Padrões de Projeto de Software

Unidade 1 – Fundamentos de Padrões de Projetos

Luiz Leão – luizleao@gmail.com

http://www.luizleao.com

Unidade 1 – Fundamentos de Padrões de Projetos

- Introdução
- O que é?
- Como descrever?
- Principais Padrões de Projetos

Unidade 2 – Padrões GoF

- PADRÕES CRIAÇÃO
  - *Abstract Factory*
  - *Builder*
  - *Factory Method*
  - *Prototype*
  - *Singleton.*
- PADRÕES ESTRUTURAIS
  - *Adapter*
  - *Bridge*
  - *Composite*
  - *Decorator*
  - *Façade*
  - *Flyweight*
  - *Proxy*

Unidade 2 – Padrões GoF (Cont.)

- PADRÕES COMPORTAMENTAIS
  - *Chain of Responsibility*
  - *Command*
  - *Interpreter*
  - *Iterator*
  - *Mediator*
  - *Memento*
  - *Observer*
  - *State*
  - *Strategy*
  - *Template Method*
  - *Visitor*

Unidade 3 – Padrões GRASP (*General Responsibility Assignment Software Patterns*)

- PADRÕES BÁSICOS
  - *Information Expert* ou *Expert* (Especialista na Informação)
  - *Creator* (Criador)
  - *Low Coupling* (Acoplamento Fraco ou Baixo)
  - *High Cohesion* (Coesão Alta)
  - *Controller* (Controlador)
- PADRÕES AVANÇADOS
  - *Polymorphism* (Polimorfismo)
  - *Indirection* (Indireção)
  - *Pure Fabrication* (Invenção Pura)
  - *Protected Variations* (Variações Protegidas)

Unidade 4 – Arquitetura em Camadas

- Definição
- Composição

Conteúdo Programático

- Introdução
- O que é?
- Como descrever?
- Principais Padrões de Projetos

Introdução

- Projetar software **orientado a objetos** é difícil
- Projetar software **orientado a objetos** e **reutilizável** é muito difícil
- O projeto deve ser específico ao problema, porém genérico o suficiente para acomodar futuras mudanças
- É difícil obter um projeto flexível e reutilizável na primeira tentativa
- Projetistas novatos levam um tempo para entender o que é um bom projeto orientado a objetos

Introdução

- O que os projetistas experientes fazem:
  - Reutilizam soluções que funcionaram no passado
  - Muitos sistemas orientados a objetos compartilham padrões de funcionamento das classes e da comunicação entre objetos
  - Estes padrões tornam o projeto mais flexível, elegante e reutilizável
  - O projetista aplica o padrão de projeto sem ter que o re-descobrir
  - Novelistas e dramaturgos aplicam padrões constantemente
  - O mesmo vale para software

Introdução

- Exemplos:
  - "Represente o estado como um objeto"
  - "Decore os objetos de modo que funcionalidades possam ser facilmente adicionadas ou removidas"
- Se você conhece o padrão, uma série de decisões de projeto surgem automaticamente
- Um padrão de projeto registra uma determinada experiência bem sucedida em projeto de software
- Cada padrão sistematicamente nomeia, explica e avalia um projeto importante e recorrente

Introdução

- Os padrões de projeto facilitam a definição de um projeto "correto" em um tempo reduzido
- A idéia de padrões foi apresentada por Christopher Alexander em 1977 no contexto de Arquitetura (de prédios e cidades):
- “Cada padrão descreve um problema que ocorre frequentemente em nosso ambiente e descreve o núcleo da solução para o problema, de uma forma que ela possa ser utilizada inúmeras vezes” (Christopher Alexander, 1977)

Introdução

- Um padrão possui quatro elementos:
  - **Nome:** Identificador utilizado para descrever, com uma ou duas palavras, o problema, sua solução e consequências
  - **Problema:** Descreve quando aplicar o padrão e o contexto do problema
  - **Solução:** Descreve os elementos que compõem o projeto, seus relacionamentos, responsabilidades e colaborações. Não descreve uma implementação ou projeto concreto em particular
  - **Consequências:** Resultados e *trade-offs* da aplicação do padrão

Introdução

- Os padrões apresentados são descrições de classes e objetos inter-relacionados que solucionam um problema de projeto em um contexto particular

- Outros padrões estão disponíveis para solucionar problemas de concorrência, computação distribuída, tempo-real e aspectos específicos de domínio