Documento de estilo e design

Camila Ferreira

Carolina Ramalho

Jads Victor

1.1 Cabeçalho

O código deve conter um cabeçalho no início da classe no estilo /* ... */ contendo o nome da classe, uma breve descrição da mesma e um aviso de direitos autorais.

1.2 Declaração de pacotes

A primeira linha não comentada é a declaração do Package e o mesmo deve seguir o seguinte padrão: nome do pacote sem nenhum caracter especial, caso o nome do pacote tenha duas palavras, escreve-las juntas com a primeira letra da segunda palavra em maiúsculo. Como segue o exemplo: nomePacote

1.3 Declaração de classes e interfaces

A tabela a seguir mostra as partes de uma classe e a ordem em que devem aparecer.

Partes da declaração Classe/interface	
Comentário de documentação da classe/interface (/** ... */)	Documentação Javadoc
Declaração **class** ou **interface**	
Comentários de implementação da classe/interface (/* ... */), se necessário	Este comentário deve conter qualquer informação não apropriada para o comentário da documentação
Variáveis de classe (**static**)	Primeiro as variáveis **public**, depois **protected**, depois **private**.
Variáveis de instância	Primeiro **public**, depois **protected**, depois **private**
Construtores	

Métodos	Esses métodos devem ser agrupados por funcionalidade

1.4 Indentação

1.4.1 Comprimento da linha

Evitar linhas com mais de 80 caracters.

1.4.2 Quebra de linha

Quando uma expressão não couber em uma linha, quebre-a com os seguintes princípios gerais:

- Quebrar depois da vírgula

- Quebrar antes do operador

- Alinhar a nova linha com o mesmo nível do início da expressão da linha anterior

1.4.3 Comentários

1.4.3.1 Da função

Deve seguir o padrão: // comentário

1.4.3.2 Do bloco

Deve seguir o padrão: /* comentário */

1.4.4 Declarações

1.4.4.1 Número por linha

Uma declaração por linha é o recomendado, visto que essa abordagem incentiva a comentar.

1.4.4.2 Localização

Colocar a declaração das variáveis o mais próximo possível da sua utilização

1.4.4.3 Inicialização

Inicializar variáveis locais onde são declaradas, exceto quando o valor inicial depende de alguns cálculos que ocorrem primeiro.

1.4.4.4 De classes e interfaces

As seguintes regras devem ser seguidas:

- Sem espaço entre o nome do método e o parênteses que inicia a sua lista de parâmetros .

- Abrir colchetes “{“ aparece no final da mesma linha que o comando da declaração.

- Fecha colchetes “}” inicia em uma linha própria recuando para combinar com a abertura de colchetes da correspondente declaração, exceto quando a declaração é nula neste caso o “}” deve seguir imediatamente depois do “{“.

- Métodos devem ser separados por uma linha em branco.

1.4.4.5 Declarações simples

Cada linha deve conter apenas uma declaração.

1.4.4.6 Declarações complexas

Declarações compostas são declarações que contem instruções entre chaves, e estas devem utilizar colchetes até mesmo em declarações individuais, quando elas fazem parte de uma estrutura de controle, como a instrução if-else ou for. Isto torna mais fácil adicionar declarações sem acidentalmente gerar erros, devido à falta de colchetes.

1.4.4.7 Declaração return

Uma declaração return com um valor não deve usar parênteses a menos que seja uma expressão.

1.4.4.8 Declaração if-else, if-else-if-else

Deve seguir a seguinte forma:

```
1. if (condição) {
2.     declaração;
3. }
4.
```

```
if (condição) {
    declaração;
} else {
    declaração;
}

if (condição) {
    declaração;
} else if (condição) {
    declaração;
} else if (condição) {
    declaração;
}
```

1.4.4.9 Declaração for

A declaração for deve seguir a seguinte forma:

```
for (inicialização; condição; update) {
    declaração;
}
```

1.4.4.10 Declaração while

A declaração while deve seguir a seguinte forma:

```
while (condição) {
        declaração;
}
```

1.4.4.11 Declaração do-while

A declaração do-while deve seguir a seguinte forma:

```
do {
```

```
    declaração;
} while (condição);
```

1.4.4.12 Declaração switch

A declaração switch deve seguir a seguinte forma:

```
switch (condição) {
case ABC:
    declaração;
    /* passa através */
case DEF:
    declaração;
    break;

case XYZ:
    declaração;
    break;

default:
    declaração;
    break;
}
```

Toda vez que um case passa através (não inclui a instrução break), adicione um comentário onde a instrução break normalmente estaria.

1.4.4.13 Declaração try-catch

A declaração try-catch deve seguir a seguinte forma:

```
try {
    declaração;
} catch (ExceptionClass e) {
    declaração;
}
```

1.4.5 Espaço em branco

Deve-se usar uma linha em branco nas seguintes situções:

- Entre métodos

- Entre as variáveis locais em um método e a sua primeira declaração

- Antes de um comentário de bloco

- Entre seções lógicas dentro de um método para melhorar sua legibilidade

Deve-se usar espaços em branco nas seguintes situações:

- Uma palavra-chave seguida por um parentese deve ser separada por um espaço em branco. Exemplo:

```
1. while (true) {
2.     ...
3. }
```

Nota: Um espaço em branco não deve ser utilizado entre o nome do método e a abertura do parentese. Isso ajuda a distinguir palavra chave de chamada de método.

- Um espaço em branco deve aparecer depois da vírgula em uma lista de argumentos.

- Todo operador binário exceto “.” deve ser separado de outros operadores por espaço. Espaço em branco nunca deve separar operadores unários, como incremento (“++”) e decremento (“--”).

- As expressões em uma declaração for deve ser separada por um espaço em branco.

1.5 Convenções dos nomes

Tipo de Identificador	Regra para Nomeclatura	Exemplo

Classes e Interfaces	Os nomes de classes devem ser substantivos, com a primeira letra de cada palavra, inclusive interna, maiúscula	class Student interface StudentOfLaw
Métodos	Métodos devem ser verbos, iniciando a palavra em minúsculo e com as palavras internas em maiúsculo.	run(); getBackground();
Variáveis	Devem iniciar com minúscula e palavras internas em maiúsculas.	int myWidth;
Constantes	Devem ter todas as letras maiúsculas, e caso necessite separar por “_”	int MIN_WIDTH;