ステップ30

Java
ワークブック

CUTT
カットシステム

もくじ

Step 00

はじめに

Javaは、オブジェクト指向を特徴とするプログラミング言語です。とはいうものの、慣れないうちは、いきなりメソッドやらフィールドやらといわれてもわからないと思います。そこで、オブジェクト指向に特有の用語や考え方はとりあえずおいて、プログラムを書いていくことにします。そして、その過程のところどころに説明をはさむことで、オブジェクト指向に慣れてもらおうと思います。

本書の構成

この本は、Javaの基礎を学ぶための、30ステップからなるワークブックです。言語を習得するには、体系的かつ網羅的に学んでいくやりかたと、実践的に使いながら必要な事柄を一点突破で学んでいくやりかたの2通りがあると思いますが、本書は後者のアプローチをとっています。こうした学び方の場合、何か目標があったほうが話の流れがわかりやすくなります。そこで、本書ではクイズゲームの作成を課題に掲げ、それに必要となる知識と関連する話題を、その都度説明していきます。

30のステップは、大きく5部に分かれています。

第1部（Step01～03）　とりあえず動くプログラムを書くことを目標に、最もシンプルなプログラムとその構成について説明します。

第2部（Step04～13）　データ型、変数、配列といったデータの格納方法、代入や加減乗除などの演算方法、そしてプログラムの流れの制御といった、Javaの基礎部分を説明します。ここまでで、ゲームに必要となる問題と回答選択肢の表示ができる程度までプログラムが書けるようになります。

第3部（Step14～23）　課題のクイズゲームを作成するのに必要な、**基本ライブラリ**とよばれる各種の機能を提供するクラスの説明をします。こうした機能の中には、文字列や数値の操作、ファイルの読み込み、ランダム数の生成などがあります。

第4部（Step24～26）　ここまでの知識を使って、完動するクイズゲームを構築します。また、このクイズゲームをより使いやすくするために、エラーの対処策やコマンドオプションについて説明します。

第5部（Step27～30）　最後に、これまで学習してきたことをオブジェクト指向の枠組みでとらえなおした上で、プログラムをクラスの集合体に書き換えます。

課題のクイズゲーム

本書で取り上げるのは、地理の問題に出てくるような、国名から首都を当てるゲームです。以下、本書ではこれを「首都名当てクイズ」とよびます。ゲームでは図0.1に示したように、問題として国名が、その下に選択肢として首都名がいくつか表示されます。プレーヤーは、選択肢の中からその国名の首都を選びます。結果は即座に採点され、次の問題へと続きます。問題を一定の数だけこなすとゲームは終了し、得点が表示されます。

このゲームを作り上げるには、大まかに以下の処理を行う必要があります。

1　国名/首都名のリストをファイルから読み込む
2　ランダムに国名と選択肢の首都を選択して、問題を表示する
3　プレーヤーからの回答を受け取る
4　正誤を判定した上で採点する

本書ではこれらをどのようにして行うか、順次解説していきます。

```
C:¥temp¥java>java Quiz
Q1: ナイジェリア
1. タシケント
2. キガリ
3. アブジャ
4. アンマン
==> 2
外れ! 正答はアブジャ
Q2: 南アフリカ
1. ポルトープランス
2. シンガポール
3. プレトリア
4. リヤド
==> 3
当たり!!
```

図0.1●首都名当てクイズの画面例

●補記

本書には、下記のような「補記」と記した枠がところどころに出てきます。ここでは、本文に関連はするものの必須とはいえない話題や、本書の範囲を超える話題を扱います。

補 記

名前の意味

この「Java」という名前はどこからきたのでしょうか？　諸説ありますが、コーヒーのジャワを採って命名されていることは、Javaのロゴが湯気の立つコーヒーカップであることからも確実でしょう（図0.2）。一説によれば、Javaを開発した人たちが足しげく出入りしていた近所のコーヒー屋から採られたとされています。

図0.2●Javaのロゴ

Javaがコーヒー由来というのは、Javaのクラスファイル（実行時に使うファイルです。**バイトコード**ともいいます）の先頭に、必ず「CA FE」と書かれていることからもわかります。16進数で書かれているので、メモ帳やWordのような普通のアプリケーションではそのようには読めませんが、コンピュータの扱う**バイト**をそのまま16進数の数値として表示できるアプリケーション（Unixならod）で読めば、この隠しメッセージをみることができます。

では、コーヒーのジャワという名前はどこからきたのでしょうか？　これは、他のコーヒーの名称同様に地名がもとで、インドネシアを形成する島のひとつからきています。Javaと書いてジャワと読むのは変だと思うかもしれませんが、現地語ではJawaになります。ちなみに、インドネシア首都のジャカルタはジャワ島にあります（図0.3）。

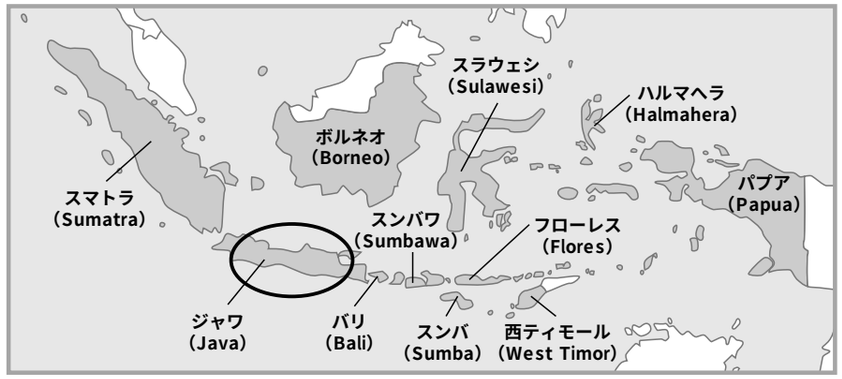

図0.3●インドネシア地図

Step 01

プログラムの作成

Javaを使ってプログラムを書くとは、どういうことでしょうか。また、どのように実行するのでしょうか。まずは、最も簡単なプログラムを例に、ソースファイルの作成とコンパイル、そして実行の方法を理解しましょう。

●プログラムの作成と実行の手順

プログラムの作成と実行は、図1.1に示した順番で行います。

図1.1●Javaプログラムの作成手順

最初にプログラムを書きます。中身については追って説明しますが、最初に知っておかなければならないのは、プログラムは**エディタ**を使って書き、ファイルとして保存しなければならないということです。このようにして作成されたファイルを**ソースファイル**といいます。「ソース（source）」は「源」という意味です（料理に使うソースはsauceです）。

エディタは、OS（オペレーティングシステム）についてくる簡単なものでかまいません。Windowsならメモ帳やワードパッド、Mac OSならテキストエディット、Unixならばviやemacsあたりが標準的です。これらを使って、メモや文書を作成する要領で書いていきます。ただし、ファイルへの保存は**テキスト形式**で行わなくてはなりません（図1.2）。たとえば、ワードパッドの標準的な保存形式はRTF（リッチテキスト）ですが、これら非テキスト形式はフォントの種類や行間の長さなどのデータを埋め込むため、書いたものそのままがファイルにならないからです。

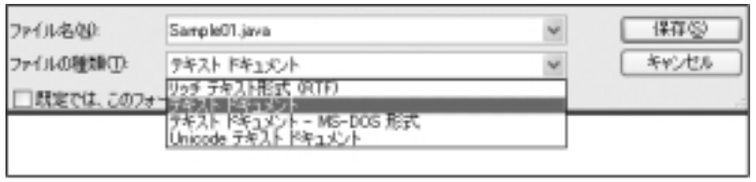

図1.2●テキスト形式での保存を指定（ワードパッドの場合）

続いて、作成したソースファイルを**コンパイル**します。コンパイルとは、ソースファイルに文字で書かれたJavaのプログラムから、Javaが実行できる別の形式のファイルを生成することです。これには**javac**というJavaコンパイラを使います。生成したファイルを**クラスファイル**といいます。

クラスファイルを生成したら、**java**コマンドを使って実行します。これは、WindowsやUnixといったOS上のアプリケーションとしてjavaを実行し、そのjavaが指定のクラスファイルを実行するという、2段階のステップが踏まれることを意味しています。通常のアプリケーションのように、目的のアプリケーションを直接実行するのではなく、間接的になっている点に注意してください。

補 記

以下本書でJava（Jが大文字）と書いた場合はプログラミング言語としてのJavaを、java（jが小文字）と書いた場合はクラスファイル実行プログラムであるjava（Windowsならばjava.exe）を指します。

●環境の確認

まずはじめに、Javaプログラムの作成と実行が自分のコンピュータ上で可能かどうかを調べましょう。

Javaの開発元であるサン・マイクロシステムズ社はJava関係のソフトウェアを各種提供していますが、ここで必要になるのはJavaプラットフォームである**J2SE**（Java 2 Standard Edition）です。これにはJREとJDKの2種類が用意されていますが、ここで必要なのは、実行環境と開発ツールの両方を備えた**JDK**（Java Development Kit）の方です。これが自分のコンピュータに正しくインストールされていれば、javacコマンドを実行できるはずです。コマンド引数に何も指定しないで実行すると、次のように操作方法が表示されます（以下、本書では端末のプロンプトを「%」記号で表します）。

```
% javac
使い方: javac <options> <source files>
使用可能なオプションには次のものがあります。
.......
```

javacコマンドを実行して、「プログラムが存在しない」や「内部コマンドまたは外部コマンド、操作可能なプログラムまたはバッチ ファイルとして認識されていません」などのエラーが表示されたら、JDKは正しくインストールされていません。http://java.sun.com/からダウンロードし、指示にしたがってインストールしてください。

JDKの最新版はバージョン1.5（2006年11月現在）です。「java -version」コマンドを実行すれば、その環境のバージョン番号を調べることができます。

```
% java -version
java version "1.5.0_04"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_04-b05)
Java HotSpot(TM) Client VM (build 1.5.0_04-b05, mixed mode, sharing)
```

9行目の「.（ドット）」で分けられた最初の2つの数字がバージョン番号で、上記では1.5になっています。それ以降の数字はアップデート等を示す補助番号なので、特に気にする必要はありません。

JDKそのものの他に、Javaが提供する基本ライブラリの使い方を説明したドキュメントは必須です。これは**APIガイド**とよばれ、先述のサイトからWebブラウザで読むことができます。また、全文をダウンロードすることもできます。なお、ソースファイルの拡張子は必ず「.java」です。

●ソースファイルを作成する

では、ソースファイルを作成しましょう。以下に掲げたプログラムを、行番号を示す各行頭の数字を除いて、エディタで正確に書き写してください。大文字小文字の違いにも意味があるので、注意してください。ファイル名は、Sample01a.javaとします。なお、ソースファイルの拡張子は必ず「.java」です。

Sample01a.java

```
1 public class Sample01a {
2   public static void main (String[] args) {
3     System.out.println ("Hello World!");
4   }
5 }
```

補 記

例題プログラム名

例題として掲げるプログラムには、いずれもSampleNNX.javaという名前をつけています。「NNX」の「NN」はStep番号、「X」はa、b、c、……と続くそのStep内での子番号です。いくつか登場するので、C:\samples（Windows）や/home/samples（Unix）といった専用のディレクトリを作っておくとよいでしょう。

●コンパイルする

コンパイルには、javacコマンドを使います。コマンド引数に作成したファイル名を指定して実行します。

```
% javac Sample01a.java
```

すると、ソースファイルとおなじディレクトリに、拡張子だけが「.class」に置き換わったSample01a.classというクラスファイルが生成されます。

ソースファイル中にJavaの文法にのっとっていない不正な内容があれば、エラーが表示されます。以下は3行目のprintlnの「l（小文字のエル）」を、「1（数字のイチ）」と誤ってタイプしたときのエラーメッセージです。

```
% javac Sample01a.java
Sample01a.java:3: シンボルを見つけられません。
シンボル: メソッド print1n(java.lang.String)
場所    : java.io.PrintStream の クラス
    System.out.print1n ("Hello World!");
```

エラーが発生したら、いま一度、サンプルソースファイルと自分のファイルを見比べてください。

●実行する

作成したプログラムはjavaコマンドを使って実行します。javaコマンドの引数に、javacコマンドで生成したクラスファイルを指定して ENTER を押し、以下のように表示されれば成功です。

```
% java Sample01a
Hello World!
```

ここで重要なのは、生成されたクラスファイルの名前がSample01a.classであるのに、末尾の拡張子（.class）は指定しない点です。拡張子をつけたまま実行するとエラーになります。これに対し、コンパイラのjavacでは、ソースファイルの指定で拡張子（.java）まで書かれていないとエラーになります。使い分けに注意してください。

演習

（1）自分のコンピュータにインストールされているJavaのバージョンを調べよ。

（2）http://java.sun.com/（英文）にアクセスし、自分の使っているJavaのバージョンに対応する日本語版APIガイドを探し、自分のWebブラウザのブックマークに登録せよ。

（3）Sample01a.javaに意図的に一部誤りを入れ、コンパイルしてみよ。

（4）クラスファイルSample01a.classのファイル名を、たとえばSample01a.txtのように拡張子を変更したり、foo.classのように拡張子以外の部分を変更してjavaコマンドを実行してみよ。

（5）OSの中には、ファイル名の大文字小文字を問わないものがある。たとえばWindowsでは、「notepad SAMPLE01a.java」を実行してもSample01a.javaが開かれる。そのようなOSで、javacやjavaコマンドに大文字小文字が異なるファイル名を指定するとどうなるか調べよ。

Step 02

ソースファイルの構成

Step01のSample01a.javaを例題に、ソースファイルの構成を説明します。より複雑な構成にすることも可能ですが、それは第5部（Step27以降）で話をするとして、まずは最もシンプルなものからスタートしましょう。

●クラス名とソースファイル名

ソースファイルは、図2.1のような構成になっています。

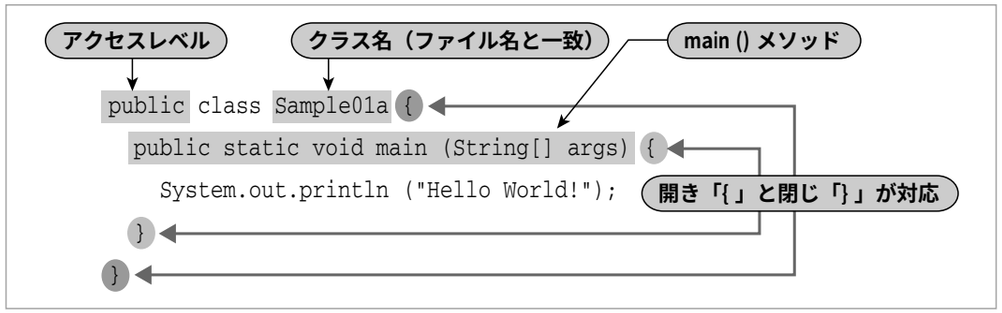

図2.1●ソースファイルの構成（最もシンプルなもの）

Javaでは、ソースファイルとは**クラス**がどのように動作するかを記述したものを指します。こうしたクラスのファイルを必要なら複数用意し、それらを連携して動作させることで1つのプログラムを構成します。しかし、今はまだ最初の段階ですので、1つのファイルに書かれた1つのクラスが1つの完結したプログラムとなるような構成にします。

ソースファイルには、そのクラスの名前を最初に書きます。これを**クラス宣言**といいます。クラス宣言の最初の1語「public」は、このクラスを誰が利用できるかを設定しています（これを**アクセスレベル修飾子**といいます。詳細はStep27で説明します）。publicは、誰でもアクセスできる、を意味しており、このクラスをjavaから実行するには必須です。続く「class」は、これがクラスの宣言であることを示しています。最後の「Sample01a」は、このクラスの名前です。

クラス宣言で示したクラス名と、拡張子部分を除いたソースファイルのファイル名は一致させる必要があります。図2.1のようにクラス名がSample01aならば、ファイル名は必ずSample01a.javaです。両者がマッチしていないと、コンパイラがエラーを報告します。

クラスの宣言のあとに「{（開き中括弧）」を書き、それ以降に、クラスが何をするのか、つまりプログラムを書きます。最後に「}（閉じ中括弧）」で閉めて、このクラスを終わります。

なお、1つのクラスファイルには1つのクラスのみを記述するのが基本です（複数書いたり、クラスの中にクラスを書くこともできますが、そうした話題は本書では扱いません）。

main()メソッド

クラス宣言の中括弧の間には、このクラスの動作を記述する**メソッド**を書きます。メソッドも、クラスと同様に、まずメソッドの宣言を書き、それに続けて具体的な動作内容を中括弧でくくって書きます。

1つのメソッドには、クラス全体の機能の一部を担当させるので、一般的には1つのクラス内に複数のメソッドを書くことになります。しかし、ここは超簡単版ということで、全機能をすべて書き込んだメソッドを1つだけ書きます。このメソッドには`main()`と名前をつけます。`main()`メソッドは、javaコマンドがクラスファイルを実行するときに起動する（呼び出す）、特殊なメソッドです。これがないと、javaからの実行はできません。

javaは、クラスファイルを実行するとき、クラス内に書かれた`main()`メソッドを探します。そして、`main()`メソッドの中括弧に囲まれた内容を、上から順に逐一実行していきます。そして終わり（閉じ中括弧）に到達すると、プログラムを終了します。

`main()`メソッドの宣言は、図2.1に示したように、`public static void main (String[] args)`と書きます。とりあえずはそのまま書き写してください。いずれこの呪文の意味は説明します。

文

クラスを宣言し、その中に`main()`メソッドを書けば、あとは実際のプログラムを書き込んでいきます。プログラムの長さはさまざまですが、いずれも**文**という最小の動作単位を組み合わせて構成されます。文を組み合わせて文章を構成するのとおなじ感じです。

文の終わりは、「;（セミコロン）」で示されます。文章にたとえれば、「;」は句点（「。」のこと）に相当します。たとえば、図2.1の3行目は「;」で終わっているので1つの文です。この文の場合は、括弧の中身を表示するという動作をします。1文はできれば1行で書いたほうが読みやすいのですが、長い文の場合は間で折り返して複数行にすることもできます。

文を構成するものには、変数の宣言や代入、加減乗除のような演算、プログラムの流れの制御、メソッドの呼び出しなどがありますが、これらについては追々説明します。ここではとりあえず、プログラムは文単位で構成され、文末には「;」が必要だという点だけを覚えてください。

プログラムを構成する文は`main()`の中括弧の内側に、`main()`はクラス宣言の内側に書かれているので、全体としては中括弧が2重の入れ子になっています。文自体も中括弧を使うことがあるので、入れ子は3重にも4重にもなります。そうしたとき、どの開き中括弧がどの閉じ中括弧に対応しているか、文がどのレベルの中括弧に属するのかをわかりやすくするため、入れ子になるたびに順に行頭を数文字下げます。これを**インデント**といいます。図2.1やSample01a.javaではインデントを2文字分にしていますが、何文字にしてもかまいません。まったくなくても、コンパイラはエラーにはしません。

コメント文

プログラム中には、プログラムそのものである文だけでなく、自分の覚え書きも書き込むことができます。これを**コメント文**といいます。コンパイラはコメント文を無視して処理を行うので、コメント文があってもプログラムの動作そのものには影響しません。

コメント文にはいくつか書き方があります。「//」と書けば、この記号から行末までがコメントとして扱われます。行単位なので、コメント中での改行はできません。「/*」と「*/」ではさんだ部分も、これらの記号を含めてコメント文とみなされます。このスタイルのコメントはどこにでも書き込むことができ、内部で改行することも可能です。コメント文の記述例を次に示します。

```
// ここで表示を行う
System.out.println ("Hello World!");

System.out.println ("Hello World!");   // ここで表示を行う

/* ここで
   表示を行う */
System.out.println ("Hello World!");

System.out.println ("Hello World!");   /* ここで表示を行う */
```

どちらのスタイルをどのように使ってもかまいませんが、本書では、ほとんどの場合//を使うことにします。なお、**javadoc**とよばれるドキュメント作成用の特殊なコメント記号として「/**」と「*/」もありますが、本書では扱わないので説明は割愛します。

表2.1に、Javaのコメント文についてまとめておきます。

表2.1●Javaのコメント

記号	使い方
//	この記号以降はコメント。短いコメントを書くときに使う。
/* */	間にコメントを書く。長かったり数行にわたったコメントに便利。
/** */	javadoc用。

演習

（1）ソースファイル名と宣言されたクラス名が異なるとき、コンパイラがどのようなエラーを出すか調べよ。

（2）以下のように、ソースファイル中に２つのクラス宣言を書き込むと、コンパイラがどのようなエラーを出すか調べよ。

```
public class Sample02a {
}
public class Sample02b {
}
```

（3）以下のように、クラス宣言だけで中身はカラのソースファイルを作成したとき、①コンパイルは可能か、②コンパイルが成功したなら実行できるかを調べよ。

```
  public class Sample02a {
  }
```

（4）Sample01a.javaを以下のように改行もインデントもなしで書き、コンパイルしてみよ（印刷では複数行になっていますが、１行に続けて書きます）。

```
public class Sample01a {public static void main (String[] args)
{System.out.println ("Hello World!");}}
```

（5）//と/* */の２種類のコメントを使って、Sample01a.java内に作成日と作成者の名前を書き込め。また、プログラムを実際に動かして、コメントがプログラムには影響しないことを確かめよ。