Javaの イケてるコード 残念なコード

# 記述するコードを減らす技術

Acroquest Technology
石田 浩司

この連載では、どのようにJavaをコーディングすればイケてるコードになるのか、そのテクニックを学んでいきます。今回は、開発者が記述するコード量を減らすことで、イケてるコードになる「コードを書かない技術」についてみていきましょう。

## コードを書かない＝イケてるコード

開発者である以上、皆さんも何らかのコードを記述していると思います。しかし、いくらイケてるコードを書いたとしても、そのコードが記述する必要のない場合はどうでしょうか。無駄なコードを記述しなければ、あれこれとイケてるコードを書くために頭を悩ませる必要もなくなります。

筆者は、どのようにイケてるコードを書くのか、と同じぐらい「いかに自分でコードを書かずに済ませるか」を考えてプログラミングしています。プログラムを記述する上で、自分で一からすべてを実装しなければまともに動く成果物を作れない、と考えている人はいませんか。昔のことを思い出すと、筆者も自分ですべてを作る必要があると考えていましたし、何よりプログラミング自体が楽しくて、何でもかんでも自分で実装していました。

しかし、イケてるコードにするためには、プログラマ本人が書かなければならない領域と、実際には書かなくても良い領域に分けることが重要なのです（**図1**）。

**図1●プログラムにはプログラマが書く領域と書かない領域がある**

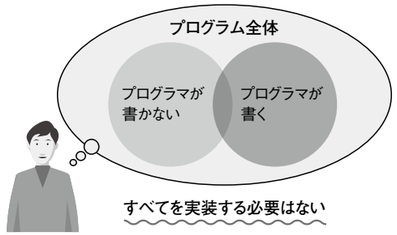



**図2●Eclipseによってmainメソッドを自動生成する**

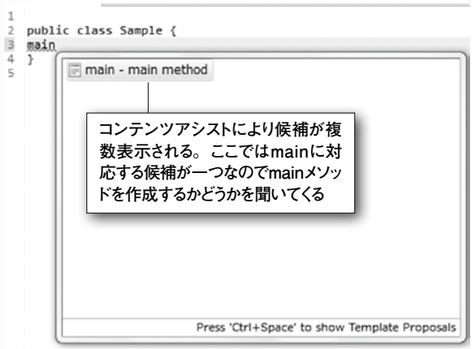
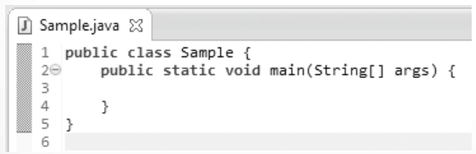



**図3●自動生成されたmainメソッド**




## 実装をライブラリやツールに頼る

プログラムは実装範囲が大きくなればなるほど、バグを埋め込んでしまう可能性が高くなります。プログラミングは人間が行うので、書き間違えや勘違い、スペルミスなどが発生してしまいます。例えば、10000文字の文章を一字一句間違えずにボールペンで書くことを想像してみてください。できると思いますか。おそらくほとんどの人はできないでしょう。プログラムでも同じです。

もし、プログラムを間違いなく正確にコーディングしてくれる、あるいは実装した結果をあらかじめ用意してくれる代替品があったらどうでしょう。コーディングしなくても機能は実装できますし、何よりヒューマンエラーを引き起こす可能性が低くなります。「コードを書かない技術」とは、プログラムをしないということではなく、正確にプログラムを記述してくれる「何か」にプログラミングを任せてしまうことを指しているのです。

例えば、皆さんがJavaを書く際には、必ずmainメソッドを記述します。このmainメソッドは、必ず書かなければならない上に、毎回同じコードを書きます。これを自動的に正しく記述する方法はないのでしょうか。

## Eclipseに実装を任せる（コンテンツアシスト）

ここで、イケてるコードを記述する方法の一つとして、統合開発環境のEclipseを使ったmainメソッドの自動生成をみていきましょう。クラスを作成したら、mainと記述して「Ctrl」キー+「Space」キーを押してみましょう。**図2**のような選択画面が現れます。これは、Eclipseの「コンテンツアシスト」機能によるものです。この状態でmainメソッドを選択すると、**図3**のmainメソッドが自動生成されます（図3はインデントをオートフォーマットで整理したもの）。このコンテンツアシストは、Eclipseを使っている人なら誰でも利用できる便利な機能の一つです。

変数だけでなく、Javaが用意したメソッドなどであれば、同じくコンテンツアシスト機能によって自動生成できます。mainメソッドを例にしましたが、他にも代表的なものとして、

1. `if / for / while`文
2. 「`sysout`」で`System.out.println`の自動生成
3. 「`syserr`」で`System.err.println`の自動生成

などをEclipseのコンテンツアシストで実装できます。コンテンツアシストを駆使することで、コンピュータに実装を任せることができ、間違いなく実装できる＝イケてるコード、に近づけられるのです。

## getter/setterの生成

データ用のクラスを作成した際に、定義しているインスタンス変数のgetterやsetterをすべて手で書いていたりしませんか。データクラスを作る際に毎回必要になるこのルーチンワークの煩雑さもEclipseなら自動生成で解決できます。

例えば、**リスト1**のようなデータ保持用のUserInfoクラスがあったとします。それぞれの変数をprivate修飾子で宣言しているので、別のクラスから利用するためにはgetter/setterを宣言しなくてはなりません。

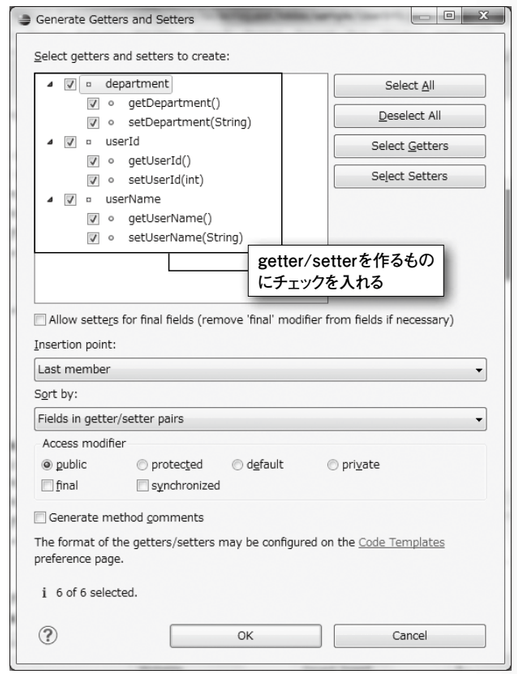
**リスト1●getterやsetterを必要とするデータ用のクラス**

```
public class UserInfo {
/** ユーザー ID */
private int userId;

/** ユーザー名 */
private String userName;

/** 部署 */
private String department;
…
}
```

**図4●Eclipseによるgetterとsetterの生成画面**




ここで、Eclipseの自動生成機能の出番になります。Eclipse上で生成したい変数を選択し、右クリックをするとメニューが出てきます。次に、「Source」→「Generate Getters and Setters…」を選択します。すると、**図4**のような画面が表示されます。

実際に作成したい変数のgetterとsetterにチェックを入れて、OKボタンを押しましょう。getter/setterが自動生成されます*1。

*1 Eclipseでは、Javadocの自動生成設定を行うだけでgetter/setter生成時に一緒にJavadocを記述してくれる機能も備えています。