Design Patterns para Java

Daniel Wildt (daniel.wildt@bluestar.inf.br)

JBuilder/Delphi Certified Instructor/Developer

http://www.geocities.com/dwildt2

Bluestar Technology

http://www.bluestar.inf.br

Daniel Wildt (daniel.wildt@bluestar.inf.br)

JBuilder/Delphi Certified Instructor/Developer

http://www.geocities.com/dwildt2

Agenda

- O que são Design Patterns e suas características
- Padrões de projeto de Mercado, visão geral
- Padrões de Projeto GoF
  - Criação
  - Estruturais
  - Comportamento

O que são Design Patterns e suas características

- Na definição de Christopher Alexander, “Cada padrão descreve um problema que acontece diversas vezes em nosso ambiente e descreve uma solução base de modo que se possa reutilizar a solução milhares de vezes...”.
- Descrição de objetos e classes que se comunicam para resolver um problema de design genérico em um contexto particular.

O que são Design Patterns e suas características

- Cada padrão de projeto possui características, como:
  - Nome: sua identificação para o mercado, normalmente o atributo mais complicado de se descrever.
  - Problema: indicar quando o uso do padrão pode ser interessante.

O que são Design Patterns e suas características

- Cada padrão de projeto possui características, como:
  - Solução: como aplicar o padrão de projeto
  - Consequências: perdas e ganhos ao aplicar o padrão. As vezes se pode perder coesão ou ganhar acoplamento no sistema ao fazer uso de determinado padrão.

Agenda

- O que são Design Patterns e suas características
- Padrões de projeto de Mercado, visão geral
- Padrões de Projeto GoF
  - Criação
  - Estruturais
  - Comportamento

Padrões de projeto de Mercado, visão geral

- Padrões de projeto GoF (serão apresentados em seguida)
- Padrões de projeto Sun J2EE
  - Data Transfer Object
  - Data Access Object
  - Front Controller
  - Value List Handler
  - Viewer Helper
  - http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html

Padrões de projeto de Mercado, visão geral

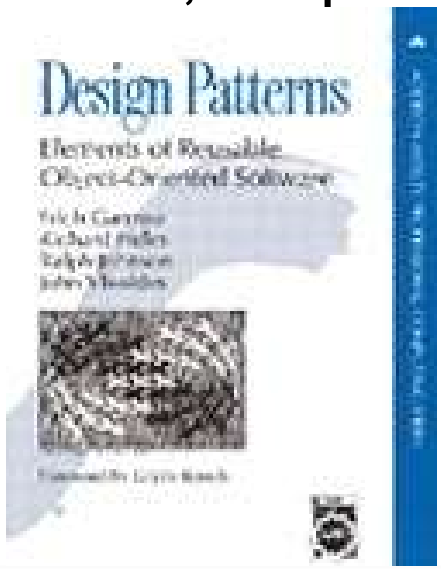
- Coad Patterns
  - http://www.thecoadletter.com/coadletter/
- Refactoring Patterns
  - http://www.refactoring.com

Agenda

- O que são Design Patterns e suas características
- Padrões de projeto de Mercado, visão geral
- Padrões de Projeto GoF
  - Criação
  - Estruturais
  - Comportamento

Padrões de Projeto GoF

- Referência:
  - Título: Design Patterns : Elements of Reusable Object-Oriented Software
  - Autores: Eric Gamma, Richard Helm, Ralph Johnson, John Vlissides
  - ISBN: 0201633612
  - Editora: Pearson
  - Publicação: 1994



Agenda

- O que são Design Patterns e suas características
- Padrões de projeto de Mercado, visão geral
- Padrões de Projeto GoF
  - Criação
  - Estruturais
  - Comportamento

Padrões de Projeto GoF

- Criação:
  - **Abstract Factory**: provê uma interface para se criar familias de objetos relacionados ou dependentes sem utilizar as classes concretas.
  - **Builder**: separa o processo de construção de um objeto complexo de sua representação. Deste modo o mesmo processo de construção pode retornar diferentes representações.

Padrões de Projeto GoF

- Criação:
  - **Factory**: define uma interface para a criação de um objeto, mas deixa as classes escolherem qual classe deve ser instanciada.
  - **Prototype**: estabelece padrão para permitir a cópia de objetos (clone)
  - **Singleton**: estabelece a regra de que só pode existir um objeto disponível na memória para a classe que implementa este padrão.

Agenda

- O que são Design Patterns e suas características
- Padrões de projeto de Mercado, visão geral
- Padrões de Projeto GoF
  - Criação
  - Estruturais
  - Comportamento

Padrões de Projeto GoF

- Estruturais:
  - **Adapter**: permite que classes trabalhem juntas onde em outros casos não poderiam por possuirem interfaces diferentes
  - **Bridge**: desacopla a implementação de uma abstração de modo que as duas possam variar de forma independente.
  - **Composite**: capacidade de utilizar relações de parte-todo (composição)

Padrões de Projeto GoF

- Estruturais:
  - **Decorator**: anexa responsabilidades a um objeto de forma dinâmica. Alternativa para especialização de classes.
  - **Facade**: prover uma interface unificada para um conjunto de interfaces. Interface de mais alto nível para se trabalhar.

Padrões de Projeto GoF

- Estruturais:
  - **Flyweight**: técnica para permitir que objetos possam ser usados em múltiplos contextos de forma simultânea.
  - **Proxy**: controla o acesso ao objeto original e em alguns casos visa diminuir a complexidade de uso de objetos.

Agenda

- O que são Design Patterns e suas características
- Padrões de projeto de Mercado, visão geral
- Padrões de Projeto GoF
  - Criação
  - Estruturais
  - Comportamento

Padrões de Projeto GoF

- Comportamento:
  - **Chain of Responsibility**: evita acoplamento entre objetos para responder a requests. Cria uma cadeia de processamento.
  - **Command**: encapsula a requisição feita para um objeto.
  - **Interpreter**: padrão para suprir a necessidade de interpretadores de linguagens através de sua gramática.

Padrões de Projeto GoF

- Comportamento:
  - **Iterator**: provê acesso sequêncial aos elementos de uma agregação sem expor sua representação.
  - **Mediator**: define objeto que encapsula o relacionamento entre outros objetos. Agrega a perda de acoplamento explícito entre classes.
  - **Memento**: sem violar encapsulação, captura e externaliza o estado interno de um objeto.

Padrões de Projeto GoF

- Comportamento:
  - **Observer**: define uma dependência de um para muitos onde quando um objeto muda seu estado, todos associados são notificados.
  - **State**: permite ao objeto modificar seu comportamento quando seu estado interno muda.
  - **Strategy**: necessidade de diferentes variantes de um algoritmo, conforme situação. Interdependência.

Padrões de Projeto GoF

- Comportamento:
  - **Template Method**: define o esqueleto de um algoritmo e permite que subclasses redefinam alguns passos sem modificar a estrutura do algoritmo.
  - **Visitor**: permite definir uma nova operação sem modificar a estrutura das outras classes que em este opera.

Obrigado!

Design Patterns para Java

Daniel Wildt (daniel.wildt@bluestar.inf.br)

JBuilder/Delphi Certified Instructor/Developer

http://www.geocities.com/dwildt2