SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA · UFV
CAMPUS FLORESTAL

UFV

CCF 322 - Engenharia de Software II

PADRÕES DE CODIFICAÇÃO

Equipe 5º Ano - “BYTECRAFT”

Florestal - MG
2025

Sumário

Esse documento indica os padrões de codificação que devem ser seguidos por todos os membros da equipe de desenvolvimento. Esse conjunto de regras tem como objetivo facilitar a análise e o entendimento dos códigos desenvolvidos em Java pela equipe. É necessário a compreensão de todos e a utilização das regras abaixo:

1. Regras de Nomeação Geral

Regra geral para nomear variáveis, métodos, classes, pacotes e outros elementos do código, garantindo clareza, consistência e manutenção.

- Não utilizar acentos, pontuação, caracteres especiais, símbolos ou espaço em branco.

Exemplo aceitável: class CalculadoraCientifica, class Maca, class Calendario

Exemplo não aceitável: class Calculadora Científica, class Maçã, class Calendário

- Evitar abreviações. O nome deve ser coeso, indicando o propósito do elemento.

Exemplo aceitável: int numeroDeAlunos, public calculaMedia

Exemplo não aceitável: int numAlun, public calcMed

2. Regras de Nomeação para Classes e Interfaces

- Devem ser nomeadas em PascalCase (primeira letra de cada palavra maiúscula), os nomes devem ser substantivos ou frase nominal clara.

Exemplo aceitável: class Usuario, class PessoaJuridica

Exemplo não aceitável: class usuario, class pessoaJuridica

- Interfaces são objetos mais gerais que são implementados por uma ou mais classes, portanto, sua nomenclatura deve remeter a essa característica.

Exemplo aceitável: interface Televisão, interface DispositivoMobile

Exemplo não aceitável: interface TelevisãoSamsung, interface Iphone8

3. Regras de Nomeação para Métodos

- Devem ser nomeados em camelCase (primeira letra minúscula, seguintes em maiúscula) e seu nome deve ser um verbo ou frase verbal, indicando ação.

Exemplo aceitável: calcularMedia, obterNomeUsuario

Exemplo não aceitável: CalcularMedia, nome_usuario

4. Regras de Nomeação para Variáveis

- Devem ser nomeados em camelCase (primeira letra minúscula, seguintes em maiúscula) e seu nome deve ser descritivo, refletindo o uso da variável.

Exemplo aceitável: int idadeAluno, double valorTotalCompra

Exemplo não aceitável: int temp, int Idade_aluno, double vltComp

5. Regras de Nomeação para Constantes

- Devem ser nomeadas em maiúsculas, com palavras separadas por underline(_).

Exemplo aceitável: int MAX_USUARIOS

Exemplo não aceitável: int MaxUsuarios

6. Regras de Nomeação para Pacotes

- Devem ser nomeados em minúsculas, não utilizando de underline ou maiúsculas.

Exemplo aceitável: com.empresa.sistema.usuario; edu.ufv.bytecraft.jogo

Exemplo não aceitável: com.Empresa.Sistema.Usuario; edu_ufv.Bytecraft.jogo

7. Regras de Nomeação para Enumerações

- Devem ser nomeadas em PascalCase (primeira letra de cada palavra maiúscula) e os valores do enum devem ser em maiúsculas.

Exemplo aceitável: public enum StatusPedido {PENDENTE, APROVADO, CANCELADO}

Exemplo não aceitável: public enum status_pedido {pendente, aprovado, Cancelado}

8. Regras de Nomeação para Comentários

Comentários devem explicar o porquê do código, não apenas o que ele faz. Não se deve escrever comentários redundantes, que apenas repetem o código

- Tipos de comentários:

- **Comentário de linha única (//):** Utilizado para observações curtas e diretas. Devem ser colocados acima da linha de código

Exemplo aceitável:

```java
// Verifica se o usuário já está ativo antes de realizar login
if (usuario.isAtivo()) {
    realizarLogin(usuario);
}
```

Exemplo não aceitável:

```java
if (usuario.isAtivo()) { // verifica se o usuário está ativo
    realizarLogin(usuario);
}
```

- **Comentário de bloco (/* ... */):** Utilizado para explicar trechos de código mais longos ou complexos. Não devem ser usados para desativar trechos inutilizados.

Exemplo aceitável:

```java
/*
 * Este bloco inicializa a conexão com o banco de dados.
 * Utiliza o padrão Singleton para evitar múltiplas conexões.
 */
Connection conexao = ConexaoBanco.getInstance().abrirConexao();
```

Exemplo não aceitável:

```java
/*
Connection conexao = DriverManager.getConnection("...");
Statement st = conexao.createStatement();
*/
```

- **Comentário de Documentação - Javadoc (/** ... */):** Utilizado em classes, métodos e atributos públicos. Devem explicar o propósito,

parâmetros, retorno e exceções. Posteriormente, pode ser utilizado para gerar documentação automática.

Exemplo aceitável:

```java
/**
 * Calcula a média de notas de um aluno.
 *
 * @param notas Lista de notas do aluno
 * @return média aritmética das notas
 * @throws IllegalArgumentException se a lista estiver vazia
 */
public double calcularMedia(List<Double> notas) {
    if (notas.isEmpty()) {
          throw new IllegalArgumentException("Lista de notas não pode estar vazia");
    }
                                                                    return notas.stream().mapToDouble(Double::doubleValue).average().orElse(0.0);
}
```

Exemplo não aceitável:

```java
/** Calcula a média */
public double calcularMedia(List<Double> notas) { ... }
```