Melhores Práticas de Programação JAVA – 2018 V. 01

Lucas Pereira de Oliveira, Scarlett Gomes Barros

Análise e Desenvolvimento de Sistemas – Faculdades Integradas Camões (FICA)
80020-040 – Curitiba – PR – Brazil

{lukspoli@gmail.com, Scarlett__gomes@hotmail.com

Abstract. *The following document had the purpose to turn the development of Java projects standardized in order to make the future code maintenance simpler and faster.*

Resumo. *Este documento tem por objetivo tornar o desenvolvimento de projetos em Java padronizados com o intuito de tornar as futuras manutenções no código mais simples e rápidas.*

1. PACOTES, CLASSES E INTERFACES

A implementação do código deve estar dentro de um pacote com mesmo nome da classe **main**. Os pacotes devem ser nomeados com letras minúsculas e em casos de nomes compridos devem ser separados por underline (_). Abaixo exemplo:

```
1
2    package dias_e_horas;
3
```

As classes e interfaces devem ser nomeadas com a primeira letra maiúscula e em caso de nomes compostos eles devem seguir o mesmo padrão sem adição de espaços, traços ou underline. A chave ({) de abertura da classe deve estar na mesma linha do nome separada por um espaço. Exemplo:

```
4    public class AcessoHoras {
5
```

Toda classe Java deve ter um pacote associado a ela.

1.1 COMENTÁRIOS

Toda classe e interface deve conter um comentário como cabeçalho informando o autor, o nome do projeto e função da classe utilizando os identificadores (/**, */). Exemplo:

```
1  /**
2   * Nome do desenvolvedor
3   * Calculador de perídos
4   * Função: Acessar as variáveis usadas na classe main
5   */
```

Os comentários devem estar presentes no decorrer de todo o código identificando funções ou variáveis importantes para a interpretação sem a documentação em mãos. Eles podem ser definidos seguindo a seguinte formatação. Duas barras para comentários em uma linha. Exemplo:

```
8
9       public double calculaArea(){
10          return area=this.lados[0]*this.lados[1]; //cálculo área retangulo
11      }
```

Os comentários que possuem mais uma linha podem utilizar duas formatações, usando duas barras em cada linha ou utilizar uma barra e um asterisco. Exemplos:

```
16
17      public void exemplo(double[] ex){//Método exemplo
18          this.lados = ex;             //Mais uma linha
19      }
```

```
17      public void exemplo(double[] ex){
18          /*Método exemplo
19          Mais uma linha
20          Mais uma linha
21          */
22          this.lados = ex;
23      }
```

1.2 VARIÁVEIS

As variáveis que podem ser instanciadas e estáticas devem obedecer a seguinte ordem.

- Variáveis default (protected);
- Variáveis públicas;
- Variáveis privadas.

As variáveis devem ser declaradas no início de um bloco (todo código que esteja entre duas chaves { e }) utilizando uma linha por declaração de variável, pois facilitam comentários e a localização delas dentro do código. Nunca declare variáveis de tipos diferentes na mesma linha.

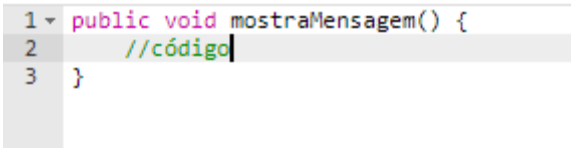
1.3 CONSTANTES

Para melhor identificação das constantes, todas as suas letras devem estar em maiúsculo e com o símbolo “_” entre as palavras.

```
1
2   static final int PI = 3.14;
3
4   static final int VALOR_MAXIMO = 1000000;
5
6
```

1.4 MÉTODOS E CONSTRUTORES

O nome do método deve começar com um verbo, onde a primeira letra é minúscula seguida de uma segunda palavra, onde a primeira letra será maiúscula. Sempre deve haver um retorno declarado, mesmo que não tenha um. Nesse caso utiliza-se o tipo void.



Os construtores devem estar dispostos antes dos métodos que por sua vez devem seguir a ordem de funcionalidade dentro do programa. Podendo existir um método privado entre dois públicos.

2. INDENTAÇÃO DO CÓDIGO

A legibilidade do código traz melhor compreensão para os envolvidos no projeto, então seguem alguns hábitos para serem aplicados em seu código e melhorar a estrutura do mesmo, tornando-o mais fácil de compreender:

- Opte por linhas menores (de até 100 caracteres por linha no máximo) para que o texto se encaixe melhor na tela;
- A quebra de linha do código deve ocorrer em um ponto que não perca o sentido ou corte palavras;
- Frequentemente utiliza-se quatro espaços para a tabulação;
- Cada linha deve conter somente uma expressão;
- Utilize um espaço antes do ‘{‘ ao invés de pular uma linha, desse modo economiza-se código;
- Evitar comentários desnecessários e muito longos;
- Agrupe as linhas de código de acordo com as suas funções;
- Não repita linhas de código.

3. OUTRAS CONSTATAÇÕES

Ao utilizar expressões que estejam entre chaves ({ }) e que possuam linhas com comandos, deixe a chave de abertura no final da linha e comece a escrever as expressões nas linhas que sucedem. Isso torna o código mais organizado e mantêm um padrão que será seguido em todas as linhas.

Espaços em branco tornam o código mais legível e organizado. Utilize ele entre seções do código, entre métodos, construtores, entre variáveis e primeira instrução. Porém não devem ser utilizados na nomenclatura de pacotes, classes, interfaces, métodos, construtores ou variáveis.

Não utilize espaços em branco em casos de variáveis incrementais (++) e decrementais (--).

Em expressões matemáticas que envolvam vários operandos, utilize os parênteses para evitar problemas de precedência, ainda que seja claro para você.

4. REFERÊNCIAS

Passei Direto (2013). “Boas práticas de programação Java”. https://www.passeidireto.com/arquivo/1807879/boas-praticas-java, Agosto.

PEDRALHO, André (2016). “Aplicando boas práticas em todo o processo do desenvolvimento”. https://www.devmedia.com.br/aplicando-boas-praticas-em-todo-o-processo-de-desenvolvimento/34407, Agosto.