

# Interpreter Design Pattern

Behavioral Pattern

By:- Yunsheng Guo  
Koushik Godbole  
Pratheek Gowda

# Intent

- A class of a problem occurs repeatedly in a well defined and well understood domain.
- Problems can be expressed in a simple language
- If a domain are characterized by a language the problem could be easily solved with an interpretation engine.

# Scenario

Musician Interpreting the musical notations by other musicians.

Musical notes are a well understood language.

Music is a well understood domain.

Musicians are our interpreters.

# How to achieve this in our problem?

Define a grammar for the language.

Match each expression in the input to the grammar rule book.

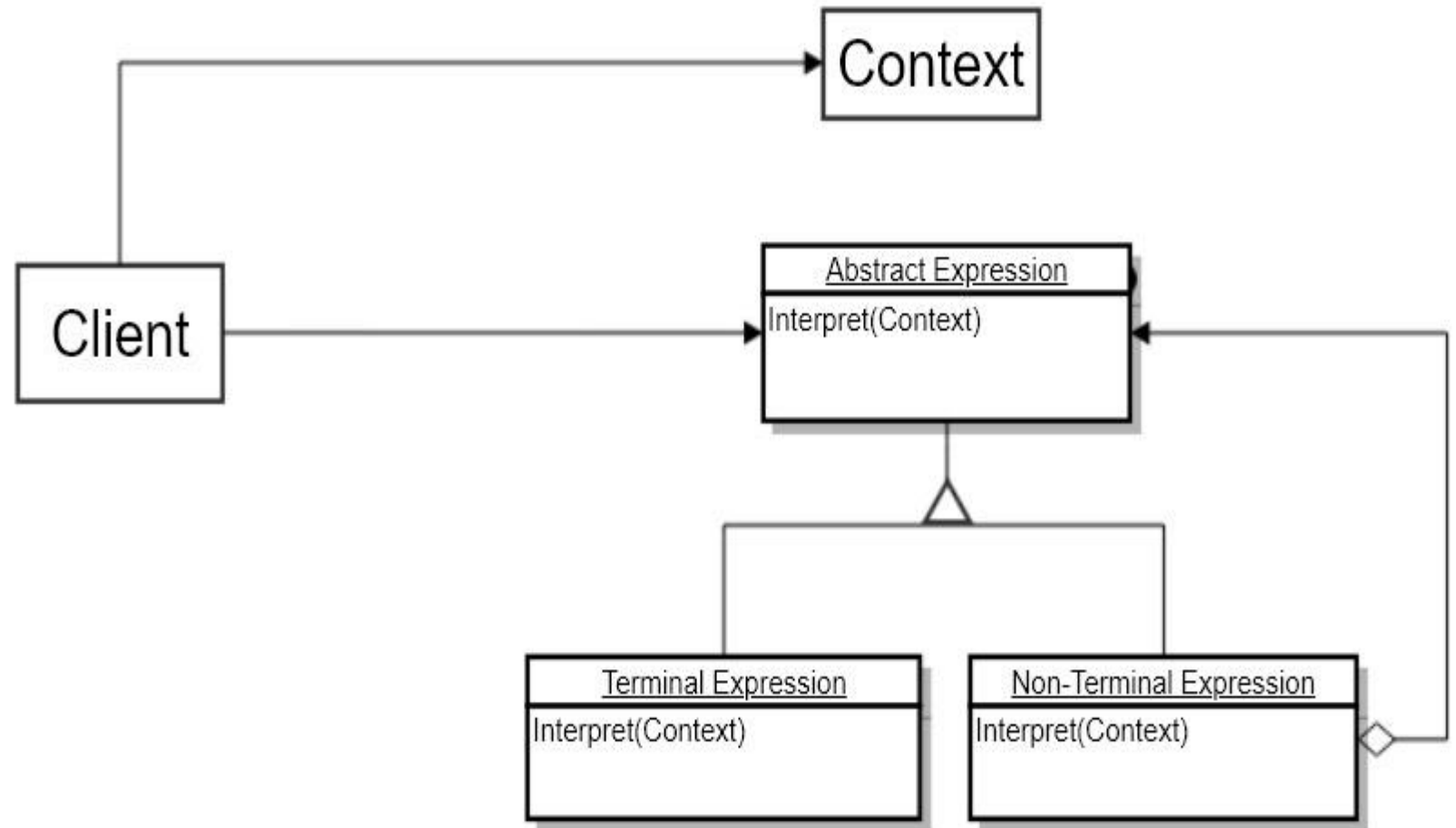
Organize the rule book into a structure.

Input the expression to the rule book and get it interpreted.

# Interpreter Pattern

The Interpreter pattern describes how to define a grammar for simple languages, represent sentences in the language, and interpret these sentences.

# Structure of the Pattern



# Our Skeleton Code

Microsoft Visual Studio

# Participants

- Abstract Expression
  - Declares an abstract Interpret operation that is common to all nodes in the abstract syntax tree.
- Terminal Expression
  - Implements an Interpret operation associated with terminal symbols in the grammar.
  - An instance is required for every terminal symbol in a sentence.
- Nonterminal Expression
  - One such class is required for every rule  $R ::= R_1 R_2 \dots R_n$  in the grammar.
  - maintains instance variables of type Abstract Expression for each of the symbols  $R_1$  through  $R_n$ .

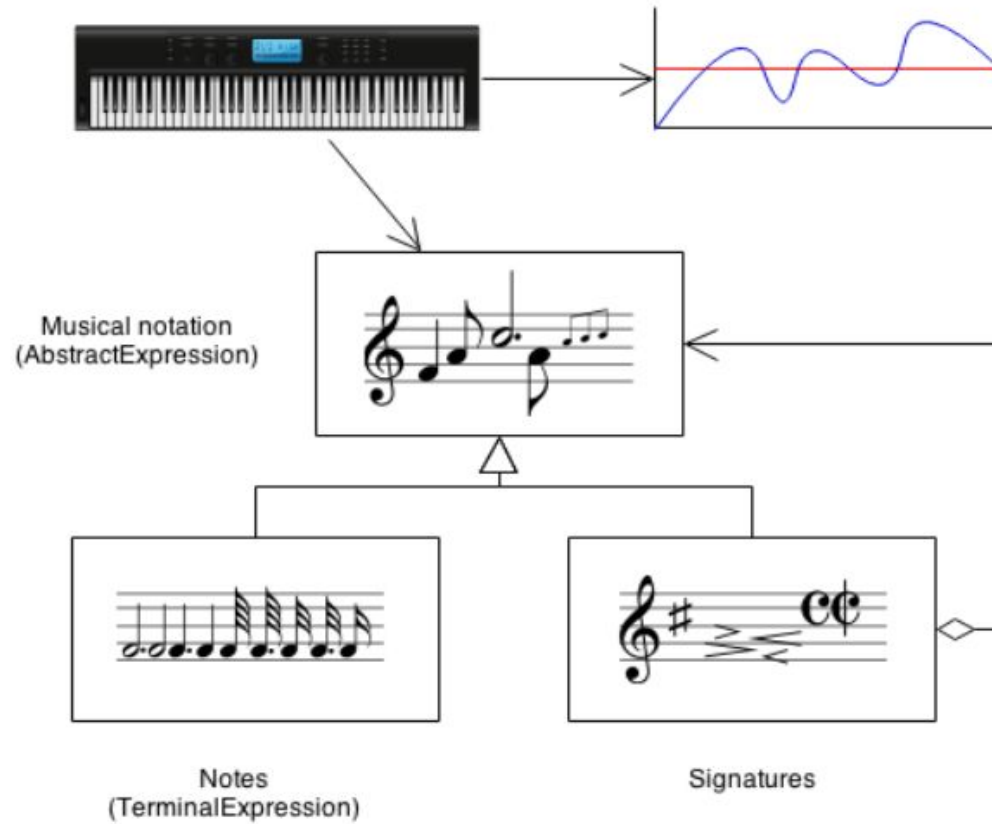
- Context
  - Contains information that's global to the interpreter.
- Client
  - Builds an abstract syntax tree representing a particular sentence in the language that the grammar defines. The abstract syntax tree is assembled from instances of the Nonterminal Expression and Terminal Expression classes.
  - Invokes the interpret operation.

# Applicability

- Language to interpret, and statements in the language can be represented as Abstract Syntax Tree.

# Scenario

Taking musician as example for interpreters



## Known uses

- Compiler implemented with Object Oriented Design.
- SQL Parsing
- Symbol Processing Engines

# Consequences

- It's easy to change, extend, and implement the grammar
- The more complex the grammars are, the harder it is to maintain
- It's easy to add new ways to interpret (with visitor pattern).

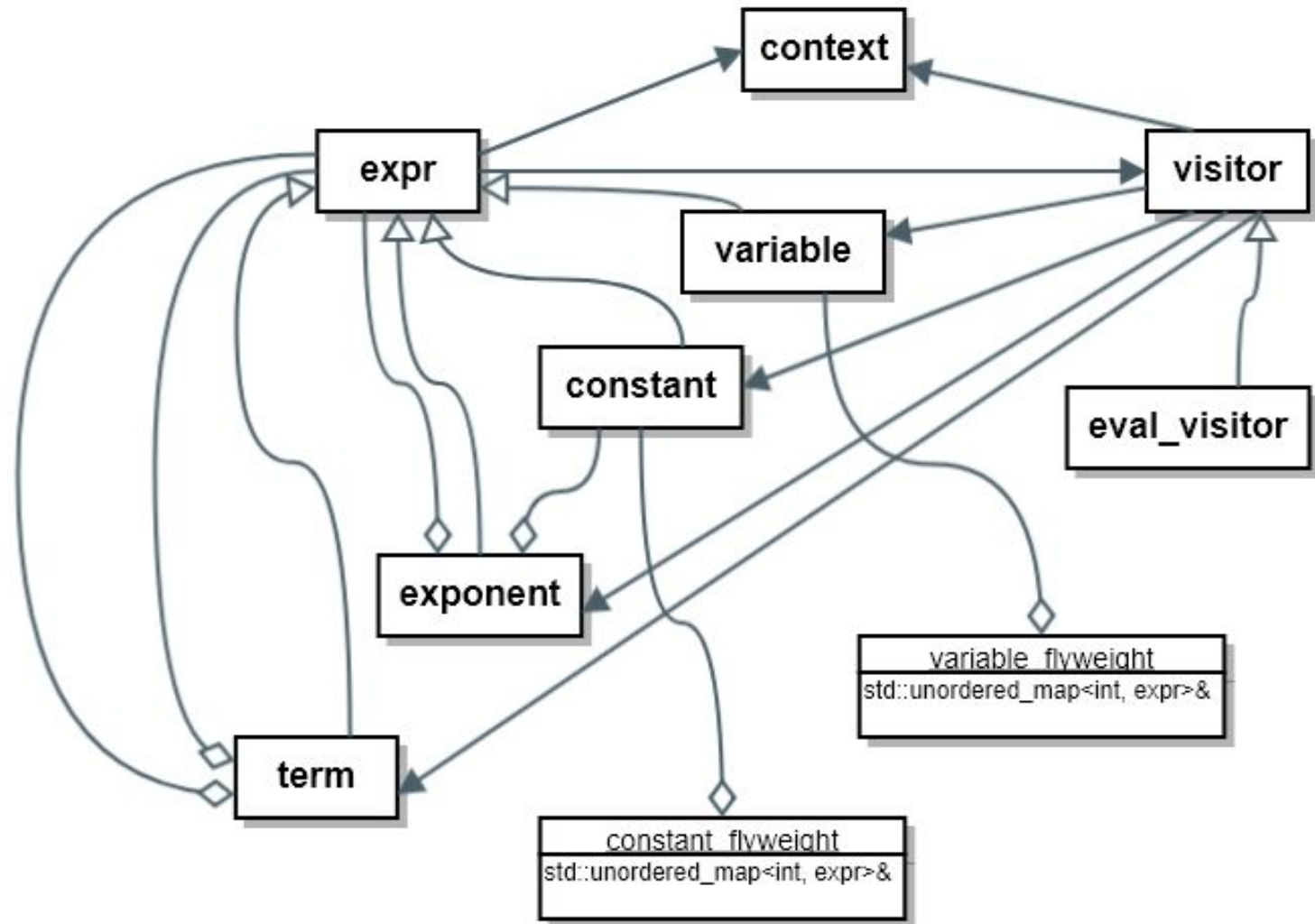
# Implementation

- How to acquire the syntax tree is not within the scope of the pattern
- Interpret operations can be put in a separate visitor object rather than expression classes
- terminal symbols can be shared with flyweight pattern
- watch out for recursion loop in a direct formal method approach

Revised Code

Microsoft Visual Studio

# Revised Code Structure



# Similar Patterns

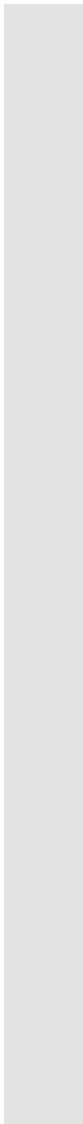
- Composite Pattern.
- Visitor Pattern.
- Flyweight Pattern.

# Reference

Design Patterns Elements of Reusable Object-Oriented Software

<https://www.geeksforgeeks.org/interpreter-design-pattern/>

[https://sourcemaking.com/design\\_patterns/interpreter](https://sourcemaking.com/design_patterns/interpreter)



Thank You