State ID

Student:

Izabela Kuźniar

Teacher:

Andrea Corradini

Course:

Software Design Patterns

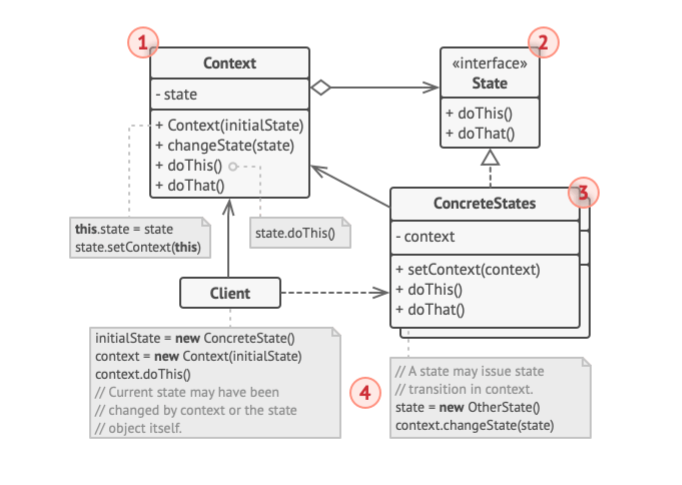
# Name and category

State is a behavioral pattern.

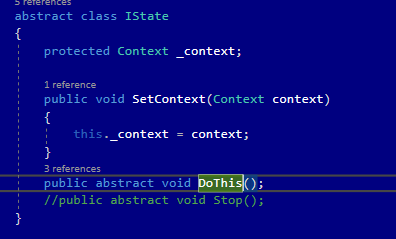
# Intent:

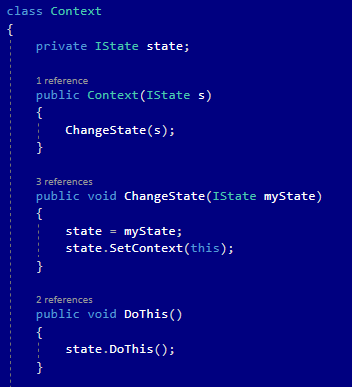
Enables an object to alter its behavior when its internal state changes. It appears as the object changed its class.

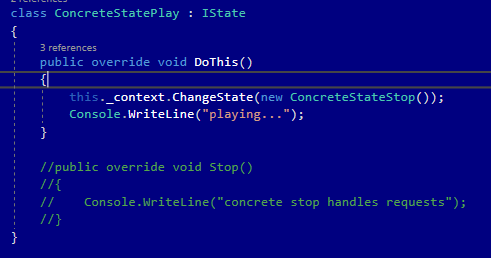
# Structure as a UML class diagram

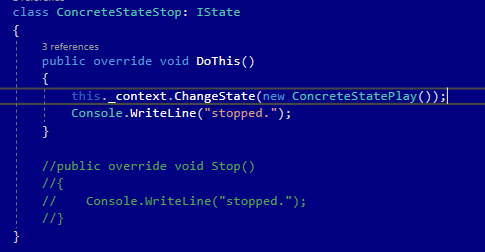


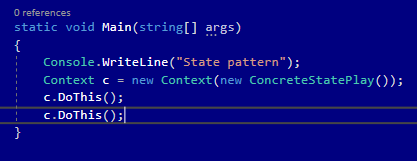
# Implementation:











# Consequences:

Benefits:

* Single Responsibility Principle. Organize the code related to particular states into separate classes.
* Open/Closed Principle. Introduce new states without changing existing state classes or the context.
* Simplifying the code of the context by eliminating bulky state machine conditionals.

Drawbacks:

* Applying the pattern can be overkill of a state machine has only a few states or rarely changes.

# Related patterns

1. Bridge, State, Strategy (and to some degree Adapter) have very similar structures. Indeed, all of these patterns are based on composition, which is delegating work to other objects. However, they all solve different problems. A pattern isn’t just a recipe for structuring your code in a specific way. It can also communicate to other developers the problem the pattern solves.
2. State can be considered as an extension of Strategy. Both patterns are based on composition: they change the behavior of the context by delegating some work to helper objects. Strategy makes these objects completely independent and unaware of each other. However, State doesn’t restrict dependencies between concrete states, letting them alter the state of the context at will.