
# SDET Course

Design Patterns - Interpreter

# 3 Types of Design Patterns

- Creational
  - Singleton
  - Builder
  - Prototype
  - Factory Method
  - Abstract Factory
- Structural
  - Adapter
  - Composite
  - Proxy
  - Flyweight
  - Bridge
  - Facade
  - Decorator
- Behavioral
  - Strategy
  - Observer
  - Command
  - Memento
  - State
  - Template Method
  - Mediator
  - Chain of Responsibility
  - **Interpreter**
  - Visitor
  - Iterator

# Agenda

- Description
- Diagram
- Code sample (Java)
- Use cases

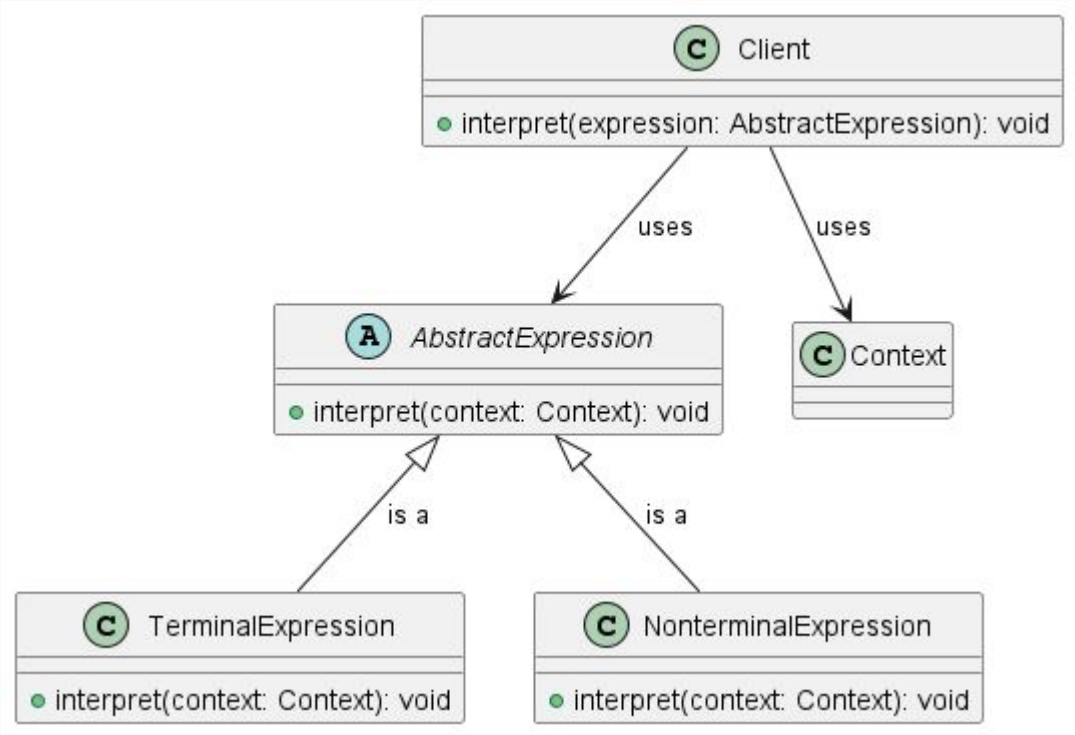
VERISOFT

Descrip
on

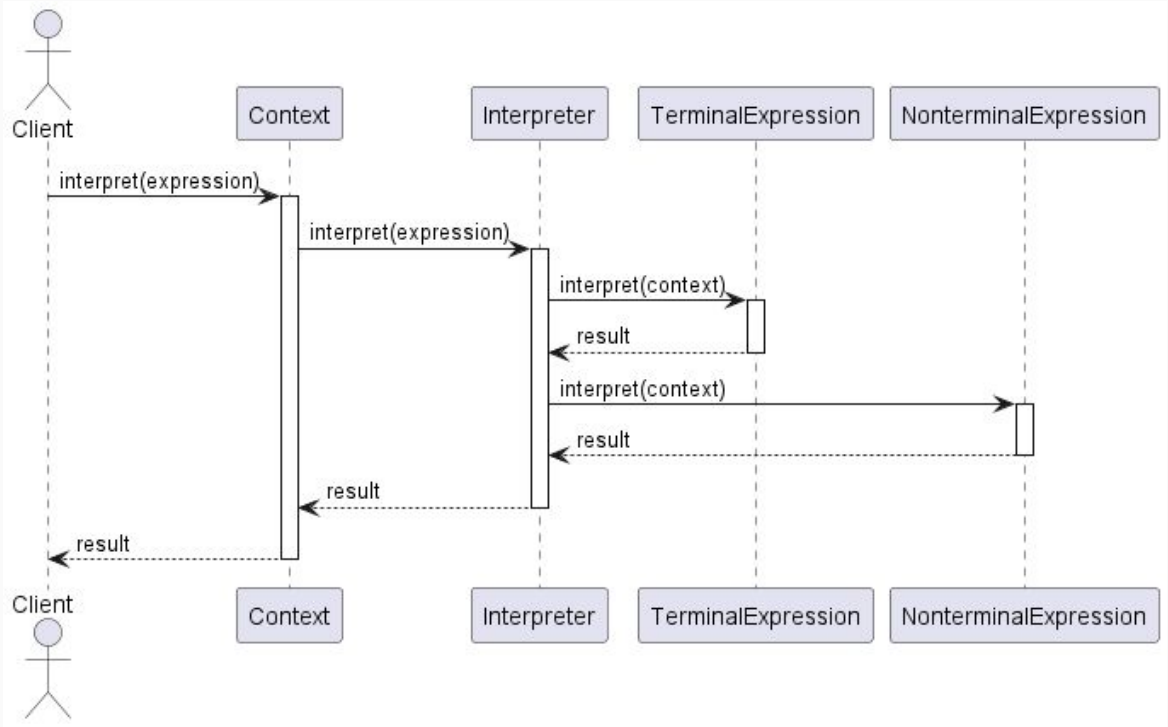
# Description

The Interpreter design pattern is a behavioral design pattern that is used to define a grammatical representation for a language and provides an interpreter to deal with this grammar. The pattern is particularly useful when there is a need to implement a language for expressing operations and statements that can be evaluated or executed by the software. This pattern involves defining an expression interface with an interpret method and implementing this interface for different kinds of expressions. The interpreter uses a tree structure to represent the syntax of the language it interprets, with each node of the tree being an instance of an expression class. This pattern allows for the parsing and interpretation of the language sentences, offering a flexible and extendable way to integrate new expressions and functionality into the system without modifying the core code structure. It is widely used in compilers, parsers, and other language processing systems.

# Class Diagram



# Sequence Diagram



# Code Sample

# Use cases

- General
  - SQL Query Interpretation
  - Programming Language Compilers

- In Test Automation
  - Cucumber Design
  - Keyword Driven Testing

cucumber

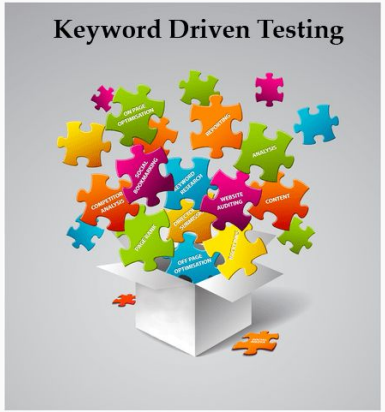



Image Source

# KDT - Keyword Driven Testing

VERISOFT

- **Definition**: Keyword-driven testing is a software testing methodology that separates test script design and execution.
- **Approach**: Tests are designed using a set of keywords representing actions or operations to be performed on the application under test.
- **Modularity**: Tests are composed of reusable modules or keywords, enhancing maintainability and scalability.
- **Flexibility**: Allows testers to create tests without scripting knowledge by using a predefined set of keywords.
- **Collaboration**: Encourages collaboration between testers and domain experts to define meaningful keywords reflecting business requirements.
- **Automation**: Enables automation by associating keywords with corresponding automation scripts or functions.
- **Maintenance**: Eases test maintenance as changes in the application require updates only to the keyword implementations, not the test scripts.
- **Example**: Keywords might include actions like "login," "search," "add to cart," etc., which are then combined to create test cases.

# Send an Email Using Gmail - KDT

VERISOFT

| Step | Keyword | Parameters |
|---|---|---|
| 1 | Open Browser | Browser: Chrome |
| 2 | Navigate | URL: https://mail.google.com/ |
| 3 | Input Text | Locator: username field, Text: [username] |
| 4 | Click Element | Locator: Next button |
| 5 | Input Text | Locator: password field, Text: [password] |
| 6 | Click Element | Locator: Sign in button |
| 7 | Click Element | Locator: Compose button |
| 8 | Input Text | Locator: To field, Text: [recipient email] |
| 9 | Input Text | Locator: Subject field, Text: [email subject] |
| 10 | Input Text | Locator: Email body, Text: [email body] |
| 11 | Click Element | Locator: Send button |
| 12 | Verify Text | Locator: Success message, Expected Text: Email sent successfully. |


# Happy Coding