**Guia de Implementação**

SUMÁRIO

[1. Introdução 2](#_Toc428177384)

[1.1. Definições, Acrônimos e Abreviações 2](#_Toc428177385)

[1.2. Referências 2](#_Toc428177386)

[2. Organização e Estilo do Código 2](#_Toc428177387)

[3. Comentários 2](#_Toc428177388)

[4. Nomeação 2](#_Toc428177389)

[5. Declaração 2](#_Toc428177390)

[6. Expressões e Sentenças 2](#_Toc428177391)

[7. Gerenciamento da Memória 2](#_Toc428177392)

[8. Tratamento de Erros e de Exceções 2](#_Toc428177393)

[9. Portabilidade 2](#_Toc428177394)

[10. Reutilização 2](#_Toc428177395)

[11. Problemas de compilação 2](#_Toc428177396)

[12. teste unitário 2](#_Toc428177397)

[13. Diretrizes Gerais 2](#_Toc428177398)

# Introdução

Este documento contem o padrão de organização, comentários, nomeação e declaração para o desenvolvimento do sistema para o Socio Pedagógico do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – Campus Salto na plataforma Android Studio, em que os desenvolvedores deverão ter conhecimento e utilizar para fazer o desenvolvimento do código na padronização.

## Definições, Acrônimos e Abreviações

Android Studio 2.3.1 – IDE utilizada para o desenvolvimento do sistema

Android SDK 25.0.5 - Ferramentas que fazem interface com a plataforma Android, estas ferramentas são necessárias para o desenvolvimento de aplicativos Android.

## Referências

Guia – Oracle Java Code Conventions.pdf

Guia – Padrões de Codificação Java.pdf

# Organização e Estilo do Código

Um arquivo consiste de seções que devem ser separadas por linhas em branco e um comentário opcional identificando cada seção.

Arquivos com mais de 2000 linhas são complicados e devem ser evitados.

O código deve ter a seguinte ordem no desenvolvimento:

* Comentário Inicial
* Packages e Imports
* Variáveis
* Declarações de classes e interface
* Construtores
* Métodos

# Comentários

**Comentário padrão/inicial**

Sempre ao criar uma nova classe deve ter o comentário inicial com todas as informações.

/\*

\* < Login.Java>

\*

\* Data de criação: < 09/06/2017 >

\*

\* Desenvolvido por Ruan.

\* Fábrica de Software – SGSP Mobile

\*/

**Comentário JavaDoc de Classes**

Antes do nome de cada classe deverá haver um comentário no padrão JavaDoc com os seguintes itens:

/\*\*

\* <Descrição da funcionalidade da Classe>

\* @author: <Ruan>

\*/

**Comentário JavaDoc de Métodos**

Antes de cada método deverá haver um comentário no padrão JavaDoc com os seguintes itens:

/\*\*

\* <Descrição completa das funcionalidades do método>

\* @return <tipo do retorno do método: String, int, etc>

\* @param <parâmetro1> <Descrição do parâmetro>

\* @param <parâmetro2> <Descrição do parâmetro>

\* @exception <exceção retornada>

\*/

# Nomeação

**Variáveis:** Não poderá começar com a primeira letra maiúscula e também não pode começar com numero ou caracter especiais. A nomeação terá a primeira palavra coincidindo com oque será utlizada e sempre que precisar de mais uma palavra tem que começar com a letra maiúscula.

**Ex:**

String nome;

String dataNascimento;

**Classes:** A primeira palavra terá que ser em maiúscula, e não poderá começar com número ou caracter especiais.

**Ex:**

Formulario.class

Login.java

**Métodos:** Não poderá começar com a primeira letra maiúscula e também não pode começar com numero ou caracter especiais. A nomeação terá a primeira palavra coincidindo com oque será utlizada e sempre que precisar de mais uma palavra tem que começar com a letra maiúscula.

**Ex:**

private void acessarWeb(){}

# Declaração

**Declaração de variável:**

Recomenda-se uma declaração por linha, incentivando o uso do comentário. E sempre colocar as variáveis no começo do bloco.

**Ex:**

int nível; // nível de indentação

int tamanho; // tamanho da tabela

Mas também poderá estar utilizando todas as variaveis na mesma linha sem está em tipo diferente.

**Ex:**

Int nível, tamanho;

Exceção para o índice de for (loops) pode ser declarado na instrução:

Ex:

for (int i = 0; i < maxLoops; i++) { ... }

**Declaração de Classe e interface:**

Ao codificar classes e interfaces Java, as seguintes regras de formatação devem ser seguidas:

* Nenhum espaço entre um nome de método e o parêntese "(" iniciando sua lista de parâmetros
* Open brace "{" aparece no final da mesma linha que a instrução de declaração
* Closing brace "}" inicia uma linha por si mesma recuada para corresponder à sua declaração de abertura correspondente, exceto quando é uma instrução nula o "}" deve aparecer imediatamente após o "{"
* Os métodos são separados por uma linha em branco

**Ex:**

Classe Sample extends Object {

int ivar1;

Int ivar2;

Amostra (int i, int j) {

ivar1 = i;

Ivar2 = j;

}

Int emptyMethod () {}

...

}

# Expressões e Sentenças

[Uma descrição de como as expressões e as sentenças devem ser escritas.]

# Gerenciamento da Memória

[Um descrição de como a memória deve ser gerenciada.]

# Tratamento de Erros e de Exceções

[Esta seção fornece regras e diretrizes sobre como os erros e as exceções devem ser tratados.]

# Portabilidade

[Esta seção descreve regras especiais para assegurar a portabilidade nas diferentes plataformas e compiladores.]

# Reutilização

[Esta seção fornece regras e diretrizes para facilitar a reutilização.]

# Problemas de compilação

[Esta seção descreve as dependências entre os módulos, diretivas de compilação etc.]

# teste unitário

[Esta seção deve descrever como os testes unitários devem ser desenvolvidos. Por exemplo, em Java deve-se usar o Framework de testes Junit.]

[Se existir um documento específico, deve-se referenciá-lo aqui.]

[Se existir uma instruções de boas práticas na biblioteca de componentes, pode-se também referenciar a biblioteca.]

# Diretrizes Gerais

[Esta seção deve descrever, caso existam, diretrizes e estratégias a serem usadas no desenvolvimento].