

## 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」シリーズの後編である。シリーズ前編の『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、変数、条件文、繰り返し文、配列などについて解説した。それらの内容は、Java 言語に限定したのではなく、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 言語を題材として取り上げてプログラミングの基礎について記述した。本書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。

オブジェクト指向プログラミングが真価を発揮するのは、対象とするアプリケーションが大規模化し、持続的に展開される開発現場においてであり、数行のサンプルプログラムでそれを実感して理解するのは容易ではない。本書では、新しい概念を説明するためのサンプルプログラムを、それに留まること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に重要な学びの手法である。また、「コーヒープレイク」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」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.138 参照）。

本書は比較的短いまとまりを章に区切って構成している。一つの章をほぼ 1 週間で理解し学習を進めることにより半年足らずで学習が完了する。また、必要に応じて「復習」を行う章を入れ、最後には理解度を測るための模擬試験を置いて、理解の確認と定着を図った。

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、オブジェクト指向プログラミングを理解し、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。

2014年8月

大野 澄雄

●編者・執筆者一覧●

○編者

おおの すみお  
大野 澄雄 (東京工科大学)

○執筆者 (執筆順)

うだ りゅうや  
宇田 隆哉 (東京工科大学) : 1, 2, 11, 15, 16章  
みつはし かおる  
三橋 郁 (東京工科大学) : 3章  
しばた ちひろ  
柴田 千尋 (東京工科大学) : 4, 5章  
おぎや みつはる  
萩谷 光晴 (東京工科大学) : 6, 7, 8章  
いとう たく  
伊東 拓 (東京工科大学) : 9章  
いのうえ あきふみ  
井上 亮文 (東京工科大学) : 10章  
おきな ゆうこ  
長名 優子 (東京工科大学) : 12章  
わたなべ のりふみ  
渡邊 紀文 (東京工科大学) : 13章  
たどころ ひろやす  
田所 裕康 (東京工科大学) : 14章  
まさくら ゆうこ  
政倉 祐子 (東京工科大学) : 17章

(2014年8月現在)

# 目 次

## 1. 『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』の復習

1.1 条 件 文 .....	1
1.1.1 if~else 文 .....	1
1.1.2 if ~ else if ~ else 文 .....	2
1.1.3 switch 文 .....	3
1.2 論 理 演 算 子 .....	5
1.3 繰 り 返 し 文 .....	6
1.3.1 for 文 .....	6
1.3.2 while 文 .....	7
1.3.3 do~while 文 .....	8
1.4 配 列 .....	9

## 2. ク ラ ス

2.1 クラスの概念 .....	11
2.2 クラスの宣言 .....	13
2.3 インスタンスの作成 .....	14
2.4 変数の利用 .....	15
2.5 複数のインスタンスの作成 .....	17
2.6 メソッド .....	18
2.7 クラスのフィールドとローカル変数 .....	20

## 3. クラスの機能

3.1 クラスメンバへのアクセスの制限 .....	23
3.1.1 アクセス修飾子 .....	23
3.1.2 カプセル化 .....	24
3.2 メソッドのオーバーロード .....	25

3.3	コンストラクタ	27
3.3.1	コンストラクタとは何か	27
3.3.2	コンストラクタのオーバーロード	28
3.3.3	別のコンストラクタを呼び出す this	29
3.3.4	コンストラクタに public/private 修飾子をつける	30

## 4. クラス変数とクラスメソッド

4.1	クラス変数	33
4.1.1	インスタンスとインスタンス変数 (復習)	33
4.1.2	クラス変数の宣言	35
4.1.3	クラス変数の意味	35
4.1.4	クラス変数へのアクセスの方法	36
4.1.5	クラス変数の利用例	38
4.2	クラスメソッド	41
4.2.1	クラスメソッドの宣言と記述	41
4.2.2	クラスメソッドの呼び出し方	41
4.2.3	クラスメソッドの利用例	42

## 5. クラスライブラリの利用 (入門編)

5.1	クラスライブラリとは	45
5.2	文字列を取り扱う: String クラス	46

## 6. クラスライブラリの利用 (応用編)

6.1	計算や乱数生成に関するクラス	50
6.1.1	Math クラス	50
6.1.2	Random クラス	52
6.2	ラッパークラス	53
6.2.1	Integer クラス	54
6.2.2	Double クラス	55

## 7. 変数の代入

7.1 基本型変数の代入	57
7.2 オブジェクト型変数の代入	57
7.3 値渡しと参照渡し	61

## 8. オブジェクト型の配列

8.1 基本型の配列	63
8.2 オブジェクト型の配列	63

## 9. 継承

9.1 クラスの拡張	66
9.2 スーパークラスとサブクラスの使用	67
9.3 スーパークラスとサブクラスの関係	70
9.3.1 アクセス権	70
9.3.2 スーパークラスの引数なしコンストラクタの暗黙的呼び出し	71
9.3.3 スーパークラスのコンストラクタの明示的呼び出し	71
9.4 オーバーライド	75
9.5 スーパークラスのメンバへの明示的なアクセス	77
9.5.1 スーパークラスのメソッドの明示的呼び出し	77
9.5.2 スーパークラスとサブクラスにおける同名フィールドへのアクセス	78
9.6 オーバーライドによる多態性	80
9.7 修飾子: final による制御	84
9.8 Object クラス	86

## 10. パッケージ

10.1 パッケージとは	90
10.2 クラスのパッケージ化	91
10.3 パッケージに含めたクラスの利用	93
10.3.1 同じパッケージのクラス	93

10.3.2	別パッケージのクラス	94
10.4	パッケージの高度な使い方	96
10.4.1	インポート	96
10.4.2	サブパッケージ	97
10.4.3	複数クラスのインポート	99

## 11. 復 習 1

11.1	『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』の復習	102
11.2	クラスの復習	105
11.3	メソッド, 修飾子の復習	106
11.4	コンストラクタの復習	107
11.5	オーバーロードの復習	108
11.6	継承の復習	108
11.7	オーバーライドの復習	109
11.8	クラスライブラリの復習	110

## 12. 抽象クラスとインタフェース

12.1	抽象クラス	111
12.1.1	抽象クラスの宣言	112
12.1.2	抽象クラスの利用	112
12.2	インタフェース	116
12.2.1	インタフェースの宣言	117
12.2.2	インタフェースの利用	117
12.2.3	複数のインタフェースの実装・抽象クラスの継承	119
12.2.4	インタフェースの拡張	121

## 13. 例 外 処 理

13.1	例外処理の基礎	122
13.1.1	例外の基本	123
13.1.2	例外処理の書き方	124
13.1.3	後 処 理	126

13.2	例外クラス	127
13.3	捕捉する例外クラスの指定	127
13.4	例外の作成と送付	128

## 14. 入 出 力

14.1	ストリーム	131
14.2	標準入出力	131
14.3	データ入出力	132
14.3.1	ファイルからの入力	132
14.3.2	ファイルへの出力	134
14.3.3	コマンドライン引数	135

## 15. 復 習 2

15.1	『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』の復習	139
15.2	修飾子の復習	143
15.3	メソッドの復習	144
15.4	コンストラクタの復習	145
15.5	オーバーロードの復習	145

## 16. 復 習 3

16.1	クラスライブラリの復習	146
16.2	継承の復習	148
16.3	抽象クラスの復習	151
16.4	例外の復習	152

## 17. 模擬試験問題

..... 154

索 引	.....	164
-----	-------	-----

# 1

## 『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』の復習

この章では Java プログラミングの基礎となる条件文、論理演算子、繰り返し文、配列について概略を説明するのみとし、詳細は割愛する。詳細については『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』を参照してほしい。

### 1.1 条件文

#### 1.1.1 if~else 文

if 文は else 文と組み合わせて使用される。if~else 文の構造を以下に示す。

```
if(条件) {
    文 1;
    文 2;
    ...
}
else {
    文 A;
    文 B;
    ...
}
```

条件が満たされた場合、文 1、文 2、…が実行され、条件が満たされなかった場合には文 A、文 B、…が実行される。else 文を書かない状態でも実行は可能であるが、条件を満たさなかった場合にどうなるのかを明示しないままにしておくのはよくない。if 文を書くときには else 文も書くことを心がけよう。なお、else を書き忘れたのではないことが明示的にわかる場合には else は省略してもよいし、本書でもそのような書き方になっている。

if~else 文の例をプログラム 1-1 に示す。

プログラム 1-1 (if~else 文の例 (IfElseSample.java))

```
01 class IfElseSample {
02     public static void main(String[] args) {
03         int x = 1;
04         char answer = 'N';
05         if(x == 1) {
06             answer = 'Y';
07         }
08     }
```

int 型の変数 x を宣言すると同時に 1 を代入する。

char 型の変数 answer を宣言すると同時に N を代入する。

x の値が 1 であれば「Y」が、そうでなければ「N」が表示されるようになっている。



## 2 1. 『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』の復習

```
09     else {
10         answer = 'N';
11     }
12
13     System.out.println(answer);
14 }
15 }
```

### アクティブラーニング 1.1

プログラム 1-1 の 04, 06, 07, 10 行目を変更し, x の値が 5 より大きければ 1 が, そうでなければ 0 が表示されるようにせよ。なお, answer を int 型の変数にすること。

[04 行目]

[06 行目]

[07 行目]

[10 行目]

#### 1.1.2 if ~ else if ~ else 文

if ~ else if ~ else 文では, if~else 文に else if 文が加わる。if ~ else if ~ else 文の構造を以下に示す。

```
if(条件 1) {
    文 1;
    文 2;
    ...
}
else if(条件 2) {
    文 a;
    文 b;
    ...
}
else if(条件...) {
    ...
}
else {
    文 A;
    文 B;
    ...
}
```

条件 1 が満たされた場合, 文 1, 文 2, …が実行され, 条件 1 が満たされなかった場合には, 条件 2 が満たされるかどうかの判定に移る。条件 2 が満たされた場合, 文 a, 文 b, …が実行され, 条件 2 が満たされなかった場合には, 次の条件が満たされるかどうかの判定に

移る。else if 文は複数記述できる。どの条件も満たされなかった場合には、文 A, 文 B, … が実行される。

if ~ else if ~ else 文の例をプログラム 1-2 に示す。

プログラム 1-2 (if ~ else if ~ else 文の例 (IfElseIfElseSample.java))

```

01 class IfElseIfElseSample {
02     public static void main(String[] args) {
03         int x = -1;
04         char answer = 'N';
05
06         if(x == 1) {
07             answer = 'Y';
08         }
09         else if(x < 0) {
10             answer = 'F';
11         }
12         else {
13             answer = 'N';
14         }
15
16         System.out.println(answer);
17     }
18 }

```

x の値が 1 であれば「Y」が表示される。

そうでなくてももし x の値が 0 未満であれば「F」が表示される。

そうでなければ「N」が表示される。

## アクティブラーニング 1.2

プログラム 1-2 の 04, 06, 07, 10, 13 行目を変更し、x の値が 5 より大きければ 1 が、0 未満であれば -1 が、そうでなければ 0 が表示されるようにせよ。なお、answer を int 型の変数にすること。

[04 行目]

[06 行目]

[07 行目]

[10 行目]

[13 行目]

### 1.1.3 switch 文

switch 文では、条件の代わりに式が評価される。switch 文の構造を以下に示す。

```

switch(式) {
case 値 1:
    文 1;
    文 2;

```

```

...
    break;
case 値 2:
    文 a;
    文 b;
    ...
    break;
default:
    文 A;
    文 B;
    ...
    break;
}

```

式の値が値 1 となる場合、「case 値 1:」の行に処理が移り、文 1、文 2、…が実行される。式の値が値 2 となる場合、「case 値 2:」の行に処理が移り、文 a、文 b、…が実行される。case の行が現れるかどうかに関係なく、break 文が現れるまでこれらの処理は継続される。つまり、もし「case 値 2:」の直前の break 文がなければ、式の値が値 1 だとしても、文 1、文 2、…に続いて文 a、文 b、…も実行され、「default:」の行の直前の break 文で switch 文の処理が終了する。式の値と一致する case が存在しない場合には、「default:」の行に処理が移り、文 A、文 B、…が実行される。

switch 文の例をプログラム 1-3 に示す。

プログラム 1-3 (switch 文の例 (SwitchSample.java))

```

01 class SwitchSample {
02     public static void main(String[] args) {
03         int x = 1;
04         char answer = 'N';
05         switch(x) {
06             case 1:
07                 answer = 'Y';
08                 break;
09             case 2:
10                 answer = 'Y';
11                 break;
12             default:
13                 answer = 'N';
14                 break;
15         }
16         System.out.println(answer);
17     }
18 }
19 }

```

x の値を評価し、1 であれば 07 行目、2 であれば 10 行目、それ以外であれば 13 行目に処理が移る。

x の値が 1 であれば「Y」が表示される。

x の値が 2 であれば「Y」が表示される。

x の値がそれ以外であれば「N」が表示される。

## アクティブラーニング 1.3

プログラム 1-3 の 04, 07, 08, 10, 11, 14 行目を変更し, x の値が 0 であれば 1 が, -1 であれば -1 が, それ以外であれば 0 が表示されるようにせよ。なお, answer を int 型の変数にすること。

[04 行目]

[07 行目]

[08 行目]

[10 行目]

[11 行目]

[14 行目]

この変更を行った場合, 03 行目の x の値を 0 や -1 に変更し, 出力がどのように変わるか説明せよ。

[出力がどのように変わるか]

## 1.2 論理演算子

Java でよく使われる論理演算子をつぎに示す。

- (1) && 両辺の論理積 (AND) を演算する。両辺が真 (true) であれば演算結果も真になる。
- (2) || 両辺の論理和 (OR) を演算する。両辺のどちらか片方が真 (true) であれば演算結果も真になる。
- (3) ! 右側の値を否定 (NOT) する。真 (true) であれば演算結果は偽 (false) に, 偽であれば真になる。

論理演算子の例をプログラム 1-4 に示す。

プログラム 1-4 (論理演算子の例 (LogicalOperatorSample.java))

```
01 class LogicalOperatorSample {
02     public static void main(String[] args) {
03         int x = 1;
04         int y = 0;
05         char answer = 'N';
06     }
```

```

07     if(x == 1 && y == 0) {
08         answer = 'Y';
09     }
10     else {
11         answer = 'N';
12     }
13
14     System.out.println(answer);
15 }
16 }

```

xの値が1, なおかつyの値が0であれば「Y」が、それ以外であれば「N」が表示されるようになっている。

### アクティブラーニング 1.4

プログラム 1-4 の 05, 07, 08, 11 行目を変更し, x の値が 5 より大きい, または y の値が 3 未満であれば 1 が, それ以外であれば 0 が表示されるようにせよ。なお, answer を int 型の変数にすること。

[05 行目]

[07 行目]

[08 行目]

[11 行目]

## 1.3 繰り返し文

### 1.3.1 for 文

for 文の構造を以下に示す。

```

for(式 1; 式 2; 式 3) {
    文 1;
    文 2;
    ...
}

```

式 1 は, for 文に処理が移った時点で実行される式である。式 2 は, for 文のループを行うかどうか判定する条件式である。条件が満たされた場合, ループ部分の文 1, 文 2, …が実行される。式 3 は, for 文のループが毎回終了する直後に実行される式である。ループが終了するたびに式 2 の条件が評価され, 条件が満たされた場合, ループ部分の文 1, 文 2, …が実行される。

for 文の例をプログラム 1-5 に示す。

プログラム 1-5 (for 文の例 (ForSample.java))

```

01 class ForSample {
02     public static void main(String[] args) {
03         int sum = 0;
04
05         for(int i=0; i<10; i++) {
06             sum += i;
07         }
08
09         System.out.println(sum);
10     }
11 }

```

0 から 9 までの整数を sum に足し、その結果が表示される。

加算代入演算子が用いられており、「sum = sum+i;」と同じ意味を持つ。

## アクティブラーニング 1.5

プログラム 1-5 の 03, 05, 06, 09 行目を変更し、1 から 5 までの整数の積が表示されるようにせよ。なお、05 行目では int 型の変数 product を宣言すること。

[03 行目]

[05 行目]

[06 行目]

[09 行目]

## 1.3.2 while 文

while 文の構造を以下に示す。

```

while(条件) {
    文 1;
    文 2;
    ...
}

```

条件が満たされた場合、ループ部分の文 1, 文 2, …が実行される。ループが終了するたびに条件が評価され、条件が満たされた場合、ループ部分の文 1, 文 2, …が実行される。

while 文の例をプログラム 1-6 に示す。

プログラム 1-6 (while 文の例 (WhileSample.java))

```

01 class WhileSample {
02     public static void main(String[] args) {
03         int sum = 0;
04         int i = 0;
05
06         while(i<10) {

```

0 から 9 までの整数を sum に足し、その結果が表示されるようになっている。

# 索引

<b>【あ】</b>		クラス変数	33	デフォルトパッケージ	95
アクセス違反	70	——のデフォルトの初期値	35	<b>【と】</b>	
アクセス権	68	クラス名	35, 37	統合開発環境	92
アクセス修飾子なし	70	クラスメソッド	41	同名フィールドへのアクセス	78
値渡し	62	クラスライブラリ	45	<b>【な】</b>	
アノテーション	76	<b>【け】</b>		名前空間	99
暗黙的呼び出し	71	継承	67	<b>【に】</b>	
<b>【い】</b>		<b>【こ】</b>		入出力	131
インスタンス	14	互換性	82	<b>【は】</b>	
インスタンス変数	33	コマンドライン引数	135	バイトストリーム	131
インスタンス名	36	コンストラクタ	12, 27, 39	パッケージ	45, 70, 90
インスタンスメソッド	41	<b>【さ】</b>		パッケージプライベート	70
インタフェース	69, 116	サブインタフェース	121	<b>【ひ】</b>	
<b>【え】</b>		サブクラス	67	標準出力	131
エラー	122	サブパッケージ	97	標準入力	131
<b>【お】</b>		参照渡し	62	<b>【ふ】</b>	
オーバーライド	76	<b>【し】</b>		フィールド	12
オーバーロード	25	自作クラス	86	<b>【へ】</b>	
オブジェクト型の配列変数	63	実装	117	変数名	35, 36
オブジェクト型変数の代入	57	<b>【す】</b>		<b>【ほ】</b>	
<b>【か】</b>		ストリーム	131	ポリモーフィズム	82
拡張	66	スーパーインタフェース	121	<b>【め】</b>	
ガベージコレクション	60	スーパークラス	67	明示的呼び出し	71, 77
完全限定名	95	——とサブクラスの互換性	82	メソッド	12
<b>【き】</b>		<b>【た】</b>		メンバ	12
木構造	86	多重継承	69, 116	<b>【も】</b>	
基本型変数の代入	57	多態性	82	文字ストリーム	131
キャスト	87	単一継承	116	<b>【ら】</b>	
キャストが実行される タイミング	89	単純名	94	ラッパークラス	53
<b>【く】</b>		<b>【ち】</b>		<b>【れ】</b>	
クラス	11	注釈	76	例外	122
——の拡張	67	抽象クラス	111		
——の宣言	13	抽象メソッド	112		
クラスパス	93	<b>【て】</b>			
		デフォルトコンストラクタ	28, 74		

例外クラス 127  
 例外処理 122

**【ろ】**

ローカル変数 21, 34

**【A】**

abstract 112  
 API 46

**【B】**

BufferedReader クラス 132  
 BufferedWriter クラス 134

**【C】**

CLASSPATH 93  
 close() メソッド 133

**【D】**

Double クラス 55

**【E】**

Error 127  
 Exception 127  
 extends 66

**【F】**

FileReader クラス 132  
 FileWriter クラス 134  
 final 84  
 finally 126

**【I】**

IDE 92  
 import 文 96  
 InputStream クラス 131

**【J】**

InputStreamReader クラス 132  
 instanceof 115  
 Integer クラス 54  
 io パッケージ 131  
 Java のクラスの継承関係 86  
 java.awt 45  
 java.io 45  
 java.lang 45  
 java.swing 45  
 javax.sound 45  
 JDK 45  
 JOptionPane 46  
 JRE 45

**【M】**

Math クラス 50

**【N】**

null 60

**【O】**

Object クラス 86  
 OutputStream クラス 131

**【P】**

package 文 91  
 print() メソッド 132  
 println() メソッド 131  
 PrintWriter クラス 134

private 70  
 protected 70  
 public 70

**【R】**

Random クラス 52  
 Reader クラス 131  
 readLine() メソッド 132

**【S】**

static 35, 85  
 String 46  
 String クラスのおもなメソッド 49  
 super 71  
 super. 変数名 78  
 super. メソッド名 78  
 System.in 131  
 System.out 131

**【T】**

this 29  
 this. 21, 78  
 throw 文 129  
 try-catch 文 124

**【W】**

Writer クラス 131

**【記号】**

@Override 76



— 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 2

Foundations of Java Programming 2 — An Active Learning Approach —

© Sumio Ohno 2014

2014年10月20日 初版第1刷発行

★

検印省略

編者 おおのすみお 大野澄雄  
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 978-4-339-02487-6 (新井) (製本:愛千製本所)

Printed in Japan



本書のコピー、スキャン、デジタル化等の無断複製・転載は著作権法上での例外を除き禁じられております。購入者以外の第三者による本書の電子データ化及び電子書籍化は、いかなる場合も認めておりません。

落丁・乱丁本はお取替えいたします