Modelos, Métodos e Técnicas de Engenharia de Software
Visão e análise de projeto Padrões Prática 3 – Chain Of Responsability(13)

Prof. Osmar de Oliveira Braz Junior

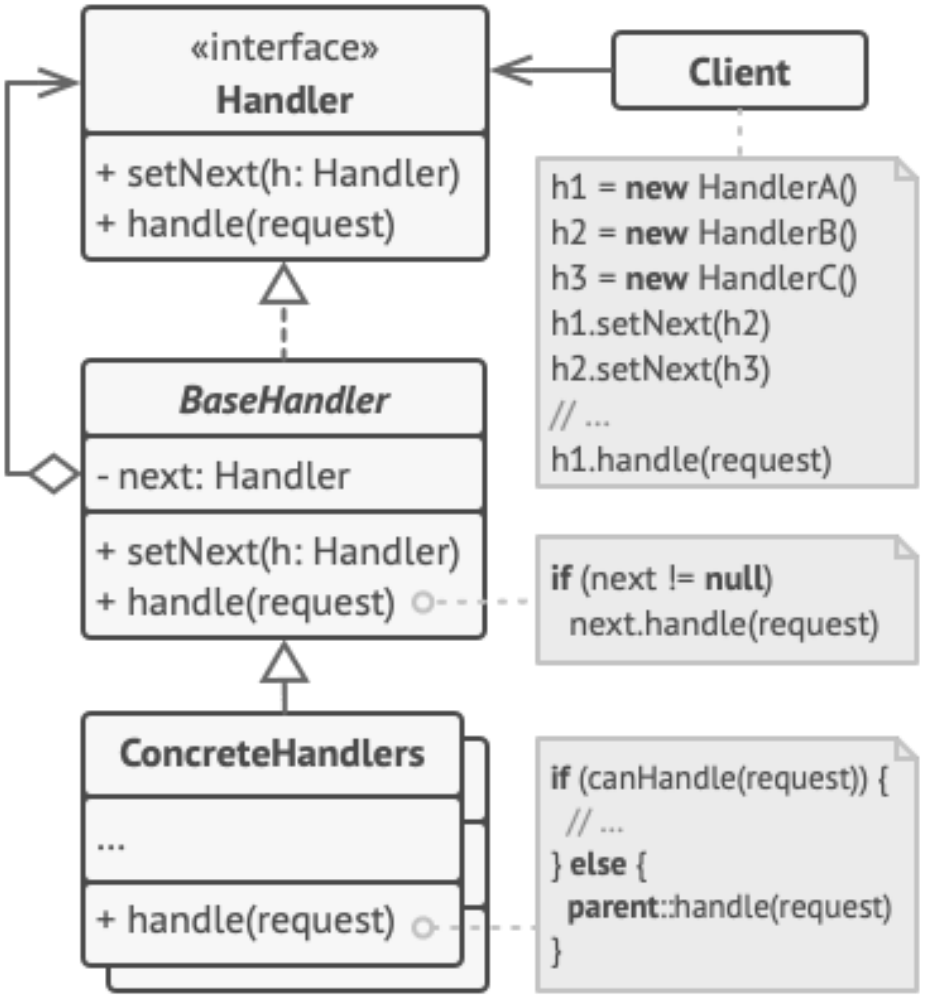
Prof. Richard Henrique de Souza

Objetivos

- Aplicar padrão comportamental ***Chain of Responsability*** em situação problema.

13. Chain of Responsability

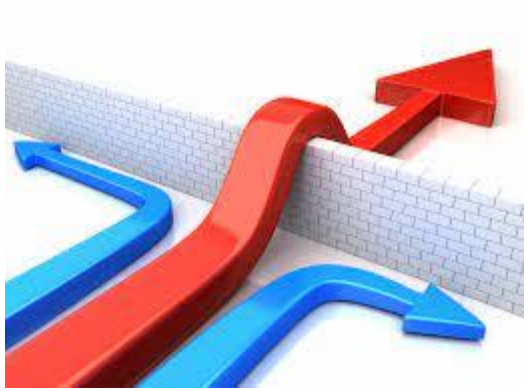
Estrutura:



https://refactoring.guru/pt-br/design-patterns/chain-of-responsibility

Importante

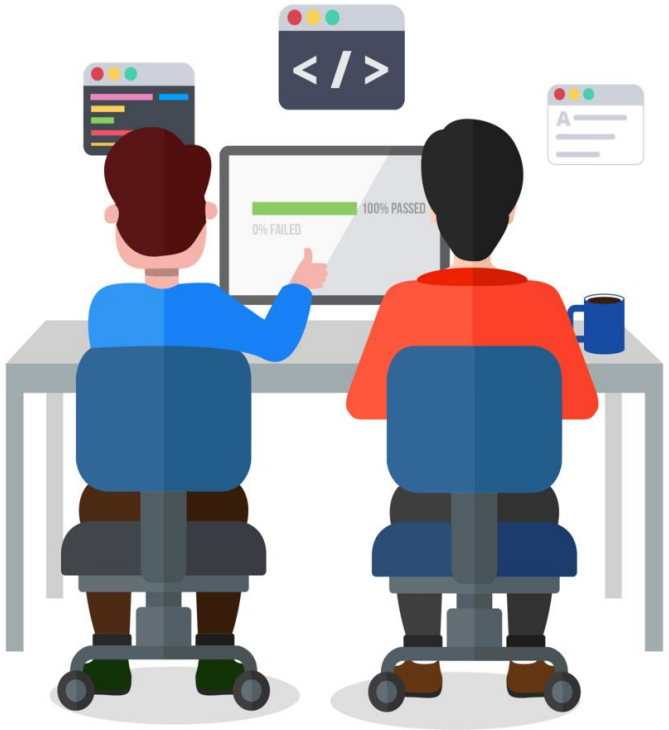
Siga os ROTEIROS !!!



Atividade em Grupo

Para esta atividade crie grupos de 2 alunos, para desenvolver a atividade segundo ***Pair Programming.***

Navegador



Piloto

Extreme Programming, metodologia ágil criada por Kent Beck em 1996.

Pair Programming

- Um é o **piloto**, responsável por escrever o código, o outro o navegador, acompanha a escrita de código e verificar se está de acordo com os **padrões do projeto** e de encontro à solução necessária.
- A intenção desta técnica é **evitar** erros de lógica, e ter um código mais confiável e melhor estruturado, utilizando-se para isso a máxima de que “**duas cabeças pensam melhor do que uma**”.

Preparação do ambiente

- Acesso a ferramenta **draw.io(https://app.diagrams.net/)** para realizar a modelagem.
- Escolha a sua linguagem de programação de preferência
- Escolha uma IDE ou o **git.dev**
- Crie um repositório no github(https://github.com/) para que todos os membros da equipe possam colaborar no desenvolvimento.

Atividade prática

1

13. Chain of Responsibility
Propósito

- Formar uma cadeia de objetos receptores e passar uma requisição pela mesma, dando a chance a mais de um objeto a responder a requisição ou colaborar de alguma forma na resposta.
- Também conhecido como: CoR, Corrente de responsabilidade, Corrente de comando, Chain of command

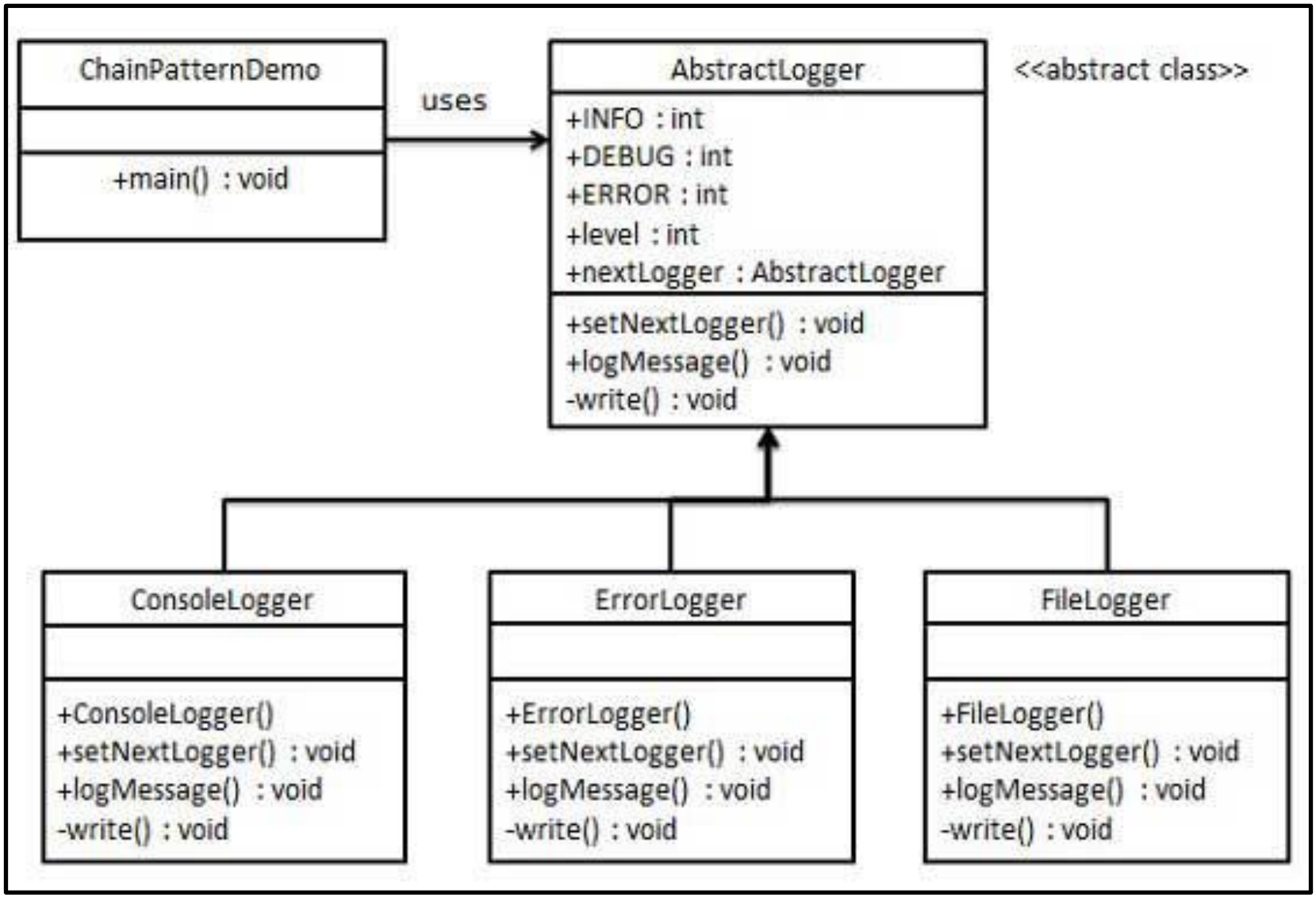
13. Chain of Responsibility

- Usar este padrão quando…
  - Mais de um objeto pode responder a uma requisição e:
    - não se sabe qual a priori;
    - não se quer especificar o receptor explicitamente;
    - estes objetos são especificados dinamicamente.

13. Chain of Responsibility

- Vantagens e desvantagens
  - Acoplamento reduzido:
    - Não se sabe a classe ou estrutura interna dos participantes. Pode usar Mediator para desacoplar ainda mais.
  - Delegação de responsabilidade:
    - Flexível, em tempo de execução.
  - Garantia de resposta:
    - Deve ser uma preocupação do desenvolvedor!
  - Não utilizar identidade de objetos.

13. Chain of Responsibility



13. Chain of Responsibility Passo 1

Crie uma classe abstrata de agente de log.

AbstractLogger.java

```
public abstract class AbstractLogger {
   public static int INFO = 1;
   public static int DEBUG = 2;
   public static int ERROR = 3;

   protected int level;

   //next element in chain or responsibility
   protected AbstractLogger nextLogger;

   public void setNextLogger(AbstractLogger nextLogger){
      this.nextLogger = nextLogger;
   }

   public void logMessage(int level, String message){
      if(this.level <= level){
         write(message);
      }
      if(nextLogger !=null){
         nextLogger.logMessage(level, message);
      }
   }

   abstract protected void write(String message);

}
```

13. Chain of Responsibility
Passo 2

Crie classes concretas estendendo o logger.

ConsoleLogger.java

```
public class ConsoleLogger extends AbstractLogger {

   public ConsoleLogger(int level){
      this.level = level;
   }

   @Override
   protected void write(String message) {
      System.out.println("Standard Console::Logger: " + message);
   }
}
```

13. Chain of Responsibility
Passo 2 - Continuação

ErrorLogger.java

```
public class ErrorLogger extends AbstractLogger {

   public ErrorLogger(int level){
      this.level = level;
   }

   @Override
   protected void write(String message) {
      System.out.println("Error Console::Logger: " + message);
   }
}
```

FileLogger.java

```
public class FileLogger extends AbstractLogger {

   public FileLogger(int level){
      this.level = level;
   }

   @Override
   protected void write(String message) {
      System.out.println("File::Logger: " + message);
   }
}
```

13. Chain of Responsibility
Passo 3

Crie diferentes tipos de registradores. Atribua os níveis de erro e defina o próximo registrador em cada registrador. O próximo registrador em cada registrador representa a parte da cadeia.

13. Chain of Responsibility
Passo 3 - Continuação

ChainPatternDemo.java

```
public class ChainPatternDemo {

   private static AbstractLogger getChainOfLoggers(){

      AbstractLogger errorLogger = new ErrorLogger(AbstractLogger.ERROR);
      AbstractLogger fileLogger = new FileLogger(AbstractLogger.DEBUG);
      AbstractLogger consoleLogger = new ConsoleLogger(AbstractLogger.INFO);

      errorLogger.setNextLogger(fileLogger);
      fileLogger.setNextLogger(consoleLogger);

      return errorLogger;
   }

   public static void main(String[] args) {
      AbstractLogger loggerChain = getChainOfLoggers();

      loggerChain.logMessage(AbstractLogger.INFO,
         "This is an information.");

      loggerChain.logMessage(AbstractLogger.DEBUG,
         "This is an debug level information.");

      loggerChain.logMessage(AbstractLogger.ERROR,
         "This is an error information.");
   }
}
```

13. Chain of Responsibility
Passo 4

- Terminamos
  - Teste sua implementação

Compile e **Mostre** o código para o professor

- Pense, o que você fez aqui ?

Lembre de salvar no seu github

Conclusão

Os padrões comportamentais tem como principal função designar responsabilidades entre objetos.

Referências

- PRESSMAN, Roger; MAXIM, Bruce. Engenharia de software: uma abordagem profissional. 8.ed. Bookman, 2016. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788580555349

- SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011. E-book. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/2613/epub/0

- LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed Porto Alegre: Bookman, 2007. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/books/9788577800476

Fim