**SISTEMA DE HORÁRIOS**

**Padrão de Codificação Java**

**Versão <1.0>**

Sumário

[1- Introdução 3](#_Toc292828389)

[2- Objetivos 3](#_Toc292828390)

[3- Organização de um arquivo “.java” 3](#_Toc292828391)

[4- Comentários de classe 3](#_Toc292828392)

[5- Declarações de pacote 4](#_Toc292828393)

[6- Instruções de importação 4](#_Toc292828394)

[7- Declaração de classes ou Interface 5](#_Toc292828395)

[8- Documentação de Interfaces 7](#_Toc292828396)

[9- Métodos 8](#_Toc292828397)

[10- Documentação de Métodos 9](#_Toc292828398)

[11- **Padrões de Espaçamento** (Recuo; Comprimento e quebra de linha; Espaços em branco) 10](#_Toc292828399)

[11-1 Recuo 10](#_Toc292828400)

[11-2 Comprimento e Quebra de Linha 10](#_Toc292828401)

[11-3 Espaços em Branco 11](#_Toc292828402)

[12- Expressões e Blocos de Comando 12](#_Toc292828403)

[12-1 Expressões 12](#_Toc292828404)

[12-2 Comando if-else 12](#_Toc292828405)

[12-3 Estrutura switch 12](#_Toc292828406)

[13- PREFIXOS 13](#_Toc292828407)

[14- Bibliografia 14](#_Toc292828408)

# Introdução

Um padrão de codificação visa **facilitar** o entendimento do código do sistema por qualquer pessoa que conheça e siga os padrões de codificação, pois estabelece regras **definindo** como o código deve ser escrito.

Seguir padrões de codificação não é difícil apenas requer atenção.

# Objetivos

Apresentar os padrões de codificação Java da SUN e recomendações de documentação de classes que deverão ser utilizados no projeto do sistema de horários.

# Organização de um arquivo “.java”

O arquivo tem o **mesmo** nome da classe pública que o contém. Não se define mais de uma classe por arquivo, exceto para as classes internas.

A linguagem java **impõe** a seguinte organização ao código fonte:

declaração de pacote.

instrução de importação.

declarações de classes.

A seqüência de declarações **recomendada** nos arquivos é:

* Comentários de classe
* Declarações de pacote
* Instruções de importação
* Documentação da Classe
* Declaração de classes
* Variáveis estáticas
* Variáveis de instância
* Construtores
* Métodos

# Comentários de classe

Todos os arquivos fontes iniciam com um comentário no estilo da linguagem C que lista o nome da classe, versão, data e informações de copyright.

**Recomendações**

Se um arquivo **possuir** mais de uma classe ou interface, é inserida uma lista com uma pequena descrição de cada classe ou interface que compõe o arquivo.

É **recomendável** uma explicação que justifique a declaração de mais de uma classe por arquivo, pois Java só permite uma classe público por arquivo, dificultando a busca de classes não públicas.

|  |
| --- |
| **/\***  **\* Nome da classe**  **\***  **\* Informações de versão**  **\***  **\* Data**  **\***  **\* Copyright**  **\*/** |

|  |
| --- |
| **/\***  **\* Professor.java   \* Versão: 1.2   \* Data de Criação : 01/06/2004 \*  \* Copyright (c) 1994-1999 Sun Microsystems, Inc.   \* 901 San Antonio Road, Palo Alto, California, 94303,   \* U.S.A.**  **\* All rights reserved.   \*   \* This software is the confidential and proprietary   \* information of Sun   \* Microsystems, Inc. ("Confidential Information"). You   \* shall not disclose such Confidential Information and   \* shall use it only in accordance with the terms of the   \* license agreement you entered into with Sun.   \*/** |

# Declarações de pacote

Tem que ser a primeira declaração válida.

Todas as letras minúsculas.

Não devem conter caracteres especiais, como *underscores*, ou caracteres específicos de um idioma.

Segue o padrão de nomeação das **URLs**, só que invertido. E são nomeados de acordo com o seu contexto.

Ex: **org.mysql.bd, dai.academico.utilitario**

# Instruções de importação

Declaração de Importação

Exemplo: **import java.awt.peer.CanvasPeer;**

# Declaração de classes ou Interface

Começa com letra maiúscula seguida por minúsculas. Exceto nos casos que a sua abreviação seja mais sugestiva que o nome completo. Exemplo: **Professor.java** ou **XML.java**

Devem ser nomeadas como substantivos.

O nome da classe é sempre no singular.

Quando a palavra for composta, a separação entre elas é feita por uma letra maiúscula.

Não usa-se artigos, preposições para conectar substantivos e adjetivos, nem caracteres específicos de uma língua como é o caso do “ç” e os acentos da língua portuguesa.

Ex: **Conta**, **ContaEspecial**, **LinkedHashMap**

A chave de abertura “**{**“ deve aparecer na mesma linha da declaração da classe.

Para efeito de legibilidade, sempre que possível, o parâmetro de retorno deve ser movido para o final do método.

|  |
| --- |
| **public** **class** **DVDDatabase {** |

**Documentação de classes ou interfaces usando Javadoc**

Cada classe começa com um comentário “**/\*\* ... \*/**” descrevendo:

O propósito da classe.

Instruções de uso.

E, opcionalmente, alguns exemplos para facilitar o uso da mesma.

Em seguida, têm-se lembretes sobre possíveis melhoramentos e defeitos existentes na classe.

No final do comentário, **adiciona-se** o nome dos autores e referências úteis para o entendimento da classe.

**Em seguida**, tem-se a declaração do nome da classe.

|  |
| --- |
| **/\*\***  **\* Descrição da classe**  **\***  **\* Exemplo de uso:**  **\* <pre>**  **\* algum Código**  **\* </pre>**  **\***  **\* Limitações:**  **\***  **\* @author**  **\* @version**  **\* @see java.awt.Component**  **\*/** |

|  |
| --- |
| /\*\*  \* O método main na classe DOSClient possibilita as seguintes  \* funções:  \* <ul>  \* <li> Exibe todos os DVDs do Banco de Dados.</li>  \* <li> Adiciona um DVD no banco de dados. </li>  \* <li> Remove um DVD do banco de dados. </li>  \* <li> Modifica o DVD, sendo localizado pelo código. </li>  \* <li> Tenta alugar um DVD. </li>  \* <li> Devolução de um DVD alugado. </li>  \* </ul>  \*  \* Limitações: Este programa apresenta uma interface de console que \* poderá ser substituída por uma interface Swing.  \*  \* @author Fulano de Tal  \* @version 1.0  \* @see sampleproject.db.DVDDatabase  \*/  public class DOSClient { |

Após a documentação e a declaração do nome da classe. As declarações dentro de uma classe seguem, respectivamente, a ordem apresentada:

Constantes.

Variáveis de classe (estáticas).

Variáveis de instância.

Construtores.

Métodos de classe (estáticos).

Métodos de instância.

Quanto aos modificadores de acesso, primeiro declaram-se as variáveis públicas, depois as protegidas, as sem modificadores, e, por último, as privadas.

# Documentação de Interfaces

As declarações de interface **seguem a ordem** apresentada

Comentários da interface “ /\*\* ...\*/ ”

Declaração da Interface.

Constantes: ***na seguinte ordem*** públicas, protegidas, sem modificadores (pacote), privadas.

Métodos: os métodos devem ser agrupados por **funcionalidade.**

Recomendações sobre a utilização de constantes, variáveis de classe e de instância:

**Constantes –** para definir uma constante uma variável deve-se **rotular** como estática e final.

Escritas com todas as letras maiúsculas.

Quando composta por duas ou mais palavras a separação é feita por um *underscore* ( \_ )

**Ex**: **TAXA**, **VALOR\_MEDIO**

A SUN sugere as seguintes regras de nomeação:

**Atributos** (variáveis) – escritas com letras minúsculas.

Mesmo podendo iniciar com ( \_ ou $) **não** o faça

E **somente** variáveis temporárias **devem** usar nome com apenas **um** caractere.

Quando a palavra for composta, a separação entre elas é feita por uma letra maiúscula

**saldo // Correto**

**strTitulo // Correto**

**floatSaldo// Correto. Palavras reservadas podem ser   
 // usadas como parte do identificador**

**lâmpada // Correto, mas inadequado**

**User\_name // Correto, mas não segue as regras de nomeação**

**Recomendações**

Fazer uma declaração por linha.

int nivel; // nível de indentação  
int tamanho; // tamanho da tabela

**Documentação de uma Variável de Instância**

|  |
| --- |
| /\*\*  \* This number uniquely identifies a DVD.  \*/  **private** String upc; // Holds the record UPC identification  /\*\*  \* Stores the release date of the film in month /day/ year format.  \*/  **private** Date year = new Date(); // Holds the movie's release date |

# Métodos

Métodos **construtores** devem ser listados **antes** de métodos estáticos e de instâncias.

Na assinatura dos métodos **não** deve haver espaços entre o nome do método e o parêntese de abertura “**(**“

A chave de abertura “**{**“ deve **aparecer** na *mesma linha* da declaração do método

Os métodos são agrupados por funcionalidade e não pela forma de acesso ou sua condição de estático ou de instância.

**Métodos** de acesso a atributos iniciam com **get** ou **set** e finalizam com o nome da variável tendo a primeira letra da variável maiúscula.

**Métodos:** Tem a mesma regra das variáveis

Normalmente são verbos no infinitivo representando a utilidade do método, com exceção dos métodos que retornam um **boolean**, que devem **começar** com um verbo no presente.

**Não** se utiliza nenhum caractere especial (**ç**, **é**, **ã**, ...)

Os nomes **não** devem ser abreviados (torna o código mais fácil de compreender).

Exemplos de nomes de métodos:

**void adicionarLivro(Livro livro)**

**void removerLivro(Livro livro)**

**boolean existeUsuario(int codigoUsuario)**

**double getSaldo() // método de acesso**

**void setNome(String nome)// método modificador**

# Documentação de Métodos

Todo método contém um cabeçalho de documentação que fornece informações suficientes para seu entendimento e uso adequado. Inicialmente, documenta-se o que o método faz e porque faz. Após isto, relaciona-se todos os parâmetros necessários para chamar o método, sua cláusula de retorno, e as possíveis exceções que pode levantar

Exemplo

|  |
| --- |
| /\*\*  \* Locates a DVD using the upc identification number.  \*  \* **@param** upc The UPC of the DVD to locate.  \* **@return** The DVD object which matches the upc.  \* **@throws** IOException Indicates there is a problem   \* accessing the data.  \* **@throws** ClassNotFoundException Indicates the DVD class   \* definition cannot be found.  \*/  **public** DVD **getDVD**(String upc) **throws** IOException,   ClassNotFoundException **{**  **return** **retrieveDVD**(upc);  **}** |

**Recomendações**

Caso a decisão de visibilidade do método possa ser questionada, documenta-se a razão pela qual foi tomada esta decisão.

Se necessário, são declaradas ao final do comentário referências a outras classes e métodos, assim como, a data da criação do método.

### Padrões de Espaçamento (Recuo; Comprimento e quebra de linha; Espaços em branco)

## 11-1 Recuo

Cada nível de recuo deve ter quatro espaços

O início de comentários de declarações de pacote, instruções de importação, declaração de interfaces e classes **não** devem ser recuados.

Variáveis estáticas, variáveis de instância, construtores, métodos e seus respectivos comentários **devem** ser recuados em um nível.

Dentro de construtores e métodos as variáveis locais, instruções e seus comentários devem ser recuados em um nível.

|  |
| --- |
| **public** **class** Indent {    **static** **int** staticVar = 7;    **public** **Indent**() { }    **public** **static** **void** **main**(String [] args) {    **int** x = 0;    **for**(**int** z=0; z<7; z++) {  x = x + z;  **if** (x < 4) {  x++;  }  }  }  } |

## 11-2 Comprimento e Quebra de Linha

A regra geral é que uma linha **não** pode ter mais que 80 caracteres

Algumas diretrizes para fazer a quebra de linha

Insira a quebra depois de vírgulas

Use a quebra antes de um operador.

A nova linha é alinhada com o começo da expressão do mesmo nível da linha anterior.

|  |
| --- |
| **/\* exemplo de uma quebra de linha \*/**  System.out.println(((x \* 42) + (z - 343) + (x % z ))  + numberOfParsecs);  **/\* example de quebra de linha para método \*/**  x = doStuffWithLotsOfArgs(coolStaticVar, instanceVar,  numberOfParsecs, reallyLongShortName, x, z); |

## 11-3 Espaços em Branco

São usados para **tornar** o código mais legível e menos amontoado.

Use **uma linha** em branco entre

Métodos e construtores.

Depois da última variável de instância.

Dentro de um método entre variáveis locais e a primeira instrução.

Dentro de um método para separar segmentos lógicos de código.

Antes de comentários de uma linha ou bloco.

Use **duas linhas** em branco entre seções maiores do código fonte.

O pacote, as instruções de importação, a classe ou a interface.

**Use espaços em branco**:

Entre operadores binários

|  |  |
| --- | --- |
| **a += c + d;** | **a = (a + b) / (d \* c);** |

Depois de vírgula em uma lista de argumentos

|  |
| --- |
| **resultado = soma(arg1, arg2);** |

**Depois de expressões em uma instrução for**

|  |
| --- |
| **for (exp1; exp2; exp3) {**  ***comandos*;**  **}** |

**Entre uma palavra reservada e um parêntese**

|  |
| --- |
| **while (true) {**  **comandos;**  **}** |

**Depois de um cast**

|  |
| --- |
| **livro = (Livro) objeto.getMidia();** |

# Expressões e Blocos de Comando

## 12-1 Expressões

Expressões simples

Cada linha deve conter uma instrução.

Exemplo:

contador++; // Correto   
 indice--; // Correto   
 contador++; indice--; // Evitar!

O comando return

Uma sentença return com valor de retorno não utiliza parêntesis, a menos que a sentença fique mais clara.

Exemplo

return lista.size();

return (tam > MAX ? tam : VALOR\_PADRAO);

## 12-2 Comando if-else

É usado com as chaves – “ **{ }**” – para evitar ambigüidade no escopo do comando.

Estilos de formatação:

|  |
| --- |
| if (condição1) {  comandos;  } else if (condição2) {  comandos;  } else {  comandos;  } |

## 12-3 Estrutura switch

|  |
| --- |
| **switch** (variável) {  **case** ABC:  comandos;  break;  **case** DEF:  comandos;  break;  **case** XYZ:  comandos;  break;  **default**:  comandos;  break;  } |

1. PREFIXOS  
   (Objetivo: manter a uniformidade do código)

Prefixo para objetos da API JDBC

|  |  |
| --- | --- |
| **Interfaces** | **Prefixo** |
| Connection | com |
| Statement | stmt |
| PreparedStatement | pstmt |
| ResultSet | rs |

Prefixos para componentes de interface gráfica

|  |  |
| --- | --- |
| **Componentes** | **Prefixo** |
| **Containers** | |
| JFrame | frame |
| JDialog | dialog |
| JPanel | painel |
| JSplitPane | splitPane |
| JScrollPane | scrollPane |
| JTabbedPane | tabbedPane |
| JToolBar | toolBar |
| JInternalFrame | iFrame |
| JDesktopPane | desktop |
| **Componentes** | |
| JButton | btn |
| JLabel | lbl |
| JTextField | tf |
| JTable | tbl |

# Bibliografia

Sun Microsystem. Code Conventions for the JavaTM Programming Language, Revisado em 20/04/1999.

Java 2 Certificação para Programador & Desenvolvedor Java 2. Autor: Kathy Sierra, Bert Bates (Capítulo 11 – Padrões de Codificação).

The SUN Certified Java Developer Exam with J2SE 1.4.

Padrão de Codificação Java. Qualiti Software Processes.