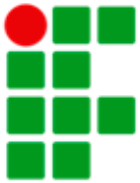


INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Bahia

Linguagem de Programação II

Introdução aos Padrões de Projeto (Design Patterns);

Roteiro

- Introdução aos Padrões de Projeto (Design Patterns):
  - O que são Padrões de Projeto (Design Patterns)?
  - De que consiste um padrão?
  - Como se descobre um padrão?
  - História dos padrões;
  - Por que devo aprender padrões?
  - Críticas ao padrões;
  - Classificação dos padrões;
  - Catálogo de padrões;
  - Definição dos Padrões;
  - Frequência de Uso dos Padrões.

O que são Padrões de Projeto?

São soluções típicas para problemas comuns em projeto de software.

Eles são como plantas de obra pré-fabricadas que você pode customizar para resolver um problema de projeto recorrente em seu código.

O que são Padrões de Projeto?

Você não pode apenas encontrar um padrão e copiá-lo para dentro do seu programa, como você faz com funções e bibliotecas que encontra por aí.

O padrão não é um pedaço de código específico, mas um conceito geral para resolver um problema em particular.

O que são Padrões de Projeto?

Os padrões são frequentemente confundidos com algoritmos.

Um padrão é mais uma descrição de alto nível de uma solução.

O código do mesmo padrão aplicado para dois programas distintos pode ser bem diferente.

De que consiste um padrão?

A maioria dos padrões são descritos formalmente para que as pessoas possam reproduzi-los em diferentes contextos.

Aqui estão algumas seções que são geralmente presentes em uma descrição de um padrão:

De que consiste um padrão?

- O **Propósito** do padrão descreve brevemente o problema e a solução;

- A **Motivação** explica a fundo o problema e a solução que o padrão torna possível;

De que consiste um padrão?

- As **Estruturas** de classes mostram cada parte do padrão e como se relacionam;

- **Exemplos de código** em uma das linguagens de programação populares tornam mais fácil compreender a ideia por trás do padrão.

De que consiste um padrão?

Alguns **catálogos de padrão** listam **outros detalhes** úteis, tais como a **aplicabilidade do padrão**, **etapas de implementação**, e **relações com outros padrões**.

Como se descrobre um padrão?

Quando uma solução é repetida de novo e de novo em vários projetos, alguém vai eventualmente colocar um nome para ela e descrever a solução em detalhes.

É basicamente assim que um padrão é descoberto.

História dos padrões

O conceito de padrões foi primeiramente descrito por Christopher Alexander em **Uma Linguagem de Padrões**. O livro descreve uma "**linguagem**" para o projeto de um ambiente urbano.

As unidades dessa **linguagem** são os **padrões**. Eles podem descrever quão alto as janelas devem estar e etc.

História dos padrões

A ideia foi seguida por quatro autores: **Erich Gamma**, **John Vlissides**, **Ralph Johnson**, e **Richard Helm**. Em 1994, eles publicaram **Padrões de Projeto — Soluções Reutilizáveis de Software Orientado a Objetos**, no qual eles aplicaram o conceito de **padrões de projeto** para **programação**.

História dos padrões

O **livro** mostrava **23 padrões** que **resolviam vários problemas** de **projeto orientado a objetos** e se **tornou um best-seller** rapidamente.

Desde então, dúzias de **outros padrões orientados a objetos** foram **descobertos**. A "**abordagem por padrões**" se tornou muito **popular** em **outros campos** de programação.

Por que devo aprender padrões?

- Eles são um **kit de ferramentas** para **soluções tentadas e testadas** para **problemas comuns** em **projeto de software**. É útil porque eles **ensinam como resolver vários problemas** usando **princípios de projeto orientado a objetos**.

Por que devo aprender padrões?

- Eles **definem** uma **linguagem comum** para se **comunicar** mais **eficientemente**. Você pode dizer, “Oh, é só usar um **Singleton** para isso,” e todo mundo vai entender. **Não é preciso explicar** o que um **singleton é se você conhece** o padrão.

Críticas aos padrões

- **Soluções ineficientes**: os padrões tentam sistematizar abordagens que já são amplamente usadas. Essa unificação é vista por muitos como um dogma e eles implementam os padrões “direto ao ponto”, sem adaptá-los ao contexto de seus projetos.

Críticas aos padrões

- **Uso injustificável**: programadores iniciantes, tendo aprendido sobre eles, tentam aplicá-los em tudo, até mesmo em situações onde um código mais simples seria suficiente.

Classificação dos padrões

Padrões de projeto diferem-se por sua **complexidade**, **nível de detalhes**, e **escala de aplicabilidade** ao **sistema** sendo desenvolvido.

Todos os **padrões** podem ser **categorizados** por seu **propósito**, ou **intenção**. A **seguir** serão apresentados os **três grupos principais de padrões**:

Classificação dos padrões

- Os **padrões criacionais** fornecem mecanismos de criação de objetos que aumentam a flexibilidade e a reutilização de código.

Classificação dos padrões

- Os **padrões estruturais** explicam como montar objetos e classes em estruturas maiores e ainda mantém as estruturas flexíveis e eficientes.

Classificação dos padrões

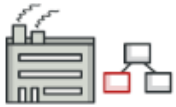
- Os **padrões comportamentais** cuidam da comunicação eficiente e da designação de responsabilidades entre objetos.

Classificação dos padrões

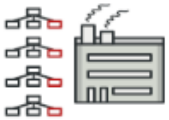
		Propósito		
		De criação	**Estrutural**	**Comportamental**
Escopo	**Classe**	Factory Method	Adapter (class)	Interpreter Template Method
	Objeto	Abstract Method Builder Prototype Singleton	Adapter (object) Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsability Command Iterator Mediator Memento Observer State Strategy Visitor

Classificação por Finalidade

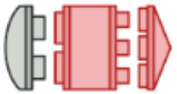
Catálogo de padrões



Factory Method



Abstract Factory



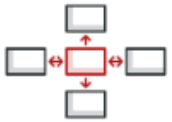
Adapter

Bridge

Chain of Responsibility

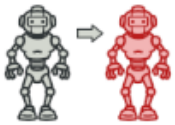
Command

Iterator



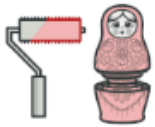
Mediator

Builder

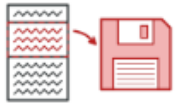


Prototype

Composite

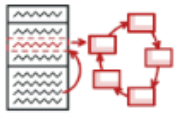


Decorator

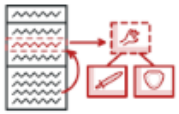


Memento

Observer

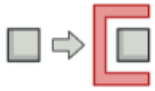


State



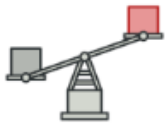
Strategy

Singleton

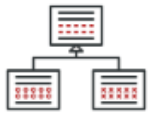


Proxy

Facade



Flyweight



Template Method

Visitor

Analisando um catálogo de padrões

https://refactoring.guru/pt-br/design-patterns/catalog

Definição dos padrões

Creational Patterns (Padrões de Criação):

1. **Abstract Factory:** Fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.
2. **Builder:** Separa a construção de um objeto de sua representação, permitindo a criação de diferentes representações do mesmo objeto.
3. **Factory Method:** Define uma interface para criar um objeto, mas permite que as subclasses decidam qual classe instanciar.
4. **Prototype:** Cria novos objetos a partir de um modelo existente, permitindo a cópia/clonagem de objetos já construídos.
5. **Singleton:** Garante a existência de apenas uma única instância de uma classe e fornece um ponto global de acesso a essa instância.

Definição dos padrões

Structural Patterns (Padrões Estruturais):

1. **Adapter:** Permite que objetos com interfaces incompatíveis trabalhem juntos, convertendo a interface de um objeto para outra que seja esperada pelos clientes.
2. **Bridge:** Separa a abstração da sua implementação, permitindo que ambas variem independentemente.
3. **Composite:** Agrupa objetos em estruturas de árvore para representar hierarquias parte-todo. Permite tratar objetos individuais e composições de objetos de maneira uniforme.
4. **Decorator:** Adiciona comportamentos adicionais a objetos dinamicamente, sem alterar a estrutura do objeto em si.
5. **Facade:** Fornecer uma interface unificada para um conjunto de interfaces em um subsistema, facilitando o uso desse subsistema.
6. **Flyweight:** Utiliza o compartilhamento para suportar um grande número de objetos granulares de forma eficiente.
7. **Proxy:** Fornece um substituto ou representante para outro objeto para controlar o acesso a ele.

Definição dos padrões

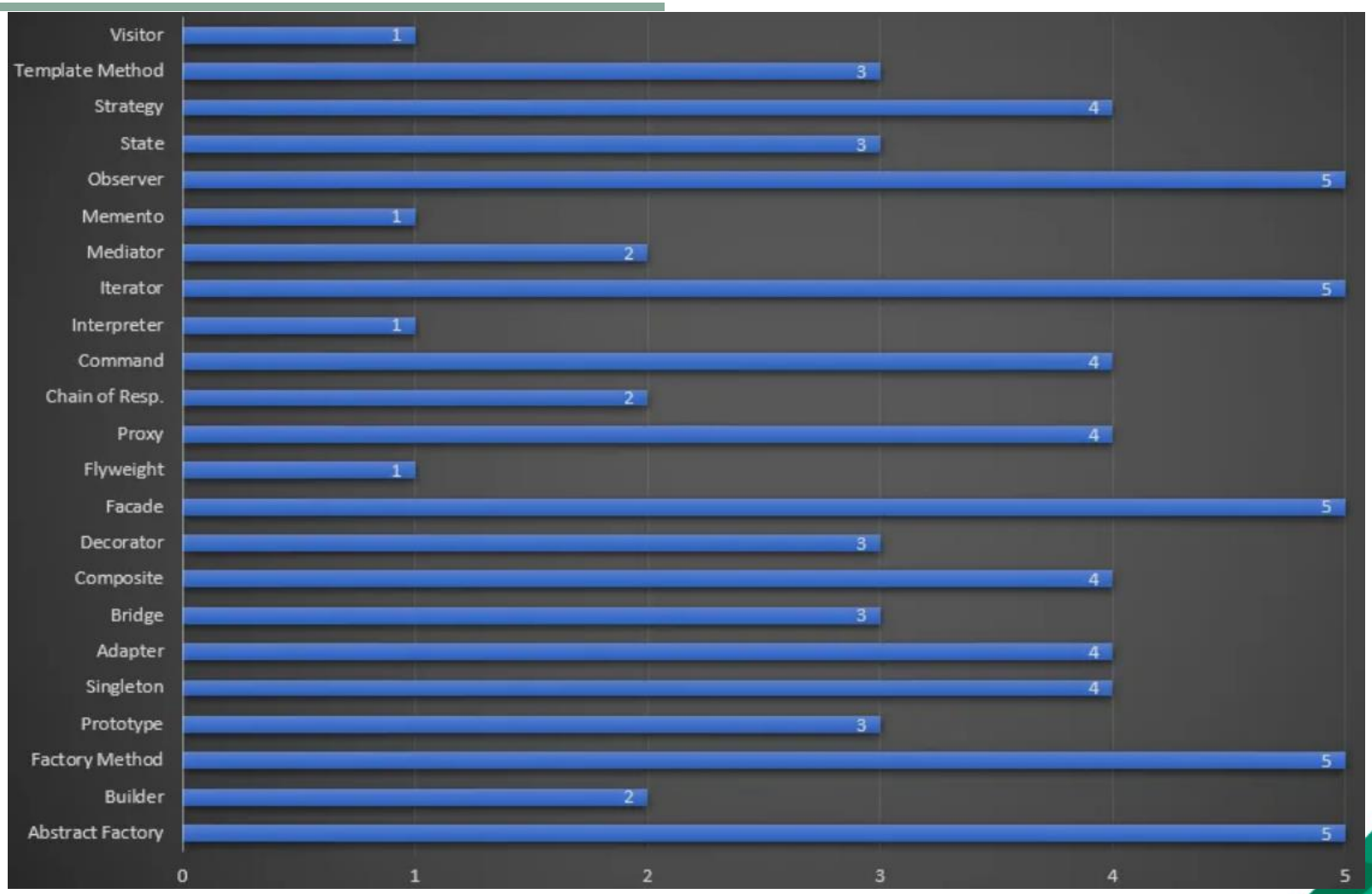
Behavioral Patterns (Padrões Comportamentais):

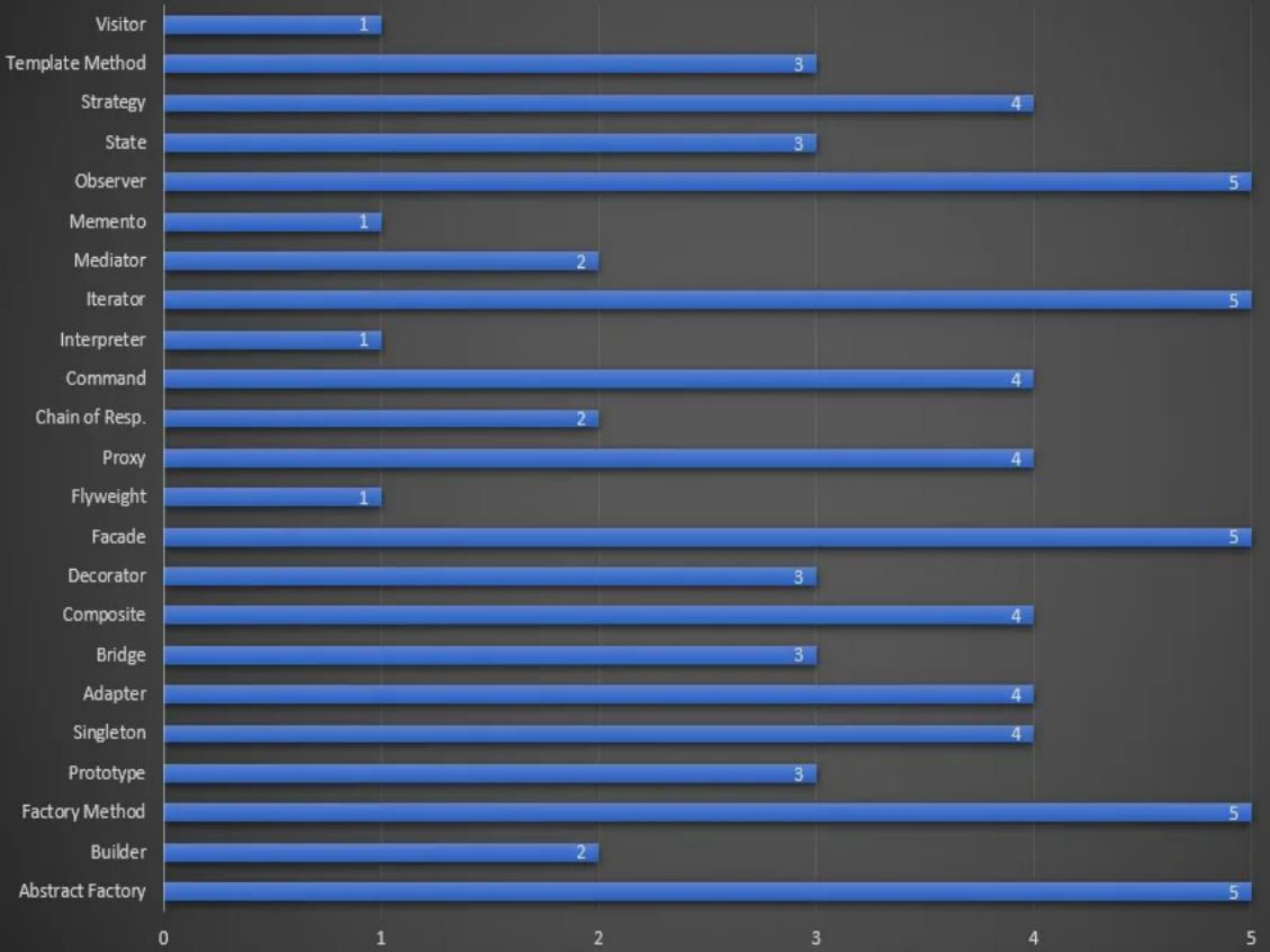
1. **Chain of Responsibility:** Permite passar solicitações por uma cadeia de objetos receptores, dando a chance a múltiplos objetos de lidar com a solicitação.
2. **Command:** Encapsula uma solicitação como um objeto, permitindo parametrizar clientes com solicitações, fila de solicitações e operações assíncronas.
3. **Interpreter:** Define uma representação gramatical para um idioma e fornece um interpretador para interpretar sentenças dessa linguagem.
4. **Iterator:** Permite acessar sequencialmente os elementos de uma coleção sem expor sua representação subjacente.
5. **Mediator:** Define um objeto que encapsula como um conjunto de objetos interage, promovendo um acoplamento fraco entre os objetos.
6. **Memento:** Captura e restaura o estado interno de um objeto, permitindo retornar ao estado anterior do objeto.

Definição dos padrões

7. **Observer:** Define um relacionamento um-para-muitos entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente.
8. **State:** Permite que um objeto altere seu comportamento quando seu estado interno muda.
9. **Strategy:** Define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis permitindo que o algoritmo varie independentemente dos clientes que o utilizam.
10. **Template Method:** Define o esqueleto de um algoritmo em uma operação, deixando alguns passos para serem preenchidos por subclasses.
11. **Visitor:** Permite adicionar novas operações a uma estrutura de objetos existente sem modificar os objetos em si.

Frequência de Uso




Visitor 1
Template Method 3
Strategy 4
State 3
Observer 5
Memento 1
Mediator 2
Iterator 5
Interpreter 1
Command 4
Chain of Resp. 2
Proxy 4
Flyweight 1
Facade 5
Decorator 3
Composite 4
Bridge 3
Adapter 4
Singleton 4
Prototype 3
Factory Method 5
Builder 2
Abstract Factory 5
0 1 2 3 4 5

Exercícios – Pesquise e Responda

Exercícios:

1) O que são Padrões de Projeto?
2) De que consiste um padrão?
3) Como se descobre um padrão?
4) Por que se deve aprender padrões?
5) Em quais tipos se classificam os padrões?
6) Cite e defina os 23 padrões separando-os (agrupando-os) em suas categorias (tipos).

Referências

- VALENTE, Marco Tulio. Engenharia de software moderna. Princípios e Práticas para Desenvolvimento de Software com Produtividade, v. 1, 2020.

- https://www.alura.com.br/artigos/design-patterns-introducao-padroes-projeto

- https://refactoring.guru/pt-br/design-patterns

Obrigado!

- Canais de Comunicação;
- Horário de Atendimento.