Baseado em: https://google.github.io/styleguide/javaguide.html#s1-introduction

1. Linguagem

Todos os nomes, quer sejam de variáveis, funções, classes, etc. têm de ser escritos em inglês. Todos os comentário devem também ser escritos em inglês, à exceção de possíveis nomes cuja tradução não faça sentido, i.*e.* nome de uma pessoa, estas palavras devem ser escritas usando exclusivamente caracteres ASCII, i.*e.* Andre Brandao.

2. Formatação

2.1 Chavetas

As chavetas devem seguir o estilo K&R. Exemplo:

```
// Acceptable
public void method() {
   // Code
   …
}
```

```
// Not acceptable
public void method()
{
    // Code
        …
    }
```

As chavetas devem ser sempre usadas mesmo quando só existe uma instrução. Exemplo:

```
// Acceptable
if (condition) {
   System.out.print();
}
```

```
// Not acceptable
if (condition)
System.out.print();
```

2.2 Indentação: 4+

Cada bloco de código novo deve conter a indentação aumentada em 4 espaços.

2.3 Uma instrução por linha

Cada instrução deve ser seguida por uma quebra de linha.

2.4 Limite de caracteres por linha

Cada linha só pode conter no máximo 100 caracteres.

2.5 Espaço em branco na horizontal

Um espaço na horizontal deve ser acrescentado nos seguintes locais:

- Separar qualquer palavra reservada do caracter ‘(‘ (parêntesis aberto).
- Separar qualquer palavra reservada do caracter ‘}‘ (chaveta fechada).
- Antes de qualquer chaveta aberta, ‘{‘, com exceção de quando esta é precedida por um parêntesis aberto, ‘(‘, ou por uma chaveta aberta, ‘{‘.
- Depois do parêntesis fechado, ‘)’, presente num *cast*.
- Entre o tipo e o nome da variável numa declaração. Exemplo: Double number;

2.5 Arrays

Os parêntesis retos fazem parte do tipo e não do nome da variável. Exemplo:

```
// Acceptable                // Not acceptable
String[] var;                String var[];
```

2.6 Switch statements

Todos os casos devem conter um terminação abrupta (break, continue, return ou levantar uma Exceção)

2.7 Anotações

Todas as anotações devem ser seguidas por uma quebra de linha. Exemplo:

```
// Acceptable                          // Not acceptable
@Override
@Nullable                         @Override @Nullable
public void method() {            public void method() {
   // Code                            // Code
   …                                  …
}                                        }
```

Exceção:

Anotações aplicadas a campos específicos. Exemplo:

```
public void testMethod(@Mock mock) {};
```

2.8 Comentários

2.8.1 Em linha

```
// Inline comment
```

2.8.2 Em bloco

```
/*
 * Block Comment
 */
```

2.8.3 Javadoc

Ver https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html

2.9 Modificadores

Classificadores de membros e classes devem aparecer pela ordem recomendada pela *Java Language Specification*:

public protected private abstract default static final transient volatile synchronized native strictfp

3 Convenção de nomes.

3.1 Regras comuns a todos os identificadores

Todos os identificadores só podem conter letras e dígitos presentes na tabela ASCII

3.2 Nomes de classes

Devem seguir o formato UpperCamelCase. Exemplo:

Container, ImmutableSet, etc.

3.3 Nomes de métodos, variáveis

Devem seguir o formato lowerCamelCase. Exemplo:

logger, number, numberOfElements, etc.

3.4 Camel Case

Para evitar ambiguidades o formato camel case deve ser construído da seguinte forma:

1. Converter todas as letras para minúsculas.
2. Converter para maiúscula a letra inicial de cada palavra.
3. Juntar todas as letras para formar o nome da variável.

Exemplo: XML HTTP request → XmlHttpRequest.

4 Práticas de programação

4.1 Não ignorar exceções

Todas as exceções *apanhadas* devem conter algum tipo de resposta. Exemplo:

```
// Acceptable
try {
   // Code
   …
} catch(Exception e) {
   System.exit(1);
}
```

```
// Not acceptable
try {
   // Code
   …
} catch(Exception e) {}
```

Quando fôr verdadeiramente apropriado não haver nenhum tipo de resposta, tal deve ser justificado usando um comentário.

4.2 Comentários desnecessários

Deve evitar-se a utilização de comentários para explicar código que seja perfeitamente compreensível. Exemplo:

```
// Se a cidade fôr igual a "Aveiro"
if (city == "Aveir") {
   // Code
   …
}
```

4.3 Agrupamento de código

Deve existir uma separação vertical, com recurso a quebras de linha para separa contextos de código. Exemplo:

```
// Get list of all employees.
EmployeeRep empRep = new EmployeRep();
Employee[] employees = empRep.getAllEmployees();
                                                    ← Quebra de linha extra.
// Increase salary in 100 for every employee.
for (Employee e : employees) {
   e.setSalary(e.getSalary() + 100);
}
```