Java Design Patterns

Interpreter

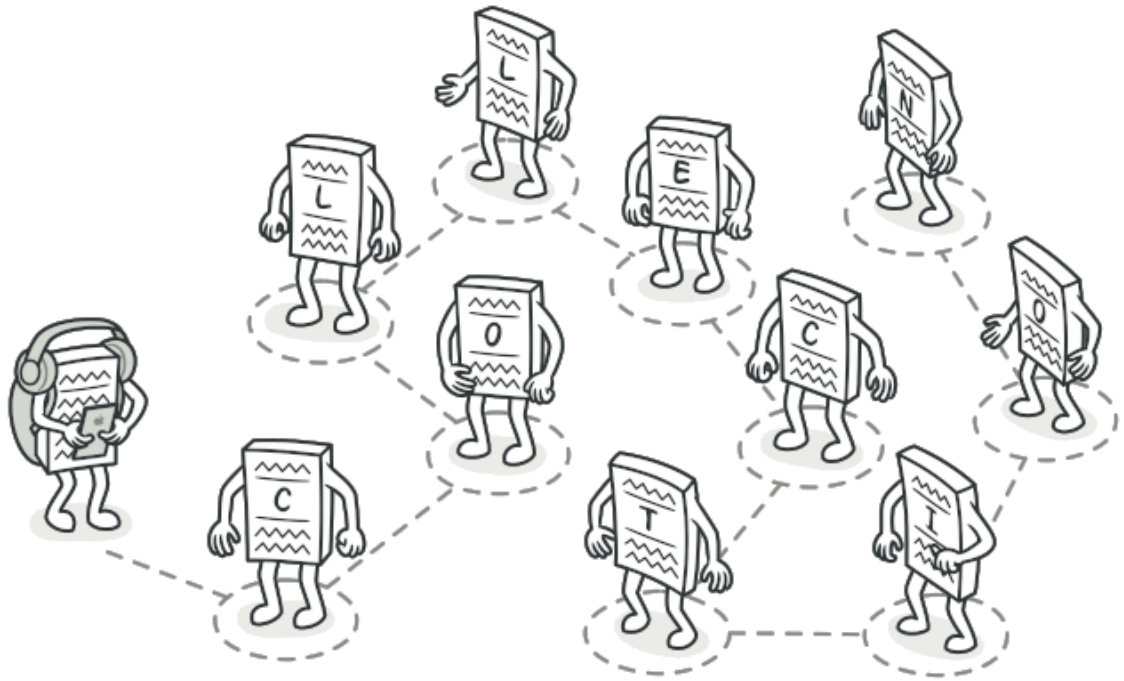
Тема

Interpreter

Суть паттерна

Интерпретатор

Интерпретатор — поведенческий паттерн проектирования, решающий часто встречающуюся, но подверженную изменениям, задачу.



Проблема

Постановка задачи

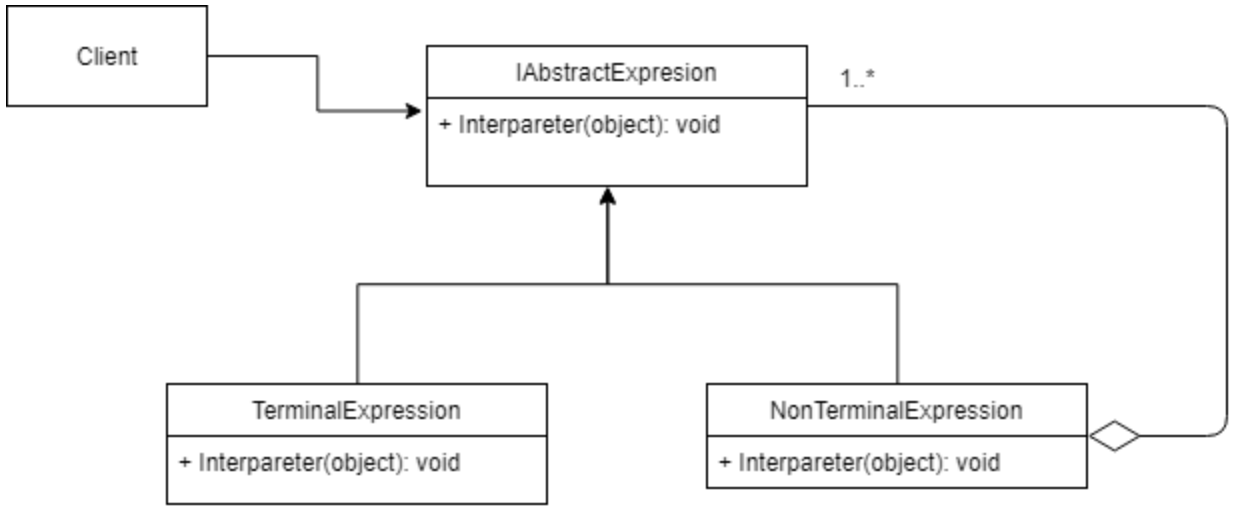
Представим что в проекте вам необходимо часто выполнять какие либо операции, вычисления, или же однообразные алгоритмы обработки данных. Данный паттерн вы рассмотрим на примере простой задачи, допустим надо совершить простые действия с простыми числами, такие как сложение и вычитание. Например: 1-2+3.

Решение задачи

Идея паттерна Интерпретатора заключается в том что бы разбить поставленную задачу, на более мелкие подзадачи. Каждую операцию мы выносим в отдельный класс, что в последствии позволит нам использовать набор уже созданных операций для выполнения более сложных задач. В нашем примере по правилам паттерна мы должны будем создать 3 класса. Каждый из классов будет отвечать за определенную операции, и 4-й класс, контекст, который будет содержать в себе алгоритм решения определённой задачи.

Структура паттерна

1. Клиент взаимодействует с системой через интерфейс, который в свою очередь может быть каким либо выражением.
2. У интерфейса есть несколько наследников.
3. Терминальное выражение это конечные выражения которые принимают в себя какие то параметры или данные, и в замен выдают результат в зависимости от решаемой задачи.
4. Не терминальные выражения это не конечные выражения, которые могут принимать в качестве параметров другие выражения, которые в свою очередь так же могут принимать другие выражения.



Применимость

Применение паттерна

1. Когда есть язык для интерпретации, предложения которого можно представить в виде абстрактных синтаксических деревьев.
2. Грамматика достаточно проста.

Шаги реализации

Алгоритм реализации паттерна

1. Разбейте задачу на простые подзадачи.
2. Создайте интерфейс через который пользователь будет взаимодействовать с решением задачи.
3. Для каждой подзадачи создайте отдельный класс который будет наследоваться от ранее созданного интерфейса.
4. Создайте класс контекст, в который поместите основную логику решения вашей задачи.

Преимущества и недостатки

Плюсы и недостатки

Плюсы:

- Грамматику становится легко расширять и изменять.
- Можно легко изменять способ вычисления выражений.

Минусы:

- Сопровождение грамматики с большим числом правил затруднительно.

Отношение с другими паттернами

- Обычно вместе с паттерном интерпретатор используется компоновщик: абстрактное синтаксическое дерево - это пример применения паттерна компоновщик.
- Далее - приспособленец показывает варианты совместного использования терминальных символов в абстрактном синтаксическом дереве.
- Итератор: интерпретатор может пользоваться итератором для обхода структуры.
- Посетителя можно использовать для инкапсуляции в одном классе поведения каждого узла абстрактного синтаксического дерева.

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association