Guillermo Fernández Sánchez
Javier Fernández Rozas

1. Ticket Management. Report

Design principles:

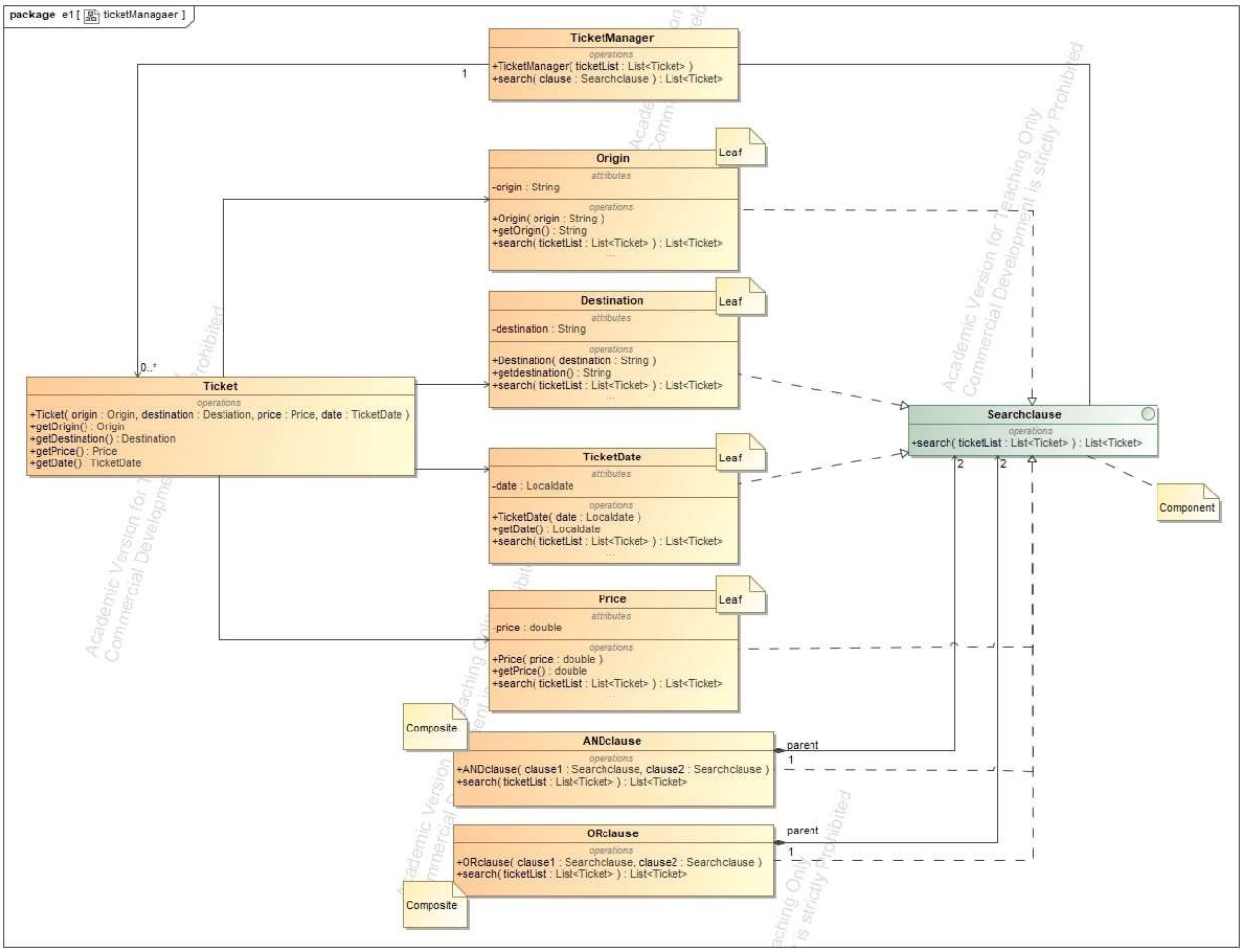
- **The Open-Closed Principle (OCP):**
  We have used the open-closed principle on the *TicketManager* class and the *Searchclause* interface since we can add more search criterias whithout knowing the source code of the classes.
- **Favor Immutability principle:**
  We used this principle on the classes *Ticket*, *Origin*, *Destination*, *Price* and *TicketDate*. This means that all this classes are immutable since there is no reason to make them mutable (no *set* methods, final and private classes, all their atributes are private and there is no access to mutable components).

Desing patterns:

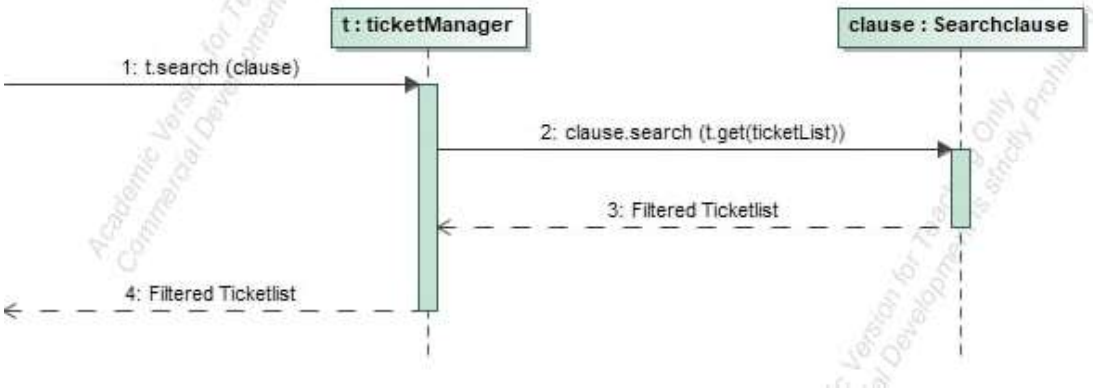
We've mainly used the **Composite pattern** with the aim of treating simple search criterias and AND/OR clauses uniformly. It also allows to introduce new search criterias, as it was mentioned above.

The interface *Searchclause* plays the role of Component, it declares the interface for objects and search() is the operation that is redirected to the subclasses.
The classes *Origin*, *Desination*, *Price* and *TicketDate* make up the leafs of the composition (this are the simple search cirterias).
The classes ANDclause and ORclause play the role of Composite which redirect the operations to their children.

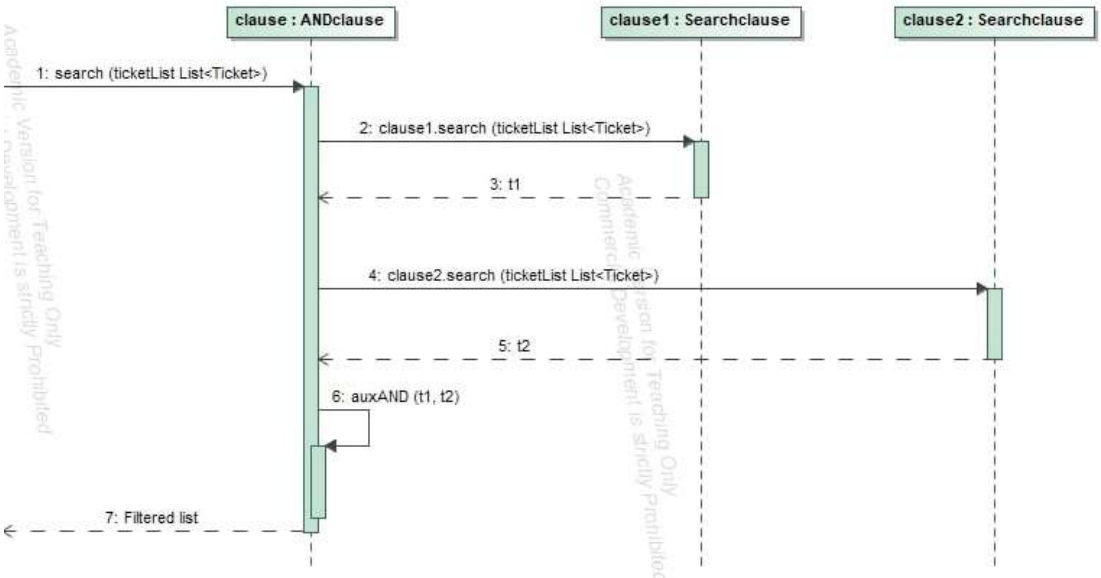
The roles of these classes and the connections between them are shown in the following class diagram:



The *TicketManager's search()* function calls the *Searchclause's search()* and, by polimorfism, the desired search is executed, as we can see in the following sequence diagram:



In the sequence diagram below, we can see the behaviour of the Searchclause in the case that it is an instance of the ANDclause composite:



In the case that we search by a simple criteria, the sequence diagram would look like this: