例題で学ぶ Java 入門

工学博士　大堀　隆文
博士(工学)　木下　正博　共著

コロナ社

ま　え　が　き

情報教育を専門とする大学ではプログラミング教育コースは全学生が受講する必修科目であり，全員がそのコースにて必要とされる最低限の能力や技量を身につけなければならない。しかし，「プログラミングに王道なし」といわれ，どんな教え方をしても学生本人のモチベーションがなければ，プログラミングのスキルを修得することはできない。例えば，著者らの大学ではプログラミング言語 Java（以下，Java）のための科目を 1 年後期から開講している。ここで，重要な要素は初めてプログラミングに接する学生が興味やおもしろさを継続して感じ，学習への意欲を保持できるか否かにある。しかしながら，学生の Java 学習へのモチベーションを保つだけではなく，さらにより興味を湧き立たせる例題や課題を用意することは難しい。従来，学生のモチベーションを保つ試みとして，ゲーム作成[1] やペア・プログラミング[2] など，いろいろな方法が試みられているが，従来の課題はおもしろく興味があるが難しすぎて理解に苦しむというようなことをよく聞く。

また，どの言語にもいえることであるが，プログラミングを教える順序にはさまざまな方法がある。Java の場合でも，アプレットから教える，オブジェクト指向から教えるなどの方法がある。どの方法も一長一短があるが，学生全員が落伍しないでプログラミングを楽しんで覚えるためには，難解なオブジェクトやクラスの概念が薄い，コンソールアプリケーションから始めるのがベストであると考える。

従来コンソールアプリケーションは，文字ベースであるためアプレットなどと比べ応用範囲が狭く，なかなか良質な課題を作ることができなかった。ま

1）Michael Koling et al.：Game Programming in Introductory Courses With Direct State Manipulation. ITiCSE05（2005）

2）Laurie Williams et al.：Pair Programming in an Introductory Computer Science Course：Initial Results and Recommendations（2002）

た，従来のテキストの例題や課題は数学系に偏り，数学が苦手な学生，特に文科系の学生には敷居の高いものであった。本書では，文字列処理を基本に据え，学生が興味の引きそうな例題や課題を多く開発し，学生のモチベーションを保ちながら Java の基礎を習得することを目的とする。本書で開発した例題と課題は，どのようにして学生の興味を引くかという観点から次の四つのカテゴリに分かれる。

（1）身近な話題を含む課題　　学生が思わず引き込まれていくような学生に密接な関係のある大学や地域の話題を課題の中に取り入れる。

（2）アイドル名を含む課題　　今はやりで学生に人気のあるアイドルを課題の中に登場させる。

（3）季節感のある課題　　クリスマスなどの講義の開講時期に合わせた課題により学生の興味を引く。

（4）ゲーム風プログラムの課題　　現在の学生の大半が興味をもつゲームそのものまたはゲームの一部を体験させる。

本書は以下の 6 章から構成される。1 章では，Java の概要を述べる。Java の歴史，Java の特徴，Java のインストールからなり，最後に Java を使ったプログラム開発の流れを説明する。2 章では，無料で世界で最も多くの人が使用している Java プログラム開発環境である Eclipse について述べる。Eclipse の概要，Eclipse のインストール，Pleiades による Eclipse の日本語化，Eclipse による簡単なプログラム作成方法について述べる。3 章では Java によるプログラミングの基礎を述べる。コメント，簡単なプログラムと注意点，文字と文字列の表示の後，最後に変数を使ったプログラムの作成方法について述べる。4 章は，プログラムの流れを制御する判断文について述べる。条件式の考え方，Java による単純な条件式 if 文の使い方，より複雑な複合条件をもつ複合 if 文，多方向分岐の switch 文について述べる。5 章では，同じような処理を繰り返す反復文について述べる。繰り返しの考え方，Java による繰り返しの仕組み，単純 for 文，for 文の入れ子，while 文によるプログラム作成について述べる。6 章では，プログラミングに欠かせない配列について述べる。配列の必要性，比較的簡単な一次元配列，少し高度な二次元配列によるプログラム作成方法に

ついて述べる。

本書は大学等の講義においても使用できるように，それぞれの章において例となるプログラムと課題を用意している。例のプログラムは実際に入力し，実行してみることでプログラムの流れを確認することができる。課題はプログラムを作成するものであるが，解答例を Web ページ†からダウンロードすることができ，自分で作成したプログラムと比較し学習することができる。

難しいと思われている Java プログラミングは，もしモチベーションが下がればさらに難しいものとなる。本書はモチベーションを保つ試みの一つとして学生が興味をもちそうな例題と課題を作成した。是非，本書を読んで一人でもプログラミングが好きな学生が現れることを願ってまえがきとする。

なお，本書の作成にあたって北海道工業大学大学院応用電子工学専攻博士課程 1 年の早坂亮佑君，電気工学専攻の木暮直記君（現在雪研スノーイーターズ勤務）には，われわれの作成した例題や課題に関して貴重な意見をいただいた。この場を借りて謝意を表したい。

最後に，われわれをいつも陰からサポートしてくれている妻の大堀真保子と木下倫子に最大の感謝の意を表したい。彼女らの支えがなければこのテキストが完成することはできなかっただろう。そして，本書の企画から完成まで，さまざまな面でご助力いただいたコロナ社の関係者の皆様に，改めて感謝申し上げます。

2012 年 9 月

著者を代表して　大堀　隆文

執筆分担

1 章〜3 章，6 章	木下正博
4 章，5 章	大堀隆文

目　　次

1. Java の 概 要

2. Eclipse による Java プログラム開発

3. プログラミングの基礎

4. 判　断　文

5. 反 復 文

6. 配 列

Java の概要

本章では，Java の特徴や歴史的背景を説明し，Java を使ったプログラム開発について簡単に説明する。Java によって初めてプログラムを学ぶ場合，他の言語との比較やなぜそのような特徴があるのかを把握しづらいかもしれない。よって，本章において説明していることがらを前提として学習を進め，徐々に理解を深めた後で再度この章を読み返すとよい。

1.1 Java の歴史

1946 年に世界初の汎用コンピュータ ENIAC が開発されてから，コンピュータの世界ではさまざまなプログラミングのための言語が開発されてきた。コンピュータそのものの基本構造はほとんど変化がないのに対して，なんらかの作業をコンピュータに処理をさせるための命令，いわゆるプログラムの仕組みは少しずつ基本的概念も含めて変化してきている。特にオブジェクト指向といわれる概念はかなり以前までは存在しない概念であった。現在，主流のプログラミング言語は 1970 年代に開発されたものから発展してきたが，Java は比較的新しい言語である。

Java は 1991 年アメリカの Sun Microsystems 社で開発が始められたといわれている。その後，1994 年に Java の仕様が完成し翌年に発表され，**JDK**（Java Development Kit）としてリリースされた。ここでは，**アプレット**（applet）と呼ばれる Web ブラウザで動作するプログラムも含んでいたため，インターネットの普及とともに Java が注目を集めるようになった。そして 1998 年に JDK

は名称を **Java 2 SDK**（Software Development Kit）に変えている。その後，2004 年には JDK（Java SE Development Kit）に再度変更になった。

Java 2 SDK では，拡張モジュールとして提供されていたさまざまな技術が開発対象ごとにまとめられ，J2SE（Java 2 Platform, Standard Edition），J2EE（Java 2 Platform Enterprise Edition），J2ME（Java 2 Platform, Micro Edition）など対象ごとに複数の開発基盤が登場した。

このように Java の開発環境である JDK はバージョンアップを重ね，すべての開発対象を網羅することを意図した Java 2 となった。その後も，変更が加えられ現在は JDK7.0 という開発環境が利用されている。

現在では，スタンドアロンやネットワーククライアント環境をはじめ，サーバサイドあるいはモバイル環境にいたる多くのアプリケーションの開発に利用されている。

1.2 Java の 特 徴

Java は**オペレーティングシステム**（OS：operating system）の種類や環境に依存しない性質とネットワーク環境に対応した機能が多いことから，インターネット利用の拡大とともに多くのプログラム作成者に受け入れられている。Java の**コンパイラ**（compiler）は，実行するコンピュータの環境に対応したコード（ある種の命令のこと）に変換するのではなく，中間の言語まで変換する。コンパイラはプログラムの**ソースコード**をコンピュータが実行できるような機械語に変換するプログラムである。中間言語は，**バイトコード**と呼ばれる形式で保存されるので，異なるさまざまな OS で実行することができる。この中間言語までの変換により OS に依存することのない開発が可能となる。また，Java はオブジェクト指向を意識して作られており，ライブラリには実行時に使える部品が豊富に用意されている。以上をまとめると，①オブジェクト指向，②プラットフォームに依存しない性質，③豊富なネットワーク関連機能，④充実した標準クラスライブラリ，という特徴がある。

さらに，Java にはプロセスを並行動作させる**マルチスレッド**，使われなくなったメモリ領域を自動的に整理する**ガベージコレクション機能**，プログラムの暴走を未然に防御する例外処理機能などいくつかの特徴がある。

1.3 オブジェクト指向

Java の基礎は**オブジェクト指向**（object-oriented）である。オブジェクト指向はオブジェクトを一義的に考える方法論である。**クラス**（class）は**オブジェクト**（object）を抽象化したもので，ここでは，簡単のためにクラスを単位としてプログラムを作成していくと考える。オブジェクトはプログラムの世界において対象となる「もの」を意味している。通常オブジェクトはデータと処理を一体にして抽象化したものと考えることができる。例えば「車」という「もの」を考えると，寸法や色はデータに，走るという動きはプログラムに相当する。「車」というオブジェクトを抽象化すると「スポーツカー」というクラスや「トラック」というクラスを作り出すことができる。しかし，このような定義はしばしばあいまいであり，オブジェクト指向を難しいものにしている。このようなオブジェクト指向にもとづくプログラミングは**オブジェクト指向プログラミング**（object-oriented programming）と呼ばれている。

オブジェクト指向プログラミング以前のプログラミング手法で最も一般的なものは**構造化プログラミング**である。プログラムを構造的に構築し，プログラムの処理を流れとしてとらえることができ，しばしば**フローチャート**（流れ図 flow chart）が使われる。例えば，**順次処理**，**分岐処理**，**繰り返し処理**などの基本プログラム構造を組み合わせてプログラムの構成を考えていく手法であり，フローチャートによって視覚的にプログラムの動きを理解することができる。しかし，構造化プログラミングだけではソフトウェアの再利用や開発労力の面で障壁があることがわかってきた。そこで，ソフトウェア開発の面からもオブジェクト指向が注目されるようになった。

オブジェクト指向では，いくつかの概念が導入されているが，主に① カプ

セル化，② 継承，③ ポリモフィズムの 3 点をあげることができる。

〔1〕 **カプセル化**　**カプセル化**（encapsulation）は，データとプログラムを一体化することである。カプセル化によって，オブジェクト以外の処理からそのオブジェクト内のデータを扱うことができなくなる。いい換えると，外部からのオブジェクト内の参照が不可能となる。これによって，小さい部分をオブジェクトとして分割し，全体を作り上げることが可能となる。例えば，ソフトウェア開発において分担して開発を行うような場合にきわめて有効な手法となることが考えられる。

〔2〕 **継承**　**継承**（inheritance）はオブジェクトから新しいオブジェクトを作成するとき，古いオブジェクトの性質を受け継ぐことができることである。このとき，クラスも継承の対象となるが，古いクラスは**スーパクラス**（superclass），新しいクラスは**サブクラス**（subclass）といわれる。このとき，データとプログラムを受け継ぐので同じ定義を繰り返し記述する必要がなくなり，新しいクラスで必要な定義のみを追加すればよい。このように，継承はソフトウェア開発において再利用を可能にする概念である。

〔3〕 **ポリモフィズム**　**ポリモフィズム**（polymorphism）は異なるオブジェクトに対して類似の処理を行う場合，オブジェクトが属するクラスによって処理が変わる仕組みである。これによって，異なるオブジェクトを同様のインタフェースで扱うことが可能になる。

1.4　アルゴリズムとフローチャート

Java に限らず，プログラミング言語を習得するためには**アルゴリズム**（algorithm）の理解が重要である。アルゴリズムとは計算手順，または算法といわれる。一般的には，これらを組み合わせて具体的問題を解く。この手順を基本的な図形や線，矢印で図式化したものがフローチャートである。Java においてもクラス単位ではフローチャートで記述することができる。さらに，記述される処理は基本構造を組み合わせたものとなるが，次のような基本構造が

ある。

・順 次 処 理

・分 岐 処 理

・繰り返し処理

順次処理は，いくつかの処理を連続的に行う構造である。分岐処理は条件の真偽により処理を選択する構造である。繰り返し処理は，同一の処理を繰り返し行う構造である。フローチャートでは，基本処理に対応した記号が用意されており，これらの記号を線で結ぶことによってアルゴリズムを表現する。フローチャートは縦に書かれ，処理は上から下へ流れる。

主なフローチャート記号として**図 1.1** のようなものがある。

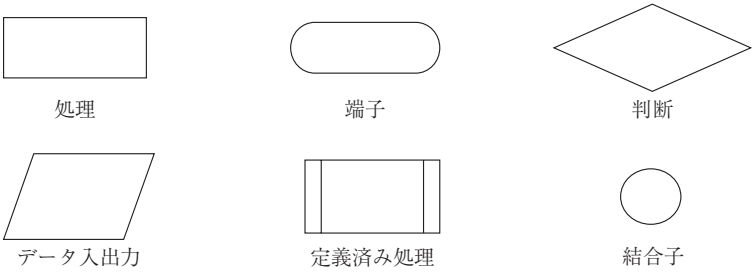

図 1.1 主なフローチャート記号

フローチャートの有効な点は，図式化することにより，プログラムの流れを視覚的に把握できることである。複数人でプログラムを開発する場合，プログラムそのものではなく，フローチャートを用いることによって，プログラムの説明や理解を容易にしている。

1.5 Java のインストール

Java JDK は **Sun Microsystems 社**が無償で提供している Java の開発，実行環境を提供する Java プラットフォームで，コンパイラ，デバッガ，クラスライブラリ，実行環境などが含まれる。現在は Sun Developer Network のサイトから入手

（http://java.sun.com/javase/ja/6/download.html），あるいは，ORACLE が提供しているサイト（http://www.oracle.com/technetwork/jp/java/javase/downloads/index.html）を利用し入手することもできる（**図 1.2**）。

図 1.2　ORACLE の Java トップページ

また，java.com のサイト（http://java.com/ja/download/）では「無料 Java のダウンロード」の部分をクリックするだけで簡単にインストールすることができる（**図 1.3**）。

図 1.3　java.com のトップページ

Java は **Java 仮想マシン**（Java Virtual Machine：Java VM）という形式で実行される。これは，バイトコードをそのプラットフォームのネイティブコードに変換して実行するソフトウェアである。Java で開発されたソフトウェアはプラットフォームから独立した独自のバイトコード形式となっており，そのままでは実行することができない。このため，プラットフォーム固有の形式であ

るネイティブコードに変換するソフトウェアを用意して変換しながら実行する。この変換と実行を行うのが Java VM である。本書では次章で説明する Eclipse がこれらの作業をすべて行っているので，JDK と Eclipse を用意することによりすべての実行環境を整えることができる。

1.6 Java を使ったプログラム開発

Java で作ることのできるプログラムには，クライアントで動作するアプリケーション，Web ブラウザ上で動作するアプレット，サーバ側で動作するサーブレットなどがある。

一般にプログラム開発は，①ソースファイルの作成，②コンパイル，③バイトコードの実行，④デバッグという流れで進められる。エラーが発生した場合には①に戻り，ソースファイルの修正作業を行い再度②〜④を行う。

① **ソースファイルの作成**　プログラムの仕様が決定したあとソースコードを作成する。このことをコーディングという。具体的には Eclipse のエディタ画面において入力し，ファイルとして保存する。これをソースファイルという。

Java ではソースファイル名とクラス名を同じにしなければならない。ここでは，英大文字小文字を区別することに注意しなければならない。ソースファイルの拡張子は .java にしなければならない。

② **コンパイル**　ソースファイルに記述したプログラム（ソースコード）をバイトコードに変換する。コンパイルはソースコードをコンピュータが理解できるコードに翻訳することである。Eclipse ではメニューの中の「実行」を選択することにより，自動的にコンパイルと実行が行われる。また，同時にファイルへの保存も行われるので，開発者は明示的なコンパイルを意識することなく作業を進めることができる。何もコンパイルの異常な状態が表示されず，実行した結果がモニター表示領域に表示されればコンパイルと実行が成功したことになる。もし，コンパイルに失敗した場合はモニター表示領域にコン

パイルエラーが表示される。エラーメッセージには行番号やその理由が表示される。これらを参考にソースファイルを修正して再度コンパイルと実行を行う。コンパイルに成功すると，拡張子が .class というバイトコードのファイルが生成される。これをクラスファイルと呼ぶ。

③ **バイトコードの実行**　**クラスファイル**とは，コンピュータ固有の環境とは完全に独立した Java 仮想マシン上の実行ファイルである。コマンドによる実行形式の場合，Javac コマンドによって Java 仮想マシンを起動し，指定されたクラスファイルを読み込んで実行する。Eclipse ではこれらの処理をすべて統合開発環境の中で行うので，プルダウンメニューの中の「実行」を選択する。

④ **デバッグ**　プログラムには誤りが存在し，実行に至らないものが出てくる。この「誤り」のことを**バグ**（bug）といい，このバグを見つけ出し修正することを**デバッグ**（debug）という。バグのことを単にエラーと呼ぶこともあり，これらを取り除く必要がある。

以上のように Java プログラム開発ではいくつかのステップを踏むが，本書では Eclipse を利用し効率的な開発を行うことを目的としており，Eclipse については次章で詳細を説明する。

索　　引

―― 著 者 略 歴 ――

大堀　隆文（おおほり　たかふみ）
1973 年　北海道大学工学部電気工学科卒業
1975 年　北海道大学大学院工学研究科修士課程修了（電気工学専攻）
1978 年　北海道大学大学院工学研究科博士後期課程修了（電気工学専攻）
工学博士
1978 年　北海道工業大学講師
1981 年　北海道工業大学助教授
1993 年　北海道工業大学教授
現在に至る

木下　正博（きのした　まさひろ）
2003 年　博士（工学）（北海道大学）
2004 年　北海道工業大学講師
2005 年　北海道工業大学助教授
2010 年　北海道工業大学教授
現在に至る

例題で学ぶ Java 入門
Introduction to Java with Examples

2012 年 11 月 30 日　初版第 1 刷発行　★

検印省略

著　者　大堀　隆文
　　木下　正博
発行者　株式会社　コロナ社
　　代表者　牛来真也
印刷所　萩原印刷株式会社

112-0011　東京都文京区千石 4-46-10
発行所　株式会社　**コ ロ ナ 社**
CORONA PUBLISHING CO., LTD.
Tokyo　Japan
振替 00140-8-14844・電話(03)3941-3131(代)
ホームページ　http://www.coronasha.co.jp

ISBN 978-4-339-02468-5　（吉原）　（製本：愛千製本所）
Printed in Japan