Visitor ID

Student:

Izabela Kuźniar

Teacher:

Andrea Corradini

Course:

Software Design Patterns

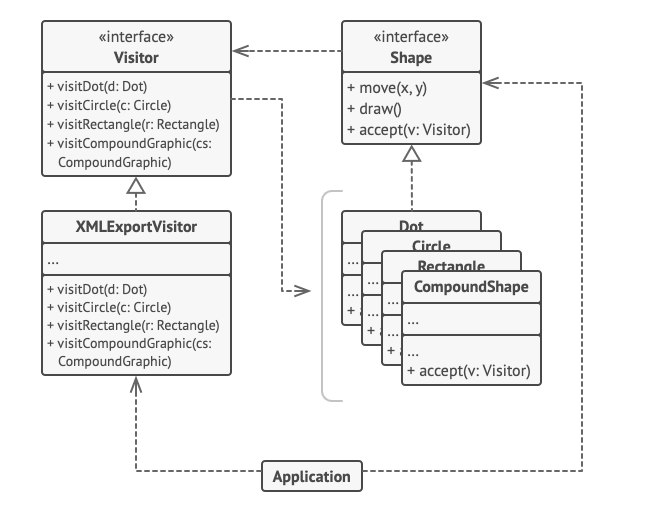
# Name and category

Visitor is a behavioral pattern.

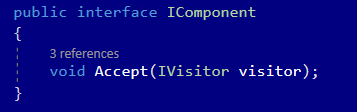
# Intent:

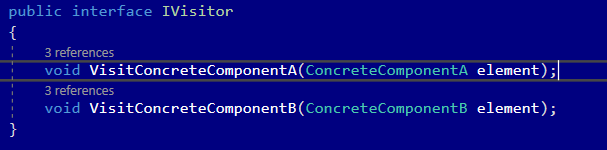
Separate algorithms from objects on which they operate.

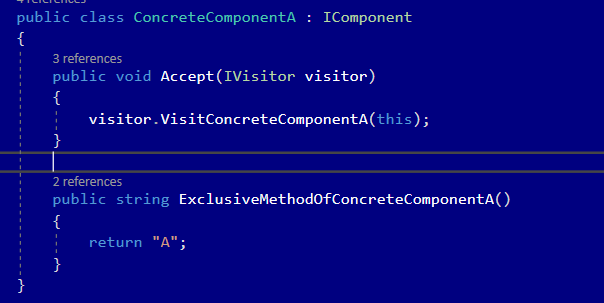
# Structure as a UML class diagram

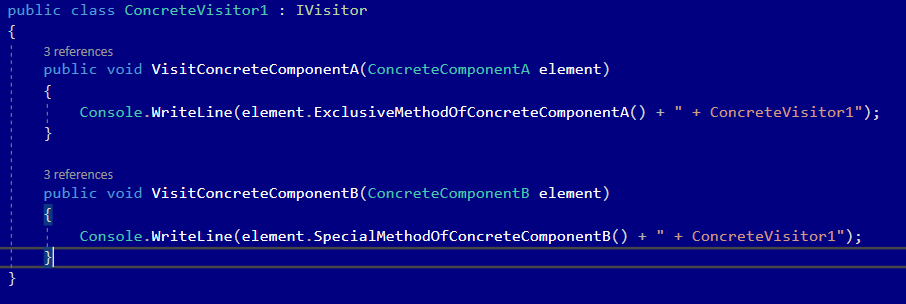


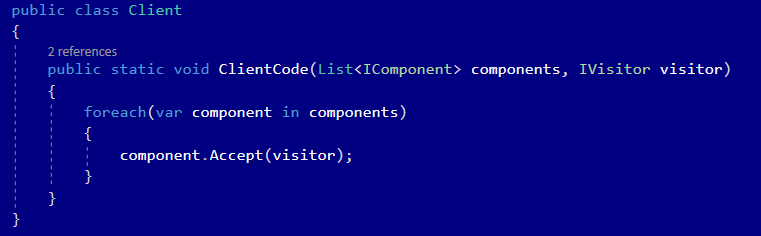
# Implementation:

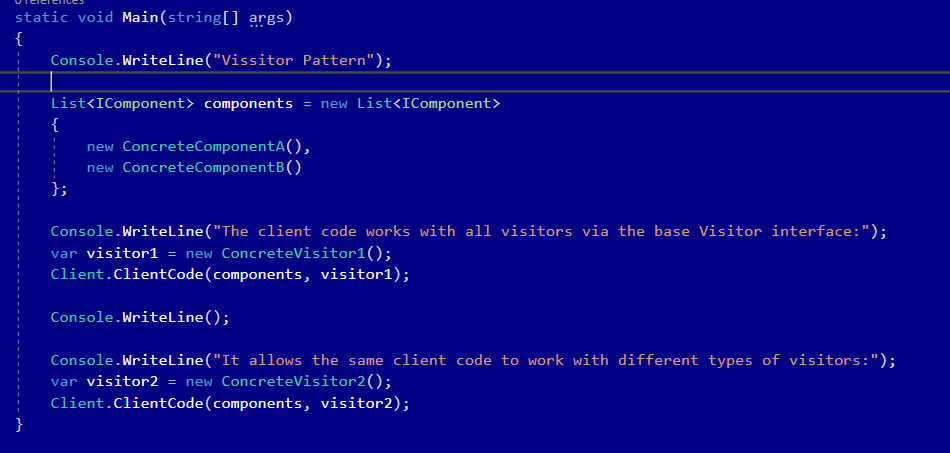












# Consequences:

Benefits:

* Open/Closed Principle. You can introduce a new behavior that can work with objects of different classes without changing these classes.
* Single Responsibility Principle. You c an move multiple versions of the same behavior into the same class.
* A visitor object can accumulate some useful information while working with various objects. This might be handy when you want to traverse some complex object structure, such as an object tree, and apply the visitor to each object of this structure.

Drawbacks:

* Need to update all visitors each time a class gets added to or removed from element hierarchy.
* Visitors might lack the necessary access to the private fields and methods of the elements that they’re supposed to work with.

# Known uses

* Since pattern separates the visitor from the object structure, new visitors can be added if new operations are needed.
* Can be applied when similar operation have to be performed on objects of different types in a structure

# Related patterns

1. Visitor can be treated as a powerful version of Command pattern. Its objects can execute operations over various objects of different classes.
2. Visitor can be used to execute an operation over an entire Composite tree.
3. Visitor can be used along with Iterator to traverse a complex data structure and execute some operation over its elements, even if they all have different classes.