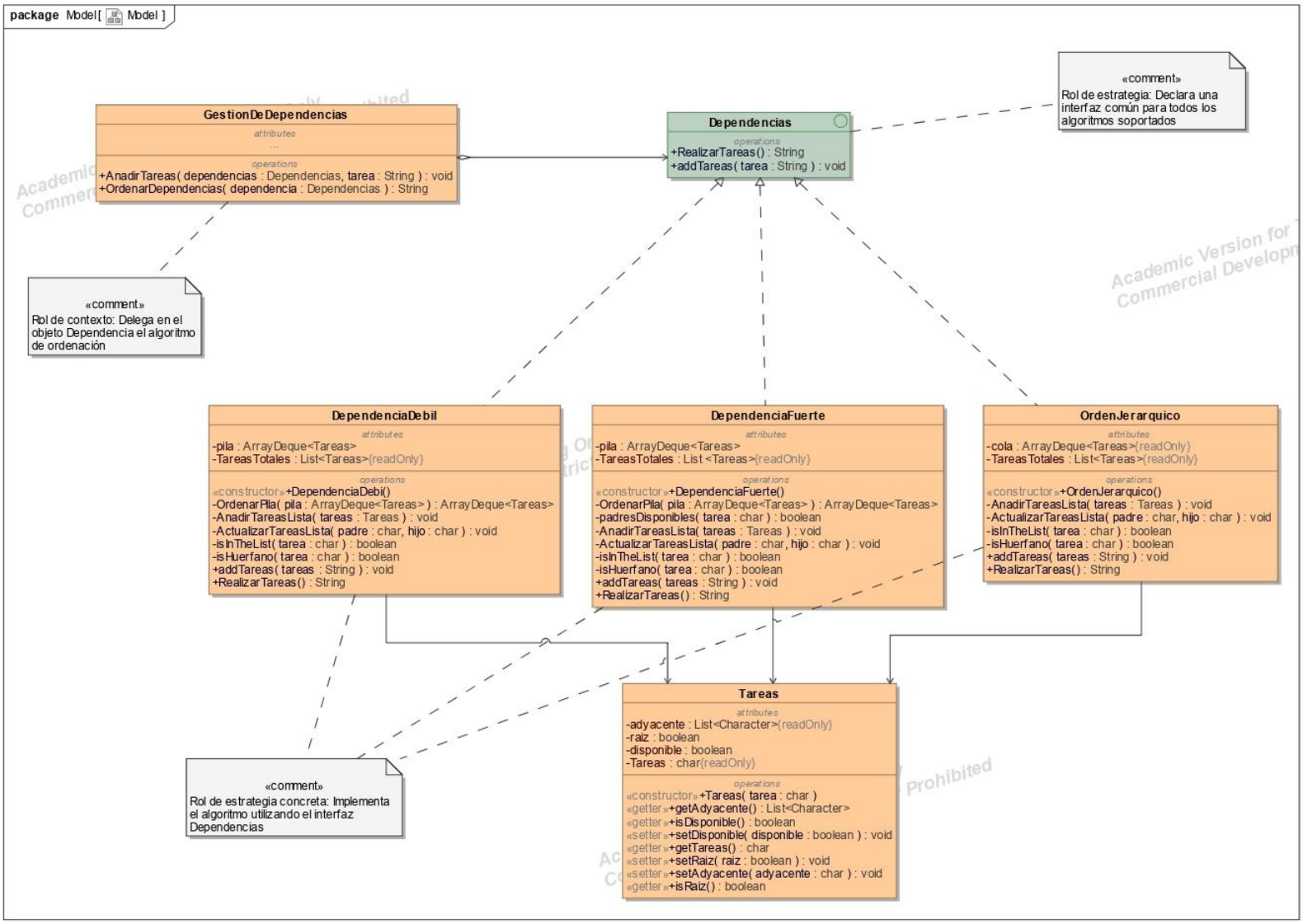
Patrón:

En este segundo ejercicio de la práctica de diseño hemos elegido utilizar un **patrón estrategia** para plantear la solución. Este patrón se basa en una familia de algoritmos que se encapsulan y de esta forma se pueden hacer intercambiables mediante una interfaz. Además, nos permite tener mucha facilidad a la hora de integrar nuevas formas de ordenación de tareas en un futuro. En nuestro caso, hemos diseñado tres algoritmos de ordenación de tareas (dependencia fuerte, dependencia débil y orden jerárquico) que son elegidos a través de una interfaz.

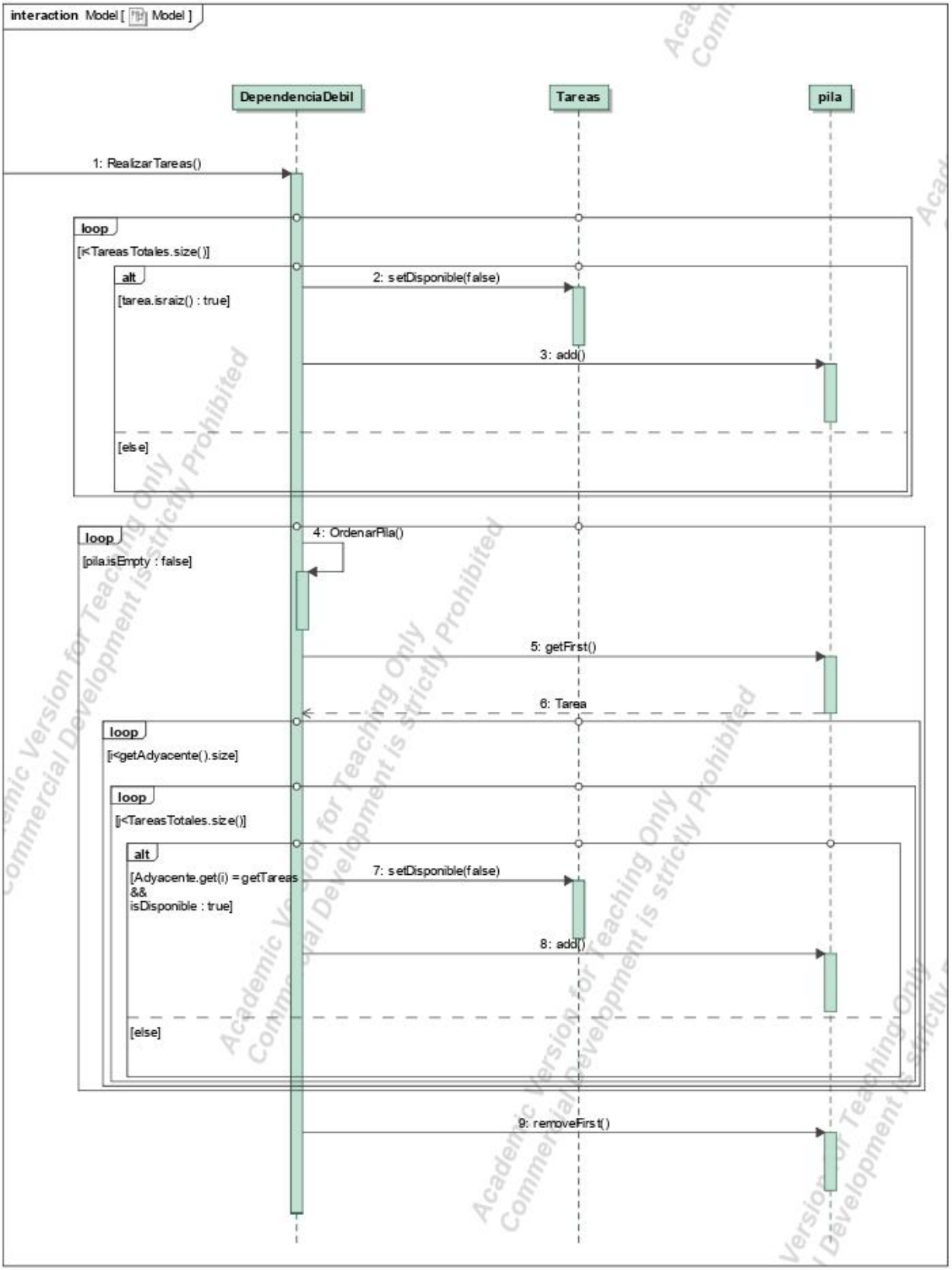
El patrón estrategia está formado por una estrategia, que sería la interfaz común a los algoritmos soportados, un contexto, que delega en el interfaz el cálculo del algoritmo, y estrategias concretas, que implementan el algoritmo utilizando el interfaz definido. En nuestro caso, la clase "GestionDeDependencias" juega el rol de contexto, la interfaz "Dependencias" juega el rol de estrategia y las clases "DependenciaDebil", "DependenciaFuerte" y "OrdenJerárquico" juegan el rol de estrategias concretas.

A continuación, dejamos los diagramas necesarios para la comprensión del código:

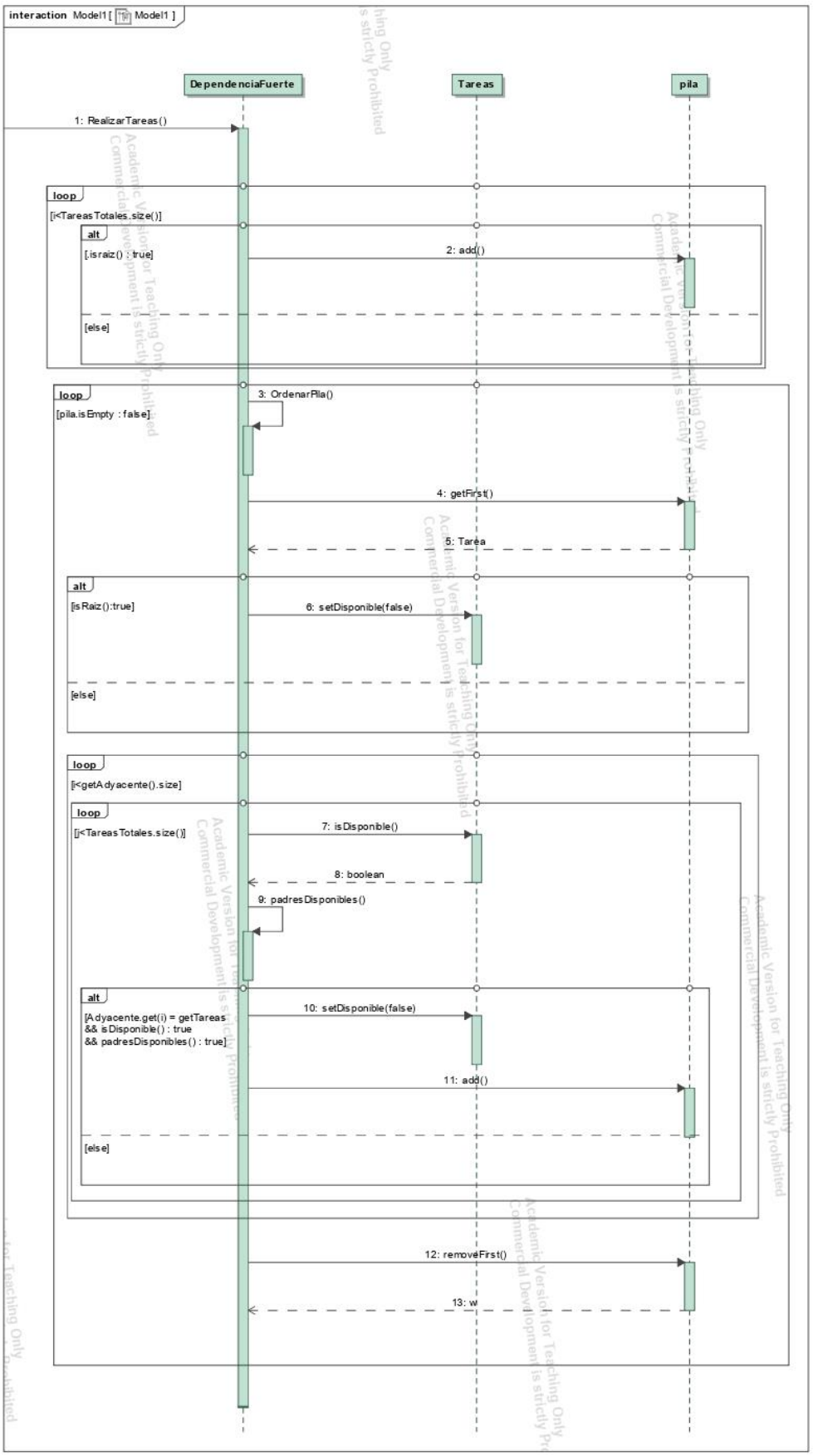
Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.



Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.



Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.



Raúl Fernández del Blanco
Brais González Piñeiro
2ºGEI
Grupo: 2.4.