Padrões de Projeto

Prof. Rafael Milbradt

Contato: rmilbradt@gmail.com

Ementa

UNIDADE 2 – PADRÕES DE PROJETO

- 2.1 – Introdução.
- 2.2 - Tipos de padrões (análise, projeto, banco de dados, programação, entre outros).
- 2.3 - Padrões de projeto de criação.
- 2.4 – Padrões de projeto de estruturais.
- 2.5 – Padrões de projeto de comportamentais.

Bibliografia

BIBLIOGRAFIA BÁSICA

- GAMMA, E. **Padrões de Projeto: soluções reutilizáveis de software orientado a objetos.** Porto Alegre: Bookman, 2007**.**
- PREISS, B**. Estruturas de dados e algoritmos : padroes de projetos orientados a objetos com Java.** Rio de Janeiro: Elsevier, 2001.
- COIMBRA, E.; GUIZZO, G.; LAMB, J. R. **Padrões de Projeto Em Aplicações Web**. Ed. Visual Books, 2013.

BIBLIOGRAFIA COMPLEMENTAR

- BLOCH, J. **Effective Java.** Addison-Wesley, 2008.
- SANDERS, William. **Aprendendo padrões de projeto em PHP**. São Paulo: Novatec, 2013.
- MAHEMOFF, Michael. **Padrões de projetos Ajax.** Rio de Janeiro: AltaBooks, 2007.
- ALUR, D. **Core J2EE Patterns: Best Practices and Design Strategies.** Prentice Hall: 2ª ed., 2003.
- PREE, W. **Design patterns for object-oriented software development**. Addison-Wesley, 1995.
- Apostilas CAELUM;

Introdução

- O uso da Orientação a Objetos na programação trouxe grandes benefícios aos desenvolvedores de software, favorecendo o desenvolvimento de software reaproveitável através de algumas características como:
  - Encapsulamento;
  - Modularidade;
  - Polimorfismo;
- O Software reaproveitável, por sua vez, tem suas vantagens:
  - Mais barato para ser mantido;
  - Pode evoluir mais facilmente;
  - Mais seguro;
  - Mais estável;

Introdução

- Porém o uso do paradigma OO na sua plenitude, é tarefa para os mais experientes apenas:
  - Existem armadilhas como a própria herança, como já visto;
- Analogias com outras ciências:
  - Arquitetura de hardware e de software;
  - Linha de montagem: fábrica de software;
  - Padrões de projeto da engenharia;
- Um engenheiro mecânico quando vai projetar uma nova máquina não reinventa componentes já existentes:
  - Mancais, rolamentos, eixos, engrenagens helicoidais, engrenagens cônicas, etc.
  - Ele apenas monta o seu projeto de forma a integrar os componentes certos, para que a nova máquina cumpra a função para que foi designada;
  - Os componentes são padronizados e fornecidos por diferentes empresas;
    - **São por sua vez, componente reaproveitáveis;**

Introdução

- Assim como num projeto de engenharia:
  - É necessário transmitir um movimento entre dois eixos interceptantes;
    - Padrão de Projeto: engrenagens cônicas.
- A ideia por detrás dos padrões de projeto OO é exatamente a mesma;
  - Padronizar problemas comuns, com as melhores soluções (que favoreçam a reusabilidade);
  - Problemas de OO: criação de objetos, comportamento dos objetos, entre outros...

Introdução

- Um padrão de projeto é uma solução consolidada para um problema recorrente no desenvolvimento e manutenção de software orientado a objetos.
- A referência mais importante relacionada a padrões de projeto é o livro "*Design Patterns: Elements of Reusable Object-Oriented Software*" (editora Addison-Wesley, 1995) dos autores **Erich Gamma**, **Richard Helm**, **Ralph Johnson** e **John Vlissides**.
  - Esses quatro autores são conhecidos como "**Gang of Four**"(**GoF**).
- Vamos conhecer os padrões de projeto seguindo a divisão em categorias propostas pelo GoF:
  - Padrões de criação;
  - Padrões estruturais;
  - Padrões comportamentais;
- Antes destes, vamos conhecer formalmente os padrões básicos...

Padrões de Projeto Básicos

- São os padrões fundamentais, usados também no desenvolvimento dos padrões GoF;
- Provavelmente já conhecidos, porém vamos formalizar a apresentação destes;

Padrões de Projeto Básicos

- **Interface:**
  - Faz parte do próprio conceito de orientação a objetos em muitas linguagens/plataformas;
    - Pode ser usado para implementar um conjunto de classes de provedores de serviços, que oferecem o mesmo serviço para que um objeto cliente possa utilizá-los sem a necessidade de alteração do código.
    - É como um contrato de prestação de serviço. Para o cliente não importa quem presta o serviço e sim qual o contrato que o prestador de serviços segue.

Padrões de Projeto Básicos

- **Abstract Parent class:**
  - Útil para projetar uma estrutura para a implementação consistente de funcionalidades comuns a um conjunto de classes relacionadas.
  - Conceito também já abordado dentro da própria orientação a objetos.

Padrões de Projeto Básicos

- **Private Methods**:
  - Fornecer uma maneira de projetar um comportamento de classe para que os objetos externos não tenham permissão para acessar o comportamento que se destina apenas para o uso interno.
  - É a própria definição do encapsulamento...

Padrões de Projeto Básicos

- **Accessor Methods**:
  - Fornecer uma maneira de acessar o estado de um objeto usando métodos específicos. Essa abordagem desencoraja diferentes objetos cliente de acessar diretamente os atributos de um objeto, resultando em uma estrutura de classe mais sustentável (manutenção mais fácil).
  - Getters e Setters são construídos conforme este padrão;

Padrões de Projeto Básicos

- **Constant Data Manager:**
  - Útil para projetar um repositório centralizado e fácil de manter para os dados constantes em um aplicativo.
  - Em vez de permitir que os dados constantes estejam presentes em objetos diferentes, o padrão "Constant Data Manager" recomenda que todos esses dados, considerados constantes em um aplicativo, sejam mantidos em um objeto separado e acessados por outros objetos no aplicativo.
  - Esse tipo de separação fornece um repositório centralizado e fácil de manter para os dados constantes em um aplicativo.

Padrões de Projeto Básicos

- **Immutable Object:**
  - Usado para garantir que o estado de um objeto não pode ser alterado.
    - Pode ser usado para garantir que o acesso simultâneo a um objeto de dados por vários objetos de cliente não resulte em condições de corrida.
    - Evitar efeitos colaterais, às vezes indesejados;

Padrões de Projeto Básicos

- **Exercício 1:**
  - Refaça o que for necessário da implementação do Exercício 1 do módulo de Introdução, porém:
    - Demonstre a aplicação de pelo menos 4 dos padrões de projeto básicos;

Padrões de criação

- Trata-se de uma das tarefas mais comumente realizadas em um aplicativo OO: **a criação de objetos**.
- Visa dar suporte a um mecanismo uniforme, simples e controlado para criar objetos.
  - Permitir o encapsulamento dos detalhes sobre quais classes são instanciadas e como essas instâncias são criadas.
  - Incentivar o uso de interfaces, o que reduz o acoplamento.
- Os padrões de projeto de criação são formas alternativas para a construção de objetos. Quando estes padrões são adotados, deve-se restringir o acesso às formas de criação padrão (construtores).

Padrões de criação

- **Factory Method**:
  - Objetivo: Encapsular a escolha da classe concreta a ser utilizada na criação de objetos de um determinado tipo.
  - Às vezes, um objeto só pode saber que ele precisa acessar uma classe de dentro de uma hierarquia de classe, mas não sabe exatamente qual classe entre o conjunto de subclasses da classe pai deve ser selecionada.
  - A escolha de uma classe apropriada pode depender de fatores como:
    - O estado da aplicação em execução;
    - Configurações do aplicativo;
    - Expansão de requisitos ou melhorias;

Padrões de criação

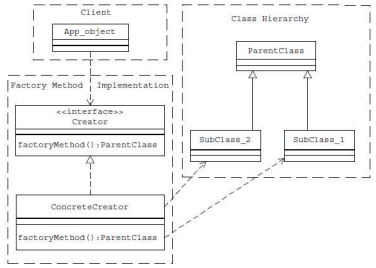
- **Factory Method**:
  - Nesses casos, o objeto precisa implementar critérios para instanciar a classe certa da hierarquia;
  - Este tipo de projeto tem as seguintes desvantagens:
    - Como cada objeto precisa implementar os critérios de seleção de classe, isto resulta em um alto grau de acoplamento entre um objeto cliente e a hierarquia de classes do provedor de serviços;
    - Sempre que os critérios de seleção de classe mudam, cada objeto de aplicativo que usa a hierarquia de classe deve sofrer uma alteração correspondente.
      - Imagine que um provedor de serviços pode ter inúmeros clientes;
    - Como os critérios de seleção de classe precisam levar em conta todos os fatores que podem afetar o processo de seleção, a implementação de um objeto de aplicativo pode conter declarações condicionais inelegantes.
      - O famoso código flecha.
      - Porém, não há nada mais inelegante que o alto acoplamento!

Padrões de criação

- **Factory Method**:
  - Nesses casos, o padrão Factory Method sugere encapsular a seleção da classe apropriada, dentro de um método designado como “método fábrica”.
    - Seleciona uma classe apropriada de uma hierarquia de classes com base no contexto do aplicativo e outros fatores de influência;
    - Instancia a classe selecionada e retorna-a como uma instância do tipo da classe pai;
  - Vantagens:
    - Reduz o acoplamento;
    - Pode ser reaproveitado por diversas classes clientes;
    - Ao precisar alterar a hierarquia de classes, apenas o Factory Method precisa ser alterado;

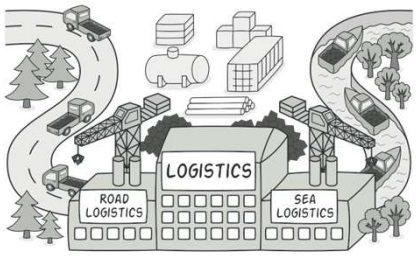
Padrões de criação

- **Factory Method**:



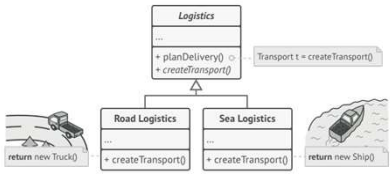
Padrões de criação

- **Factory Method**:



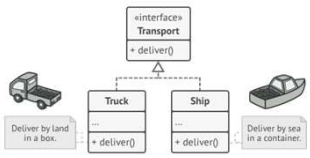
Padrões de criação

- **Factory Method**:



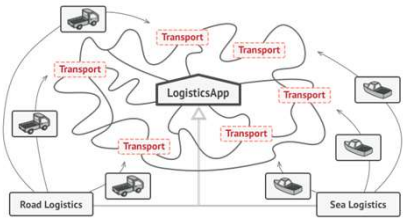
Padrões de criação

- **Factory Method**:



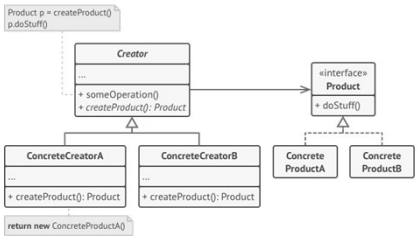
Padrões de criação

- **Factory Method**:



Padrões de criação

- **Factory Method**:



https://refactoring.guru/design-patterns/factory-method

Padrões de criação

- **Singleton**:
  - Às vezes, pode haver a necessidade de ter uma e somente uma instância de uma determinada classe durante o tempo de vida de uma aplicação.
  - Por exemplo, podemos precisar de um único objeto de conexão de banco de dados.
  - O padrão Singleton é útil em tais casos porque garante que existe uma e apenas uma instância de um objeto particular.
  - Além disso, sugere que os objetos de cliente devem ser capazes de acessar a única instância de forma consistente.
    - A própria classe Singleton deve garantir que só existirá uma única instância e deve fornecer acesso à instância para os objetos clientes.

Padrões de criação

- **Multiton (não GoF)**:
  - Uma pequena variação do Singleton que permite a criação de uma quantidade limitada de instâncias, de acordo com alguma regra de negócio;
  - P. ex. DAO é Singleton, porém eu posso ter um DAO para cada classe concreta do Modelo de Negócio.
    - class DAO<T>
      - **private** Class<T> **persistedClass**;

        public DAO(Class<T> persistedClass) {
        this.**persistedClass** = persistedClass;
        }
      - public T getObjeto(Long id) ...
      - public void criaObjeto(T objeto) ...
      - public void removeObjeto(T objeto) ...
  - Outro ex.:
    - Para uma classe Cor temos uma instância para representar cada cor Primária (neste caso considere usar enumeração);

Padrões de criação

- **Object Pool (não GoF)**:
  - O objetivo é possibilitar o reaproveitamento de objetos;
  - Imagine que podemos ter um custo alto para criação e/ou destruição de objetos;
  - Desta forma implementar este padrão pode ser útil para organizar como os usuários compartilham instâncias;
  - É necessário considerar concorrência;
  - Ex. de uso: pool de conexões com o banco de dados;

Padrões de criação

- Exercício 2:
  a. Implemente a instanciação do Banco e Contas Bancárias do Exercício 2 (introdução) através de FactoryMethod.
  b. Implemente um Singleton (qualquer classe).
  c. Implemente um Pool Genérico (classe concreta) de objetos a partir da interface:

```
public interface Pool<T> {
    T acquire();
    void release(T t);
}
```

O pool deverá manter no mínimo 3 instâncias criadas e deverá instanciar no máximo 20. Caso algum cliente tente fazer acquire e o máximo já tenha sido atingido, deverá esperar até que algum outro cliente devolva outro objeto (problema dos produtores vs. Consumidores).