Design patterns

Interpreter pattern

not-matthias

March 8, 2020

What is the Interpreter Pattern?

Wikipedia says:

[...] the interpreter pattern is a design pattern that specifies how to evaluate sentences in a language. [1].

How does it work?

General

- One class for each symbol
  - Terminal
  - Nonterminal

Terminal Symbol

Write something

Nonterminal Symbol

Write something

What problems can the pattern solve?

Write something

What solution does the pattern describe?

Abstract Syntax Tree

- Define

Usage

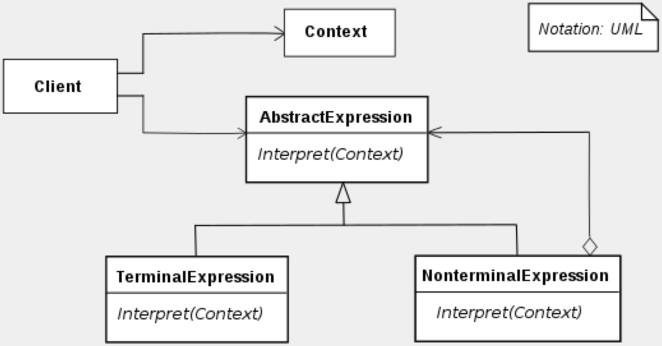
- Define a grammar for a simple language.
  - By defining a *Expression* class hierarchy with an *interpret()* function.
- Represent a sentence in the language with an AST made up of *Expression* instances.
- Interpret a sentence by calling *interpret()* on the AST.

How can the pattern be used?

Applicability

- Used when there’s a language to interpret.
  - Represent statements as AST
- Works best when **the grammar is simple**.
  - Use parsers for a large class hierarchy.
  - Doesn’t use an AST. Saves space and time.
- Works best when **efficiency is not a critical concern**.
  - More efficient when translating the parse tree to another form.

UML Class Diagram



Examples

- java.util.Pattern
- java.text.Normalizer
- javax.el.ELResolver
- All subclasses of java.text.Format

Other Examples

- Specialized database query languages (e.g. SQL)
-

Demo.

```
System.out.prinln("Thanks.");
```

References

Wikipedia.
Interpreter pattern.