ま　え　が　き

本書はプログラミング初級者を対象として，読者がプログラミング，Java 言語，オブジェクト指向の基礎を理解し，活用できることを目的とした「アクティブラーニングで学ぶ Java プログラミングの基礎」シリーズの前編である。この『アクティブラーニングで学ぶ Java プログラミングの基礎 1』では，プログラミング経験がほとんどない学習者を対象としており，変数，条件文，繰り返し文，配列などプログラミングの基礎について学ぶ。これらの内容は，Java 言語に限定したものではなく，手続き型プログラミング言語全般に共通する概念・構文であり，Java 言語を題材に取り上げて解説したものである。なお，シリーズの後編『アクティブラーニングで学ぶ Java プログラミングの基礎 2』では，それに引き続いて，クラス，継承といったオブジェクト指向言語である Java 言語の真髄を中心に学ぶことになる。

本書で学ぶ範囲のプログラミングの概念や作法は，今後学ぶおおよそのプログラミング言語に共通するものであり，プログラミングの入り口として基礎となると同時に重要な内容であるといえる。逆にいうと，この内容を理解し，難しさを乗り越えることが，さまざまなプログラミング言語の理解・活用に役立つものとなる。本シリーズでは，新しい概念を説明するためのサンプルプログラムを示すだけではなく，読者の理解を確実なものにするため「アクティブラーニング」を随所に取り入れた。アクティブラーニングとは書籍や教科書を単に読んで理解するだけでなく，読者が主体的・積極的に自ら物事を学ぶことであり，知識を定着させ，応用力を身につけるための重要な学びの手法である。また,「コーヒーブレイク」では初級者が陥りやすい誤りや，より深い理解のための知識をできる限り平易に解説した。なお,「アクティブラーニング」と演習問題の解答例を Web ページからダウンロードすることができるので，ぜひ自習することをお勧めする（詳細は p. 28 参照）。

本書の著者は，大学 1 年生の講義・演習を担当する若手の講師陣により構成されている。著者らは研究活動を通してプログラミング技術を現実の問題を解決するツールとして活用する研究者である一方で，担当講義の中で実際に学生に接して，理解が難しい概念をどう伝えたらよいかを日々工夫して解説をしている。本書はそういったノウハウを結集し，新しい学習者のために再構成を行ったものである。

読者は，本書を活用して，Java 言語を題材としたプログラミングの基礎を理解し，つぎのステップに進むべき基礎力を身につけていただきたい。また，自ら学ぶ力を習得することによって，プログラミングの世界での応用力を養ってくれることを期待している。

最後に，宇田隆哉講師をはじめとする東京工科大学の教員からなる本書の執筆陣一同の協力に心から感謝する。また，本書の出版にあたりコロナ社にも感謝の意を表する。

2015 年 1 月

大野　澄雄

●編者・執筆者一覧●

○編者

大野（おおの）　澄雄（すみお）（東京工科大学）

○執筆者（執筆順）

荻谷（おぎや）　光晴（みつはる）（東京工科大学）：1，2，3 章

田所（たどころ）　裕康（ひろやす）（東京工科大学）：4，5 章

宇田（うだ）　隆哉（りゅうや）（東京工科大学）：6 章

加藤（かとう）　秀行（ひでゆき）（東京工科大学）：7 章

長名（おさな）　優子（ゆうこ）（東京工科大学）：8 章

政倉（まさくら）　祐子（ゆうこ）（東京工科大学）：9 章

（2015 年 1 月現在）

目　　次

1. Java を始める前の準備

2. はじめての Java

3. Java 言語の簡単な出力

4. 変　　数

5. 演 算

6. 条 件 文

7. 繰り返し文

8. 配　　列

9. メ ソ ッ ド

『アクティブラーニングで学ぶ Java プログラミングの基礎 2』主要目次

1 Java を始める前の準備

この章では，Java プログラムを使うための環境を整える方法について記述する。Java プログラムを行ったことがない人は，まずはこの章の内容を見て環境を整えてほしい。

Java プログラムは JDK がインストールされていれば，どの環境でも実行することが可能である。本書では，Linux の Ubuntu 上で emacs を用いたプログラムの説明を行うが，Windows 上で Java プログラムを作成する場合についての環境設定も説明する。

1.1 Windows での環境設定

Windows の場合は，JDK を Oracle の HP からダウンロードしてインストールすることで，Java プログラムを実行することが可能となる (2014 年 9 月現在)。

http://www.oracle.com/technetwork/java/javase/downloads/index.html

上記の URL にアクセスして，自分の使っているパソコンの OS や環境に対応した JDK をダウンロードしインストールする。インストール後，2 章で示すプログラム 2-1 の「`Hello.java`」のようなソースコードをメモ帳などで記述し，コマンドプロンプトを起動して，コンパイル，インタプリタを実行すればよい。Windows 8 での，コマンドプロンプトの起動は「Windows ロゴキー」+「x」を押すと，プルアップメニューが表示されるので，その中から「コマンドプロンプト」を選択すると起動できる。コマンドプロンプト上で `java -version` や `javac -version` と入力して，つぎのようにバージョン情報が表示されれば実行可能である。環境により，「`\`(バックスラッシュ)」は「￥(円記号)」として表示されることもあるが，同じ意味である。

バージョン確認

```
C:\Users\アカウント名>java -version
java version "1.8.0_20"
java(TM) SE Runtime Environment (build 1.8.0_20-b26)
java HotSpot(TM) 64-Bit Server VM (build 25.20-b23, mixed mode)

C:\Users\アカウント名>javac -version
javac 1.8.0_20
```

もし，どちらかのコマンド結果で以下のようにエラーが出た場合は，システムの環境変数 PATH を書き換えて，Java の環境を認識させる必要がある。

バージョン確認のエラー例

```
C:\Users\アカウント名>javac -version
'java' は，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。
```

例えば JDK が「C:\Program Files\Java\jdk1.8.0_20」の場所にインストールされた場合は，環境変数 PATH に「;C:\Program Files\Java\jdk1.8.0_20\bin」を書き加えるなどすればよい。

しかし環境変数の変更は，間違えるとパソコンが起動しなくなるので，自信がない人には統合開発環境（IDE) をインストールすることをお勧めする。IDE は，いくつかあるので好きなものを選ぶとよい。先ほどの URL には NetBeans という IDE があるので，インストールしてもよい。IDE をインストールした場合は，コンパイルやインタプリタを明示的に実行しなくてもよくなる。

1.2 Linux での環境設定

本書では Linux の Ubuntu を用いて説明している。Ubuntu は，以下の URL から無料でダウンロードできる。

http://www.ubuntu.com/download/desktop/

2014 年 9 月現在では，Ubuntu 14.04 LTS がダウンロード可能である。また Java プログラムをコンパイルなどするために，JDK と emacs のインストールが必要となる。

まずは Ubuntu のインストールを行い，ログインする。デスクトップが表示された後,「Ctrl（コントロール)」+「Alt（オルト)」+「t」を同時に押すとシェル・ウィンドウを起動できる。

JDK と emacs をインストールするためには，インターネットに接続した状態で，以下のコマンドをシェル・ウィンドウで実行すればよい。

emacs, JDK, JRE のインストール

```
$ sudo apt-get update
$ sudo apt-get upgrade

$ sudo apt-get install emacs

$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

その後，シェル・ウィンドウで，`java -version` と `javac -version` コマンドを実行して，Windows の場合と同じ結果が表示されれば，JDK のインストールは正常に完了している。あとは，emacs を用いて次章以降を参考に Java プログラムを作成してほしい。

2 はじめての Java

この章では，Java 言語を用いたプログラムの書き方について説明する。Java 言語は，OS に依存せずに使用することができる。本書では，OS としてオープンソースの Ubuntu を用いた Java プログラムの書き方について取り上げる。プログラムはエディタで記述することができる。Ubuntu には emacs というエディタがあるので，emacs を用いてプログラムを書いてみることにする。

2.1 プログラムを書いてみる

まずは Ubuntu 上で端末（シェル・ウィンドウ）を起動してみよう。Ubuntu を起動して，自分のユーザー名でログインした後に,「Ctrl（コントロール）」キーと「Alt（オルト）」キーを押しながら,「t」のキーを押そう。ウィンドウで端末が開き，つぎのように表示されているはずである。

シェル・ウィンドウの表示の意味

```
ユーザー名@コンピュータ名：~$
```

以降は，ユーザー名が「user」であり，コンピュータ名が「laptop」として説明をする。emacs を起動する前に，覚えておくと便利ないくつかの Unix のコマンドを紹介する。

- cd（チェンジディレクトリ）：ディレクトリを移動する。
- ls（リスト）：現在のディレクトリの中にあるファイルやディレクトリを表示する。
- pwd（プリントワーキングディレクトリ）：現在いるディレクトリを表示する。
- mkdir（メイクディレクトリ）：ディレクトリを作成する。
- emacs（イーマックス）：emacs を起動する。

はじめに，ディレクトリを作成して，プログラムをまとめる場所を作成してみよう。

```
user@laptop:~$ mkdir prog1      ← prog1 というディレクトリを作成
user@laptop:~$ ls
prog1
user@laptop:~$ cd prog1         ← prog1 ディレクトリへ移動
user@laptop:~/prog1$
```

1 行目では「prog1」という名前のディレクトリを作成している。2 行目で，ディレクトリが作成できたか確認している。3 行目は「ls」コマンドの実行結果を示しており，4 行目で

「prog1」ディレクトリへ移動している。現在のディレクトリが移動したかどうかは，5 行目の「$」の前が「~」から「~/prog1」と変更されていることからも確認できる。

つぎに emacs を起動してプログラムを作成してみよう。今回は,「Hello.java」という名前のファイル名でプログラムを作成してみる。

```
user@laptop:~/prog1$ emacs Hello.java &
```

emacs コマンドでは「emacs Hello.java」のように，半角スペースの後に文字列を指定すると，指定した文字列名でファイルを開くことができる。では，上記のコマンドを端末に入力して Enter を押してみよう。emacs 用のウィンドウが開くので，後はその中にプログラムコードを記述すればよい。なお，入力したコマンドの最後についている「&」は，emacs をバックグラウンドで起動するためのコマンドである。「&」をつけて実行すると，emacs を起動した後でも端末にさらにコマンドを入力できる。この状態で起動すると emacs を閉じなくてもコンパイルができるので,「&」をつけてコマンドを実行することを推奨する。それでは，プログラム 2-1 を emacs 上で入力してみよう。

プログラム 2-1 (文字を表示する Java プログラムの例)

```
1 class Hello {
2     public static void main(String[] args){
3         System.out.println("はじめての Java プログラム");
4     }
5 }
```

プログラム 2-1 の構造は以下の通りである。

```
class Hello {          ← クラス名の宣言とクラスの始まり
    public static void main(String[] args){   ← main() の始まり
        System.out.println("はじめての Java プログラム");   ← 出力部分
    }                  ← main() の終わり
}                      ← Hello クラスの終わり
```

`class Hello {}` の部分は**クラス名の宣言**部分であり,「`class` クラス名 `{ }`」のようにクラスの名前を記述している。Java プログラムはクラス名を指定して，`{ }` の中に行いたい処理を記述する。`public static void main(String[] args) {}` は，プログラムの主となる部分である。この部分は **main()** と呼ばれ，Java プログラムではこの main() の後ろにある `{ }` 内のソースコードを上から順番に実行することになる。3 行目が，このプログラムで実際に実行される処理である。ここでは,「はじめての Java プログラム」という文字列を出力するものになっている。

プログラムを実行したい場合，emacs でソースコードを記述しただけでは実行ができない。プログラムを実行するためには，記述したファイルを保存してからコンパイルやインタプリ

タを実行する必要がある。emacs で使用できる便利なショートカット機能を列記しておく。

- C-g：ショートカット入力を中止する（何度か押すとよい）。
- C-x C-s：ファイルを上書き保存する。
- C-x C-w：ファイルを名前をつけて保存する。
- M-w：選択している範囲をコピーする。
- C-w：選択している範囲を切り取る。
- C-y：コピーした内容を貼りつける。
- C-x RET f：文字コードと改行コードを変更する。
- C-c C-q：プログラムを整形（インデントを整える）する。
- C-x C-c：emacs を終了する。

ここで

C-○：「Ctrl」キーを押しながら○キーを押す。

M-○：「Esc」キーを押した後に○キーを押す。

RET： Enter キーを押す。

例えば C-x C-s なら「Ctrl」キーを押しながら「x」キーを押し，その後「Ctrl」キーを押しながら「s」キーを押す。

を表している。

emacs で作成したファイルを保存したい場合は，emacs で開かれたウィンドウを選択した状態で,「Ctrl」キーを押しながら「s」を押せばよい。emacs で正しく保存できた場合は，左下の表示が「`-U:---  Hello.java`」のようになるので確認すること (文字コードが UTF-8 の場合の表示)。もし保存がされていない場合は，「`-U:**-  Hello.java`」のように表示されている。

プログラムを保存したら，実際にプログラムを実行してみよう。emacs はプログラムを保存した後であれば，終了しないまま実行可能である。実行するには，端末に以下のように入力すればよい。

```
user@laptop:~/prog1$ javac Hello.java   ← コンパイルを実行
user@laptop:~/prog1$ ls
Hello.class Hello.java
user@laptop:~/prog1$ java Hello   ← インタプリタを実行
はじめての Java プログラム   ← プログラムの実行結果
```

1 行目は，Java プログラムのコンパイルを行っている。コンピュータには，Hello.java に書いてある命令文のままだと理解できないため，コンパイラにコンピュータでも読める命令文に

変換してもらう必要がある。そのため「javac Hello.java」を実行する必要がある。コンパイラは「Hello.java」というファイルを読み込み，コンピュータでも理解できる「Hello.class」というファイルを作成する。2 行目の「ls」コマンドは，そのファイルがあるかどうかを確認している。コンパイルが失敗していると「Hello.class」が作成されないので，確認してみるとよい。実際にプログラムを実行しているのは，4 行目の「java Hello」である。このコマンドでは,「Hello.class」 をコンピュータに読ませることでプログラムを実行している。その結果として表示されているのが，5 行目の「はじめての Java プログラム」の文字である。

2.2 コンパイラとインタプリタ

コンピュータが直接理解できる機械語 (machine language) は,「0」と「1」の数字の羅列で表されたものである。これに対して Java 言語はプログラム言語であり，人間が理解できるように高級な命令によって構成されている。ここでいう高級という意味は，人間が実際に話している言語に近いという意味である。

Java 言語で書かれたプログラムを実行するためには，コンピュータが理解できるように翻訳を行う必要がある。コンパイラとインタプリタは，そのために用いられる。**コンパイラ** (compiler) とは,「javac ***.java」の形式で実行され，人間が見てわかりやすい高級な命令である Java 言語から，コンピュータが理解しやすい単純な命令の組み合わせに翻訳してくれるものである。**インタプリタ** (interpreter) とは,「java ***」の形式で実行され，単純な命令の組み合わせに翻訳されたものを読み，実際に命令を実行してくれるものである。

プログラムを実行する際には，この二つの操作をただ暗記するだけでなく，**図 2.1** のような流れにより，われわれが理解する言語からコンピュータが理解する言語へ 2 段階の手順により翻訳していることを理解して実行してほしい。

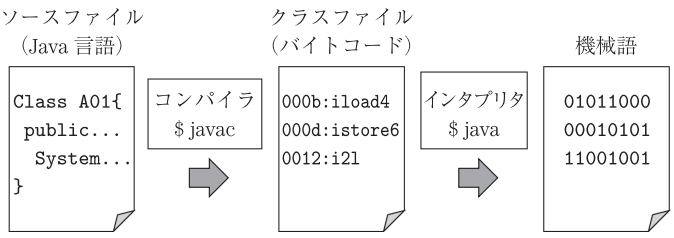

図 2.1 コンパイラとインタプリタ

2.3　println() と print() の違い

プログラムでは実行結果を出力することが頻繁にある。出力の方法には，**println()** と **print()** がある。この違いについて，プログラム 2-2 により説明をする。

プログラム 2-2 (println() と print() の違い)

```
1 class Hello {
2     public static void main(String[] args){
3         System.out.println("はじめての");
4         System.out.println("Java！");
5         System.out.print("はじめての");
6         System.out.print("Java！");
7     }
8 }
```

4 行目：改行が入る。
6 行目：改行は入らない。

実行結果 2-2

```
user@laptop:~/prog1$ javac Hello.java
user@laptop:~/prog1$ java Hello
はじめての
Java!
はじめての Java!user@laptop:~/prog1$
```

「はじめての」の後：改行が挿入
「Java!user@laptop:~/prog1$」：改行が入らない。

プログラム 2-2 の 3, 4 行目は，`println()` により出力している。`println()` は出力後に改行を行うので，実行結果において「はじめての」と「Java!」の後ろにはともに改行が挿入されている。5, 6 行目は，`print()` により出力されているので，実行結果では「はじめての Java!」と連続表示されている。さらに,「`user@laptop:~$`」もその後に改行がされずに表示されている。

2.4　コメントアウト

プログラムの中には，どのような処理を行うか説明をソースコードの中に記述したり，処理を実行しないようにすることができる。プログラムを実行する際に，ソースコードを読まないようにする処理を**コメントアウト**という。コメントアウトするためには，ソースコード上で「`//`」または「`/* */`」を用いればよい。実際にどのように記述するとコメントアウトが使用できるのかプログラム 2-3 とプログラム 2-4 で説明する。

索　　引

—— 編者略歴 ——

1988 年　東京大学工学部電気工学科卒業
1993 年　東京大学大学院工学系研究科博士課程修了
（電子工学専攻），博士（工学）
1993 年　東京理科大学助手
1999 年　東京工科大学講師
2002 年　東京工科大学助教授
2010 年　東京工科大学教授
現在に至る

アクティブラーニングで学ぶ **Java プログラミングの基礎 1**

Foundations of Java Programming 1 — An Active Learning Approach —

2015 年 3 月 20 日　初版第 1 刷発行　★

検印省略

編　者　大野澄雄（おおのすみお）
発行者　株式会社　コロナ社
代表者　牛来真也
印刷所　三美印刷株式会社

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 978–4–339–02486–9 （新井）（製本：愛千製本所）
Printed in Japan