Singleton

Objective

Ensure the creation of a single instance for a given class.

Function

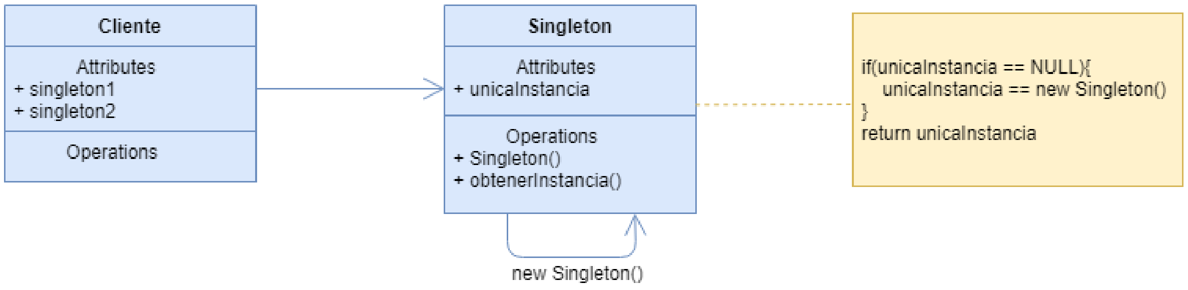
Ensure the existence of a single instance for a class and the creation of a mechanism for global access to it.

Structure

As shown in figure 1

The Singleton class declares the static method obtainInstance that returns the same instance of its own class. The Singleton constructor must be hidden from the customer's code. Calling the getInstance method should be the only way to get the Singleton object.

The structure that meets this pattern is shown in Figure 1



```
if(unicaInstancia == NULL){
   unicaInstancia == new Singleton()
}
return unicaInstancia
```

Figure 1: UML Diagram Singleton Pattern

Applications

- The system requires exactly one instance of a class, which must be accessible to customers from a well-defined access point.

- The single instance must be extended to subclasses and clients must be able to use it without modifying their code.

Design Patterns Collaborators

- A Singleton pattern is often related to and used in the implementation of the Abstract Factory pattern, especially when it concerns a specific factory.

Scope of action

Applied at the object level.

Problem

Ensuring the creation of a single instance requires high levels of validation and a global variable; which is difficult to control when there are multiple instances of objects.

Solution

The Singleton pattern creates a class that instances the only object that will be responsible for the creation, initialization and access; this instance must be a private, since the instance creation operation must be hidden. In addition, the requires a static public function that takes care of the encapsulation of the initialization and provides a global access point. In the implementation process The employer must ensure that each class is responsible for monitoring that no allow more than one instance.

Diagram or Implementation

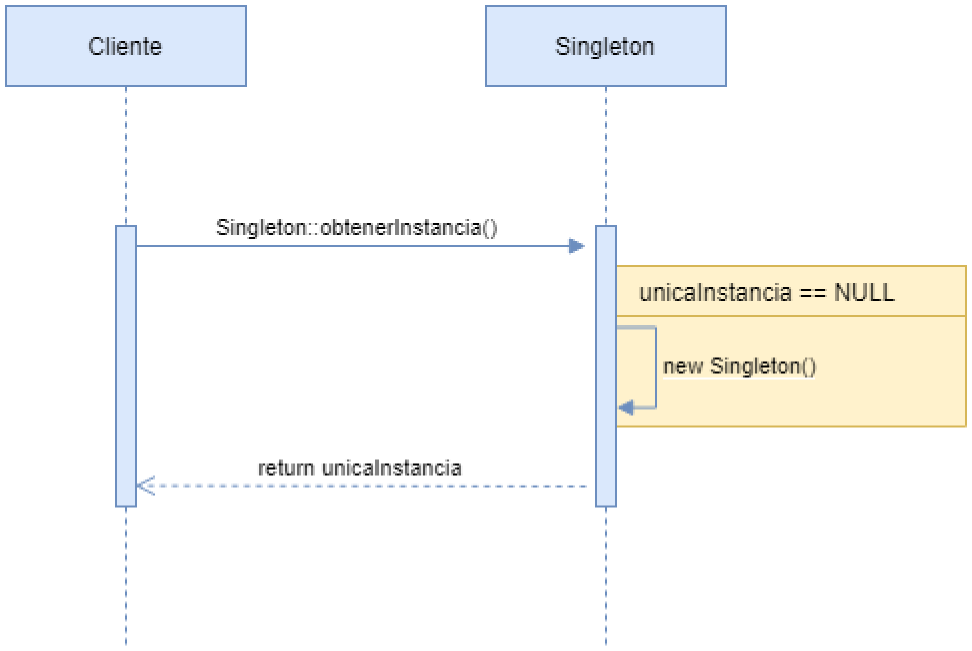


Figure 2: UML Diagram Singleton Pattern

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

- Client class requests the instance from the Singleton class by means of the static method obtainInstance().

- The Singleton class will validate if the instance was already created before, if not then a new one is created.

- The instance created in the previous step is returned or the existing instance is returned in another case.

Study Cases

Printer System

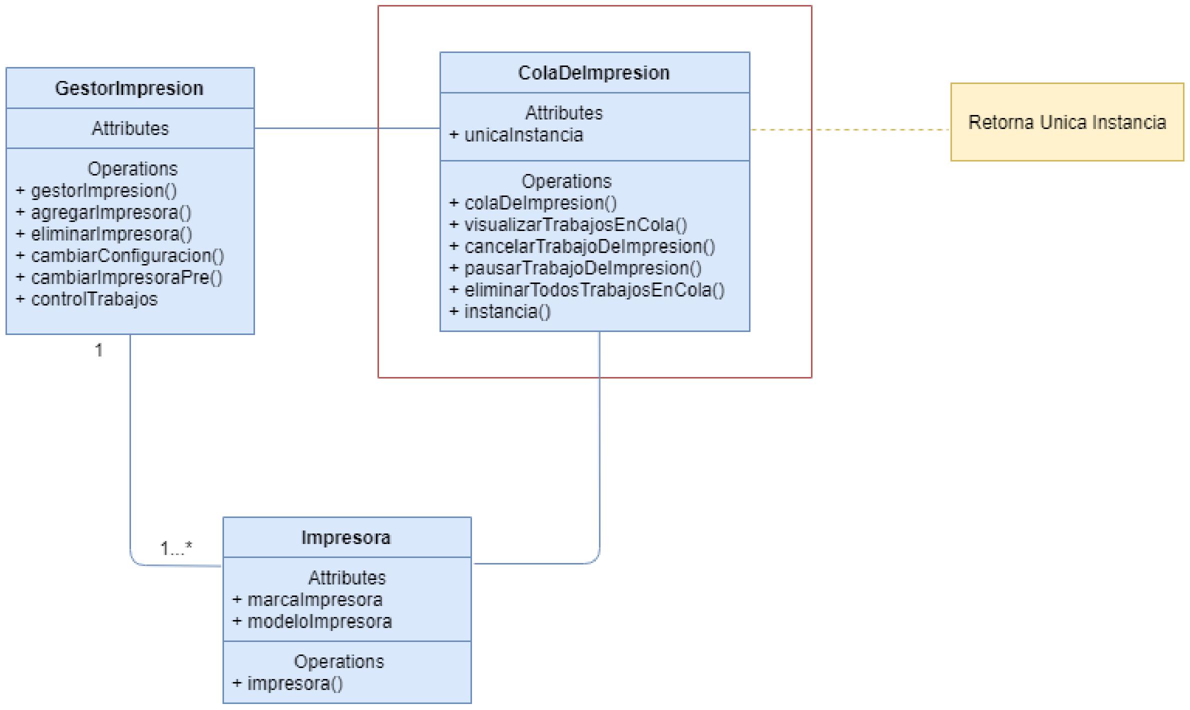


Figure 3: UML Diagram Printer System

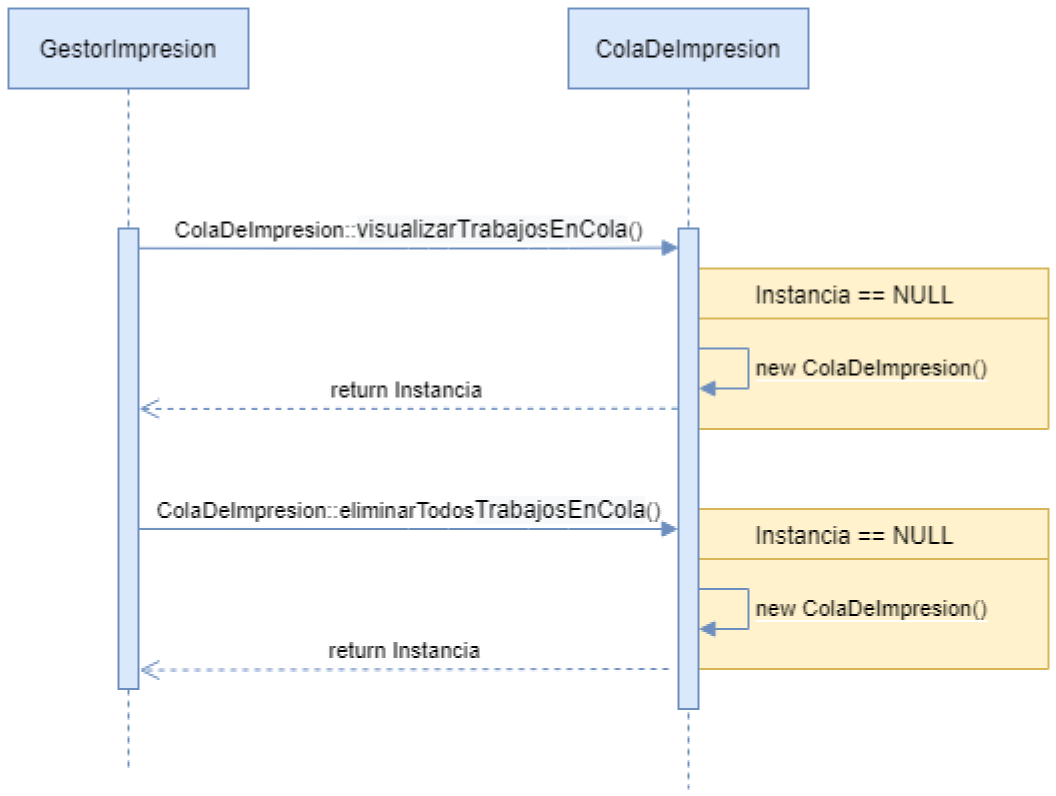


Figure 4: UML Diagram Printer System

Bank System

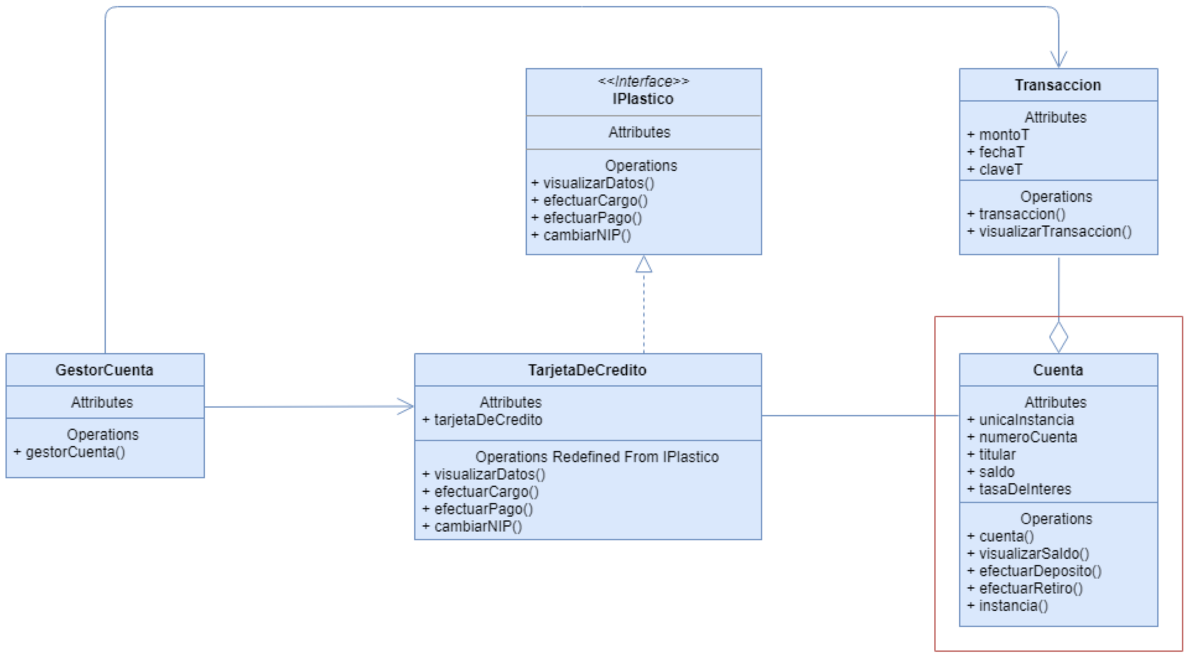


Figure 5: UML Diagram Bank System

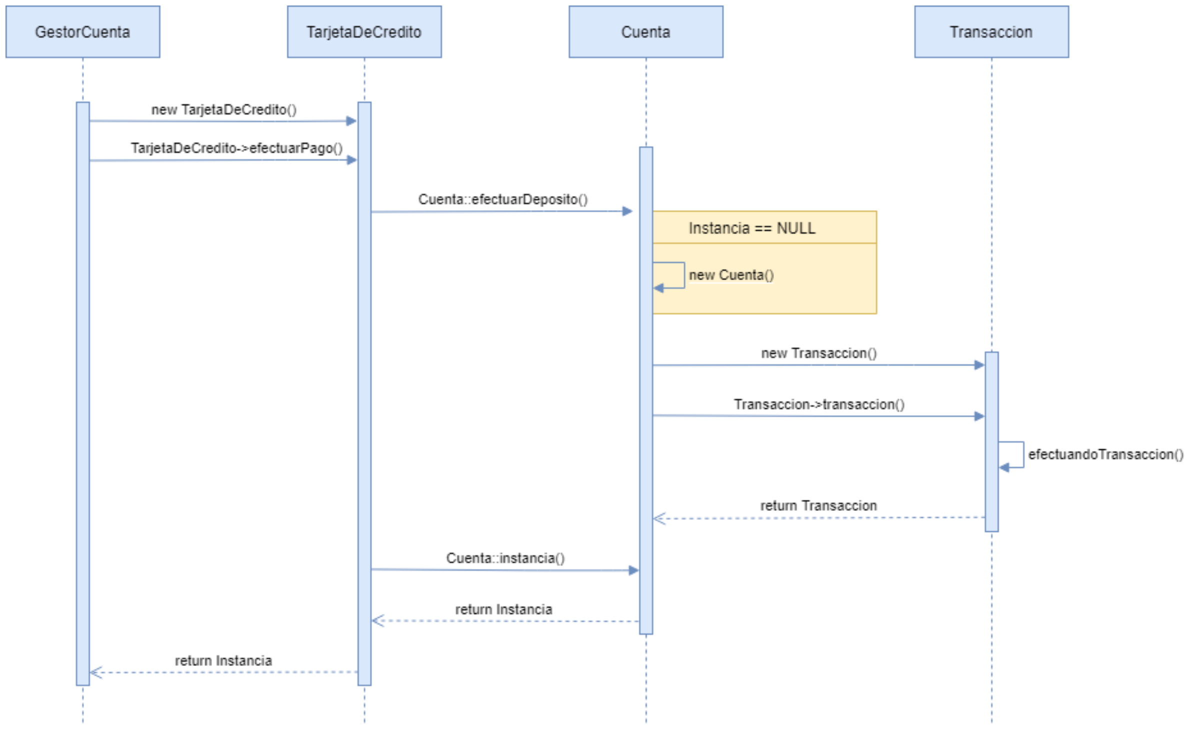


Figure 6: UML Diagram Bank System