Informe de “Planificador de Tareas”

Grupo 24-08

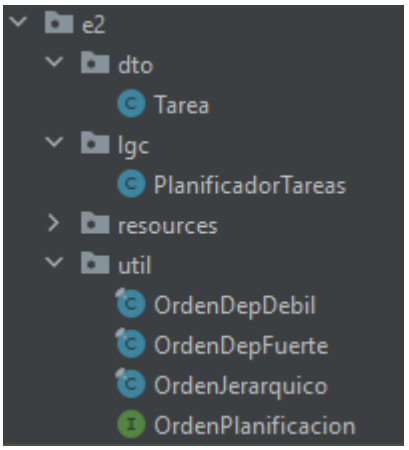
Bosco Suárez-LLanos Outeiriño

Samanta K. Machuca Montalvo

1. Principios de diseño

- Principio de Responsabilidad Unica

  En este ejercicio vemos que tiene dicha responsabilidad porque en cada package hay clases que se centran en un solo objetivo por ende tiene alta cohesión.



- Principio de Sustitución de Liskov

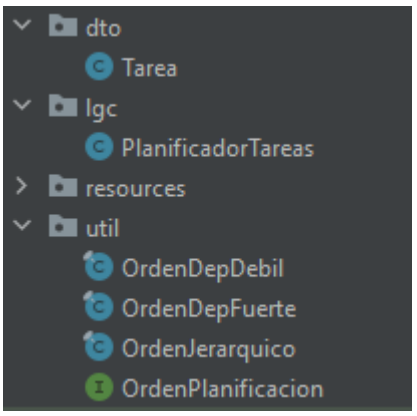
```
public String ordenaSegun(OrdenPlanificacion metodo)
```

  En la clase PlanificadorTareas existe el principio de sustitución de Liskov porque cumple el principio de subcontratación. Recibe por parámetro un método de planificación, la cual al ser una interfaz acepta que le pasen todas sus distintas implementaciones o una nueva.

- Principio de Inversión de la Dependencia

```
private final List<Tarea> cabeceras = new ArrayList<>();
```

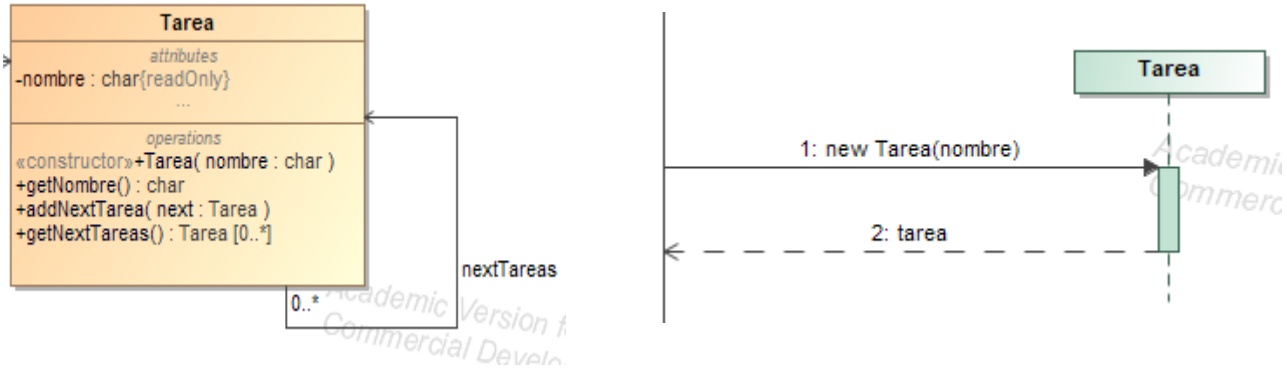
  En todo el ejercicio se hace uso de este principio en la definición de los List, así como el uso de los órdenes de planificación.



2. Patrones de diseño

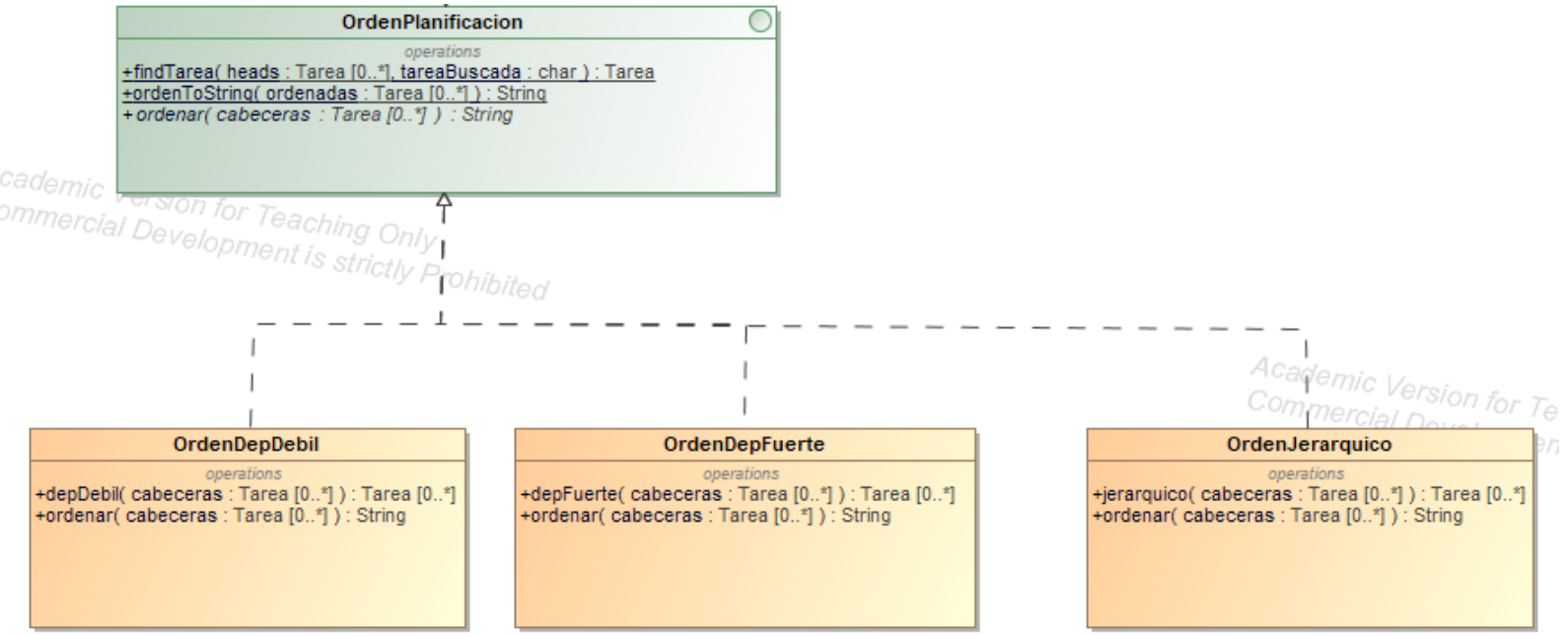
- **Principio “Favorece la inmutabilidad” (Patron Inmutable)**

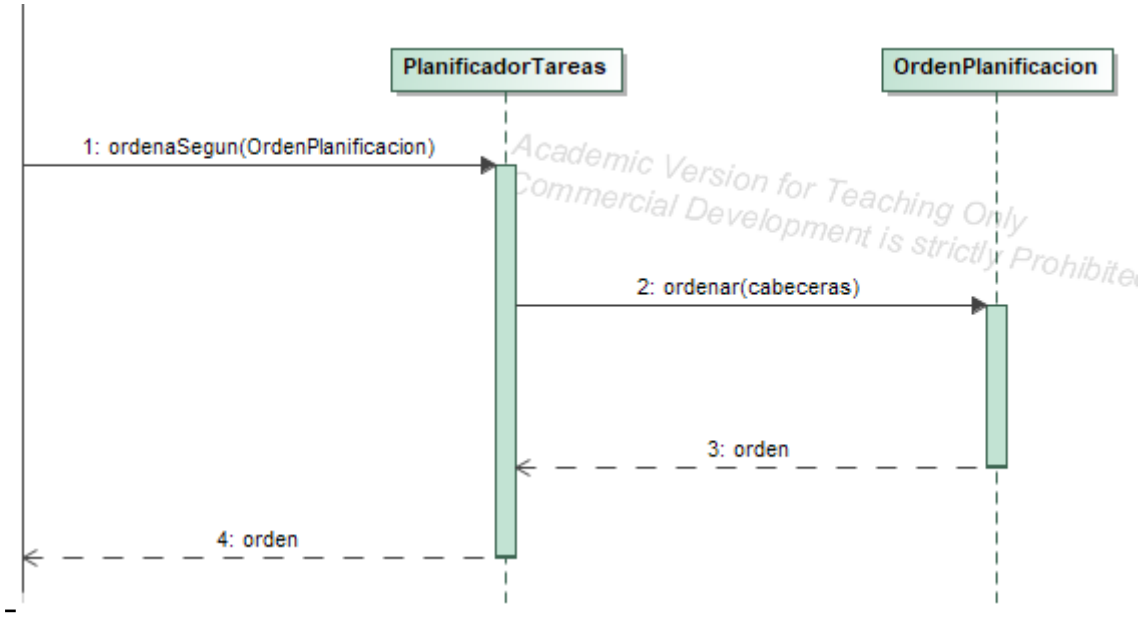
Utilizamos el **patrón inmutable** en la clase Tarea declarando los atributos privados y finales.



- **Principio “Encapsula lo que varía” (Patrón Estrategia)**

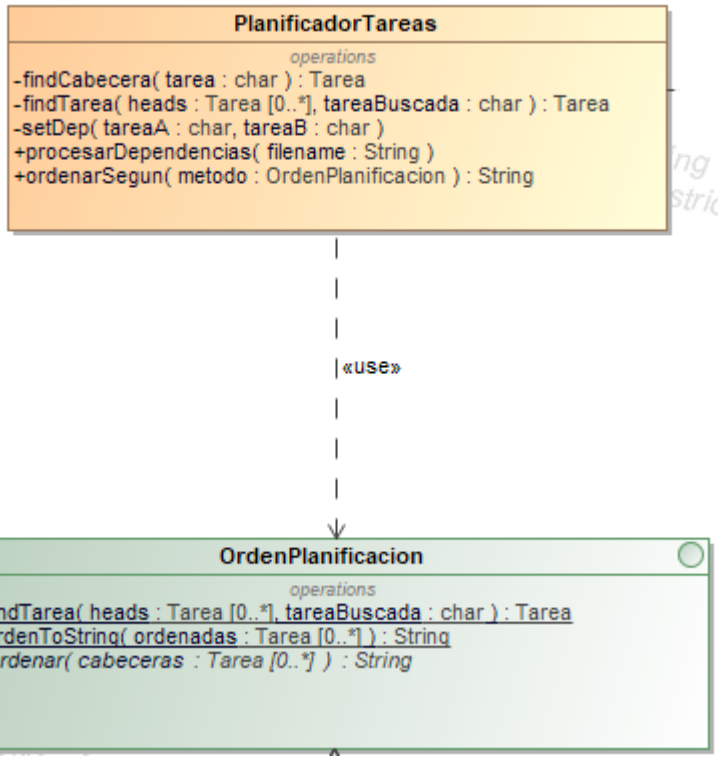
En el package util usamos el **patrón estrategia** declarando la interfaz OrdenPlanificacion que sea común para los tipos de clase de orden que hay en el ejercicio, ya que luego lo implementa según las condiciones del entorno que decide cual es el mejor orden de ejecución de las mismas.





- **Principio de "Bajo Acoplamiento"**

La clase PlanificadorTareas trabaja sobre objetos abstractos y no concretos, como la interfaz OrdenPlanificacion. Tiene el mismo diagrama de secuencia que el anterior.



- **Principio “Tell, Don’t Ask”**

Definimos el método equals en la clase Tarea para que se encargue de hacer las comparaciones, en este ejercicio dos tareas son iguales si tienen el mismo nombre.

- **Principios DRY , KISS y YAGNI**

El principio DRY lo usamos por ejemplo en la interfaz OrdenPlanificacion al definir los métodos findTarea y ordenToString los cuales sirven para no repetir código.

El principio KISS se cumple en todo el programa manteniéndolo simple y funcional sin ninguna complejidad innecesaria actualmente.

El YAGNI en todas las clases dentro de este ejercicio cumplen un propósito y son usadas, no se ha realizado sobreingeniería, de modo que no hay clases que no se necesitan.

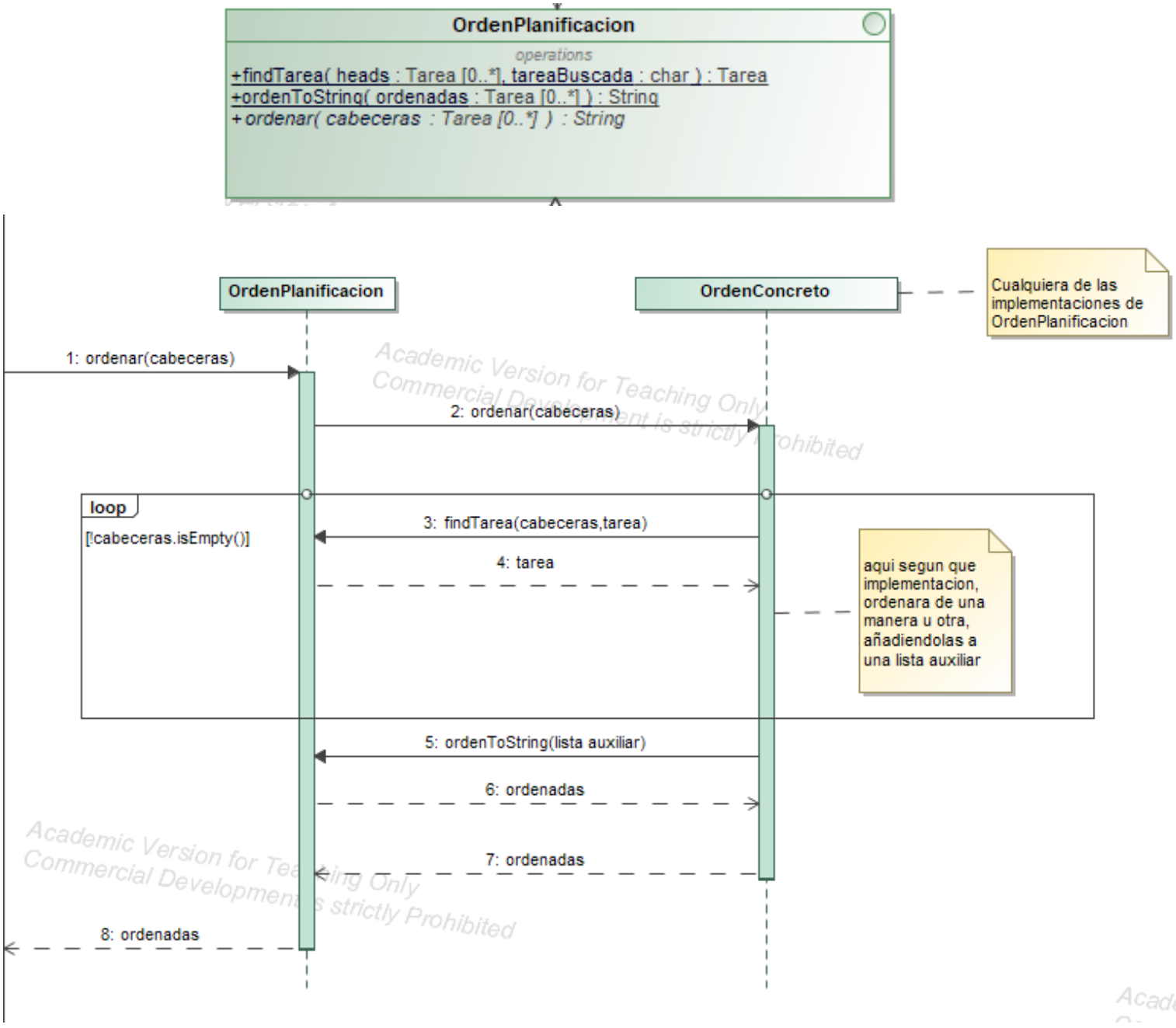


DIAGRAMA DE CLASES COMPLETO