




Woodland
Academy

3.4 コーディング規約



Shape Your Future

Java コーディング規約

- コードのスタイルには明確な公式定義はなく、暗記する必要はありません。常に「**可読性**」と「**統一性**」の2つを意識してください。
- 実際の開発において、各社が自らのコーディング規約を作ることが少なくありません。開発する前に、必ず確認してください。Java 以外の言語にも規約があるはずです。
- 本講義で使われる Java の規約について説明します。主に、Oracle の公式なアドバイスと、実際の一般的なスタイルが考慮されます。
- 参考 : Google Java Style Guide :
 <https://google.github.io/styleguide/javaguide.html>

キャメルケース

- **キャメルケース**[\[Camel Case\]](#) :
 - **英文字、数字**のみを含む。
 - 最初の単語を除く、各単語の**最初の文字を大文字**にします。
 - 数字は独立した単語として扱う。
- 一般的にキャメルケースは、**最初の単語の頭文字が小文字**の場合を指します。英語では「**camelCase**」で表記することもあります。頭文字を大文字にするキャメルケースは、**パスカルケース**[\[Pascal Case\]](#)ともいって、英語では「**CamelCase**」で区別を付けます。
- 元々全部大文字の単語であっても、頭文字だけ大文字にしてください。例えば「getHTTP」は「getHttp」で書くべき。

スネークケース

- **スネークケース**^[Snake Case] :
 - 英文字、数字、アンダースコア「_」のみを含む。
 - アンダースコアで複数の単語を区切る。
 - 数字は独立した単語として扱う。
- 一般的に、全部小文字か、全部大文字のどちらにします。
英語では、それぞれ「**snake_case**」「**SNAKE_CASE**」と表記します。

パッケージ、ファイルシステム

- パッケージ名は**小文字のスネークケース**で命名すること。
(会社によってはすべて小文字で、アンダーバーなしで表記することもあります。)
- 各コードファイルにはトップレベルクラス（内部クラスじゃないクラス）が **1 つ**だけ含まれるように。
- コードファイルは **UTF-8** 形式で保存すること。

識別子のネーミング

- クラス名は**大文字で始まるキャメルケース**にすること。
- 変数名やメソッド名は、**小文字で始まるキャメルケース**にすること。
- 定数（static final）と列挙子は**大文字のスネークケース**にすること。
- 簡単で有意義な英語の名称が望まれる。

特別な識別子

- セッターは「**set**」で始まり、ゲッターは「**get**」で始まる。
 - ブール変数のゲッターとセッターは、次のように書いてもいい：

```
1 public void setOpened(boolean isOpened) {
2     this.isOpened = isOpened;
3 }
4
5 public boolean isOpened() {
6     return isOpened;
7 }
```

- インタフェースの実装クラスが 1 つしかない場合、「**インタフェース名+Impl**」と命名する：

```
interface UserDao {}
class UserDaoImpl {}
```

インデント

- インデントはスペースのみにすること。
- 1 レベルのインデントは **4 スペース** とする。(ただし、ウェブ開発において 2 スペースにすることも多い。)
- switch 文の各 case ブロックの文も 1 レベルのインデントすること：

```
1 switch (input) {  
2     case 1:  
3     case 2:  
4         method1( );  
5         break;  
6     default:  
7         method2( input );  
8 }
```


スペース

- ほとんどの単語や記号は、スペース 1 つで区切ること。
- スペースを入れる必要のない位置：
 - 括弧（「()」 「{}」 「[]」）の**内側**。
 - クラス名またはメソッド名の**後**に付く括弧：
 - メソッド：method()。
 - 配列：int[][]。
 - ジェネリクス：List<T>。
 - コンマ「,」、セミコロン「;」とコロンの「:」の**前**。
 - 例外として、for-each 文の中のコロンの前もスペースを入れる。
 - ピリオド「.」とメソッド参照記号「::」の**両側**。

中括弧

- コードブロックの中括弧の書き方には **K & R style** がある：
 - 「{」の前で改行しない
 - 「{」の後にはすべて改行
 - 「}」の前は改行
 - 文が終了していない限り、「}」の後に改行

```

1  if (condition()) {
2      try {
3          something();
4      } catch (ProblemException e) {
5          recover();
6      }
7  } else if (otherCondition()) {
8      somethingElse();
9  } else {
10     lastThing();
11 }

```

複数行表記

- 1 行の長さは **60 文字**以内に納める。(基準は各社によって異なります。)
- 以下の位置で改行を加え、一つの文を複数行にしてもいい：
 - 演算子の**前**（代入演算子を除く）
 - 代入演算子「=」の**後**
 - カンマ「,」の**後**
 - 括弧の**前後**
- 改行したら、**2 レベル以上**にインデントすること。

複数行の文の例

Example



```
1 List<String> list = data.stream( )  
2     .sorted( )  
3     .map(i -> "" + i)  
4     .filter(s -> s.length( ) > 10  
5         || s.contains("7"))  
6     .toList( );
```

その他

- 複数の（非アクセス）修飾子を同時に使用する場合は、次の順番に従い：

abstract default static final transient volatile synchronized native strictfp

- メソッドをオーバーライドする際は、必ず @Override アノテーションを付けておくこと。
- 「*」での一括インポートや、一括静的インポートを避ける。
- 中括弧は（ブロックが一行だけであっても）省略しない。ただし、ラムダ式を書くときだけは省略してもいい。
- 長い数字は「_」で区切り、3桁の間隔にすることができる。long タイプの数字は「L」で終わる：

```
long longNumber = 1_234_567_890L;
```



Q&A

