State

Objective

Allowing an object to alter its behavior when its internal states change; it will seem to change its classes.

Function

Altering the internal behaviour of an object.

Structure

As shown in figure 1

- Context: Represents the component that can change state, which has among its properties the current state. In the example of the soda machine, this would be the machine as such.

- AbstractState: Base class for the generation of the different states. It is recommended that it is an abstract class instead of an interface because we can define default behaviors and thus affect the operation of all states.

- ConcreteState: Each of these components represents a possible state through which the application can pass, so we will have a ConcreteState for each possible state. This class must inherit from AbstractState.

The structure that meets this pattern is shown in Figure 1

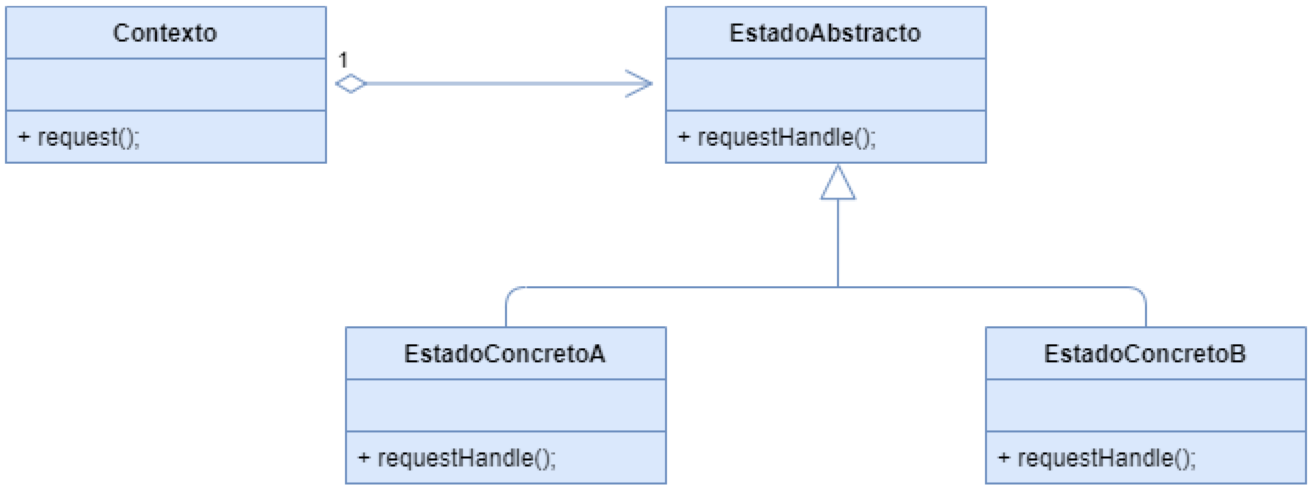

Figure 1: UML Diagram State Pattern

Applications

The use of the State pattern is recommended when:

- The behavior of an object depends entirely on its state, which must change at runtime according to that state.

- The operations to be performed have large declarations with many conditioning parts that depend on the state of the object; which is usually represented by one or several numbered constants.

Design Patterns Collaborators

- The Flyweight pattern helps to implement the State pattern; it explains when and where States objects should be shared.

Scope of action

Applied at the object level.

Problem

The application requires that the variation of a monolithic object be controlled at runtime; since this is defined according to its state, if it varies the object must also do so; which implies creating a class of declarations within which it is determined what behavior should be carried out, employing an impractical process.

Solution

The State pattern allows you to have a class that varies considering the numerous related classes; that is, the pattern changes between the different internal classes in such a way that the attached object seems to change classes. When condition statements are presented, the pattern places each branch of the conditioner in a separate class by treating the state of the object as a variable independent of others.

Diagram or Implementation

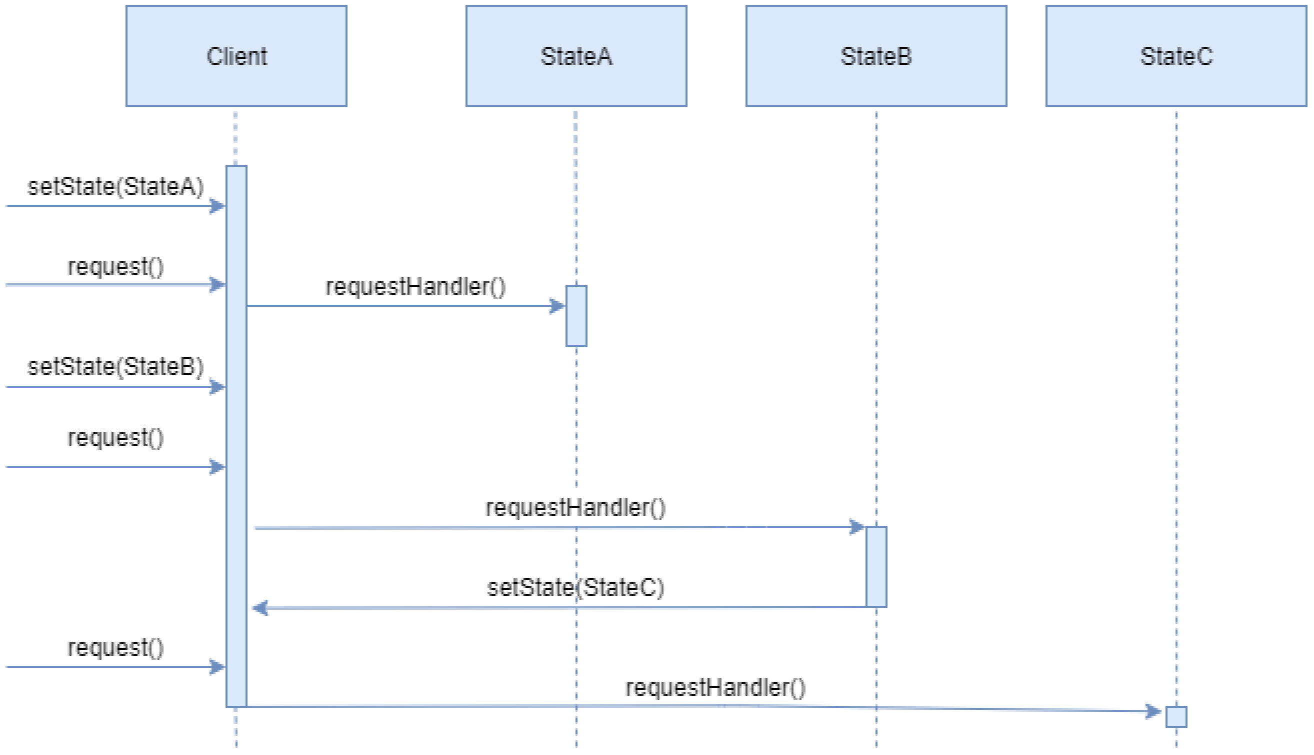

Figure 2: UML Diagram State Pattern

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

- A default state is set for the Client class, which is StateA.

- The request operation is executed on the Client class, which delegates the execution to the current state (StateA).

- Client class changes from state A to state B.

- The request operation is executed again on the Client class that delegates the execution to the current state (StateB).

- The execution of StateB results in a change of status to StateC.

- The request operation is executed again on the Client class that delegates the execution to the current state (StateC).