Universidade Federal de Uberlândia
Faculdade de Computação
Programação Orientada a Objetos II
Prof. Fabiano Dorça

Padrões de Projeto
Padrão Chain of Responsibility (Cadeia de Responsabilidade)

Padrão Chain of Responsibility

Objetivo: Evitar o acoplamento do remetente de uma solicitação ao receptor fornecendo uma cadeia de objetos para tratar uma solicitação.

- O objeto que fez a solicitação não tem conhecimento explícito de quem a tratará – essa solicitação é dita ter um receptor implícito.
- Representa um encadeamento de objetos receptores para o processamento de uma série de solicitações diferentes.
- Esses objetos receptores passam a solicitação ao longo da cadeia até que um ou vários objetos a tratem.

Padrão Chain of Responsibility

- Cada objeto receptor possui uma lógica descrevendo os tipos de solicitação que é capaz de processar e como passar adiante aquelas que requeiram processamento por outros receptores.

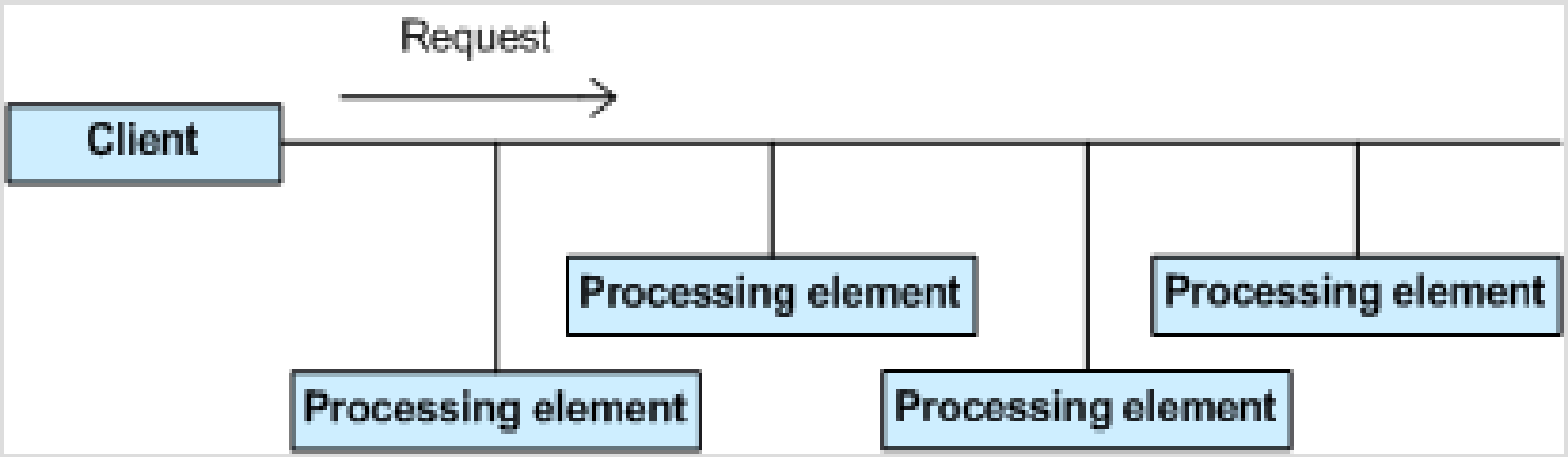
Padrão Chain of Responsibility

- Dessa forma, fornece um acoplamento mais fraco por evitar a associação explícita do remetente da um receptor concreto, permitindo a mais de um objeto a oportunidade de tratar a solicitação.

Padrão Chain of Responsibility

- É montada uma ***lista simplesmente encadeada*** de objetos que podem servir um determinado pedido.
- Em vez de acoplar o cliente a um objeto específico para a execução de um determinado método, o pedido é enviado à cadeia.
- O pedido vai passando pelos objetos até encontrar o mais adequado para satisfazê-lo.
- Cada objeto pode realizar uma parte do serviço e passar o pedido ao objeto seguinte na cadeia (soluções distribuídas).

Padrão Chain of Responsibility



Padrão Chain of Responsibility

- **Quando usar este padrão de projeto??**

  - quando mais de um objeto puder lidar com uma solicitação;
  - quando o manipulador (handler) não é conhecido antecipadamente;
  - quando o grupo de objetos que podem lidar com a solicitação deve ser especificado e modificado de forma dinâmica, em tempo de execução;
  - Quando se deseja evitar o acoplamento do rementente de uma solicitação ao seu destinatário.

Padrão Chain of Responsibility

Vantagens:

- Permite determinar quem será o objeto que irá tratar a requisição dinamicamente, durante a execução;
- A cadeia pode ser configurada em tempo de execução;
- Evita o uso de estruturas condicionais para decidir qual objeto deve tratar a requisição;
- O acoplamento é reduzido, dando ainda flexibilidade adicional na atribuição de responsabilidades a objetos.

Padrão Chain of Responsibility

Participantes:

As classes que participam do padrão são:

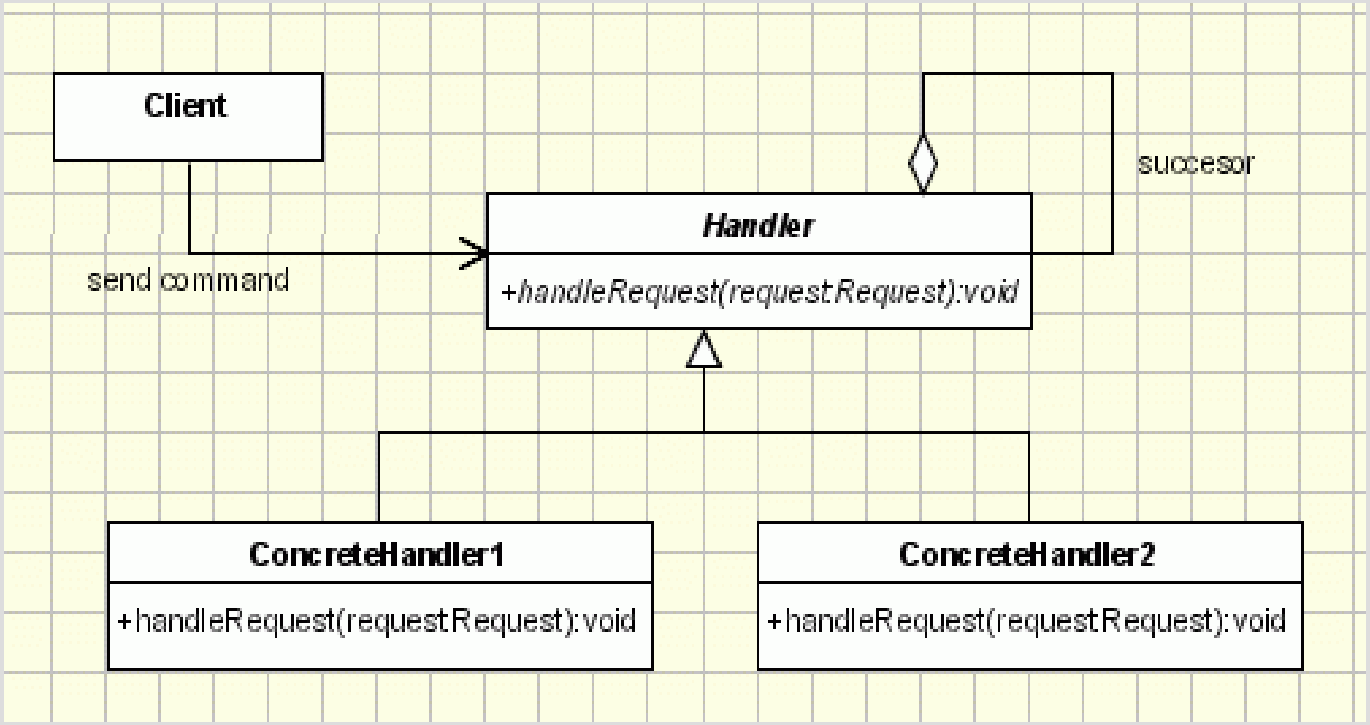
Handler
- Define uma interface para tratar os pedidos
- Implementa a ligação ao sucessor

ConcreteHandler
- Trata os pedidos pelos quais é responsável
- Se o ConcreteHandler pode tratar o pedido, trata-o; caso contrário envia-o ao seu sucessor

Client
- Inicia o pedido a um objeto ConcreteHandler na cadeia.

Padrão Chain of Responsibility

Diagrama de classes:



Padrão Chain of Responsibility

Exemplo:

Utilizando este padrão de projeto, crie um caixa eletrônico capaz manipular requisições de saque corretamente, sempre considerando o **menor número de notas possível.**

Por exemplo, em uma solicitação de saque no valor de R$475, o caixa deve entregar:

4 notas de R$100

1 nota de R$50

1 nota de R$20

1 nota de R$5

Para isto, crie manipuladores para contar a quantidade de cada tipo de nota. Ao terminar sua contagem, cada manipulador deve passar ao próximo o cálculo da quantidade de notas relativas ao montante restante.

Cada manipulador deve exibir a sua contagem.

Padrão Chain of Responsibility

Resolução:

```
//Handler
public abstract class Saque
{
    private Saque sucessor;

    public void setSucessor(Saque sucessor) {
       this.sucessor = sucessor;
    }

    public Saque getSucessor(){
        return this.sucessor;
    }
    public abstract void processaSaque(int valor);
}
```

Padrão Chain of Responsibility

```
//ConcreteHandler
public class Saque100 extends Saque
{
    public  void processaSaque(int valor){
        int notas = valor/100;
        int resto = valor%100;


        if (notas != 0)
            System.out.println("Quantidade de notas de 100: "+notas);


        if (resto != 0 && getSucessor() != null)
            getSucessor().processaSaque(resto);

    }
}
```

Padrão Chain of Responsibility

```
//ConcreteHandler
public class Saque50 extends Saque
{
    public  void processaSaque(int valor){
        int notas = valor/50;
        int resto = valor%50;


        if (notas != 0)
            System.out.println("Quantidade de notas de 50: "+notas);


        if (resto != 0 && getSucessor() != null)
            getSucessor().processaSaque(resto);

    }
}
```

Padrão Chain of Responsibility

```
//ConcreteHandler
public class Saque20 extends Saque
{
    public  void processaSaque(int valor){
        int notas = valor/20;
        int resto = valor%20;


        if (notas != 0)
            System.out.println("Quantidade de notas de 20: "+notas);


        if (resto != 0 && getSucessor() != null)
            getSucessor().processaSaque(resto);

    }


}
```

Padrão Chain of Responsibility

```
//ConcreteHandler
public class Saque10 extends Saque
{
    public  void processaSaque(int valor){
        int notas = valor/10;
        int resto = valor%10;


        if (notas != 0)
            System.out.println("Quantidade de notas de 10:
"+notas);


        if (resto != 0 && getSucessor() != null)
            getSucessor().processaSaque(resto);
    }
}
```

Padrão Chain of Responsibility

```
//ConcreteHandler
public class Saque05 extends Saque
{
    public  void processaSaque(int valor){
        int notas = valor/05;
        int resto = valor%05;


        if (notas != 0)
            System.out.println("Quantidade de notas de 05: "+notas);


        if (resto != 0 && getSucessor() != null)
            System.out.println("Não existem notas de "+resto);

    }
}
```

Padrão Chain of Responsibility

```
//Aplicação cliente
public class CaixaEletronico
{
    public static void main(String args[]){
        //instanciar objetos da cadeia
        Saque saque100 = new Saque100();//inicio da cadeia
        Saque saque50 = new Saque50();
        Saque saque20 = new Saque20();
        Saque saque10 = new Saque10();
        Saque saque05 = new Saque05();//final da cadeia

        //criar cadeia – encadear os objetos da cadeia
        saque100.setSucessor(saque50);
        saque50.setSucessor(saque20);
        saque20.setSucessor(saque10);
        saque10.setSucessor(saque05);
        saque05.setSucessor(null);

        //processar saques
        saque100.processaSaque(575);
        saque100.processaSaque(175);
        saque100.processaSaque(120);
        saque100.processaSaque(155);
        saque100.processaSaque(930);
    }
}
```

Padrão Chain of Responsibility

Vantegens

- Acoplamento reduzido: Não se sabe a classe ou estrutura interna dos participantes.
- Delegação de responsabilidade: Flexível, em tempo de execução.

Padrão Chain of Responsibility

Observação: Com o encadeamento dos *handlers* desta forma (ordenado, do maior para o menor) garantimos que a menor quantidade de notas possível seja fornecida.

Exercícios:

1) Insira handlers concretos para fornecer notas de 2 e 1.

2) Otimize esta solução parametrizando no construtor o valor da nota no *handler* concreto. Desta forma, é necessário apenas 1 *handler* concreto.

Padrão Chain of Responsibility

Aplicabilidade

Utilize Chain of Responsibility quando:

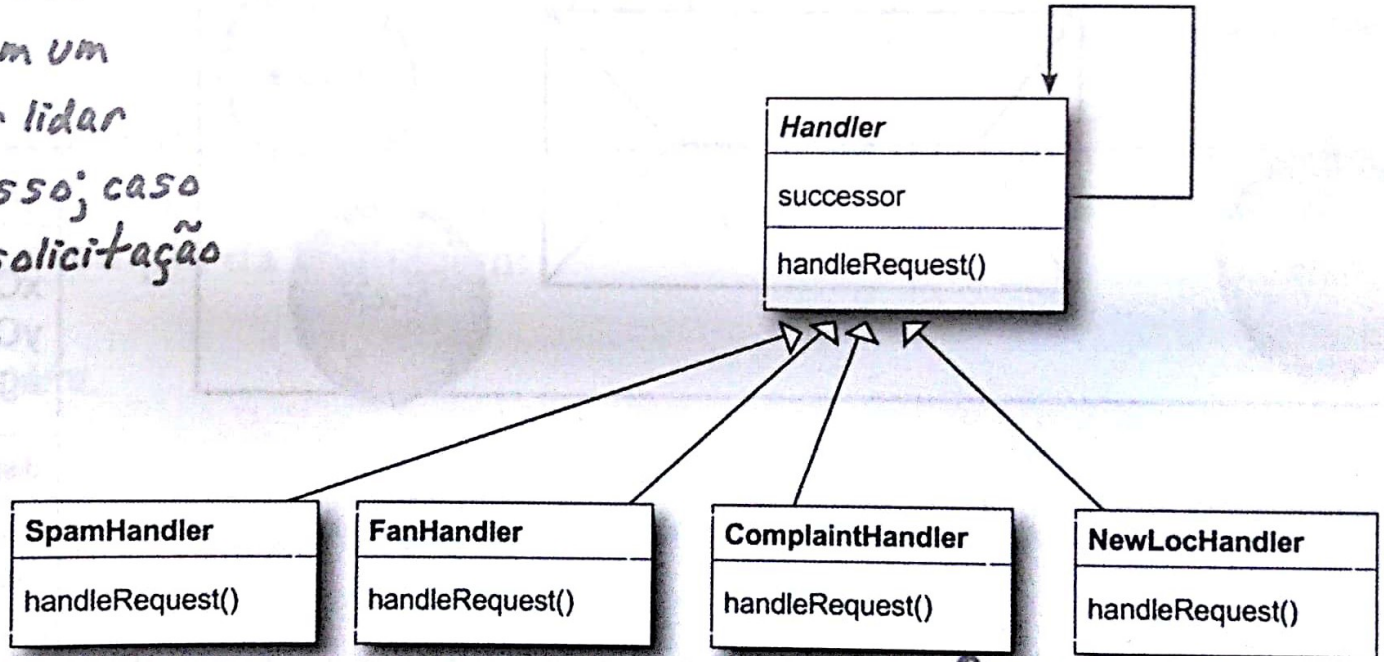
- mais de um objeto pode tratar uma solicitação e o objeto que a tratará não conhecido *a priori*. O objeto que trata a solicitação deve ser escolhido automaticamente;
- você quer emitir uma solicitação para um dentre vários objetos, sem especificar explicitamente o receptor;
- o conjunto de objetos que pode tratar uma solicitação deveria ser especificado dinamicamente.

Gamma et al. Padrões de Projeto – Soluções reutilizáveis de software orientado a objetos. Bookman, 2000.

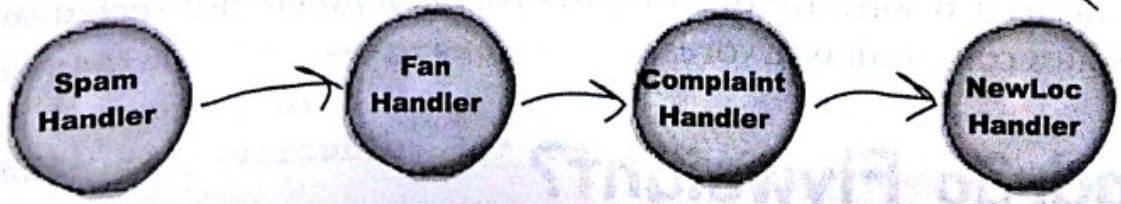
Padrão Chain of Responsibility

- Exemplo Freeman. Padrões de Projeto – Use a cabeça.

Cada objeto na cadeia funciona como um processador e tem um objeto sucessor. Se puder lidar com a solicitação, ele faz isso; caso contrário, ele repassa a solicitação para o seu sucessor.



Cada e-mail é passado para o primeiro processador.



O e-mail que não for processado sairá na outra ponta da cadeia, mas nada impede que você implemente um processador genérico para coletar tudo o que sobrou.

Padrão Chain of Responsibility

- Conclusão…
  - A ideia do padrão é desacoplar remetentes e receptores fornecendo a múltiplos objetos a oportunidade de tratar uma solicitação.
  - A solicitação é passada ao longo de uma cadeia de objetos até que um deles a trate.

Padrão Chain of Responsibility

Observação: Com o encadeamento dos *handlers* desta forma (ordenado, do maior para o menor) garantimos que a menor quantidade de notas possível seja fornecida.

Exercícios:

1) Insira handlers concretos para fornecer notas de 2 e 1.

2) Otimize esta solução parametrizando no construtor o valor da nota no *handler* concreto. Desta forma, é necessário apenas 1 *handler* concreto.

Padrão Chain of Responsibility

- Ok!