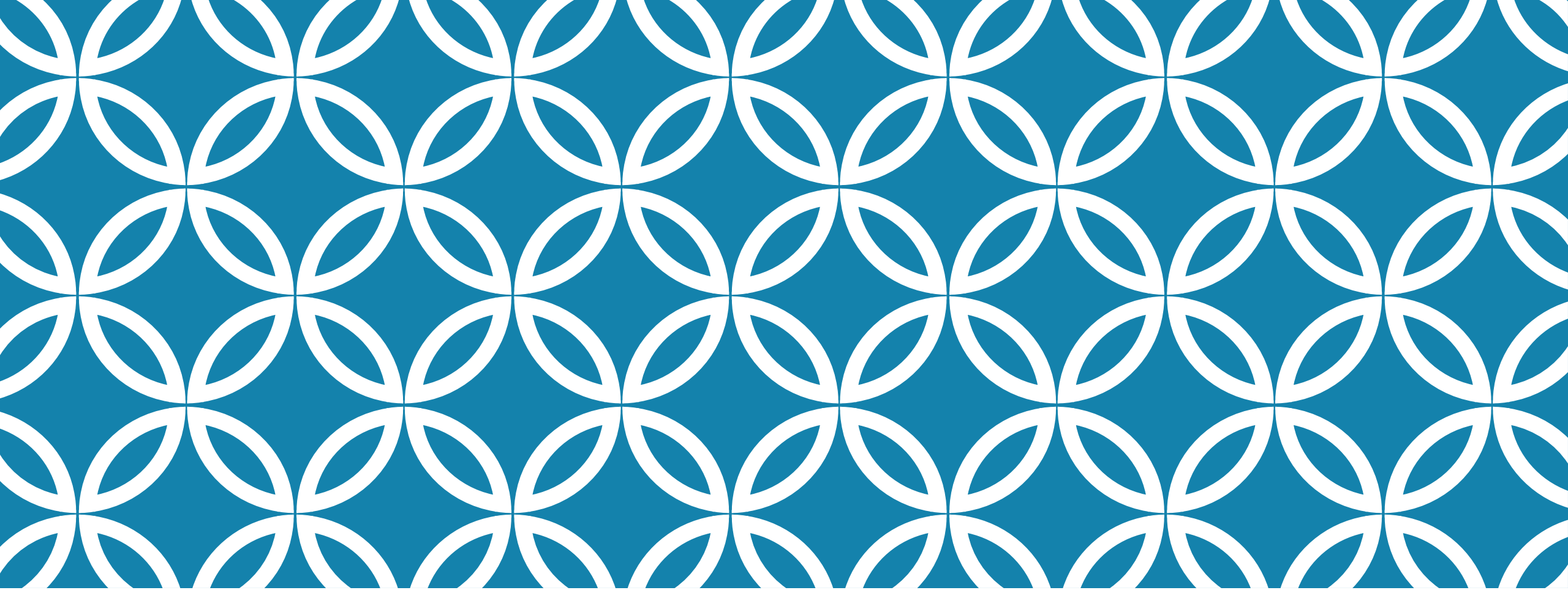

MÉTODOS AVANÇADOS DE PROGRAMAÇÃO

Padrões de Projeto

Drª. Alana Morais

O QUE É UM PADRÃO DE PROJETO?

O QUE É UM PADRÃO DE PROJETO?

Diretrizes que descrevem problemas recorrentes no projeto de sistemas e sua solução em termos de interfaces e objetos

- Nome, problema, solução, consequências…

- É reusar projetos e arquiteturas de sucesso, ou seja, técnicas comprovadas, em forma de um catálogo, num formato consistente e acessível para projetistas;

O QUE É UM PADRÃO DE PROJETO?

Maneira testada e documentada de alcançar um objetivo qualquer

- Padrões são comuns em várias áreas da engenharia

Design Patterns, ou Padrões de Projeto*

- Padrões para alcançar objetivos na engenharia de software usando classes e métodos em linguagens orientadas a objeto
- Inspirado em "*A Pattern Language*" de Christopher Alexander, sobre padrões de arquitetura de cidades, casas e prédios

RESPONSABILIDADES

.Booch e Rumbaugh

. “Responsabilidade é um contrato ou obrigação de um tipo ou classe.”

.Dois tipos de responsabilidades dos objetos:

. De conhecimento (*knowing*)

- Sobre dados privativos e encapsulados; sobre objetos relacionados; sobre coisas que pode calcular ou derivar.
- Estão relacionadas à distribuição das características do sistema entre as classes

. De realização (*doing*)

- Fazer alguma coisa em si mesmo; iniciar uma ação em outro objeto; controlar e coordenar atividades em outros objetos.
- Estão relacionadas com a distribuição do comportamento do sistema entre as classes

RESPONSABILIDADES

Responsabilidades são atribuídas aos objetos durante o planejamento

RESPONSABILIDADE E MÉTODOS

A tradução de responsabilidades em classes e métodos depende da granularidade da responsabilidade

Métodos são implementados para cumprir responsabilidades

- Uma responsabilidade pode ser cumprida por um único método ou uma coleção de métodos trabalhando em conjunto

Responsabilidades do tipo *knowing* geralmente são inferidas a partir do modelo conceitual (são os atributos e relacionamentos)

PADRÃO DE PROJETO

Padrões de Projeto são um repertório de soluções e princípios que ajudam os desenvolvedores a criar software e que são codificados em um formato estruturado consistindo de

- Nome
- Problema que soluciona
- Solução do problema
- Consequências

O objetivo dos padrões é codificar conhecimento (*knowing*) existente de uma forma que possa ser reaplicado em contextos diferentes

E OS BENEFÍCIO DE UTILIZAR OS PADRÕES DE PROJETO?

E OS BENEFÍCIO DE UTILIZAR OS PADRÕES DE PROJETO?

Padrões capturam a estrutura estática e a colaboração dinâmica entre objetos participantes no projeto de sistemas

São especialmente bons para descrever como e por que resolver problemas não funcionais

Facilitam o reuso de soluções arquiteturais que deram certo antes

Aumentam a coesão, diminuem o acoplamento

QUAIS OS GRUPOS DE PADRÕES MAIS FAMOSOS?

PADRÕES MAIS CONHECIDOS

- Padrões de **Larman** = GRASP
  - General Responsability Assignment Software Patterns
    - Information Expert.
    - Creator.
    - Fraco acoplamento.
    - Alta coesão.
    - Controller.

- Padrões **GOF**
  - **Gang of Four** descreve 23 padrões que não são os únicos mas são os mais utilizados.

INTRODUÇÃO

"Cada padrão descreve um problema que ocorre repetidas vezes em nosso ambiente, e então descreve o núcleo da solução para aquele problema, de tal maneira que pode-se usar essa solução milhões de vezes sem nunca fazê-la da mesma forma duas vezes”

Christopher Alexander, sobre padrões na arquitetura e engenharia civil

INTRODUÇÃO

Na Engenharia de Software, quatro autores (*Gang of Four - GoF*) se basearam em Christopher Alexander para criar Padrões de Projeto de *software*.

Em 1994 descreveram 23 padrões em seu livro

- Hoje ele já está na quadragésima edição
- Mais de 500 mil cópias vendidas, traduzido para 13 línguas

INTRODUÇÃO

"Descrição de uma solução para resolver um problema genérico de projeto em um contexto específico. [...] Um padrão de projeto dá nome, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la reutilizável"

Erich Gamma, et. al, sobre padrões de projeto de software

INTRODUÇÃO
Benefícios

- Padrões capturam a estrutura estática e a colaboração dinâmica entre objetos participantes no projeto de sistemas
- São especialmente bons para descrever como e por que resolver problemas não-funcionais
- Facilitam o reuso de soluções arquiteturais que deram certo antes
- Aumentam a coesão, diminuem o acoplamento

INTRODUÇÃO

Apresentar cada um dos 23 padrões do catálogo do GoF descrevendo:

NOME

Um identificador utilizado para resumir

- O problema em questão
- Suas soluções
- Suas consequências

Aumenta o vocabulário e melhora a comunicação

“A parte mais difícil de programação é dar bons nomes às variáveis”

NOME

O nome permite projetar em um nível mais alto de abstração

Conversa entre desenvolvedores:

"Cara, acho melhor usar o Template Method aqui!!!"

"Não sei não... um Strategy funcionaria melhor!!!"

"Maria, coloca um Observer aí que resolve!!!"

NOME

Permite documentar código

- Evita longas descrições.

```
/** Nesta classe implementa-se o
 * padrão Singleton...
 * @author Hyggo
 * @version 1.0
 */
public class Calendar{
   …
```

```
/** Nesta classe utiliza-se um construtor
 * privado, com um método estático que
 * retorna a única instância desta classe,
 * sincronizado para evitar que outra
 * instância seja recuperada por outra
 * linha de execução...
 * @author Hyggo
 * @version 1.0
 */
public class Calendar{
   …
```

PROBLEMA

Descreve quando aplicar o padrão

- Em que situações o padrão pode ser aplicado?
- Em que situações o padrão traz flexibilidade/ elegância ao projeto?
- Quando não utilizá-lo?

Explica o problema e seu contexto

Pode conter uma lista de pré-condições presentes antes de levar em consideração a aplicação do padrão

PROBLEMA

Geralmente o problema tem exemplos específicos de aplicação

- Como definir uma instância única de uma classe???
- Como representar algoritmos como objetos???
- Como implementar hierarquias parte-todo???

SOLUÇÃO

Solução não descreve uma implementação concreta, apenas um modelo genérico para que possa ser reutilizado

- Dependendo do seu problema, deverá ser adaptado
- Tem-se um exemplo de solução, mas apenas para guiar o reuso

Descreve os elementos que compõem o projeto da solução, suas responsabilidades e colaborações

- Modelo conceitual
- Diagrama de classes
- Diagramas de interação

SOLUÇÃO

- Descrição abstrata de como o padrão resolve o problema em questão
- Descreve os elementos que compõem
  - Relacionamentos
  - Responsabilidades
  - Colaborações
- Inclui algum exemplo concreto de implementação
  - Porém o padrão deve ser adaptado ao seu contexto específico

CONSEQUÊNCIAS

- Vantagens e desvantagens de aplicar o padrão
- Esta seção serve para
  - Avaliar várias alternativas de padrões
  - Entender os custos e desafios
  - Entender os benefícios de aplicar o padrão
- Inclui análise de impacto envolvendo
  - Flexibilidade
  - Extensibilidade
  - Portabilidade

PADRÕES MAIS CONHECIDOS

- Padrões de **Larman** = GRASP
  - General Responsability Assignment Software Patterns
    - Information Expert.
    - Creator.
    - Fraco acoplamento.
    - Alta coesão.
    - Controller.

- Padrões **GOF**
  - **Gang of Four** descreve 23 padrões que não são os únicos mas são os mais utilizados.

GRASP – GENERAL RESPONSABILITY ASSIGNMENT SOFTWARE PATTERNS

- Os padrões GRASP descrevem os princípios fundamentais para a atribuição de responsabilidades em projetos OO
- Responsabilidades:
  - Fazer algo
    - a si mesmo
    - a outros objetos
  - Conhecer/lembrar de algo
    - dados encapsulados
      objetos relacionados
- Responsabilidade != método
  - Métodos implementam responsabilidades
    - Objetos colaboram para cumprir responsabilidades

GRASP – GENERAL RESPONSABILITY ASSIGNMENT SOFTWARE PATTERNS

- Discutiremos 4/5 dos 9 padrões GRASP:
  - Information Expert
    - Quem é o especialista da informação?
  - Creator
    - Quem deve criar uma determinada instância?
  - Baixo acoplamento (Low Coupling)
    - Como diminuir o acoplamento entre as classes?
  - Alta coesão (High cohesion)
    - Como aumentar a coesão no sistema?

- Os outro são Controller, Polymorphism, Pure Fabrication, Indirection e Protected Variations

PADRÕES GOF CLASSIFICAÇÃO

- *Várias formas de classificar os padrões. Gamma et al.[2] os classifica de duas formas:*
  - *Por propósito:*
    1) *criação de classes e objetos* - Padrões de Criação,
    2) *alteração da estrutura de um programa* - Padrões Estruturais,
    3) *controle do seu comportamento* - Padrões Comportamentais
  - Por escopo: classe ou objeto
- *Metsker [1] os classifica em 5 grupos, por intenção (problema a ser solucionado):*
  - *oferecer uma interface,*
  - *atribuir uma responsabilidade,*
  - *realizar a construção de classes ou objetos*
  - *controlar formas de operação*
  - *implementar uma extensão para a aplicação*

PADRÕES GOF CLASSIFICAÇÃO– *METSKER*

Intenção	Padrões
1. Interfaces	Adapter, Facade, Composite, Bridge
2. Responsabilidade	Singleton, Observer, Mediator, Proxy, Chain of Responsibility, Flyweight
3. Construção	Builder, Factory Method, Abstract Factory, Prototype, Memento
4. Operações	Template Method, State, Strategy, Command, Interpreter
5. Extensões	Decorator, Iterator, Visitor

PADRÕES GOF
CLASSIFICAÇÃO- *GAMMA ET AL.*

		Propósito		
		1. Criação	2. Estrutura	3. Comportamento
Escopo	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

PADRÕES GOF CLASSIFICAÇÃO

- Podem ser classificados por propósito:
  - Padrões de Criação
    - Abstraem o processo de criação de objetos a partir da instanciação de classes
  - Padrões Estruturais
    - Tratam da forma como classes e objetos estão organizados para formar estruturas maiores
  - Padrões Comportamentais
    - Preocupam-se com algoritmos e responsabilidades dos objetos

PADRÕES GOF
CLASSIFICAÇÃO- *GAMMA ET AL.*

		Propósito		
		1. Criação	2. Estrutura	3. Comportamento
Escopo	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

PADRÕES GOF CLASSIFICAÇÃO

- Podem ser subclassificados por escopo
  - Padrões de Classe
    - Tratam de relações entre classes e subclasses (herança)
    - São estáticos, definidos em tempo de compilação
  - Padrões de Objeto
    - Tratam das relações entre objetos, que podem mudar em tempo de execução

DÚVIDAS