Interpreter

Objective

Given a programming language, it defines a representation with an interpreter that is used for the statements.

Function

Define a grammar for a given language, as well as the tools needed to interpret it.

Structure

As shown in figure 1

- Client: Actor who triggers the performance of the interpreter.

- Context: Object with global information that will be used by the interpreter to read and store global information among all the classes that make up the pattern, this is sent to the interpreter who replicates it throughout the structure.

- AbstractExpression: Interface that defines the minimum structure of an expression.

- TerminalExpression: Refers to expressions that have no more continuity and when evaluated or interpreted they end the execution of that branch. These expressions mark the end of the execution of a sub-tree of the expression.

- ExpressionsNoTerminal: These are composite expressions and within them there are more expressions to be evaluated. These structures are interpreted using recursion until arriving at a Terminal expression.

The structure that meets this pattern is shown in Figure 1

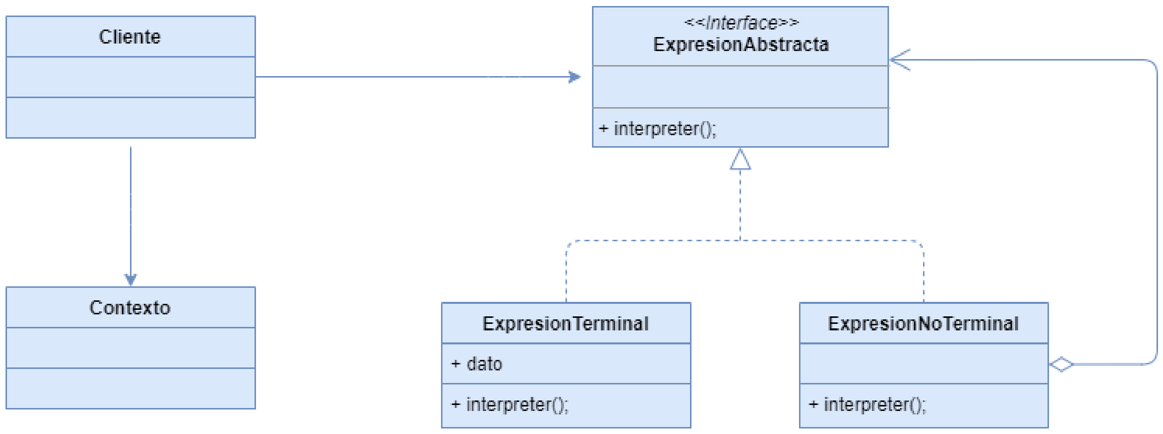

Figure 1: UML Diagram Interpreter Pattern

Applications

The use of the Interpreter pattern is recommended when:

- You want to interpret a grammar, and this is not very complex.
- The expected efficiency of the application is not demanding.

Design Patterns Collaborators

- Tree-structured statements are generally implemented as a composite.
- The flyweight pattern shows how to share terminal symbols within the abstract tree syntax.
- An Iterator pattern can be used to navigate the structure.
- The visitor pattern can be used to maintain the behavior of each node in the tree structure.

Scope of action

Applied at the object level.

Problem

To execute various commands or instructions, these are defined in a basic structure so that the application interprets them correctly; by means of an object that interprets the sentences by means of if or Switch, case: conditions, which considerably complicates programming.

Solution

The Interpreter design pattern defines the instructions as objects, which allows each implemented instruction to identify itself as a statement or another one through a tree structure analysis; all this process through an interface.

Diagram or Implementation

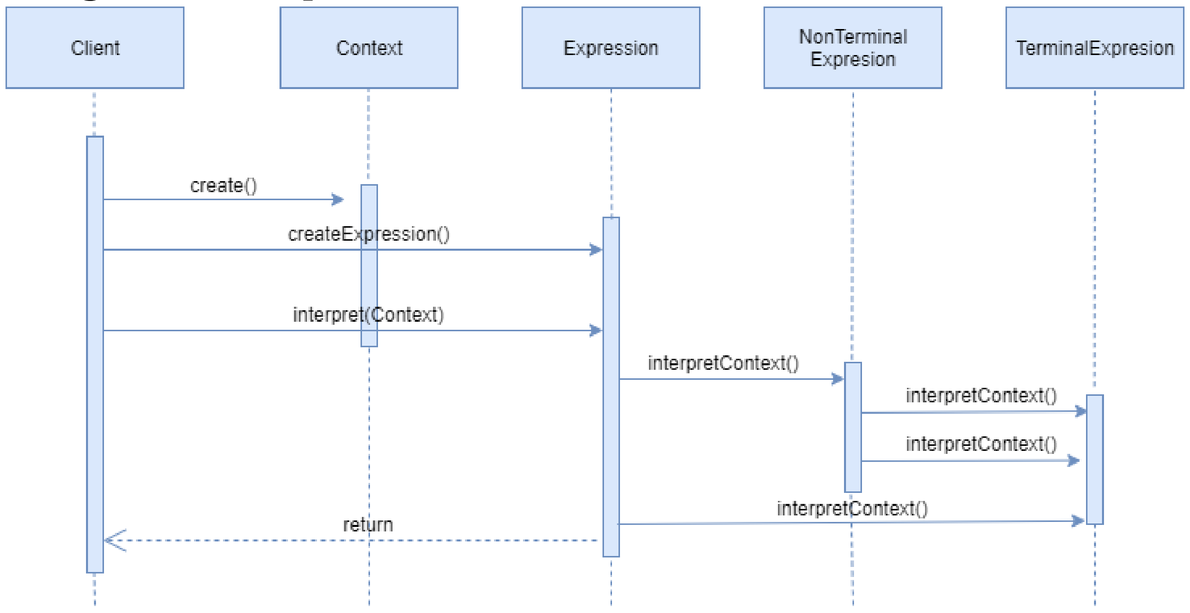

Figure 2: UML Diagram Interpreter Pattern

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

- The client class creates the context for the execution of the interpreter component.
- The client class creates or obtains the expression to be evaluated.
- The client class requests the interpretation of the expression from the interpreter component and sends it the context.
- The Expression calls the Non-Terminal Expressions it contains.
- The Non-Terminal Expression calls all Terminal Expressions.
- The Root Expression requests the interpretation of a Terminal Expression.

- The expression is fully evaluated and you have a result from the interpretation of all terminal and non-terminal expressions.