Command

Padrões de projeto

Aluno: Amaro Miranda Neto

Professor(a): Dr.ª Alana Morais

Disciplina: Métodos avançados de programação

Descrição

Intenção:

Command é um padrão comportamental que transforma uma solicitação em um objeto independente que contém todas as informações sobre a solicitação. Essa transformação permite parametrizar métodos com diferentes solicitações, atrasar ou enfileirar a execução de uma solicitação e oferecer suporte a operações que podem ser desfeitas.

Também conhecido como:

Action, Transaction.

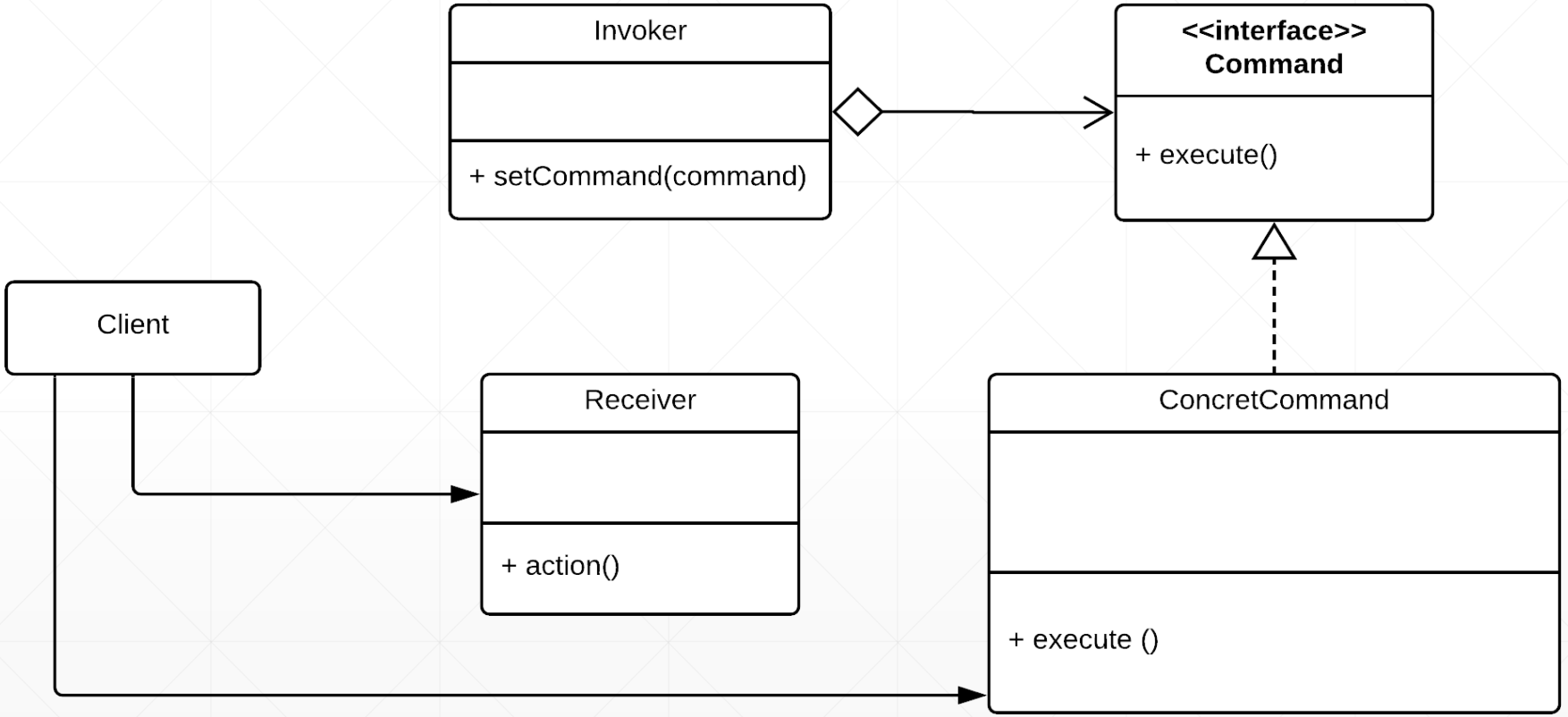
Motivação

A principal motivação do Command é que algumas vezes é necessário emitir solicitações para objetos sem nada saber sobre a operação que está sendo solicitada ou sobre seu receptor. Exemplificando, imagine um portão eletrônico que você queira abrir, não é necessário saber como funciona o mecanismo que faz o portão abrir ou de que o portão é feito, você apenas quer abrir o portão e por isso aperta um botão que faz ele abrir.

Aplicabilidade

- Parametrizar as ações a serem executadas pelos objetos.
- Especificar, enfileirar e executar solicitações em tempos diferentes.
- Registrar e eventualmente desfazer operações realizadas.
- Estruturar um sistema com base em operações de alto nível construídas sobre operações básicas.

Estrutura


Invoker
+ setCommand(command)
<<interface>>
Command
+ execute()
Client
Receiver
+ action()
ConcretCommand
+ execute ()

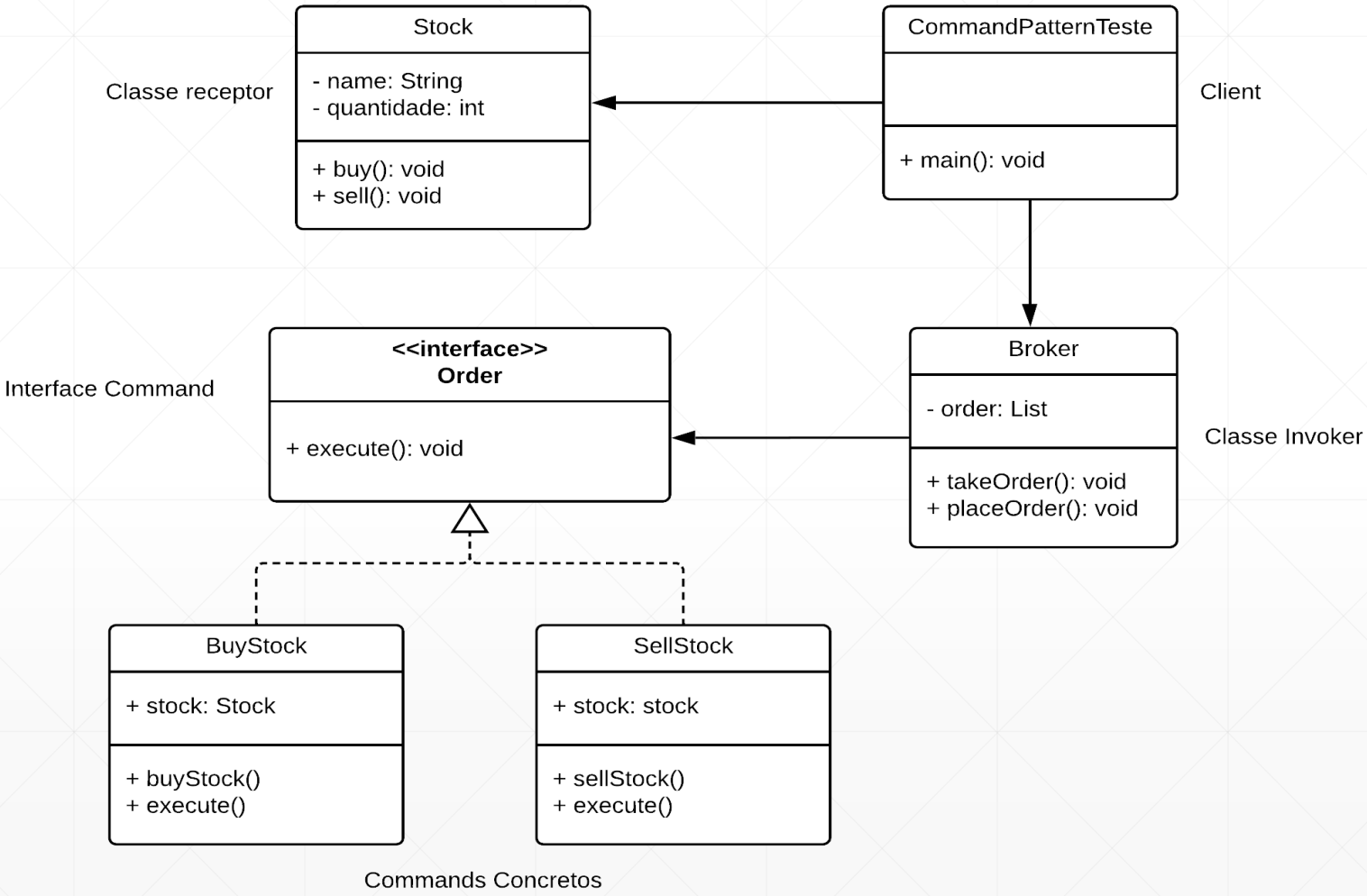
Participantes

- **Command:** Declara uma interface para a execução de uma operação.
- **ConcreteCommand:** Define uma vinculação entre um objeto, um Receiver e uma ação. E também implementa “execute” através da invocação da operação correspondente no Receiver.
- **Client:** Cria um objeto ConcreteCommand e estabelece o seu receptor.
- **Invoker:** Solicita ao Command a execução da solicitação.
- **Receiver:** Sabe como executar as operações associadas a uma solicitação. Qualquer classe pode funcionar como um Receiver.

Consequências

- Desacopla o objeto que invoca a operação do que realizá-la
- Comandos podem ser reunidos para fazer um comando composto
- Facilidade de adicionar novos comandos

Exemplo de uso



Participantes

Stock: Atua como uma solicitação

CommandPatternTeste: Funciona como cliente.

Order: É a interface que que está atuando como comando

BuyStock e SellStock: Classes que implementam a interface Order e funcionam como comandos concretos.

Broker: Funciona como invocador, e serve para identificar qual objeto executará qual comando com base no tipo de comando.


The End