コーディング規約

[1. はじめに 3](#_Toc177545631)

[2. ソースファイル 3](#_Toc177545632)

[2.1. Javaソースファイル名 3](#_Toc177545633)

[2.2. JSPファイル名 3](#_Toc177545634)

[2.3. ファイルの文字コード（UTF-8） 4](#_Toc177545635)

[2.4. ソースコードの整形 4](#_Toc177545636)

[3. 命名規則 5](#_Toc177545637)

[3.1. パッケージ名 5](#_Toc177545638)

[3.1.1. パッケージに含まれるソースコードの意味を表す単語 6](#_Toc177545639)

[3.2. クラス名とインターフェース名 6](#_Toc177545640)

[3.2.1. DAOクラス名 6](#_Toc177545641)

[3.2.2. DTOクラス名 7](#_Toc177545642)

[3.2.3. Serviceクラス名 7](#_Toc177545643)

[3.2.4. Servletクラス名 7](#_Toc177545644)

[3.2.5. Filterクラス名 7](#_Toc177545645)

[3.2.6. ユーティリティクラス名 7](#_Toc177545646)

[3.2.7. インターフェース名 8](#_Toc177545647)

[3.2.8. 抽象クラス名 8](#_Toc177545648)

[3.2.9. 例外クラス名 8](#_Toc177545649)

[3.2.10. テストクラス名 8](#_Toc177545650)

[3.3. メソッド名 9](#_Toc177545651)

[3.4. 定数名 9](#_Toc177545652)

[3.5. （定数でない）フィールド変数名 9](#_Toc177545653)

[3.6. パラメータ変数名 10](#_Toc177545654)

[3.7. ローカル変数名 10](#_Toc177545655)

# はじめに

　本書は、総合演習である電子商取引システム開発において、Java言語でコーディングをする際のルールや指針を示すものです。

本書の目的は、プロジェクトチーム全体で同じルール、指針を共有し、メンテナンス性を重視した、読みやすいプログラムコードを実現し、チーム内やレビュー担当者とのコミュニケーションを円滑にすることです。

# ソースファイル

## Javaソースファイル名

　Javaソースファイルのファイル名は、**トップレベルのクラス名に拡張子[.java]を加えたもの**にしてください。

例：public class TableListView のファイル名は、TableListView.java にする。

## JSPファイル名

　JSPファイルのファイル名は、**すべて小文字とし、単語の区切りをアンダースコア「 \_ 」**にしてください。特に**Servletクラスをforwardした先となるJSPファイル**の場合、**Servletクラスを連想できるような名前**にして、**拡張子[.jsp]を加えた**ものにしてください。

例：header.jsp、user\_Servicedy.jsp

例：TableListViewServletクラスのforward先となるJSPファイルの場合

table\_list\_view.jsp

## ファイルの文字コード（UTF-8）

　ソースファイルの文字コードは**UTF-8**にしてください。

## ソースコードの整形

　ソースコードは、**プロジェクトで使用する統合開発環境における、標準で設定されているコードスタイルのフォーマッターに準じて、整形する**ようにしてください。

例：

* 演算子や制御構文の両サイドに半角空白を1つ入れる

for (int i = 0; i < listArray.size(); i++) {

System.out.println(listArray.get(i));

}

* 中括弧は宣言と同じ行を開始位置にする

if (this.isExit() == true) { // 中括弧の開始は宣言と同じ行にする

}

特に、Eclipseを利用している場合、**Eclipseの書式のフォーマット機能（Ctrl + Shift + F）を活用**してください。

# 命名規則

## パッケージ名

　パッケージ名は、すべて**小文字**にしてください。また、特別な場合を除き、**記号やダラー「 $ 」、アンダースコア「 \_ 」などは使用しない**でください。

例：com.bh.ecsite

　パッケージ名は、ユニークなパッケージ名とするため、**組織のトップレベルドメイン名とサブドメインリストを逆順にして始め、その後に、システム名や、パッケージに含まれるソースコードの意味を表す単語を付与**してください。

例：トップレベルドメインが bh.comで、サブドメインが ecsite の場合、パッケージ名は com.bh.ecsiteで始める。

例：システム名が xyz で DAOクラスを含むパッケージ名は com.bh.ecsite.xyz.daoとする。

### パッケージに含まれるソースコードの意味を表す単語

　パッケージに含まれるソースコードの意味を表す単語は次のようにしてください。

DAOクラスを含むパッケージ：dao

DTOクラスを含むパッケージ：dto

Serviceクラスを含むパッケージ：service

Servletクラスを含むパッケージ：servlet

Filterクラスを含むパッケージ：filter

ユーティリティクラス（staticな共通処理をまとめたクラス）を含むパッケージ：util

**その他のクラスについては、クラスのカテゴリを意味する小文字**にしてください。単語から**クラスのカテゴリが想像しにくいような単語は控える**ようにしてください。（othersなどカテゴリが想像しにくい単語は控える）

例：データベースの接続を管理する

コネクションマネージャクラスを含むパッケージ：database

## クラス名とインターフェース名

　クラス名は、**UpperCamelCase（先頭を大文字とし、単語の区切りを大文字）**にしてください。**記号やダラー「 $ 」、ハイフン「 - 」、アンダースコア「 \_ 」は使用しない**でください。

例：テーブルリストを表示する（tabel list view）クラスの場合、クラス名はTableListViewとする。

### DAOクラス名

　DAOクラスを表すクラス名は、**末尾をDAO**にしてください。

例：TableListViewDAO

### DTOクラス名

　DTOクラスを表すクラス名は、**末尾をDTO**にしてください。

例：TableListViewDTO

### Serviceクラス名

　Serviceクラスを表すクラス名は、**末尾をService**にしてください。

例：TableListViewService

### Servletクラス名

　Servletクラスを表すクラス名は、**末尾をServlet**にしてください。

例：TableListViewServlet

### Filterクラス名

　Filterクラスを表すクラス名は、**末尾をFilter**にしてください。

例：TableListViewFilter

### ユーティリティクラス名

　ユーティリティクラス（共通処理を1か所にまとめるためのクラス）を表すクラス名は、**末尾をUtil**にしてください。

例：TableListViewUtil

### インターフェース名

**先頭に I を付けて、以降はクラス名と同様**にしてください。

例：テーブルリストを表示する（tabel list view）ことを意味するインターフェースの場合

ITableListViewとする。

また、クラスに能力を加えるための mix-in として利用させたい場合、**その能力を形容詞とし、末尾をable**にしてください。

例：Runnable、Serializableなど

### 抽象クラス名

**先頭にAbstractを付けて、実装するサブクラス名を連想させる名前**にしてください。

例：TableListViewクラス、ImageListViewクラスで実装をする抽象クラスの場合

AbstractListViewとする。

### 例外クラス名

　例外クラスは、独自に作成しないようにしてください。やむを得ず例外クラスを独自に作成することとなった場合、**末尾をException**にしてください。

### テストクラス名

　ホワイトボックステストを実施するためのテストクラスを作成する場合は、**テスト対象のクラス名の末尾にTestを加えて**ください。

例：TableListViewService クラスをテストするためのテストクラス名は

TableListViewServiceTest とする。

## メソッド名

　メソッド名は、**lowerCamelCase（先頭を小文字とし、単語の区切りを大文字）**にしてください。記号やハイフン「 - 」、アンダースコア「 \_ 」は使用しないでください。

例：メッセージを送信するメソッドの場合sendMessageとする。

停止をするメソッドの場合stopとする。

## 定数名

　定数名（static final）は、**すべて大文字とし、単語の区切りをアンダースコア「\_」**にしてください。

例：

static final int NUMBER = 5;

static final String ERROR\_MESSAGE = “エラーメッセージ”;

※修飾子がstaticのみ、もしくはfinalのみの場合、定数とならないため、後述する定数でないフィールド変数を参照してください。

## （定数でない）フィールド変数名

　（定数でない）フィールド変数名は、**lowerCamelCase（先頭を小文字とし、単語の区切りを大文字）**にしてください。記号やハイフン「 - 」、アンダースコア「 \_ 」は使用しないでください。

## パラメータ変数名

　コンストラクタやメソッドに渡すパラメータの変数名は、**lowerCamelCase（先頭を小文字とし、単語の区切りを大文字）**にしてください。記号やハイフン「 - 」、アンダースコア「 \_ 」は使用しないでください。

例：

private void updateTableView(int tableViewId , TableViewDTO tableViewDTO) {

～処理～

}

## ローカル変数名

　ローカル変数名は、**lowerCamelCase（先頭を小文字とし、単語の区切りを大文字）**にしてください。記号やハイフン「 - 」、アンダースコア「 \_ 」は使用しないでください。

ただし、変数のスコープが狭いループカウンタなどには、単純な名前を利用してもよいこととします。

例：for文の場合

for(int i = 0;i < xArray.length; i++) {

for(int j = 0;j < yArray.length; j++) {

～処理～

}

}

例：拡張for文の場合

List<String> strList = this.getStrList();

for(String s : strList) {

～処理～

}

List< TableViewDTO > tableViewList = this.getTableViewList();

for(TableViewDTO list : tableViewList) {

～処理～

}

以上