Disciplina: LPOO

Prof. Anderson V. de Araujo

Convenções de Código Java

Introdução

- As convenções de código são uma boa prática de programação
- Ajudam, por exemplo, na manutenção de um sistema
- A violação dessas “regras” prejudica em muito o entendimento de um código
- Com as convenções, é possível produzir um código legível, “limpo”, que demonstra qualidade e de maneira profissional

Motivação

- 80% do custo total de uma parte de um software é de manutenção
- Dificilmente um software vai ser mantido pelo autor original pela vida toda
- Deve-se seguir uma convenção, pois outros programadores podem dar manutenção no que vier a ser construído depois
- Uma aplicação possui um vida útil, e queremos que essa vida seja longa
- Fácil entendimento por uma equipe de desenvolvimento
- Profissionalismo
- Um software é um produto que deve possuir qualidade, codificação limpa, fácil entendimento, fácil manutenção

Pacotes

- Deve obedecer a uma certa hierarquia, separada por pontos
- Deve-se usar letras minúsculas e raramente se faz usos de números
- Se o pacote a ser utilizado é de fora da empresa ou organização, este deve começar com o domínio de internet redigido de forma contrária
  - Por exemplo, o pacote *com.sun*.
- Essas “partes” devem ser curtas, com até oito caracteres;
- As bibliotecas padrão e pacotes opcionais que começam com *java* e *javax*, são exceções a esta regra e não são permitidos esses nomes para uso do desenvolvedor
- São permitidas abreviações, como por exemplo, o pacote *util* ao invés de *utilities*
- O uso de acrônimos também é uma boa prática

Arquivos

- Pede-se que seja evitado arquivos com mais de 2000 linhas de código
- Cada arquivo fonte Java contém uma única classe ou interface pública
- A classes privadas e as interfaces associadas a classe pública podem ser colocadas em um mesmo arquivo que uma classe pública
- A classe pública deve ser a primeira classe ou interface no arquivo

Comentários Iniciais

- Devem ser escritos no início da classe, antes da declaração do pacote (*package*)

```
/*
 * Nome da Classe
 *
 * Versão
 *
 * Alterações na classe
 *
 */
```

Importações de código

- Evitar o uso de * no import:
  - Certo: `import javax.servlet.http.HttpSession`
  - Errado: `import javax.servlet.*`

Declaração de Classe ou Interface

- Os nomes devem consistir em uma ou mais palavras, de preferência substantivos ou frases substantivas
- Com a primeira letra de cada palavra em letra maiúscula
- Abreviações devem ser evitadas
- Exemplo:
  - `public class NomeDaClasse {`

Declaração de Classe ou Interface (2)

- Comentário Javadoc da classe ou interface:

```
/**
 * Nome da Classe e descrição
 *
 * @version
 * @author
*/
```

Declarações

- Declarar uma declaração por linha:
  - Preferir:
    - `int a;`
    - `int b;`
  - Evitar:
    - `int a, b;`
- Se possível, inicializar a variável no momento da sua criação;
  - Exemplo:
    - `int a = 0;`
    - `int b = 0;`

Declarações (2)

- Evitar múltiplas atribuições na mesma linha:
  - Preferir:
    - `a = 10;`
    - `b = 10;`

    - `a = b + c;`
    - `d = a + r;`
  - Evitar:
    - `a = b = 10;`

    - `d = (a = b + c) + r;`

Declarações (3)

- Nomes de variáveis (ou atributos) começam com minúsculas e o início de cada palavra seguinte com maiúscula:
  - Exemplo:

```
public int nomeDaVariavel;
```

- Constantes:
  - O nome deve ser todo em letra maiúscula
  - Exemplo:

```
static final int DISTANCIA_MINIMA = 30;
static final String NOME_SISTEMA = “BUSAO”;
```

Declaração de Métodos

- Nomes dos métodos também seguem a padronização similar a de variáveis
- Os métodos normalmente utilizam verbos ou frases verbais
- Métodos que retornem um tipo booleano devem usar prefixo "is" seguido de um substantivo ou adjetivo
- Exemplos:

```
public void nomeDoMetodo() {
}

public boolean isEmpregado(){
}
```

Declaração de Métodos (2)

- Os métodos que retornam o resultado de uma função ou um atributo do objeto são nomeados com substantivos com o prefixo "`get`"
- Os métodos para atribuir o valor a um atributo do objeto são nomeados com substantivos com o prefixo "`set`"
- Em relação a JavaBeans, o uso do `get` e `set` são obrigatórios

Declaração de Métodos - Comentários

- Comentário Javadoc do método:

```
/**
 * Nome do método e descrição
 *
 * @param nomeParam Descrição do Parâmetro
 * @return Descrição do retorno
*/
```

Localização das Declarações

- Todas as declarações devem ser realizadas no inicio do bloco do seu escopo
  - Exemplo:

```
public void nomeMetodo() {
      int a;  //Essas variáveis serão usadas em todo
o escopo do método
      int b;

      if (condição) {
            int c; //Variável utilizada apenas
dentro do if.
      }
}
```

Localização das Declarações (2)

- Deve-se evitar colocar o mesmo nome em variáveis de escopos diferentes
  - Exemplo:

```
int a;

void metodo() {
      int a;
}
```

Retorno de Métodos

- Evitar:

```
if (booleanExpression) {
      return true;
} else {
      return false;
}
```

- Preferir:

```
return booleanExpression;
```

Comentários Especiais

- Usar comentários especiais para definir pontos de atenção dentro do código
- //FIXME:
  - Usar quando foi identificado um erro, que deve ser corrigido mais tarde
- //TODO:
  - Usar quando alguma tarefa ainda está pendente de ser executada naquele ponto do código
- //XXX:
  - Corresponde a uma parte do código que está mal feita, precisa ser refeito, mas está funcionando

Código Fonte Oracle

- Para verificar a utilização das convenções, podemos abrir o código fonte do Java escrito por programadores da Oracle
- É interessante associar os códigos da API padrão do Java ao seu editor (IDE) preferido
  - Assim, sempre que tiver dúvidas pode consultar o código