Lição 5

Implementação – Parte 1

Objetivos

Ao final desta lição, o estudante será capaz de:

- Conhecer dicas de programação
- Conhecer práticas de engenharia de software para escrever códigos
- Oferecer exemplos de implementação do projeto, particularmente, pacotes, classes de bancos de dados e classes persistentes, e interfaces gráficas com o usuário

Padrões e Procedimentos de Programação

- Definem como o código-fonte deve ser escrito
- Tornam o código compreensível para outras pessoas
- Permitem que os programadores organizem seus pensamentos e evitem erros
- Ajudam outros times de desenvolvimento, como os testadores de software e mantenedores do sistema

Padrões Específicos de Implementação

- Eles têm que decidir por:
  - Plataforma Rígida
  - Versão de Software
- Documentação do Código-Fonte
- Correspondência entre o Projeto e o Programa

Dicas de Programação

- Usando Pseudo-códigos
- Dicas de Estruturas de Controle
- Dicas de Documentação
  - Documentação Interna
  - Documentação Externa

Bloco Cabeçalho de Comentário

- Age como uma introdução ao código-fonte
- Identifica:
  - Nome do Componente
  - Autor do Componente
  - Data da última criação ou modificação do Componente
  - Lugar onde o componente se encaixa no sistema em geral
  - Detalhes da estrutura de dados do componente, algoritmo e fluxo de controle
- Histórico da Modificação
  - Quem fez a modificação?
  - Quando o componente foi modificado?
  - Qual foi a modificação?

Dicas na Escrita de Códigos

- Use nomes significativos para variáveis e métodos
- Use formatação para aumentar a legibilidade
- Coloque comentários adicionais
- Tenha métodos separados para entrada e saída
- Evite GOTO's
- A escrita do código é um processo interativo

Mapeando Produtos de Implementação com MRR

- Não há componentes implementados no MRR além dos componentes do software e as classes são monitoradas baseadas em seus estados

Métricas de Implementação

- A maioria das métricas do programa correspondem às métricas do projeto

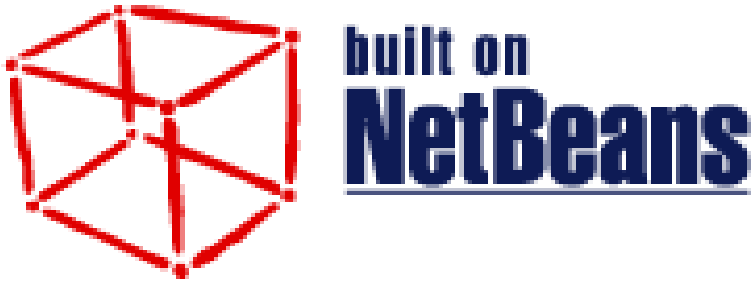
- Métricas Históricas do Projeto devem ser guardadas:
  - Linhas de Código (LDC)
  - Número de Classes
  - Número de Páginas de Documentação
  - Custo
  - Esforço

Pacotes

- Provê mecanismo para a reutilização de códigos de software
- A Linguagem de Programação Java provê um mecanismo para a definição de pacotes

Passos para definição de Pacotes

- Passaremos agora para o NetBeans



Controladores

- Controladores são responsáveis pela implementação dos requisitos comportamentais ou funcionais do sistema
- Implementação de controladores é semelhante à escrita de códigos em outros cursos de linguagem de programação
- É melhor implementar controladores utilizando classes abstratas e interface

Classes Abstratas

- Classe que não pode ser instanciada
- Geralmente aparece no topo de uma hierarquia de classes na programação orientada-a-objeto, definindo uma variedade de ações possíveis com os objetos de todas as subclasses de uma classe

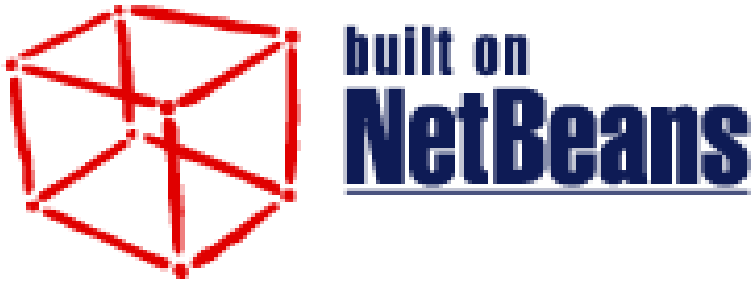
Métodos Abstratos

- Métodos na classe abstrata que não têm implementação
- Para criar um método abstrato, simplesmente escreva a declaração do método sem corpo e use a palavra-chave abstract
- Por exemplo,

```
public abstract void someMethod();
```

Exemplo de Classe Abstrata

- Passaremos agora para o NetBeans



Classes Abstratas

- Quando uma classe estende a classe abstrata LivingThing, é necessário que implemente o método abstrato walk() ou, caso contrário, esta subclasse também deverá ser uma classe abstrata e, portanto, não poderá ser instanciada

- Por exemplo,

```
public class Human extends LivingThing {
   public void walk(){
      System.out.println("Human walks...");
   }
}
```

Dicas de Codificação

- Use classes abstratas para definir vários tipos de comportamentos no topo de uma hierarquia de classes na programação orientada-a-objetos, e use suas subclasses para prover detalhes da implementação da classe abstrata

Interfaces

- Tipo especial de bloco contendo somente assinatura de métodos (e possivelmente constantes)
- Define as assinaturas de um conjunto de métodos, sem corpo
- Define uma maneira padrão e pública de se especificar o comportamento das classes
- Permite às classes, independentemente da sua localização na hierarquia de classes, implementarem comportamentos comuns

Por que usamos Interfaces?

- Para termos classes não relacionadas implementando métodos similares
  - Exemplo:
    - Classe Line e MyInteger
      - Não relacionadas
      - Ambas implementam métodos de comparação
        - isGreater (é Maior)
        - isLess (é Menor)
        - isEqual (é Igual)

Por que usamos Interfaces?

- Para revelar uma interface de um objeto de programação sem revelar sua classe
- Para modelar herança múltipla o que permite que uma classe tenha mais de uma superclasse

Interface vs. Classe Abstrata

- Métodos de interface não têm corpo
- Uma interface só pode definir constantes
- Interfaces não têm nenhuma relação direta de herança com qualquer classe em particular, são definidas independentemente

Interface vs. Classe

- Semelhança:
  - Interfaces e classes são ambos tipos
  - Isto significa que uma interface pode ser usada em lugares onde uma classe pode ser usada
  - Por exemplo:

```
PersonInterface pi = new Person();
Person pc = new Person();
```

- Diferença:
  - Você não pode criar uma instância de uma interface
  - Por exemplo:

```
PersonInterface pi = new PersonInterface();
```

Interface vs. Classe

- Semelhança:
  - Interface e Classe podem ambos definir métodos
- Diferença:
  - Interface não tem qualquer implementação dos métodos

Criando Interfaces

- Para criarmos uma interface, escrevemos:

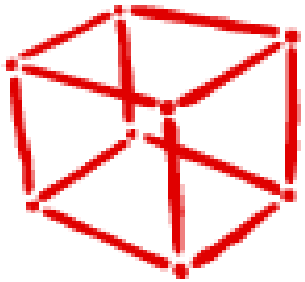
```
public interface [NomeDaInterface] {
      //alguns métodos sem o corpo
}
```

Criando Interfaces

```
public interface Relation {
   public boolean isGreater(Object a, Object b);
   public boolean isLess(Object a, Object b);
   public boolean isEqual(Object a, Object b);
}
```

Criando Interfaces

- Passaremos agora para o NetBeans



built on NetBeans

Relacionamento de uma Interface com uma Classe

- Uma classe só pode ESTENDER UMA super classe, mas pode IMPLEMENTAR VÁRIAS interfaces.
- Por exemplo:

```
public class Person implements PersonInterface,
     LivingThing, WhateverInterface {
   //algum código aqui
}
```

Relacionamento de uma Interface com uma Classe

- Outro exemplo:

```
public class ComputerScienceStudent
   extends Student
   implements PersonInterface, LivingThing {
    //algum código aqui
}
```

Final da Parte 1

- Continua...

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**