Composite

Objective

Forming hierarchical structures so that individual components can be treated as well as groups of components The operations Typical components include add, delete, display, find and group.

Function

Handle composite objects as if they were simple.

Structure

The Component interface describes operations that are common to simple and complex elements. The Client works with all the elements to through the component's interface. As a result, the customer can work in the same way with simple or complex elements.

The structure that meets this pattern is shown in Figure 1

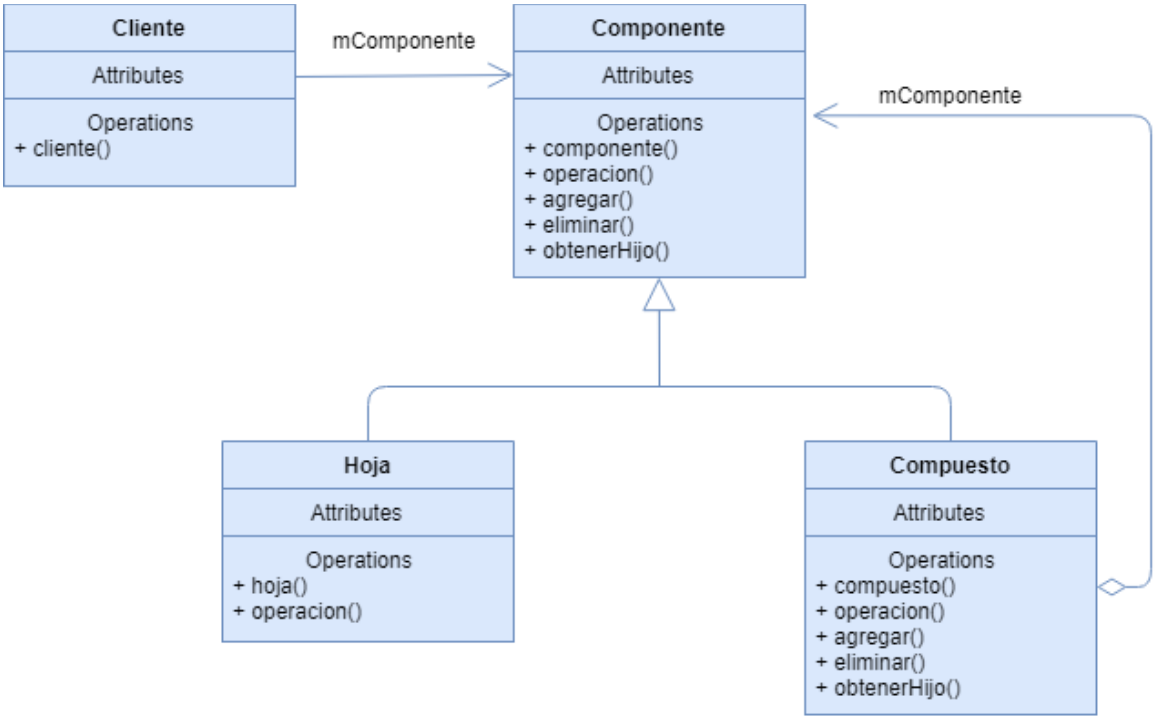

Figure 1: UML Diagram Composite Pattern

Applications

The use of the Composite pattern is recommended in the following cases:

- You want the client classes to handle all objects in a hierarchical structure, not knowing whether they are managing a single object or a composite one.

- You want to represent entire object hierarchies or just a part of them of them.

Design Patterns Collaborators

- A composite is generally ideal for Chain of Responsibility responsibilities.

- A decorator pattern usually uses a composite.

- The iterator pattern can be used to go through the composites.

Scope of action

Applied at the object level.

Problem

To create objects, individuals or a grouping, similar to a tree structure with compound or single nodes; lists should be used The difficulty in these cases, however, is knowing how to recognize a composite object or a simple one, in order to determine which operations to apply in each case.

Solution

The composite design pattern allows individual objects to be referenced or sheet, using the same "Component" interface that directly executes the operation in case of a sheet object, while in case of a composite object it replicates all operations to each of the component or sheet objects.

Diagram or Implementation

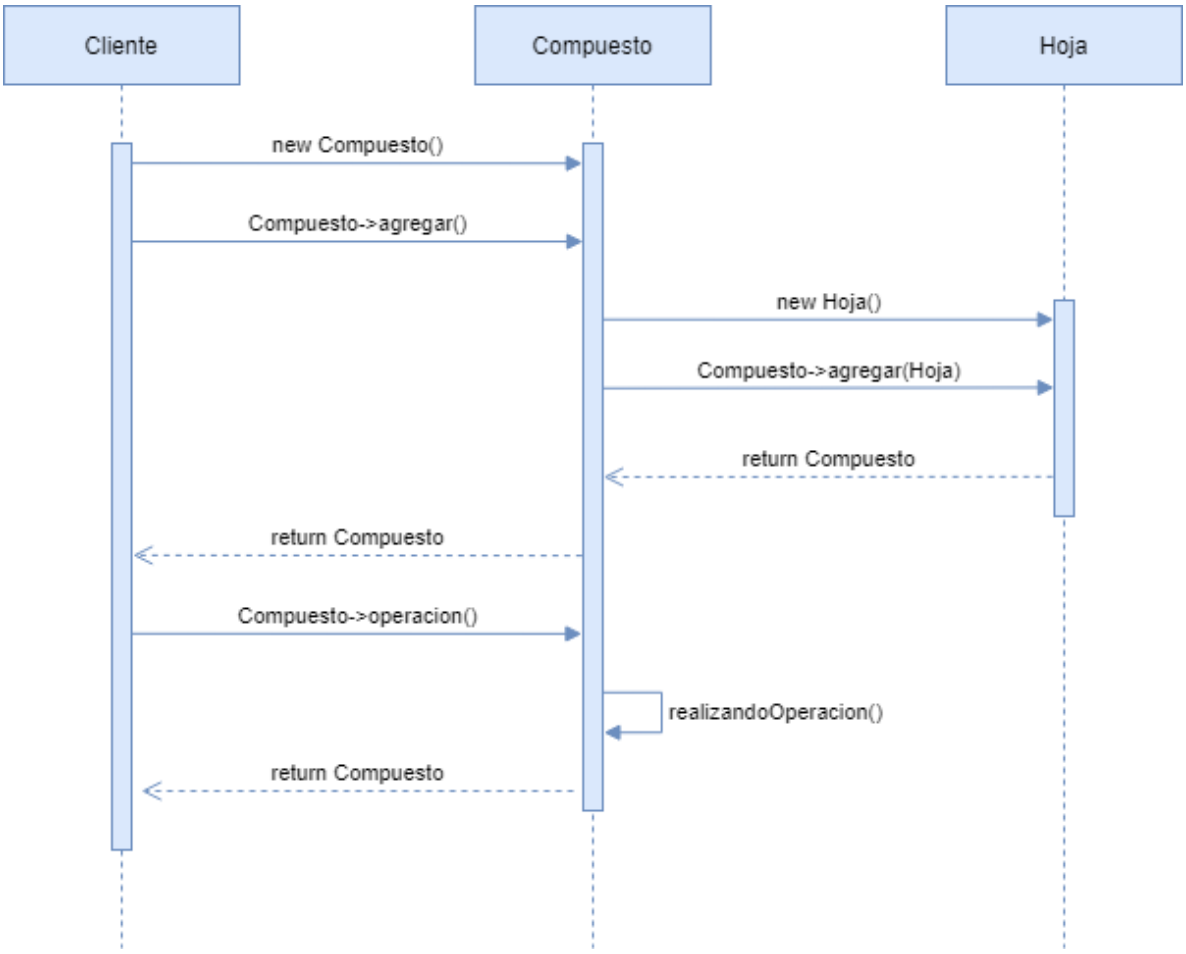

Figure 2: UML Diagram Composite Pattern

Figure 2 explains the behaviour of the Composite pattern by means of a sequence diagram.

- The client class perform an action on the CompositeA class.
- CompositeA class in turn performs an action on CompositeB class.
- Class CompositeB performs an action on class LeafA and class LeafB and the result is returned to class CompositeA.
- The CompositeA class spreads the action on LeafC, which returns a result.

- The CompositeA class obtains a final result after the evaluation of all the structure and the client class gets a result.

Study Cases

Drawing Editor System

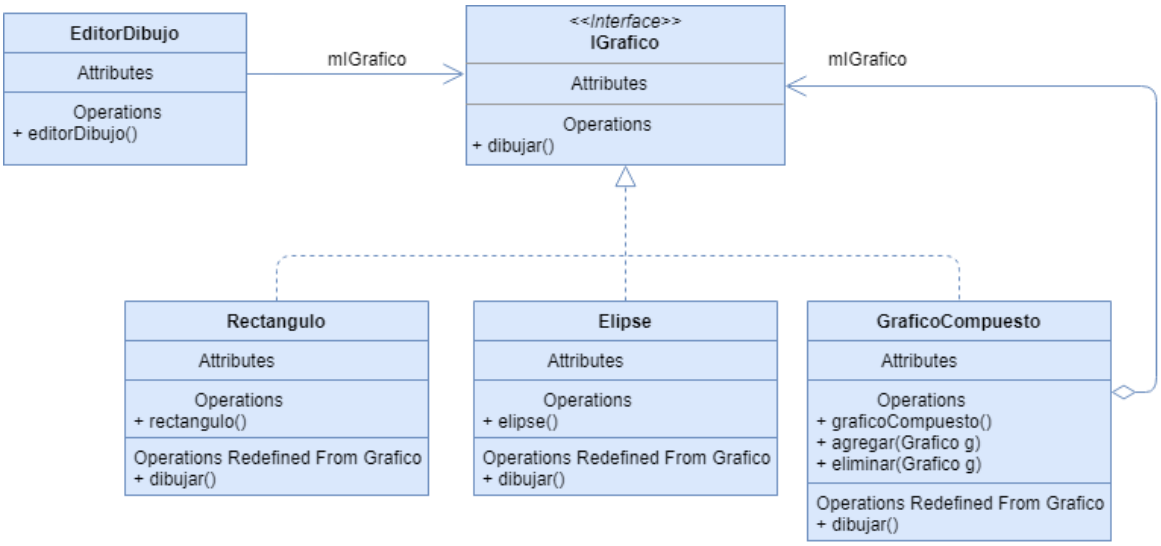

Figure 3: UML Diagram Drawing Editor System

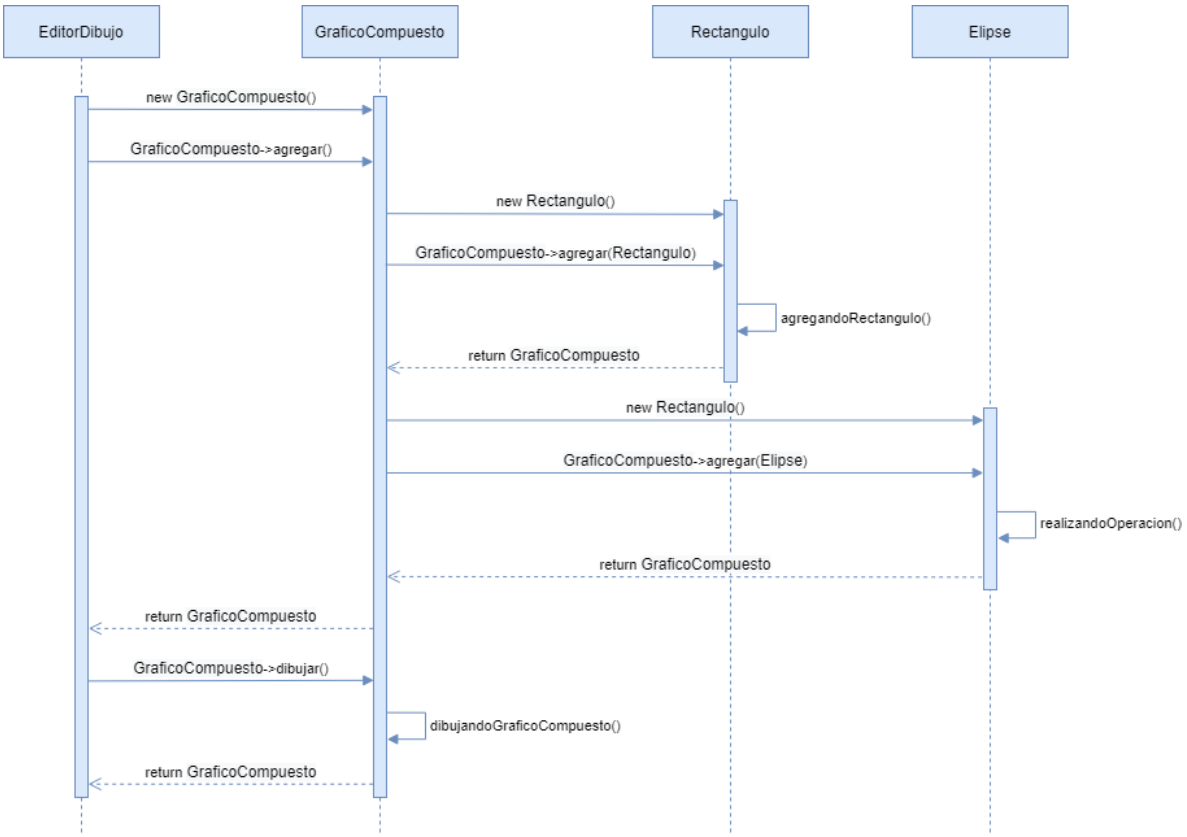

Figure 4: UML Diagram Drawing Editor System

Tourist Reservation System

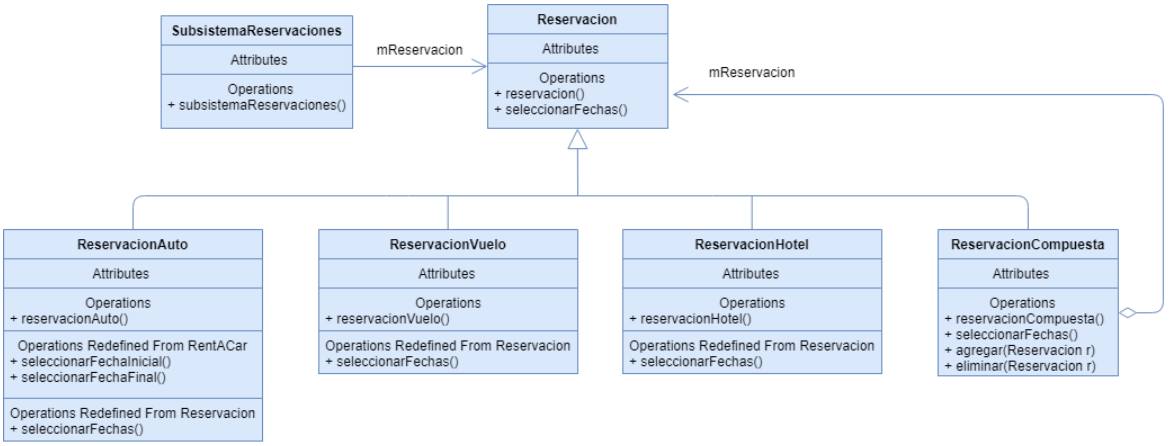

Figure 5: UML Diagram Tourist Reservation System

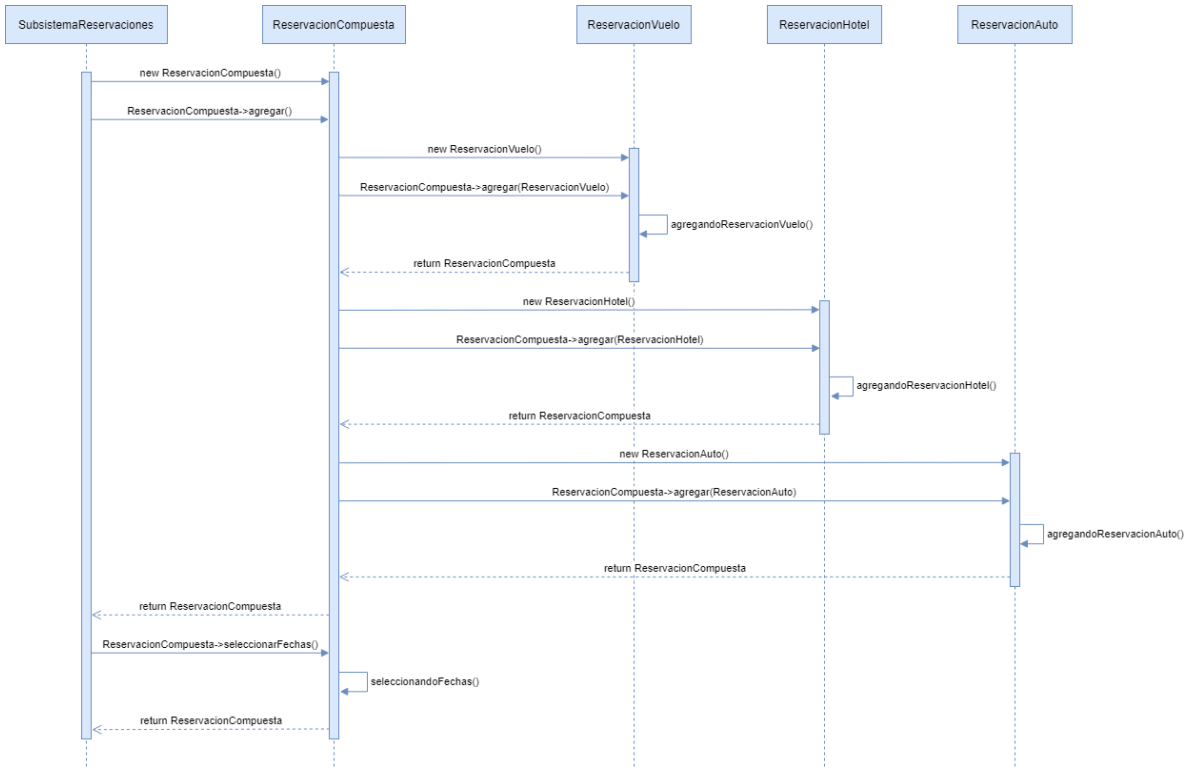

Figure 6: UML Diagram Tourist Reservation System