PADRÕES DE PROJETO GOF

Introdução: Motivação

- Projeto de software orientado a objetos é tarefa complexa
  - *Simples uso da OO não garante que se obtenha sistemas confiáveis, robustos, extensíveis e reutilizáveis*
- É difícil achar algo reusável na primeira tentativa
  - ***Reusabilidade** real não se obtém de técnicas de "copiar&colar" nem do simples reaproveitamento de módulos de software*
    - É preciso encontrar objetos pertinentes, fatorá-los em classes na granularidade certa, definir interfaces de classes e hierarquias de herança, e estabelecer relacionamentos chave entre elas
- Ainda assim boas soluções são produzidas:
  - *Boas soluções que já funcionaram devem ser reutilizadas (reuso de experiências anteriores)*

Introdução

- Existem classes de problemas que se repetem em uma diversidade de domínios
  - *Consequentemente: soluções se repetem!*
- **Padrões de Projeto**: 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*

Introdução

- Existem classes de problemas que se repetem em uma diversidade de domínios
  - *Consequentemente: soluções se repetem!*
- **Padrões de Projeto**: 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*

Introdução

- Definições de Padrões de Projeto

  "Cada padrão descreve um problema que se repete várias vezes em um determinado meio, e em seguida descreve o âmago da sua solução, de modo que esta solução possa ser usada milhares e milhares de vezes"

  ALEXANDER, C., **ISHIKAWA**, S., **SILVERSTEIN**, M., **JACOBSON**, M., **FIKSDAHLKING**, I., **ANGEL**, S. "A Pattern Language". New York, NY (USA): Oxford University Press, 1977.

  "Um padrão de projeto sistematicamente nomeia, motiva e explica um projeto genérico, que endereça um problema de projeto recorrente em sistemas orientados a objetos. Ele descreve o problema, a solução, quando é aplicável e quais as consequências de seu uso."

  GAMMA, E., **HELM**, R., **JOHNSON**, R., **VLISSIDES**, J. "Design Patterns: Elements of Reusable Object-Oriented Software". Reading, MA: Addison Wesley, 1995.

Introdução

- Os padrões de projeto beneficiam os desenvolvedores de um sistema
  - *Ajudando a construir um software confiável com arquiteturas testada e perícia acumulada pela indústria*
  - *Promovendo a reutilização de projetos em futuros sistemas*
    - Permitem compartilhar experiências bem sucedidas na resolução de problemas recorrentes.

Introdução

- Os padrões de projeto beneficiam os desenvolvedores de um sistema (cont.)
  - *Ajudando a identificar equívocos comuns e armadilhas que ocorrem na construção dos sistemas*
  - *Estabelecendo um vocabulário comum de projeto entre os desenvolvedores*
  - *Encurtando a fase de projeto no processo de desenvolvimento de um software*
    - Permitem que os desenvolvedores concentrem seus esforços nos aspectos inéditos do problema.

Introdução

- *Design patterns* são aplicados a uma situação específica. Não dá para dizer “para um sistema de locadora, utilize tais padrões”.
- Alguns padrões podem ter maior probabilidade de uso – pode ser que o Singleton tenha mais chances de ser necessário em um projeto do que o Strategy, mas mesmo assim não justifica que seja mais importante que o outro.
- Também é importante ressaltar, que em alguns casos, a utilização excessiva de padrões pode não trazer exatamente resultados positivos.

Padrões GoF

- Em 1995, ***Erich Gamma, John Vlissides, Ralph Jonhson e Richard Helm*** descreveram 23 padrões que podem ser aplicados ao desenvolvimento de sistemas de software orientados a objetos.
  - *Eles são conhecidos como a Gangue dos Quatro (Gand of Four, GoF).*
- Estes padrões não são invenções. São documentação de soluções obtidas através da experiência, coletados de experiências de sucesso na indústria de software, principalmente de projetos em C++ e SmallTalk

Categorias de padrões GoF

- Padrões de projeto estão relacionados a questões de comportamento de objetos, ciclo de vida de objetos, a interface dos objetos e a relacionamentos estruturais entre os objetos
- Permitem desenvolver software de melhor qualidade uma vez que utilizam eficientemente polimorfismo, herança, modularidade, composição, abstração para construir código reutilizável, eficiente, de alta coesão e baixo acoplamento
- Ajuda na documentação e na aprendizagem. O conhecimento dos padrões de projeto torna mais fácil a compreensão de sistemas existentes

Categorias de padrões GoF

- Os padrões GoF foram divididos em:
  1. Padrões Criacionais
     - *Todos os Padrões Criacionais tratam da melhor maneira como instanciar objetos;*
       - "Classe Criadora" especial pode tornar o programa mais flexível e geral.
  2. Padrões Estruturais
     - *Descrevem como classes e objetos podem ser combinados para formar grandes estruturas;*
       - Objetos padrões descrevem como objetos podem ser compostos em grandes estruturas utilizando composição ou inclusão de objetos com outros objetos.
  3. Padrões Comportamentais
     - *Descreve padrões de comunicação entre objetos ou classes*
       - Caracteriza o modo como classes e objetos interagem e compartilham responsabilidades.

Categorias de padrões GoF

		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

Implementações em java em http://www.fluffycat.com/java-design-patterns/

Categorias de padrões GoF - Metsker

- Um padrão GoF pode ser classificado segundo o seu escopo; de classe ou de objeto.
- Nos padrões com escopo de classe os relacionamentos que definem este padrão são definidos através de herança e em tempo de compilação. Nos padrões com escopo de objeto o padrão é encontrado no relacionamento entre os objetos definidos em tempo de execução.

Categorias de padrões GoF - Metsker

- Metsker classifica os padões GoF em 5 grupos, por intenção (problema a ser solucionado):
  - *(1) oferecer uma interface*
  - *(2) atribuir uma responsabilidade*
  - *(3) realizar a construção de classes ou objetos*
  - *(4) controlar formas de operação*
  - *(5) implementar uma extensão para a aplicação de Padrões*

Categorias de padrões GoF - 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

Catálogo de Padrões Gof

- Abstract Factory:
  - *Provê uma interface para criar uma família de objetos relacionados ou dependentes sem especificar suas classes concretas.*
- Adapter:
  - *Converte a interface de uma classe em outra, esperada pelo cliente. Permite que classes que antes não poderiam trabalhar juntas, por incompatibilidade de interfaces, possam agora fazê-lo.*
- Bridge:
  - *Separa uma abstração de sua implementação, de modo que ambas possam variar independentemente.*

Catálogo de Padrões Gof

- Builder:
  - *Provê uma interface genérica para a construção incremental de agregações. Um Builder esconde os detalhes de como os componentes são criados, representados e compostos.*
- Chain of Responsibility:
  - *Encadeia os objetos receptores e transporta a mensagem através da corrente até que um dos objetos a responda. Assim, separa objetos transmissores dos receptores, dando a chance de mais de um objeto poder tratar a mensagem.*
- Command:
  - *Encapsula uma mensagem como um objeto, de modo que se possa parametrizar clientes com diferentes mensagens. Separa, então, o criador da mensagem do executor da mesma.*

Catálogo de Padrões Gof

- Composite:
  - *Compõe objetos em árvores de agregação (relacionamento parte-todo). O Composite permite que objetos agregados sejam tratados como um único objeto.*
- Decorator:
  - *Anexa responsabilidades adicionais a um objeto dinamicamente. Provê uma alternativa flexível para extensão de funcionalidade, sem ter que usar Herança.*
- Façade:
  - *Provê uma interface unificada para um conjunto de interfaces em um subsistema. O Façade define uma interface alto nível para facilitar o uso deste subsistema.*

Catálogo de Padrões Gof

- Factory Method:
  - *Define uma interface para criação de um objeto, permitindo que as suas subclasses decidam qual classe instanciar. O Factory Method deixa a responsabilidade de instanciação para as subclasses.*
- Flyweight:
  - *Usa o compartilhamento para dar suporte eficiente a um grande número de objetos com alto nível de granularidade.*
- Interpreter:
  - *Usado para definição de linguagens. Define representações para gramáticas e abstrações para análise sintática.*

Catálogo de Padrões Gof

- Iterator:
  - *Provê um modo de acesso a elementos de um agregado de objetos, sequencialmente, sem exposição de estruturas internas.*
- Mediator:
  - *Desacopla e gerencia as colaborações entre um grupo de objetos. Define um objeto que encapsula as interações dentro desse grupo.*
- Memento:
  - *Captura e externaliza o estado interno de um objeto (captura um "snapshot"). Não viola o encapsulamento.*

Catálogo de Padrões Gof

- Observer:
  - *Provê sincronização, coordenação e consistência entre objetos relacionados.*
- Prototype:
  - *Especifica os tipos de objetos a serem criados num sistema, usando uma instância protótipo. Cria novos objetos copiando este protótipo.*
- Proxy:
  - *Provê projeto para um controlador de acesso a um objeto.*
- Singleton:
  - *Assegura que uma classe tenha apenas uma instância e provê um ponto global de acesso a ela.*

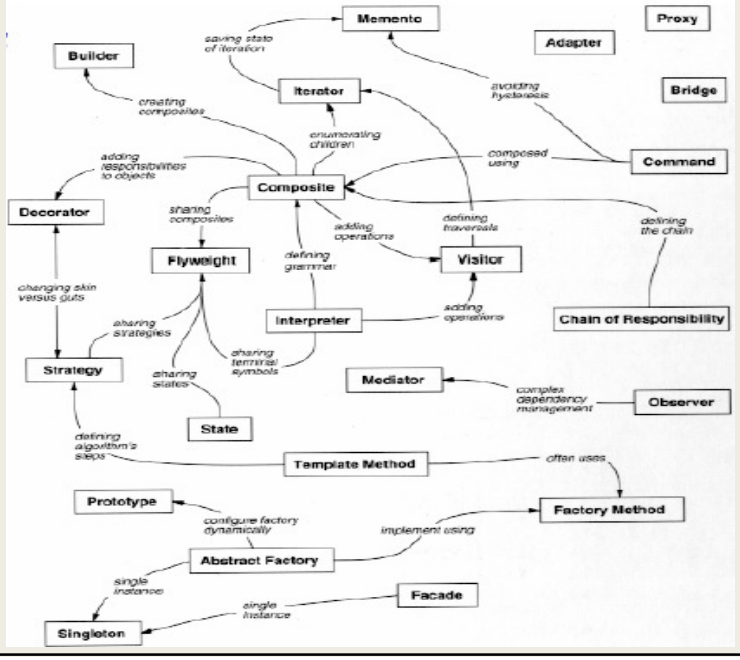
Catálogo de Padrões Gof

- State:
  - *Deixa um objeto mudar seu comportamento quando seu estado interno muda, mudando, efetivamente, a classe do objeto.*
- Strategy:
  - *Define uma família de algoritmos, encapsula cada um deles, e torna a escolha de qual usar flexível. O Strategy desacopla os algoritmos dos clientes que os usa.*

Catálogo de Padrões Gof

- Template Method:
  - *Define o esqueleto de um algoritmo em uma operação. O Template Method permite que subclasses componham o algoritmo e tenham a possibilidade de redefinir certos passos a serem tomados no processo, sem contudo mudá-lo.*
- Visitor:
  - *Representa uma operação a ser realizada sobre elementos da estrutura de um objeto. O Visitor permite que se crie um nova operação sem que se mude a classe dos elementos sobre as quais ela opera.*

Relacionamento entre os Padrões



Exemplificando - Factory Method

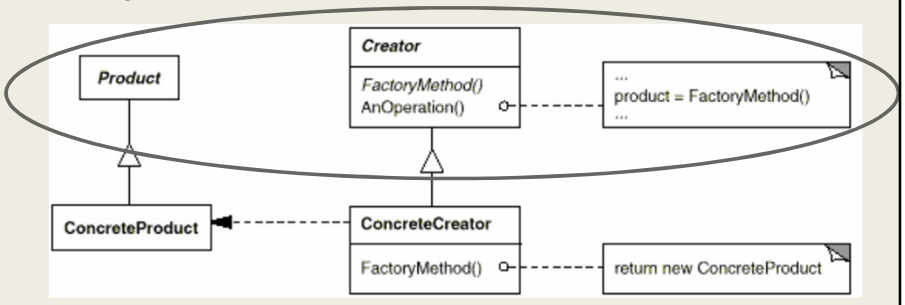
- Padrão de criação
- Exemplo de Aplicação:
  - *Sempre que houver muitas classes diferentes a serem instanciadas*
- Vantagens:
  - *Instanciação indireta*
  - *Fácil adição de novas classes*
  - *Desacoplamento*
  - *Pretinho básico: vai bem com tudo*
- Desvantagens:
  - *Uma classe concreta criadora para cada item produzido*
  - *Nem sempre o desacoplamento é garantido*

Exemplificando - Factory Method

- Nome: **Factory Method**
- Descrição do problema: Uma classe não sabe antecipar o tipo dos objetos que a mesma precisa criar
  - *É preciso adiar a instanciação de objetos para as subclasses*
  - *Exemplo: suponha uma aplicação lida com diversos tipos de documentos*
    - Ela sabe quando os documentos devem ser criados, mas não sabem que documentos criar
- Descrição da solução: próximo slide
- Consequências: Eliminam dependências de classes específicas (código lida com as interfaces)

Exemplificando - Factory Method

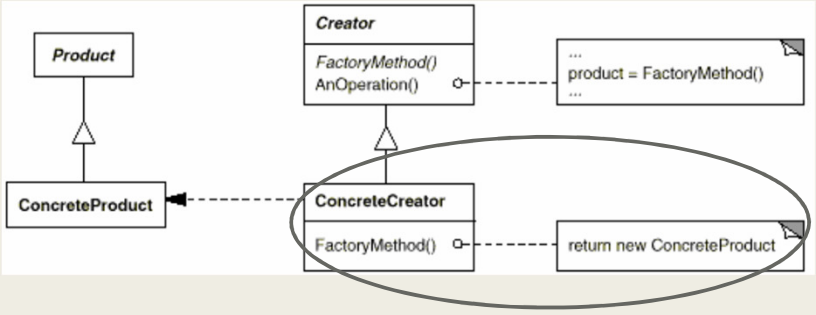
- Solução:



Superclasses não conhecem o produto específico

Exemplificando - Factory Method

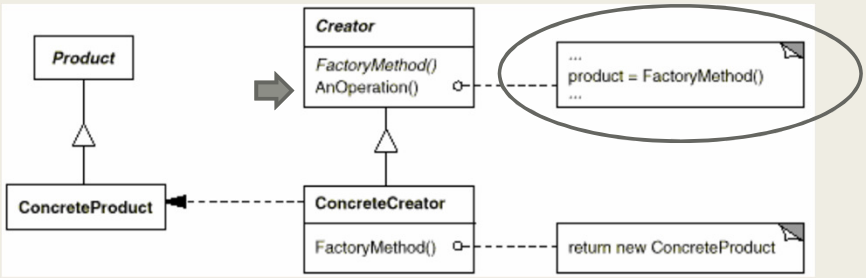
- Solução:



O FactoryMethod cria e retorna o objeto no momento adequado

Exemplificando - Factory Method

- Solução:



O produto geral pode seu usado pela superclasse, mesmo sem conhecer o produto específico

Exemplificando - Strategy

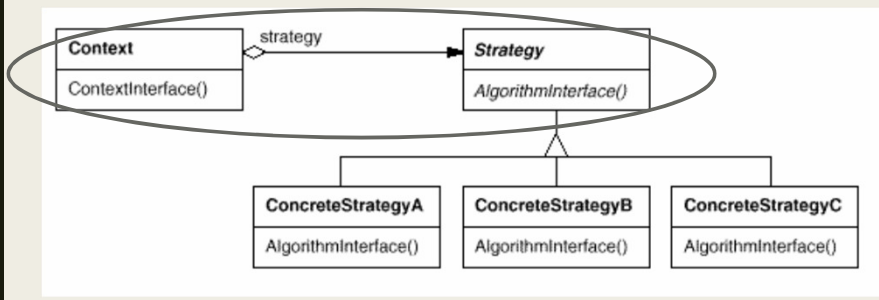
- Vantagens:
  - *Simples*
  - *Alto desacoplamento*
  - *Intercâmbio de Algoritmos*
- Desvantagens:
  - *Precisa construir um objeto de uma classe concreta*
  - *Nem sempre é a solução para tudo*

Exemplificando - Strategy

- Nome: **Strategy**
- Descrição do problema: Definir uma família de algoritmos
  - *Existem vários algoritmos para um mesmo problema*
    - Ex.: vários algoritmos de ordenação de array (BubbleSort, QuickSort, etc.)
  - *O objetivo é implementar e executar diferentes algoritmos usando uma mesma interface*
    - Encapsular os diferentes algoritmos e torná-los intercambiáveis
- Descrição da solução: próximo slide
- Consequências: Permite alternar entre diferentes algoritmos sem o uso de condicionais

Exemplificando - Strategy

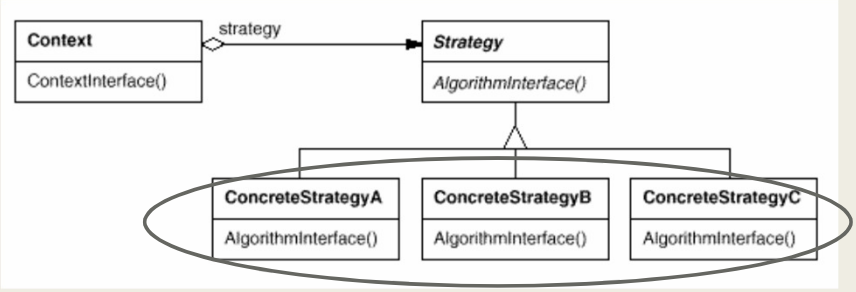
- Solução:



A aplicação usa o algoritmos sem conhecer sua real implementação.

Exemplificando - Strategy

- Solução:



Diferentes implementações do algoritmo

Bibliografia

- Erich Gamma, *Padrões de Projeto: soluções reutilizáveis de software orientado a objetos*, Bookman, 2000
- Craig Larman, *Utilizando UML e Padrões: uma introdução a análise e ao projeto orientados a objetos*, Bookman, 2000*
- Steven John Metsker, *Padrões de Projeto em Java™,* Bookman, 2004

(*) Edição mais atual (2007) na BCC Itajaí

Seminário - temas

- 19H PADRÃO SINGLETON - ANDRÉ E GUSTAVO
- 19H40 PADRÃO FACADE - HILSON
- 20H20 PADRÃO DECORATOR - ANDRIGO E GABRIEL
- 21H PADRÃO ADAPTER – KHALIL E GUSTAVO
- 21H40 ARQUITETURA MVC - KATIA E EDUARDO

Seminário

- Preparar seminário de 30-40 min
- Parte teórica e prática
  - Ex. de diagramas e de implementação usando o padrão escolhido
- Nota M3: slides + apresentação