



Behavioral Design Patterns

Ms. Nehal Adhvaryu

Introduction

concerned with the interaction and responsibility of objects.

The interaction between the objects should be in such a way that they can easily talk to each other and still should be loosely coupled.

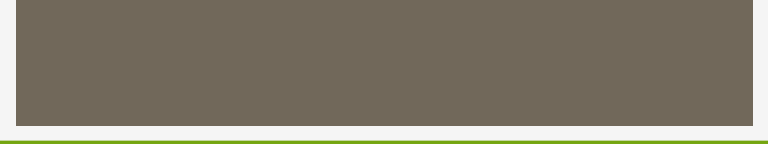
Introduction

The implementation and the client should be loosely coupled in order to avoid hard coding and dependencies

Chain of Responsibility Pattern

Sender sends a request to a chain of objects. The request can be handled by any object in the chain.

A Chain of Responsibility Pattern says that just ‘avoid coupling the sender of a request to its receiver by giving multiple objects a chance to handle the request’.



In other words, we can say that normally each receiver contains reference of another receiver. If one object cannot handle the request then it passes the same to the next receiver and so on.

Advantage:

- It reduces the coupling.

- It adds flexibility while assigning the responsibilities to objects.

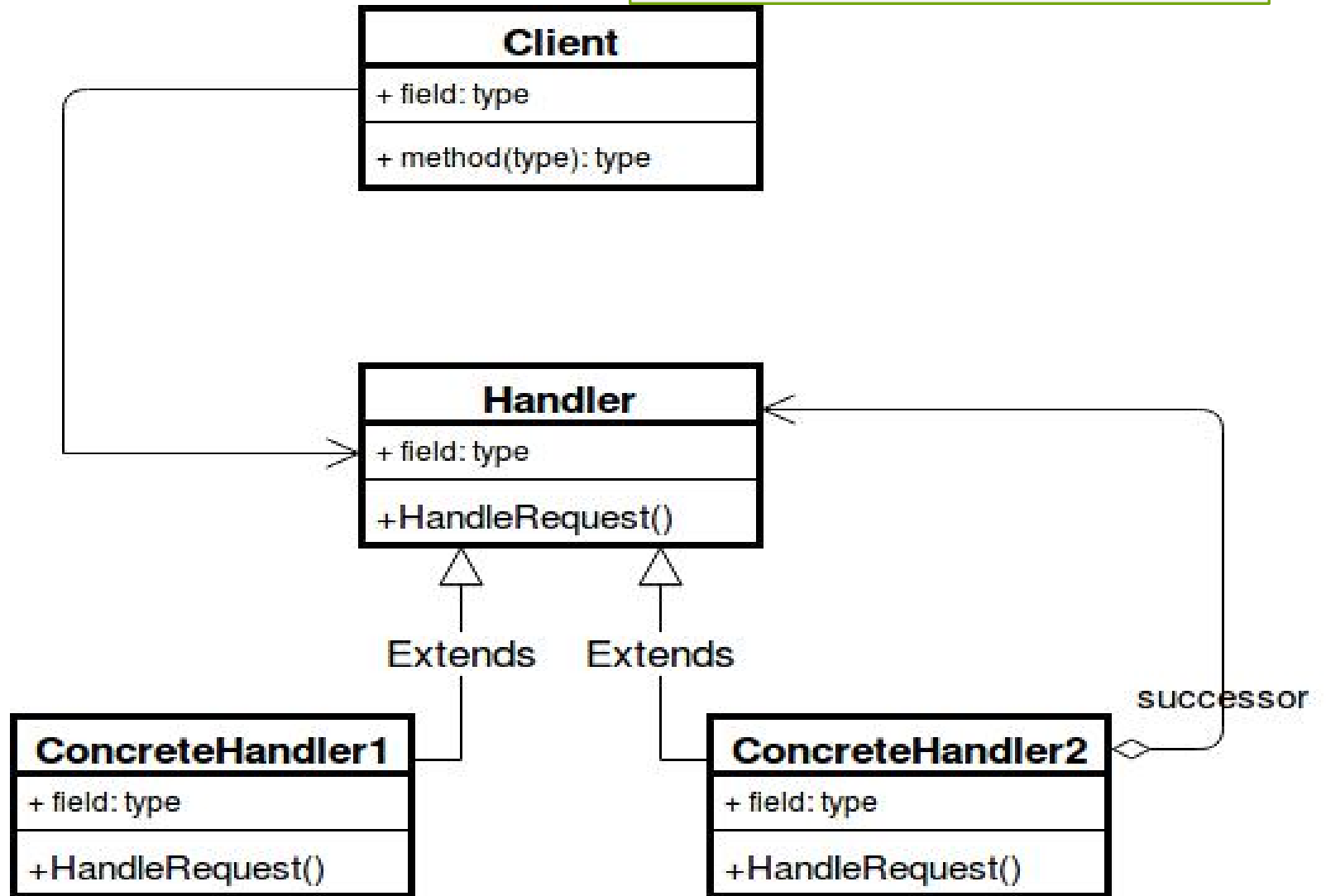
- It allows a set of classes to act as one; events produced in one class can be sent to other handler classes with the help of composition.



It is used:

When more than one object can handle a request and the handler is unknown.

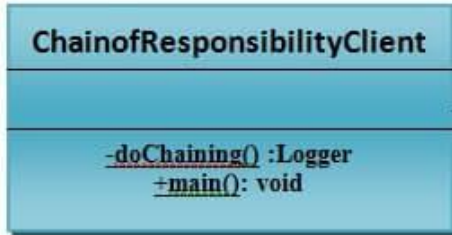
When the group of objects that can handle the request must be specified in dynamic way.



Handler : This can be an interface which will primarily receive the request and dispatches the request to chain of handlers. It has reference of only first handler in the chain and does not know anything about rest of the handlers.

Concrete handlers : These are actual handlers of the request chained in some sequential order.

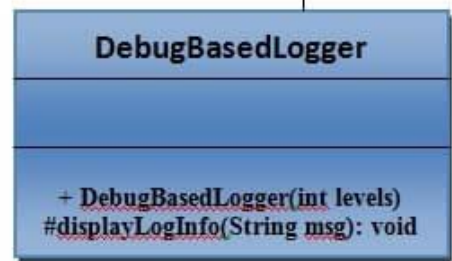
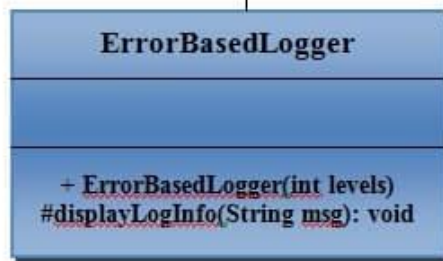
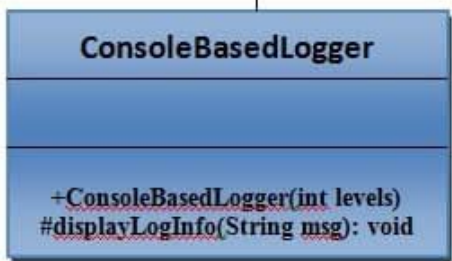
Client : Originator of request and this will access the handler to handle it.



<<abstract class>>

uses

extends



javatpoint.com

Ms. Nehal Adhvaryu

Command Pattern

A Command Pattern says that ‘encapsulate a request under an object as a command and pass it to invoker object’.

Invoker object looks for the appropriate object which can handle this command and pass the command to the corresponding object and that object executes the command’.

It is also known as Action or Transaction.

Command Pattern

Advantage

It separates the object that invokes the operation from the object that actually performs the operation.

It makes easy to add new commands, because existing classes remain unchanged.

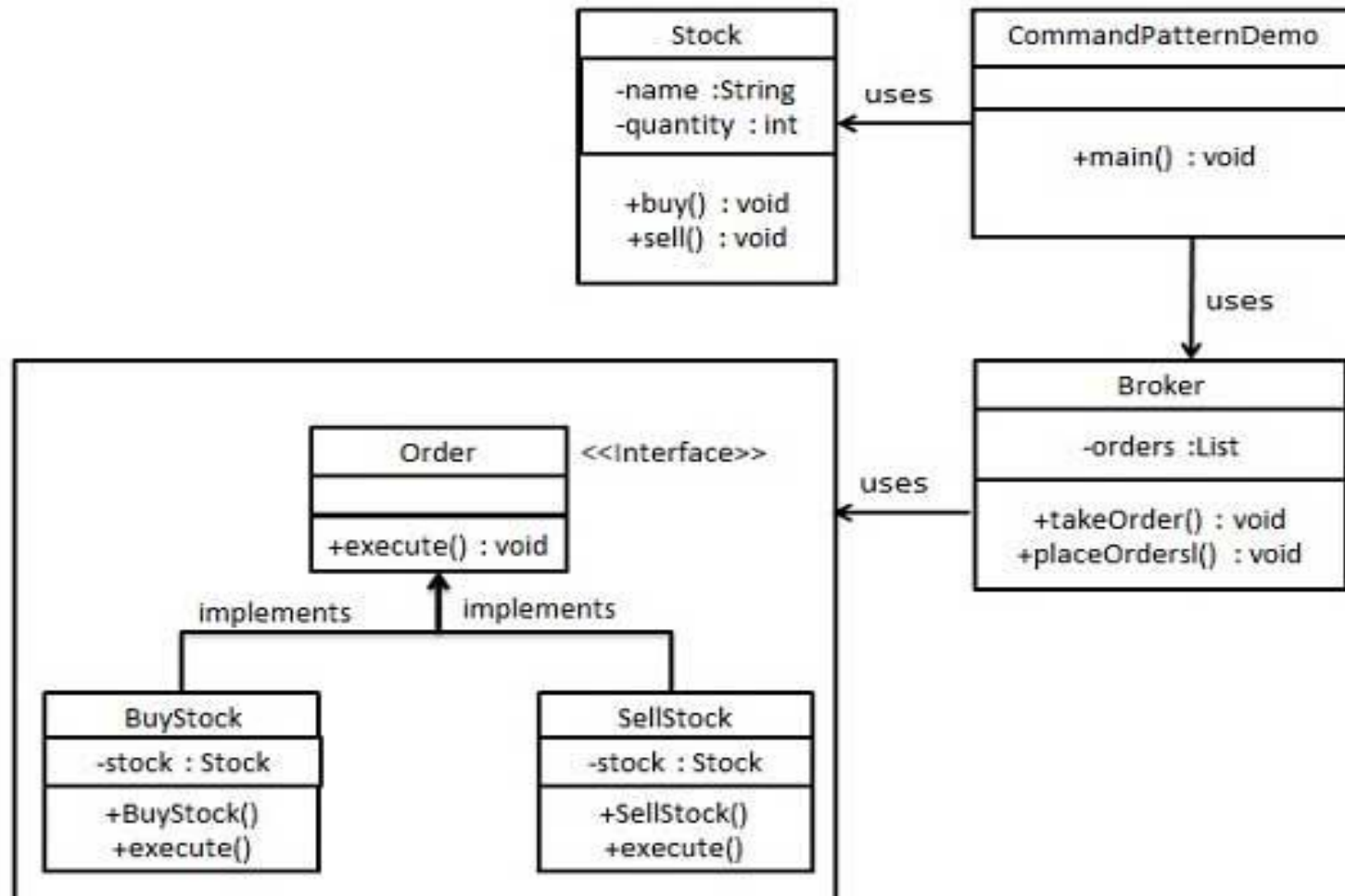
Command Pattern

It is used:

When you need parameterize objects according to an action perform.

When you need to create and execute requests at different times.

When you need to support rollback, logging or transaction functionality.



Ms. Nehal Adhvaryu

Interpreter Pattern

An Interpreter Pattern says that ‘to define a representation of grammar of a given language, along with an interpreter that uses this representation to interpret sentences in the language‘.

Interpreter pattern provides a way to evaluate language grammar or expression.

Interpreter Pattern

Basically the Interpreter pattern has limited area where it can be applied.

This pattern is used in SQL parsing, symbol processing engine etc.

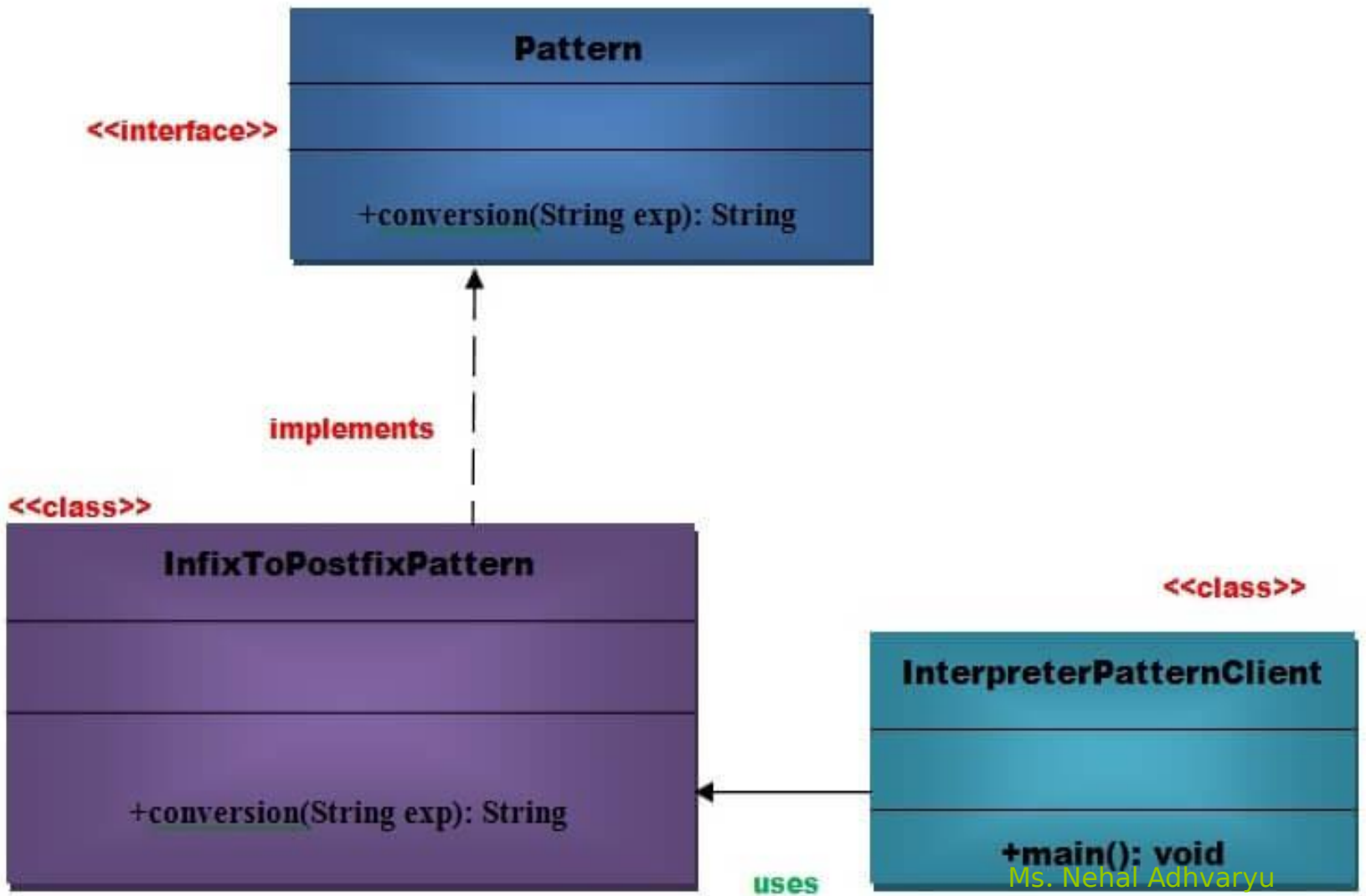
Interpreter Pattern

Advantage

It is easier to change and extend the grammar.
Implementing the grammar is straightforward.

Usage

When the grammar of the language is not complicated.
When the efficiency is not a priority.



Ms. Nehal Adhvaryu