第16章

まだまだ広がるJavaの世界

第14章以降では重要なJavaのAPIについて学んできました。
本書で解説したAPIはJava全体から見ればごく一部ですが、
本書を理解した方であれば、自力でAPIを調べ、
自分のプログラムに取り入れて活用することができるでしょう。
この最終章では読者のみなさんへの「はなむけ」として、さらに高度な
Javaプログラミングの世界と可能性を紹介します。

CONTENTS

16.1 ファイルを読み書きする

16.1.1 ストリーム

プログラムからコンピュータの中にあるファイルを読み書きする場合、ファイルの内容をすべて一度に変数へ読み込むことは通常、行いません。なぜなら、ファイルがとても大きかった場合、一度に読み込むとメモリが足りなくなってしまうからです。そこでJavaをはじめとする多くのプログラミング言語では、ファイルを「少しずつ読んだり書いたり」するための機能を備えています。このとき必要となるのがストリーム(stream)という考え方です。

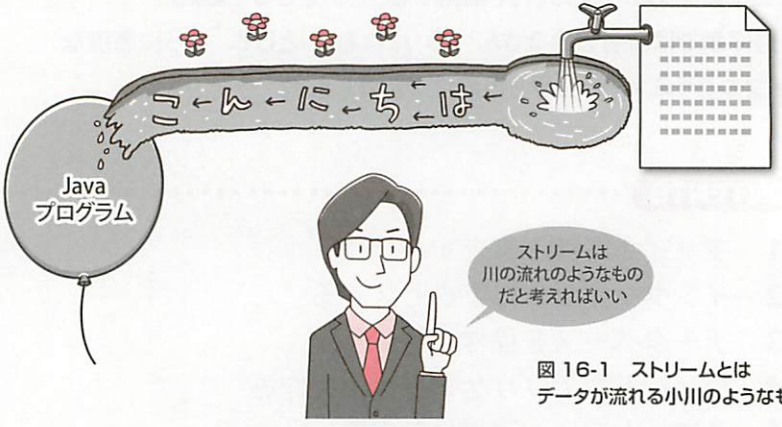


図16-1　ストリームとは
データが流れる小川のようなもの

ストリームとは、情報が流れてくる小川のようなものだと想像してください。Javaプログラムは、この小川を通してファイルを読み書きします。たとえばファイルを読み込む場合は、図16-1のように、「小川の上流にあるファイルから1文字ずつ流れてくる」というようなイメージで文字を読み込みます。

16.1.2 ファイルから文字を読み込む

ファイルから文字を1文字ずつ読み込むコードを紹介しましょう(リスト16-1)。テキストファイルから文字を読み込む場合には、java.io.FileReaderを使います。

リスト 16-1

Main.java

```
1  import java.io.*;
2  public class Main {
3    public static void main(String[] args) throws Exception {
4      String filename = "c:¥¥test.txt";
5      FileReader fr = new FileReader(filename);
6      char c1 = (char) fr.read();
7      char c2 = (char) fr.read();
8      fr.close();
9    }
10 }
```

ファイル名をセットする
最初の 1 文字を読む
次の 1 文字を読む
ファイルを閉じる
ファイルを開く

FileReader は「指定されたファイルが源流にある小川」のようなもので、read() メソッドを呼ぶたびに 1 文字ずつ文字を取り出せます。read() の結果が -1 の場合、ファイルを最後まで読み終わったことを意味します。そして読み終わったら最後に必ず close() してファイルを閉じておきます。

なお、FileReader のコンストラクタや read()、close() は IOException を送出する可能性があります。このリスト 16-1 はサンプルですので例外処理を記述していませんが、開発の現場では必ず例外処理を行ってください。

close() は必ず finally の中に書くようにね。その理由がわからない人は第 15 章を読み返してほしい。

16.1.3 ファイルへ文字を書き込む

ファイルに文字や文字列を書き込むには、FileWriter を使います。このクラスは「下流にあるファイルにつながっている小川」のようなもので、ストリームに流した文字がファイルに書き込まれていきます。

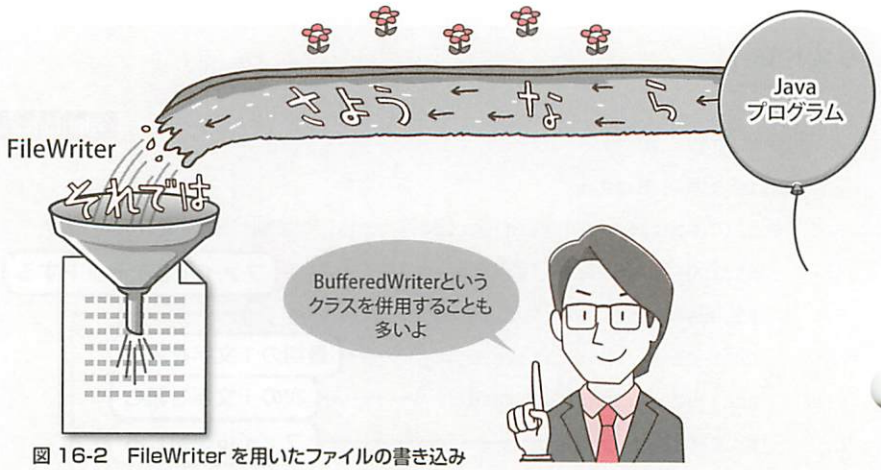


図 16-2　FileWriter を用いたファイルの書き込み

実際に FileWriter を使ってファイルに書き込む例がリスト 16-2 です。

リスト 16-2

Main.java

```
1  import java.io.*;
2  public class Main {
3    public static void main(String[] args) throws Exception {
4      String filename = "c:¥¥test.txt";
5      FileWriter fw = new FileWriter(filename);
6      fw.write('そ');
7      fw.write('れ');
8      fw.close();
9    }
10 }
```

ファイルを開く
最初の 1 文字を書く
次の 1 文字を書く
ファイルを閉じる

今回は FileReader と FileWriter の 2 つのクラスだけを紹介しましたが、java.io パッケージにあるさまざまなクラスを利用すると、より高度な入出力処理が可能になります。

16.2 インターネットにアクセスする

16.2.1 Web ページを取得する

従来のプログラミング言語では、インターネット上の Web ページの内容を取得するために、ネットワークプログラミングに関する多くの知識とプログラミング作業が必要でした。しかし、Java では java.net パッケージのクラスを使うことで、同じ機能のプログラムを、ほんの数行で書くことができます（リスト 16-3）。

リスト 16-3

Main.java

```
1  import java.io.InputStream;
2  import java.net.URL;
3  public class Main {
4    public static void main(String[] args) throws Exception {
5      URL u = new URL("http://www.impressjapan.jp/");
6      InputStream is = u.openStream();
7      int i = is.read();
8      while (i != -1) {
9        char c = (char) i;
10       System.out.print(c);
11       i = is.read();
12     }
13   }
14 }
```

6 行目：ネットへ接続
8 行目：ページの終わりまで繰り返す
10 行目：読んだ内容を画面に表示

実行結果

```
<!DOCTYPE html PUBLIC ···>
```

インプレスのサイト内容

```
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Impress</title>
 …
```

リスト 16-3 を実行すると、Web ページを構成している HTML のテキストが画面に表示されます。このプログラムのポイントは 5 行目で java.net.URL クラスのインスタンスを生み出し、openStream() メソッドを呼び出している部分です。このメソッドを呼ぶと、インターネット上のページを上流に持つストリームが取得できますので、1 文字ずつ読みながら画面に出力しています。

図 16-3　openStream() メソッドにより
Web ページを上流に持つストリームが構成される

さまざまなモノにつながるストリーム

Java のストリームはファイルや Web ページ以外にもさまざまなモノを接続できます。実は今までにも私たちはストリームを何度も使ってきました。それは下流が画面につながっている小川である「System.out」と、上流がキーボードにつながっている「System.in」です。System.out.println() とは、画面につながっている小川に情報を流す命令だったのです。

16.3 データベースを操作する

16.3.1 データベースと SQL

データベースに情報を入れたり出したりするのにも、ストリームが使えるんですか？

残念だが、データベースの操作にストリームは使えない。その代わりに SQL という専用の指示をデータベースとやりとりするよ。

データベースとは、データを整理して格納したり、高速に取り出したりするためのソフトウェアとデータの集合体を指します。

一般的なデータベースの中には多くの「表」があり、その表の中の値を取得したり書き換えたりして利用します。実際に表の中身を読んだり書いたりするためには、「SQL」というデータベース専用の言語でデータベースに指示を送る必要があります。

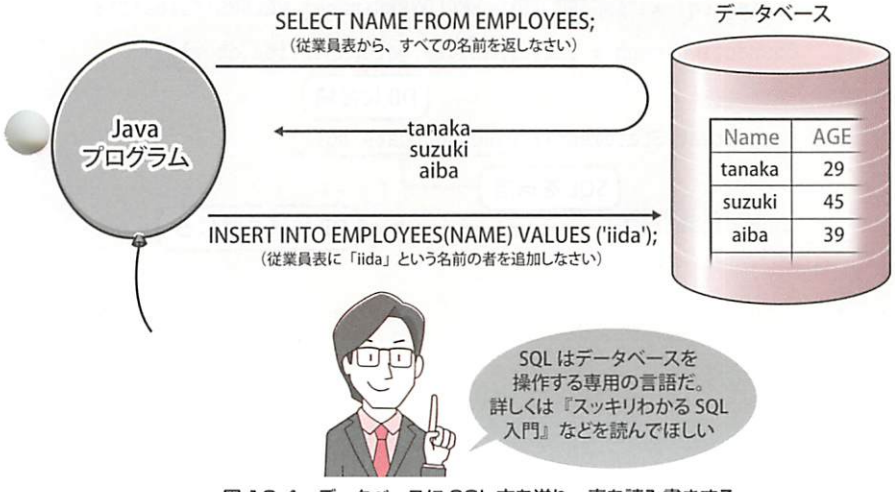


図 16-4　データベースに SQL 文を送り、表を読み書きする

図 16-4 の中の「SELECT ～」や「INSERT ～」が SQL の命令なんですね。

そうだよ。SQL 文をデータベースに送れば、その指示に従ってデータを処理してくれるんだ。

Java では java.sql パッケージのクラスを用いることで、データベースに SQL 文を送ることができます。詳細は割愛しますが、リスト 16-4 のサンプルコードを眺めて、「データベースアクセスも決して難しくない」ということを実感してみてください。

リスト 16-4

Main.java

```
1  import java.sql.*;
2  public class Main {
3    public static void main(String[] args) throws Exception {
4      Class.forName("org.h2.Driver");
5      String dburl = "jdbc:h2:~/test";
6                          接続先 DB を指定
7      String sql = "INSERT INTO EMPLOYEES(name) VALUES('iida')";
8      Connection conn = DriverManager.getConnection(dburl);
9                                        DB に接続
10     conn.createStatement().executeUpdate(sql);
11                       SQL を送信
12     conn.close();                          DB 接続を閉じる
13   }
14 }
```

16.4 ウィンドウアプリケーションを作る

16.4.1 CUI と GUI

湊くんの RPG はおもしろいけど、表示が文字だけでカワイくないのが残念ね…。

少しガンバれば、ウィンドウを持ったグラフィカルなプログラムも Java で作ることができるんだよ。

プログラムの見た目・操作性のことを**ユーザーインタフェース**(UI:User Interface)といいます。その中でも本書で開発してきたような文字ベースのユーザーインタフェースは **CUI**(Character User Interface)と呼ばれます。

一方、Windows や Mac などのパソコンで使われる Web ブラウザやメールソフトのような「窓枠があってグラフィカルな表示とマウスを使って操作できるソフトウェア」は **GUI**(Graphical User Interface)と呼ばれます。

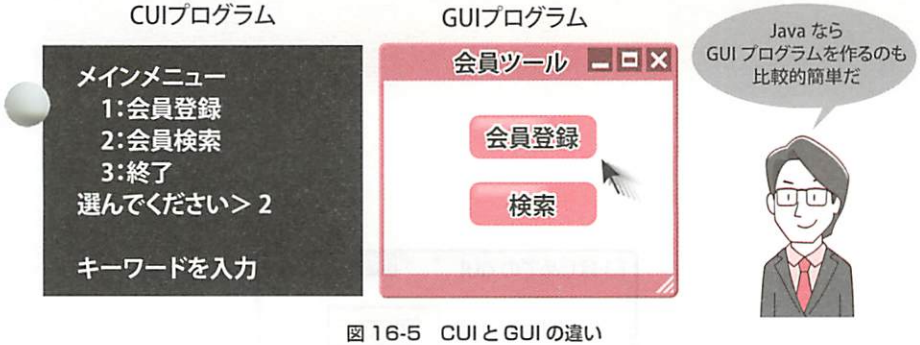


図 16-5　CUI と GUI の違い

java.awt と javax.swing パッケージには、GUI 開発に用いるボタンや入力ボックスなど、さまざまな部品がクラスとして提供されています。紙幅に制限があり

ますので、本格的な GUI プログラムの作り方の解説は専門書に譲り、ここでは簡単なサンプルコードの紹介にとどめます（リスト 16-5）。

しかし、Java を使えば Windows、Mac、Linux、そのほか Java が動作するコンピュータであれば、どれでも同じように動作するウィンドウアプリケーションが作れるということを、ぜひ知っておいてください。

リスト 16-5

Main.java

```
1  import java.awt.FlowLayout;
2  import javax.swing.*;
3  public class Main {
4    public static void main(String[] args) {
5      JFrame frame = new JFrame("はじめてのGUI");
6      JLabel label = new JLabel("Hello World!!");
7      JButton button = new JButton("押してね");
8      frame.getContentPane().setLayout(new FlowLayout());
9      frame.getContentPane().add(label);
10     frame.getContentPane().add(button);
11     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12     frame.setSize(300, 100);
13     frame.setVisible(true);
14   }
15 }
```

図 16-6　リスト 16-5 の実行結果

16.5 スマートフォンのアプリを作る

16.5.1 携帯端末で動く Java

菅原さん！ Java でスマートフォン用のアプリケーションも作れるって本当ですか！？

そうだよ。湊くんの RPG がスマートフォンでも動いたら、遊んでくれる人は増えるんじゃないかな。

最近のスマートフォンをはじめとする携帯端末には、Java で開発したアプリケーションを動かせるものがあります。たとえば NTT ドコモの「i アプリ」や、Android 端末用のアプリケーションは Java で開発します。

ただし、Windows や Mac 用に開発した Java プログラムが、そのまま携帯端末で動くわけではありません。携帯端末を開発しているメーカー各社から、それぞれの機種専用の「プログラミングで利用するクラス」が SDK（Software Development Kit）として提供されており、それらの専用クラスを用いてプログラミングを行う必要があります。

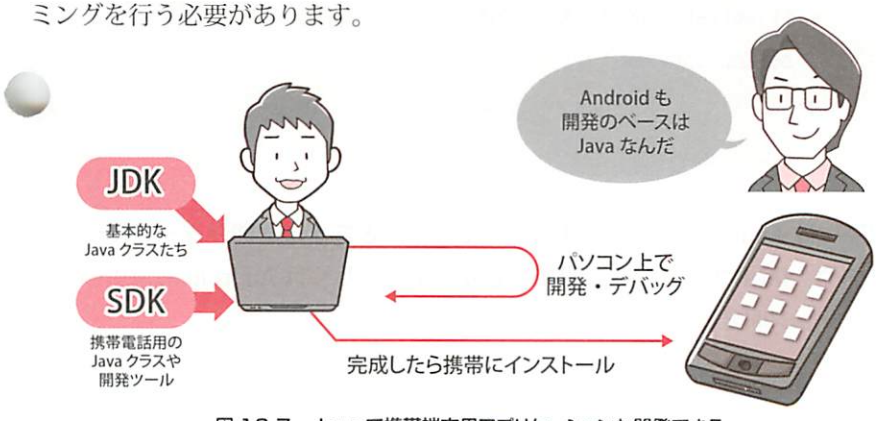


図 16-7　Java で携帯端末用アプリケーションも開発できる

通常、携帯端末用のアプリケーション開発には、SDK に同梱される「エミュレータ」と呼ばれる携帯端末の動作を擬似的に再現するパソコン用ソフトを使います。開発中のプログラムをエミュレータで実行して動作を確認し、プログラムが完成したら実際に携帯端末にインストールして利用するわけです。

16.5.2 Android 端末用の HelloWorld

参考までに Android 端末で「Hello Android」と画面に表示する Java コードを掲載します（注：このリスト 16-6 をコンパイルするには Android 用の SDK が必要です）。

リスト 16-6

HelloAndroid.java

```
1  package my.packages;
2  import android.app.Activity;
3  import android.os.Bundle;
4  import android.widget.TextView;
5  public class HelloAndroid extends Activity {
6    public void onCreate(Bundle savedInstanceState) {
7      super.onCreate(savedInstanceState);
8      TextView tv = new TextView(this);
9      tv.setText("Hello Android");
10     setContentView(tv);
11   }
12 }
```

実際には Android 端末用の SDK だけでなく、さまざまな開発ツールのセットアップや、Java コード以外の開発も必要です。詳しくは Android 端末用のアプリケーション開発に関する解説書を参照してください。

16.6 Web サーバで動く Java

16.6.1 Web アプリケーションとは

インターネットが普及し始めた当時の Web サイトは、ブラウザで Web サーバにアクセスして格納されているページ内容を読むためだけのものでした。しかし、最近の Web サイトには、利用者が入力した情報に応じてサーバ側で必要な処理が実行され、画面の表示内容が変化するようなものが多くあります。

たとえば新幹線の指定席を予約できる Web サイトでは、希望の乗車時間と出発駅・目的駅を入力すれば、データベースから新幹線のダイヤと空席情報を検索して結果を表示してくれます。さらに予約ボタンをクリックすれば、予約情報が鉄道会社のデータベースに登録され、席の予約もできます（図 16-8）。

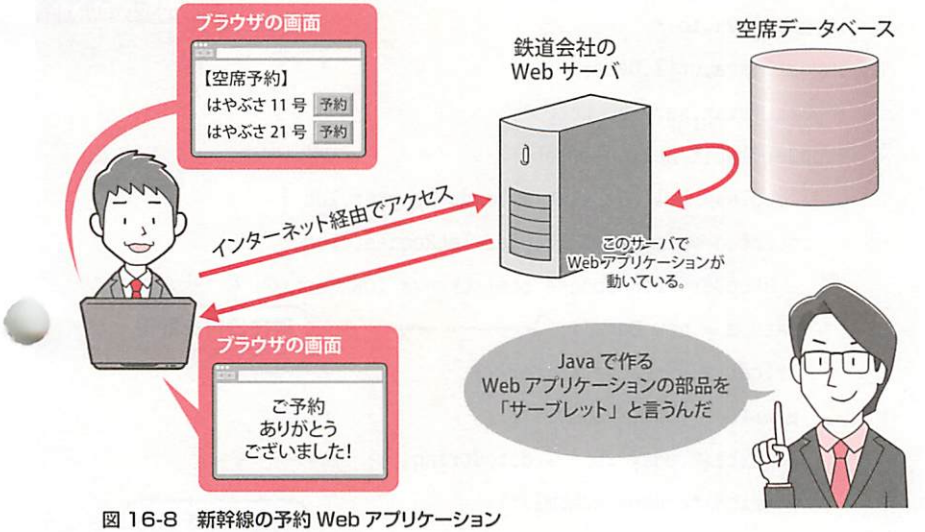


図 16-8　新幹線の予約 Web アプリケーション

この新幹線の Web サイトのように、利用者がブラウザから入力した情報をサーバ側のプログラムで処理するしくみを備えた Web サイトを Web アプリケーション（Web Application）といいます。検索サイトやショッピングサイト、あるいは

SNS など、読者の方々も数多くの Web アプリケーションを利用したことがあるはずです。

16.6.2 Java で作る Web アプリケーション

Web アプリケーションは、さまざまなプログラミング言語を使って開発できますが、Java を使って Web アプリケーションを開発する場合にはサーブレット(Servlet)というクラスを開発します。

サーブレットを開発、動作させるためにはさまざまな準備が必要ですが(参考書籍:『スッキリわかるサーブレット& JSP 入門』(インプレス)など)、ここでは「アクセスされたら現在時刻を取得して Web ページとして返す」簡単なサーブレット(HelloServlet)のサンプルコードを紹介します(注:次のリスト 16-7 をコンパイル・実行するにはサーブレットの開発環境が必要です)。

リスト 16-7

HelloServlet.java

```
1  import java.io.*;
2  import java.util.Date;
3  import javax.servlet.http.*;
4  @WebServlet("/HelloServlet")
5  public class HelloServlet extends HttpServlet {
6    protected void doGet(HttpServletRequest req,
         HttpServletResponse res) throws IOException {
7      Date d = new Date();
8      Writer w = res.getWriter();
9      w.write("<html><body>");
10     w.write("Today is " + d.toString());
11     w.write("</body></html>");
12   }
13 }
```

現在日付を取得

現在日付を出力

ソースコードができたらコンパイルし、いくつかの設定ファイルなどとともに、Java を動かせる機能がある Web サーバに登録します。

たとえば dokojava.jp サーバに、この HelloServlet を登録したとすると、世界中どこからでも「http://dokojava.jp/HelloServlet」というアドレスにブラウザでアクセスすることで現在日付を表示させることができます。

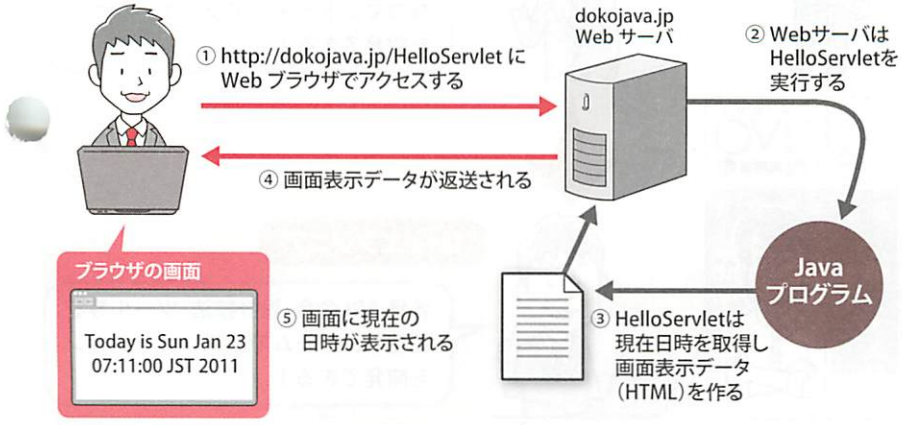


図 16-9　リスト 16-7 の実行の流れ

Java をさらに学んでいけば、もっと多くのいろんなプログラムが作れるようになるんですね。

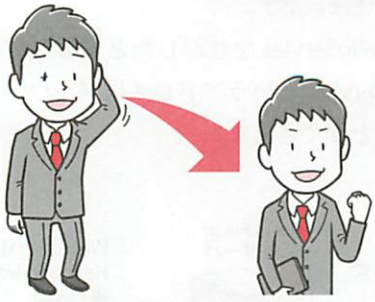
まだまだ学べることはたくさんあると思うと、なんだか楽しみです。

そうだね。初心者を卒業した 2 人なら大丈夫だから、これからも試行錯誤しながら Java プログラミングを楽しんでほしい。

はい！

さらなる高みを目指して——

湊くんの成長の旅は続きます



Java プログラマ（本書を学習し終えた現在）

オブジェクト指向を活用して、簡単なコマンドラインアプリケーションを開発できる！

Java エンジニア

各種 API 命令・設計技法・ツール等を駆使し、チームでアプリケーションを開発できる！

Java ＋ DB エンジニア

SQL やデータ設計の知識も活用し、DB を利用した本格アプリケーションを開発できる！

Web アプリエンジニア

SNS やショッピングサイトのような Web ブラウザで動くアプリケーションを設計・開発できる！