第1章
はじめに

1.1 インターネット環境でのプログラミングとJava

プログラミングという行為は，プログラムを作成するだけでは終わらない。自分で楽しむためだけ，自分で必要な問題を解くためだけならそれでも構わないが，もし人にも使ってもらいたいのなら，プログラムを宣伝し，配布し，実行してもらい，その結果のフィードバック（実行結果，バグ情報，感想など）を集め，バージョンアップを行ったならそれをユーザに反映する，そういったことも考えなくてはならない。そのためには，できるだけ多くの環境（機械やOSなど）で動作可能なようにプログラムを作成することや，利用者がインストールする手間をできるだけ減らす仕組みを組み込むことが必要となる。他人に使ってもらうからには，わかりやすい，使いやすいユーザインターフェースを提供すること[1]，バグ[2]をできるだけなくすこと，それに，たとえ自分のプログラムにバグがあったとしても，ユーザに多大な迷惑をかけないような仕組みの上で動作させることも重要だろう。

また，これだけネットワークが広まった現在，プログラミングを行う人も，プログラミングの目的も，大きく様変わりしているのではないか。昔は，必要なソフトウェアは自分で作らなくてはならず，エディタや便利なツールも自作していた。また，計算機の利用者も理系の人間がほとんどで，研究上必要なデータ処理など，一々プログラムを組んでいた。しかし，現在では，ツール類を自作するのは稀だし，表計算などの出来合いのソフトを用いれば，簡単なデータ処理ならマクロを書く程度で済んでしまうことも多い。その一方で，ネットワークを通じて，情報，楽しみを人と共有したいという要求は高まってきている。自分のホームページに動きを加えてアクセスしてくれた人と楽しみを共有したいとか，自分が研究成果や自分の蓄積したデータベースをネットワークのインターラクティブ性を活かしてわかりやすく公開したいといった要求は，文系も含めた多くの人がもっている。また，ゲームなどのエンターテイメントも，通信型，情報共有型のものが増えてきており，それ

[1] わかりやすいマニュアルも必要だが，マニュアルを見なくても動かせるくらいのわかりやすいユーザインターフェースのほうが重要だ。

[2] プログラムの間違いのこと。

が動作するプラットフォームも，携帯電話をはじめとする，コンピュータ以外のものが増えてくると予想される。

Java 言語は，こういったことをすべてコンセプトに入れた言語である。言語ではなく，ネットワークを想定したプログラムの動作環境と言ってもいい。

Java は仮想マシン方式を採用しているので，Java で作成されたプログラムは，マシンや OS の違いにかかわらず実行可能である。さらに，Web のブラウザの中に実行環境が組み込まれているので，アプレットと呼ばれる種類のプログラムを作成しておけば，ユーザが Web ページを訪問するだけでプログラムがダウンロードされて実行できるようになる。これだと，ユーザは常に最新のバージョンにアクセスできるし，インストールの手間もかからない。C や C++ 言語に比べて，バグが入りにくい言語設計になっているし，プログラムからアクセスできるリソースを制限できるので，たとえプログラムにバグがあっても，その影響を制限することができる。

Java には，ネットワークを用いたプログラムを作成しやすいライブラリが用意されている。グラフィカルなユーザインターフェースを構築するためのライブラリ (Swing) や，データベースへのアクセスのためのライブラリ (JDBC) も用意されている。また，Java は，アプレットのようにブラウザ側で動くだけではなく，サーブレット，あるいは JSP といった，Web のサーバ側で動く仕組みも用意されている。それとデータベース・ライブラリを組み合わせれば，自分のデータベースを Web ページとして公開したり，あるいは，Web 上でアンケートを行い，その結果をデータベースに蓄えることもできる[3)]。

Java 言語のこのような機能は，オブジェクト指向や並列実行（マルチスレッド）といった言語の枠組みを上手に利用することにより実現されている。よって，アプレットを利用した簡単なプログラムを書くのにも，これらの言語の仕組みについて理解しておく必要がある[4)]。本書の前半は，Java 言語自体について，オブジェクト指向や，マルチスレッドの機能を中心に勉強する。後半は，これらの機能を利用してライブラリとして実現されている，グラフィックユーザインターフェースやアプレットについて説明する。次章からの具体的なプログラミングに関する説明に先だって，もう少し Java 言語の実行の仕組みについて説明しておこう。

3) Java の実行環境が載っているのは，Web のサーバやブラウザだけではない。携帯電話をはじめとする情報家電にも Java の実行環境が載せられようとしている。もともと，このような組み込みシステムを制御するための言語を目指して Java は開発された。Java で書いたプログラムを自分の携帯電話やカーナビに載せたり，携帯電話同士で対戦型のゲームで遊んだりといったことが可能になるのも，遠い将来ではないだろう。また，言うまでもなく，Java 言語はアプレットやサーブレットだけではなく，普段のプログラミング言語としても有用である。

4) もちろん，そのような理解を抜きにして，“書き方” だけを説明することも不可能ではないが，それでは，何かの目的をもって書いたプログラムを動かす所まで到達できないだろう。

1.2 仮想マシンとバイトコード

人間がエディタなどで書くプログラムのことを**ソースプログラム**という[5)]。ソースプログラムを直接実行する処理系は，**インタープリタ**と呼ばれている。インタープリタによって実現された言語ではエディタで書いたプログラムが

5) **ソースコード**，あるいは単に**ソース**ともいう。

すぐに実行できるため，プログラムの開発が容易である。また，1つのプログラムが機種に依存せずどこでも動作が可能であるという利点もある。その反面，プログラムの実行が遅いという欠点がある[6]。

実行速度が重要な場合は，ソースプログラムから**オブジェクトプログラム**[7]と呼ばれる計算機で直接実行可能なプログラムを生成し，それを実行するという方式を用いる。ソースプログラムからオブジェクトプログラムへの変換のことを**コンパイル**といい，コンパイルを実行するプログラムのことを**コンパイラ**という。オブジェクトプログラムは，そのプログラムを実行する計算機のマシン語であるのが普通である。よって，オブジェクトプログラムは高速に実行可能な反面，機種に依存し，1つの CPU，1つの OS でしか実行できないのが普通である。

Java 言語は，コンパイラ方式を採用している。しかし，1.1 節で述べたように，Java ではインターネットでオブジェクトを配布することを目的としており，オブジェクトコードが機種に依存せず，どこでも動作する必要がある。それを実現するために，Java では**仮想マシン**という方式を採用している。Java のコンパイラ（**javac コマンド**により起動される）は，個々の計算機のマシン語を生成しない。その代りに，**JVM**[8]と呼ばれる仮想的な計算機のマシン語（**バイトコード**と呼ばれている）をオブジェクトとして出力する。このバイトコードのプログラムファイル（**クラスファイル**と呼ばれている）を作成する作業はプログラムの開発者によって行われ，生成されたクラスファイルがネットワークなどを用いて，ユーザに配布される。この概念を図 1.1 に示した[9]。

クラスファイルを実行するには，JVM という仮想的な計算機の実行環境をソフトウェアにより実現する必要がある。それは，Netscape や Internet Explorer といった，主要な Web のブラウザの中に組み込まれており，アプレットと呼ばれる形式で作られた Java のプログラムは，ブラウザ上で実行することができる。また，それ以外の通常のプログラムは，JVM の実行環境を提供するプログラムである **java コマンド**の上で実行できる。

実行環境は，通常，バイトコードを解釈実行するインタープリタとして実現されている[10]。このインタープリタが実行するのはソースではなく仮想的な計算機のマシン語なので，通常のプログラミング言語のインタープリタに比べて高速に実行されるが，それでも，C 言語などのコンパイラの出力したマシン語とは比較にならない。そこで，実行時に，バイトコードをその実行中のマシンのマシン語に変換しながら実行する JIT [11]と呼ばれるコンパイラも開発されており，主要な OS 上の java コマンドやブラウザで利用されている。

[6] プログラムの利用者がソースコードを読めてしまうことも，場合によっては問題となろう。

[7] **オブジェクトコード**，あるいは単に**オブジェクト**ともいう。

[8] Java Virtual Machine の略である。

[9] 携帯端末などの上の Java 実行環境もこれから実用化されていくだろう。この図では，あたかも同じプログラムが複数のプラットフォームの上で動作しているように描いているが，実際には，組み込みシステム用のコンパクトな実行環境では使えるライブラリが違うので，パソコン上と同じプログラムが動作するとは限らない。

[10] Java チップといって，Java のバイトコードを直接実行する CPU も開発されている。

[11] Just in time compiler の略である。

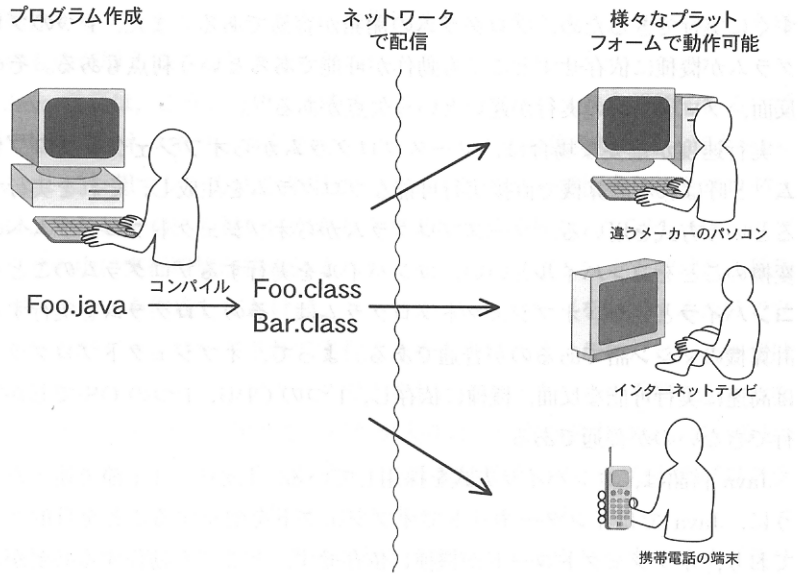

図 1.1 Java のプログラム開発／配布／実行の流れ

1.3 準備 – ソフトウェアなどの入手 –

Java 言語は，1995 年の発表以来，何度もバージョンアップがなされてきた[12]。本書は，Java2 (Java2 Platform. Standard Edition, v 1.3) に基づいて書かれている。本書のサンプルコードは，Java2 SDK, Standard Edition, v 1.3 という処理系で動作を確認している。Java2 以降の Java の処理系なら，問題なく動作するはずである。

本書は，通読するだけで，Java 言語の基本的な考え方，プログラミングの方法が身につくように書かれている。しかし，できるだけ，Java 言語の処理系を手元に用意して，サンプルプログラムを実行し，演習問題を解きながら読みすすめてほしい。そのために，3 つのものを用意して頂きたい[13]。

まず，Java2 以降の Java 言語の処理系である。

```
http://www.java.sun.com/j2se
```

から，Sun Microsystems,Inc. の配布している Java2 SDK という処理系が入手可能である。また，処理系は，様々な雑誌，書籍の付録 CD からも入手できる。

次に，出来れば，Java2 の標準クラスライブラリの仕様書である「Java2 プラットフォーム API 仕様」というドキュメントを，Web で閲覧できるようにして頂きたい[14]。これは，上記サイトの「Java2 SDK ドキュメント」の中の「Java2 プラットフォーム API 仕様」というリンクの先をたどって見ることができる。また，同じサイトから，ダウンロードすることもできる。

[12] 1998 年 12 月に発表された，Java2 以降，大きな仕様変更は行われていない。

[13] もちろん，それ以外に，ソースプログラムを作成するのにテキストエディタも必要である。Windows 環境に付属しているメモ帳でも構わないが，Mule などの高機能エディタは Java の文法に合わせた括弧の照合とか自動インデントとかの機能を利用できるので便利である。

[14] バージョンによって異なるが，次章の図 2.2 のような画面である。

最後に，本書を学習するために作られたプログラムのアーカイブを，まえがきに述べた本書のサポートページから入手して頂きたい。本書は，このアーカイブが読者のコンピュータのディレクトリ[15]に展開されていることを仮定して書かれている。例題は，そのディレクトリで動作可能なように作られているし，演習問題も，そこに置かれているサンプルプログラムを書き換えたり，そこに新たにファイルを作成するように指示を与える形で作られている。以下，javabook というディレクトリにこのアーカイブが展開されているものとして話をすすめることにする。

[15] ファイルを格納する場所のことを，UNIX ではディレクトリ，Windows や Mac ではフォルダという。本書では，ディレクトリで統一することにする。

1.4 準備 – 必要なクラスのコンパイル –

本書の前半は，タートルグラフィックスの例題を用いて学習をすすめる。それは，javabook ディレクトリの下の turtle というディレクトリで行うので，そこに移動しよう[16]。

このディレクトリには，Java 言語のソースプログラムが置かれている。Java のソースプログラムは，.java という拡張子のついた名前をしている。このディレクトリのソースプログラムのほとんどはこのテキストに載っている例題であるが，Turtle.java，TurtleFrame.java，TurtlePanel.java の 3 つは，例題ではなく，Turtle， TurtleFrame，TurtlePanel という，本書を通して利用する 3 つのクラスを定義したものである[17]。これらはあらかじめコンパイルを行い，クラスファイルの形にしておく必要がある[18]。

コンパイルは，javac というコマンドを利用して行う。ソースプログラムをコンパイルすると，そこに定義されている個々のクラスに対応してクラスファイルが作成される。クラスファイルは，クラスと呼ばれる Java のプログラムの構成要素ごとに，“クラス名.class” という名前のファイルとして作成される。OS のプロンプトの元で[19]，

```
% javac Turtle.java
```

を実行することにより Turtle.java がコンパイルされ，クラスファイル Turtle.class が作成される。また，

```
% javac TurtleFrame.java
```

を実行することにより TurtleFrame.java がコンパイルされ，クラスファイル TurtleFrame.class と TurtleFrame$Line.class が作成される[20]。同様に，TurtlePanel.java もコンパイルし，クラスファイル TurtlePanel.class を作成しよう[21]。これらのファイルが作成されたことを確認した上で，次章からの学習を始めよう。

[16] Java 言語のコンパイル／実行は，Windows 環境でも，コマンドプロンプトを起動し，その中で行う。コマンドプロンプト内での，cd でのフォルダの移動，dir や dir/w でのファイルの確認などの基本的な操作に慣れて頂きたい。

[17] これらのソースプログラムを本書を読み終わった後に眺めれば，勉強になるところが多いと思う。

[18] 1.2 節でも述べたように，クラスファイルはマシンに依存しないので，最初からクラスファイルを配布してもよかったが，javac コマンドに慣れてもらうことと，読者の環境で javac が正常に動作していることを確認する意味を兼ねて，読者に行ってもらうことにした。

[19] % は，OS の出す**プロンプト**（コンピュータが，次のコマンドを受け付ける準備ができたことを示す合図の文字列）だとする。Windows のコマンドプロンプトなら，
C:¥javabook¥turtle>
といった文字列になっている。

[20] TurtleFrame$Line も TurtleFrame.java で定義されているクラスだが，これはユーザが直接利用するものではない。

[21] 実際には，Turtle.java のコンパイルを行えば，その中で利用されている TurtleFrame.class，TurtlePanel.class を作成するため，TurtleFrame.java，TurtlePanel.java もコンパイルされる。