5章
Javaで書く「文」の形

Chapter
05

5-0 この章で学ぶこと

前の4章までは主に，Javaのソースプログラムの中に書いていく文字と名前と演算子記号について基本的なルールを学びました。たとえば，名前の最初の文字が数字であってはいけない，というルールがありましたね。

この章では，文字や名前や演算子から一歩前進して，プログラムの中に書いていく「文」というものを考えます。プログラムがやる仕事，つまりあなたがコンピュータにやらせる仕事はすべて，文として書きます。たとえば，次のようなものも文です：

```
int a;         //宣言文(変数を宣言している)
a = 3 + 128;  //代入文(変数に値を代入している)
```

文の最後に必ず書くセミコロン';'は，“ここがひとつの文の終わりだよ”を意味する重要なシルシです。

Javaプログラムの文は，それをコンピュータの命令に翻訳するコンパイラ（javacなど）が正しく解釈できる文であると同時に，そもそもソースプログラムは人間が読んで理解できるために存在しているのですから，人間にとって読みやすく，理解しやすい書き方であることが重要です。

往々にして，プログラムとして正しく書けているけれど非常に読みづらい，わかりにくい，という文を，基礎を勉強していない初心者は書きがちです。人間が読んでわかりづらいソースプログラムは，バグが発生しやすく，今後の改良作業も困難です。

プログラムの文の書き方の，「Java コンパイラ javac にとってはこれが正しい」という面は，必ずそう書かなければならない『ルール』です。一方，「人間にとってこのほうが読みやすく理解しやすい」という面は，先人たちの多くの経験をふまえて推奨されている『慣習』です。

この章では，『ルール』と『慣習』の両方を勉強します。『慣習』については，この章に書かれているのと違うことを主張する先輩に出会うこともあるかと思います。しかしとりあえずプログラミングの学習の初期には，あまりよそ見をしないで，ここで私がおすすめする慣習を覚えて実践してください。

5-1 文の基本的な形

5-1-1 文末のセミコロン

Java プログラムの文の第一のルールは，「最後にセミコロンを書く」ことです。

たとえば次のようにセミコロンを書き忘れると：

```
a = 3 + 128
```

javac コンパイラは，この文がまだ先へ続いていると解釈して，続きを探そうとします。そして，最初に出会ったセミコロンまでの全体が正しい文の形をしていたら，それをひとつの文と解釈します。だからほとんどの場合 javac は，あなたの意図とは違うなにかヘンなものを文として把握してしまうでしょう。

その，コンパイラが文として把握したヘンなものの内容と性質によっては，javac は「文末に;がありません」というわかりやすいエラーメッセージではなく，初心者が理解に苦しむ奇妙なエラーメッセージを吐きますから，ご用心ください。javac のエラーメッセージを和英対象で併記し，その一部を解説したページがここにあります：http://homepage1.nifty.com/algafield/javac-emsg.html。

プログラムのひとつの文が，ひとつの完結した処理内容を表しますから，人間が見ると正常に思える次のような文も，javac から見るとエラーになります：

```
a = 3 + 128 b =  4879 + 34;
```

これは「a という変数に 3+128 の結果を代入する」処理と，「b という変数に 4879 + 34 の結果を代入する」処理という，計二つの処理を書き表していますから，そのひとつひとつが，セミコロンで終わる完結した文でなければなりません。ですから，正しくはこう書きます：

```
a = 3 + 128;
b = 4879 + 34;
```

5-1-2 文は原則として 1 行にひとつ

javac コンパイラにとって（というより，すべての Java コンパイラにとって），セミコロンが文の終わりを示し，改行は特別の意味を持ちません。改行は Java コンパイラにとって，スペース ' ' と同じものですから，次のように，ひとつの行に複数の文を書いてもエラーにはなりません：

```
a = 3 + 128; b =  4879 + 34;  //これは二つの正しい文だ
```

しかし慣習としては，複数の文を 1 行に書くと人間にとって読みづらくなるので，1 文 1 行主義が推奨されています。とても短い文でも，ひとつの文を 1 行に書いたほうが読みやすいのです：

```
a = 3 + 128;
b = 4879 + 34;
```

以上述べたように，1 行に複数の文を書くことは原則として禁止ですが，逆に，ひとつの文を複数の行にまたがって書くことはあります。それは，その文があまりにも長すぎるときです。たとえば，こんな場合です：

```
//ひとつの文を 2 行にたたむ例
Sytem.out.println
("じゅげむじゅげむごこうのすりきれぱいぽぱいぽふーりんがんのちょうすけ");
```

このように，ひとつの文を複数の行にたたむときは，ではどこでたたむかについて，やはり一般的に勧められている慣習があります。その慣習の具体例は，今後徐々に見ていきましょう。Java の本家である Sun Microsystems 社の，この文書（http://java.sun.com/docs/codeconv/）の 4.2 節が，“長い行をどこで折りたたむか”という話題を扱っています。

5-1-3 各要素間のスペース

たとえば演算子記号（+, −, *, /など）は，名前の中に使えないルールになっていますから，コンパイラが次のようなものを見たとき，これを一つの名前と誤解することはありません:

```
a=3+128;
```

コンパイラはこれを正しく，「3 + 128 の結果を a に代入する」という意味の文として読みます。

しかし，ここでの問題はコンパイラではなく人間の目です。ほとんどの人にとって，名前や演算子記号など文の各要素が，すきまなくギッシリつながっている上のような形よりも，あいだにスペースのある次のような形が読みやすくて，瞬間的に理解しやすいでしょう：

```
a = 3 + 128;
```

ただし，文の終わりとセミコロンの間にスペースがあると，逆に，視覚的に終わりが明確でなくなるおそれがありますから，上の 128 とセミコロンの場合のように，スペースがないほうがよいと思います。こうなっていると：

```
a = 3 + 128 ;
```

あれ？，128 のあとにまだ何かあったのかな？，という曖昧な雰囲気になってしまいます。

スペースについて重要なことは，個々の独立した要素間にはスペースがあってもよいし，また，次の 5-2-1 節で説明する宣言文の場合のように，二つの異なる名前やキーワードを併記するときは，間にスペースがないと“ひとつの名前”になってしまいますから絶対にスペースが必要です。

しかしまた逆に，ひとつの要素がスペースによって区切られていてはいけません。たとえば次のように書くと，コンパイラは奇妙なエラーメッセージをつぶやくでしょう：

```
//ひとつの数がスペースで分断されている
a = 3 + 12 8 ;
```

上の例では，整数値の 128 というひとつの要素が，スペースで分断されて 12 8 になっています。コンパイラは，この文が a=3+12 という代入文であるとみなし，そ