Padrão para Códigos Java

Rafael G. M. Souza[1]

rafael.gorski@gmail.com

Abstract. *A source code standard helps development team to have a common code style to improve maintenance and refactoring of the code. This document presents a group of rules, notations and references for java coding. Additionally a checklist is available.*

Resumo. *Um padrão de código ajuda a equipe de desenvolvimento a ter um estilo comum que facilita a manutenção e refatoração de código. O seguinte documento apresenta um conjunto de regras, notações e referencias para codificação em Java. Adicionalmente um checklist é disponibilizado.*

1. Introdução

Este documento tem como objetivo apresentar padrões de código Java . De acordo com a Universidade Central de Washington [CWU 2014] cerca de 80% do custo do ciclo de vida de um *software* é gasto com manutenção. É muito raro que um código seja mantido pelo seu autor portanto os padrões de código ajudam na clareza de um *software*, isto causando que o código seja entendido mais rápido.

2. Contextualização

O presente documento é uma compilação de diversas fontes e tende a ser dinâmico sofrendo modificações e melhorias ao longo do tempo.

3. Convenção de Nomes

As seguintes regras gerais aplicam para escolha de nomes:

- Utilize descrição completa, não utilize abreviações. Por exemplo: primeiroNome, cnpjFilial.
- Evite nomes longos. Sugestão que seja menor que 15 posições.
- Evite nomes similares como por exemplo: produto, produtos.
- Auto-documentação, o código deve ser entendido por outros utilize nomes significativos como impostoICMS, impostoCOFINS no lugar de imposto1, importo2.

4. Formatação

4.1 Indentação

Todas as indentações devem ser com quatro espaços, utilizando espaços e não *tabs*.

Alinhamento vertical de chaves na mesma coluna que foram criadas.

Todos os *if, else, while* devem usar chaves mesmo sendo de uma linha.

É opcional a largura de linha de 80 ou 100 colunas.

Código em Java **com indentação**	Código em Java **sem indentação**
void foo() *{* *while (bar > 0)* *{* *//code* *bar--;* *}* *if (oatmeal == tasty)* *{* *//code* *}* *else* *{* *//code* *}* *switch (suckFactor)* *{* *case 1:* *//code* *break;* *default:* *//code* *break;* *}* *}*	*void foo()* *{* *while (bar > 0)* *{* *//code* *bar--;* *}* *if (oatmeal == tasty)* *{* *//code* *}* *else* *{* *//code* *}* *switch (suckFactor)* *{* *case 1:* *//code* *break;* *default:* *//code* *break;* *}* *}*

Figura 1. Exemplo de código indentado e não indentado.

4.2 Continuação de quebra de linha com mais 4 espaços.

Quando a linha de código tiver uma quebra é necessário começar o código na linha seguinte com acréscimo de 4 colunas a frente do alinhamento da linha superior.

4.2 Notação CamelCase

Todos objetos devem implementar a notação CamelCase [Wikipedia 2014, CamelCase], a tabela 1 apresenta exemplos por tipo de identificadores.

Tabela 1. Convensão CamelCase

Identificador	Regra	Exemplo
Classes, Interfaces, Enum e Anotações	Sentenças convertidas para nomes devem ser substantivos, com a primeira letra maiúscula. Evite Siglas, abreviações e mnemônicos.	*class Produto;* *class EnriquecimentoFatura;*
Métodos	Métodos devem ser verbos. Com a primeira letra em minúscula.	*executar();* *getNewClient();*
Variáveis	Devem iniciar com minúsculas. Não devem começar com (_, $, %, &). Variáveis temporárias dentro de estruturas como *for, while* devem ser i, j, k, m, n.	*int i;* *char cq;* *float containerWidth;*
Constantes	Devem ser todas as letras maiúsculas.	*int TIMEOUT = 60;* *int MAX_VALUE = 5;*

4.2 Espaços em branco

Todos os nomes de métodos devem seguir com o parênteses a esquerda.

```
nomeDoMetodo (i, j); // incorreto
nomeDoMetodo(i, j);  // correto
```

Todos os nomes de *arrays* devem seguir colchetes a esquerda.

```
args [0]; // incorreto
args[0];  // correto
```

Operadores binários devem ter espaços de separação em ambos os lados.

```
a=b+c;      // incorreto
a = b+c;    // incorreto
a=b + c;    // incorreto
a = b + c;  // correto
```

```
w = 2*x + 3*y;         // incorreto
w = 2 * x + 3 * y;     // correto
w = (2 * x) + (3 * y); // correto
```

Operadores de Incremento e Decremento imediatamente precedentes a variável.

```
contador ++; // incorreto
contador++;  // correto
i --;     // incorreto
i--;      // correto
```

Virgulas e Ponto-Virgulas sempre seguidos de espaços em branco.

```
for (int i = 0;i < 10;i++)   // incorreto
for (int i = 0; i < 10; i++) // correto
getCountryCode(type,country);  // incorreto
getCountryCode(type, country); // correto
```

Todos os *casts* devem ser sem espaço.

```
(MinhaClasse) v.get(3);  // incorreto
(MinhaClasse )v.get(3); // incorreto
(MinhaClasse)v.get(3);   // correto
```

5. Arquivo fonte

5.1 Codificação de Caracteres em UTF-8

Todos os arquivos fonte devem estar codificados com o padrão UTF-8.

5.2 Estrutura

Um código fonte consiste em:

1. Licença;
2. Declaração do pacote;
3. Declaração das importações;
4. Uma única declaração de classe

6. Comentários

Cada pacote, classe, interface, método e objeto deve ter uma explicação do seu propósito.

6.1 Javadoc

Todo o código deve seguir diretivas de javadoc da Oracle [Oracle 2014 Javadoc].

6.2 API

Caso o código for uma API então deve seguir diretivas de Requisitos para escrever uma especificação de API Java da Oracle [Oracle 2014 API].

6. *Checklist*

O seguinte *checklist* deve ser executado a cada nova classe gerada ou solução entregue.

1. Geral

[ ] Cada arquivo fonte java contém uma única classe ou interface dentro?

[ ] Cada arquivo fonte java contém a seguinte ordem: pacote, *imports*, declaração da classe?

[ ] A primeira linha, apos comentário, de cada arquivo fonte java contém declaração de *package*?

2. Estilo do Código

[ ] São utilizados quatro espaços em branco como identação?

[ ] Existem linhas maiores que 100 colunas?

[ ] Quando linhas quebram, as mesmas estão alinhadas com quatro colunas adicionais?

[ ] Cada linha contém uma única sentença?

[ ] Uma linha em branco para as seguintes ocorrências:

- Entre métodos.
- Entre uma variável local e um métodos.
- Entre blocos de lógica dentro do código.

[ ] Espaços em branco para as seguintes ocorrências:

- Separar palavras chaves e parênteses.
- Após virgulas em listas de argumentos.
- Todos os operadores lógicos.
- Na expressão *for*.
- Em *Casts*.

[ ] Nunca usar *break* ou *continue* em lógicas?

[ ] Uma declaração de variável por linha?

3. Comentários

[ ] Cada arquivo fonte java contém javadoc descrevendo a implementação?

4. Convenção de Nomes

[ ] Classes e Interfaces, nomes sendo substantivos. Não utilizar mnemônicos ou abreviações. Notação *CamelCase* com a primeira letra maiúscula?

[ ] Métodos, nomes sendo verbos. Notação *CamelCase* com a primeira letra minúscula?

[ ] Variáveis, significado claro. Notação *CamelCase* com a primeira letra minúscula?

[ ] Constantes, nomes em letras maiúsculas?

Bibliografia

[JavaRanch 2014] *Java Programing Style Guide*, <http://www.javaranch.com/styleLong.jsp> , Acesso em 28 Outubro 2014.

[Wikipedia 2014, Indentação] Indentação, <http://pt.wikipedia.org/wiki/Indentação>, Acesso em 28 Outubro 2014.

[Google 2014] Google Java Style. <http://google-styleguide.googlecode.com/svn/trunk/javaguide.html>, Acesso em 29 Outubro 2014.

[Wikipedia 2014, CamelCase] Camel Case, <http://pt.wikipedia.org/wiki/CamelCase>, Acesso em 8 Novembro 2014.

[CWU 2014] *Java programming Style Guide*, Central Washington University, <http://www.cwu.edu/~gellenbe/javastyle/index.html> , Acesso em 8 Novembro 2014.

[Oracle 2014, Javadoc] *How to Write Doc Comments for the Javadoc Tool*, Oracle, <http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html > , Acesso em 10 Novembro 2014.

[Oracle 2014, API], *Requirements for Writing Java API Specifications,* Oracle < http://www.oracle.com/technetwork/java/javase/documentation/index-142372.html > , Acesso em 10 Novembro 2014.

[Oracle 2014, Code conventions] , , Oracle, < http://www.oracle.com/technetwork/java/javase/documentation/codeconventions-141855.html > , Acesso em 04 Novembro 2014.

[Dalbery 2014] *Java coding standrads cheklist*. < http://users.csc.calpoly.edu/~jdalbey/SWE/CodeStdCheck.html >, Acesso em 10 Novembro 2014.