1-3 JDKのインストール手順

ここでは、Javaのプログラムを作成する上で必要なプログラムなどの導入の仕方について説明していきます。

1-3-1 プログラム開発環境

Javaのプログラムを作るためには、まずPCの環境を整えなければなりません。パソコンで文章を作成しようとしたらワープロソフトが必要なのと同じように、Javaのプログラムを作成したり実行するためには、そのためのソフトウェアをインストールする必要があります。

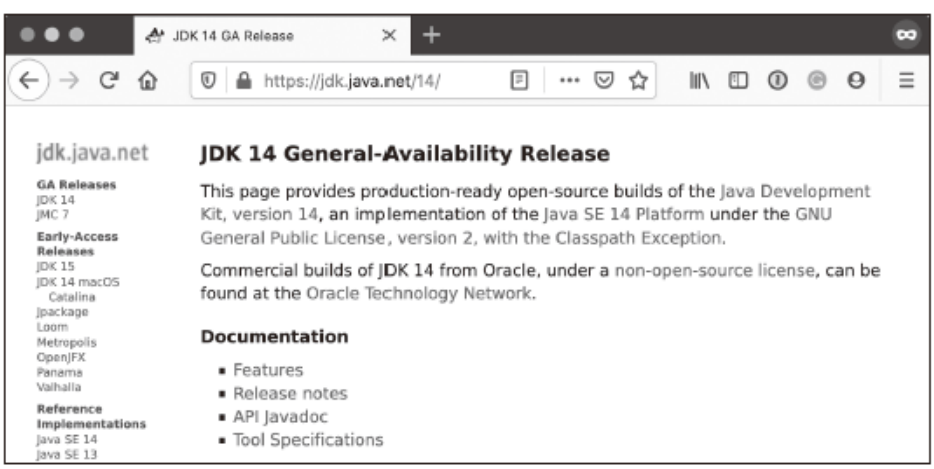
EclipseやNetBeansなど、Javaプログラムの開発を行うための様々なソフトウェアが公開されていますが、本書では、無償でOracle社が提供しているJDK（Java SE Development Kit）のインストール方法について説明します。

TIPS
Eclipse（エクリプス）などについてはP.24を参照してください。

1-3-2 JDKのダウンロード

本書では、Javaのリファレンス実装であるOpen JDKを使用していきます。Google ChromeやMicrosoft Edgeなどのwebブラウザを立ち上げて、次のURLを入力してください。図1-05のようなページが表示されます。

```
https://jdk.java.net/14/
```



●図1-05　JDK 14のページ

TIPS
Open JDKはオープンソースソフトウェアです。ビルド（ソースプログラムから実行可能なプログラムを作成すること）をした会社や団体によって、Oracle Java SEやAmazon Correttoなどの様々なOpen JDKを使用することができます。

2-3-3 コンパイラーの実行

Javaプログラムの翻訳を行うコンパイラーは**javac**という名前が付けられています。このjavacにJavaのソースファイル名を知らせることで、JavaVM用のマシン語プログラムである**クラスファイル**(拡張子が.class)が生成されます。

コンパイルを行うためには、以下のようにタイプします。

TIPS

javacはジャヴァシーと呼ばれますが、一部ではジャヴァックと呼ばれることもあるようです。

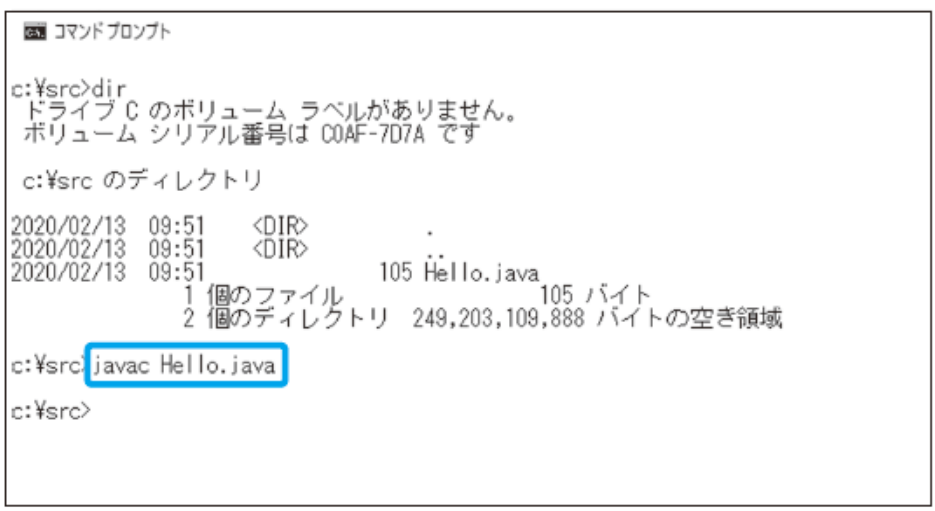
構文 ●コンパイラーの実行方法

```
javac ソースファイル名
```

2-2では、Hello.javaという名前のソースファイルを入力、保存しましたね。コマンドプロンプトから以下の実行例のとおりに入力してください(図2-13)。

●Hello.javaのコンパイルの実行例

```
javac Hello.java
```



●図2-13 Hello.javaのコンパイル

入力が完了したら、Enterキーを押してください。

コンパイルが成功すると、拡張子が.classのクラスファイル(Hello.class)が作成されます。Javaのプログラムを実行させるには、このクラスファイルが必要です。念のため、このファイルが作成されていることを、dirコマンドで確認しましょう(図2-14)。

2-4 コンパイルでエラーが出たら

Javaコンパイラーのjavacの使い方を間違えていたり、ソースプログラムの入力に誤りがあった場合は、エラーメッセージが表示されます。エラーが存在している間は翻訳を完了することができませんから、エラーメッセージをよく読み、エラーの箇所を修正して、再びコンパイルを行う必要があります。以下、コンパイル時によく出力されるエラーメッセージとその対処法を示します。

2-4-1 ソースファイル名の指定間違い

```
コマンド プロンプト

c:¥src>javac Hallo.java
エラー: ファイルが見つかりません: Hallo.java
使用方法: javac <options> <source files>
使用可能なオプションのリストについては、--helpを使用します

c:¥src>_
```

●図2-15 ソースファイル名の指定間違いによるエラー

前ページで、コンパイルを実行する際に指定したソースファイル名が間違えています。「Hallo.java」ではなく「Hello.java」と正しく入力しましょう。

また、クラス名の後に「.java」を付け忘れてはいないか注意しましょう。さらにHello.javaでは、l(小文字のL)やo(小文字のO)の表記は、I(大文字のi)や1(数値)、0(数値)やO(大文字のo)と混同しないようにしましょう。

2-4-2 シンボルを見つけられません：入力ミス

```
コマンド プロンプト

c:¥src>java Hello.java
Hello.java:3: エラー: シンボルを見つけられません
        System.out.println( Hello! );
                   ^
  シンボル:   メソッド println(String)
  場所: タイプPrintStreamの変数 out
エラー1個
エラー: コンパイルが失敗しました

c:¥src>
```

●図2-16 タイプミスによるエラー

2-5 プログラムの実行

いよいよプログラムを動かしてみましょう。しかし、ここにも様々な落とし穴があります。ここでは、プログラムの実行方法と、実行時に発生するエラーについて説明します。

2-5-1 プログラムを実行する

コンパイルを実行して、何もエラーメッセージが表示されなければ、コンパイルは完了です。次に、コンパイルしたプログラムを実際に動かしてみましょう。

プログラムを動かすためには、javacでコンパイルしたJavaVM用のマシン語をJavaVM上で動作させる必要があり、通常はコマンドプロンプトから次のようにタイプします。

構文 ● プログラムの実行

```
java クラス名  ← クラス名なので拡張子.javaは付けない
```

Hello.javaのプログラムを実行するには、図2-20の実行例のとおりに入力して Enter を押してください。

```
コマンドプロンプト

c:¥src>javac Hello.java ←― Hello.javaをコンパイルしてHello.classを作る
c:¥src>java Hello ←――― Hello.classを実行
```

●図2-20 プログラムの実行例

画面上に「Hello!」と表示されれば成功です（図2-21）。

```
コマンドプロンプト

c:¥src>javac Hello.java
c:¥src>java Hello
Hello!
c:¥src>_
```

●図2-21 プログラム実行の結果

練習問題

問題1. 次のクラス名の中からクラス名として使用できないものを選びなさい。

① Hello
② doItYourself
③ newinterface
④ volatile

問題2. 次のうち文字列としてタブを意味しているものはどれか選びなさい。

① ¥'
② ¥¥
③ ¥t
④ ¥n

問題3. 次のプログラムをコンパイルするとエラーが発生する。エラーの原因となっている箇所を修正し、プログラムを書き換えなさい。

▼リスト3-17　練習問題3

```
1. class Renshu33 {
2.     public static void main(String[] args){
3.         System.out.println("Renshu!!!");
4. }
```

問題4. プログラム中の空欄を埋め、次のメッセージを表示させるプログラムを完成させなさい。

```
Hello Java World!!
```

▼リスト3-18　練習問題4

```
1. class Renshu34 {
2.     public static void main(String[] args){
3.         System.out.[　　]("Hello Java World!!");
4.         System.out.[　　]();
5.     }
6. }
```

4-1 演算子

コンピューターの内部では、文字も数値として扱われます。3章のメッセージ表示も、コンピューターにとっては数値の表示と同じことだったのです。まずは、計算を行う際に必要な演算子について学んでいきましょう。

4-1-1 様々な演算子

「＋」や「−」など、数値に対してどのような操作（演算）を行うのかを示すものを、**演算子**と呼びます。「2＋1」のように足し算では＋の演算子を使って、「2−1」のように引き算では−の演算子を使って、演算子の左側の被演算子と右側の被演算子の数値に対して、どのような計算を行うかを示すものです。

演算子は、その働きから大きく分けて、「**算術演算子**」、「**代入演算子**」、「**関係演算子**」の3つに分類できます。

TIPS

この他にも、論理を扱う論理演算子も存在します。論理演算子に関して、詳しくは第6章をご覧ください。

- **算術演算子**

  数値を足したり引いたりする、いわゆる加減乗除の演算を行うための演算子。この演算の結果は数値となる。

- **代入演算子**

  数値や計算結果を「変数」に代入するための演算子。代入演算子と算術演算子を組み合わせることも可能。代入演算子は変数と一緒に使用する（詳説は5章で行う）。

TIPS

Javaにはバイト単位でデータを操作するためのシフト演算子と呼ばれるものや、バイト単位での論理を扱うビット演算子と呼ばれるものも存在します。

- **関係演算子**

  大きい、小さい、等しいといった、2つの数値の関係を判断する演算子（表4-01）。算術演算子と違い、演算の結果は「正しい」か「正しくない」かの2つのどちらかとなる（詳説は6章で行う）。

●表4-01 関係演算子

演算子	意味	使用例	実行結果
<	大きい	1 < 2	true（正しい）
>	小さい	4 > 3	false（正しくない）
==	等しい	5 == 5	true（正しい）

4-4-2 文字列と数値の足し算

次は、文字列に数値や式を足してみましょう。次のプログラムを見てください（リスト4-13）。

▼リスト4-13　文字列と数値の足し算プログラム(Mojiretu3.java)

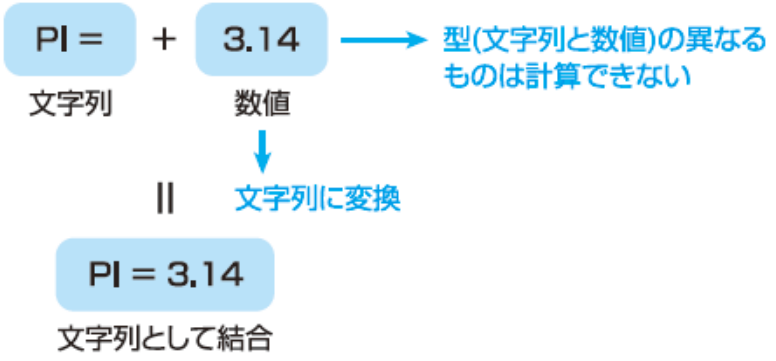
```
1. class Mojiretu3 {
2.     public static void main(String[] args){
3.         System.out.println("PI = " + 3.14);
4.     }
5. }
```

3行目で文字列「PI = 」と数値3.14を足しており、その実行画面は図4-15のようになります。

```
コマンド プロンプト

c:¥src>javac Mojiretu3.java

c:¥src>java Mojiretu3
PI = 3.14

c:¥src>
```

●図4-15　Mojiretu3.javaの実行結果

種類が異なるものをそのまま足すことはできません。文字列と数値の足し算の場合、数値の部分は自動的に文字列に変換され、文字列同士の足し算が行われます。この例だと、数値3.14が文字列"3.14"に変換され、文字列"PI = "と"3.14"の足し算として計算（結合）されます。このような文字列と数値の足し算の答えは**文字列**になるわけです（図4-16）。

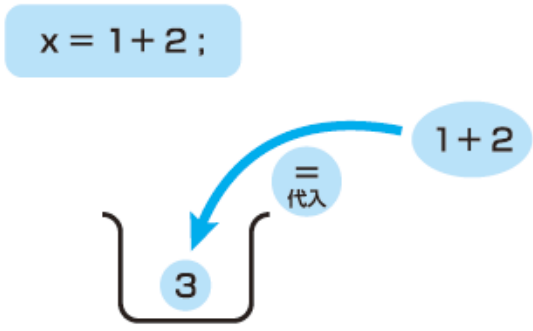


●図4-16　文字列と数値の演算

TIPS
文字列を含む計算では、結果は必ず文字列になります。

算数では「=」は代入以外に、「等しい」という意味で使用されますが、Javaでは代入以外の意味はありませんので、「**xと3が等しいことを示しているのではない**」ということに注意してください。なお、変数を宣言した後、はじめてその変数に値を代入することを**初期化**と呼びます。

TIPS
Javaでは、「等しい」を「==」で表現します。



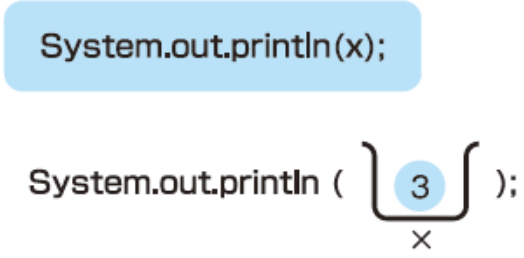
● 図5-04　x = 1 + 2による計算結果を変数xへ代入

5-1-4 変数の表示

● 変数の表示（リスト5-02　6行目）

```
System.out.println(x);
```

文字列の表示は、文字列を"でくくり、文字の表示は文字を'でくくることで行いました。しかし、**6行目**のxは、"や'でくくられていません。このxは文字ではなく、3行目で宣言した変数xなのです。printlnに、表示させるものとして変数を渡すと、その変数が持っている値が表示されます。この例では、5行目で変数xの値は3になっていますので、3が表示されます（図5-05）。



● 図5-05　System.out.println(x);による変数xの値の表示

もちろん、6行目を次のようにすると変数xではなく文字xとして扱われます。この場合、3ではなく、文字xが表示されます。

```
System.out.println('x');
```

▼リスト5-11　byte型変数を使ったプログラム(Hensu_byte.java)

```
1. class Hensu_byte {
2.   public static void main(String[] args) {
3.     byte age;
4.
5.     age = 52 - 128;
6.     System.out.println("あなたは" + (age + 128) + "歳です。");
7.   }
8. }
```

Hensu_byte.javaの実行結果は、図5-20のとおりです。

●図5-20　Hensu_byte.javaの実行結果

5行目で変数ageに年齢から、128を引いた値を入れ、**6行目**のprintlnで表示するときに128を足しているのがポイントです。また、数値の計算を先に行うため6行目で「(age + 128)」と括弧を使っています。

short型は最大32,767までの値を扱うことができます(**リスト5-12**)。

▼リスト5-12　short型変数を使ったプログラム (Hensu_short.java)

```
1. class Hensu_short {
2.     public static void main(String[] args){
3.         short age;
4.
5.         age = 300;
6.
7.         System.out.println("この木の樹齢は" + age + "歳です。 ");
8.     }
9. }
```

Hensu_short.javaの実行結果は図5-21のとおりです。

6-1 判断（if）

ifはプログラムに幅を持たせるための重要なキーワードです。条件によってプログラムの流れを変える方法を学びましょう。

6-1-1 もし～ならば

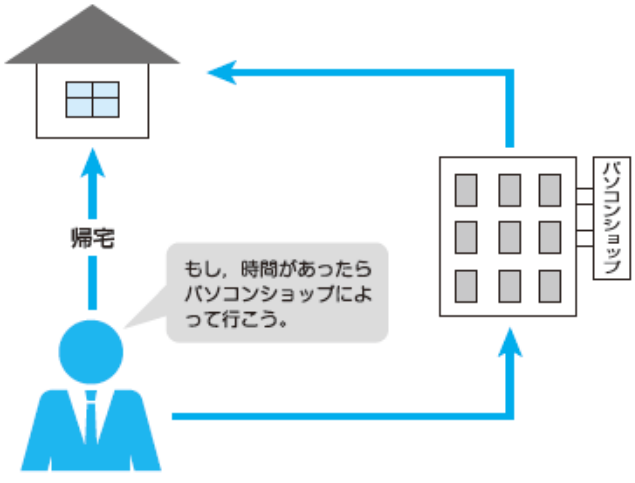
これまでに紹介したプログラムでは、処理の流れが上から下に順番に実行される一本道のプログラムでした。ここからは、処理の流れをある特定の条件によって変化させるプログラムの作成法について学んでいきます。

プログラムの流れを変化させるものを**制御構造**といいます。制御構造には、これまでの上から下へ順番に処理が行われる**順次**、ある特定の条件で処理を行う**判断**、同じ処理を何度も行う**繰り返し**の3つがあります。

6章では、その1つである**判断**について説明を行います。

「もし、時間に余裕があったらパソコンショップによってみよう。」とか、もし、おこづかいが余っていたらJavaの入門書を買ってみよう。」など、「もし～ならば」という「判断に基づく動作の決定」は私たちの普段の生活でもよくあることです。

「時間に余裕があったら」とか「おこづかいが余っていたら」のように、ある特定の条件が成り立ったときにだけ特別の処理を行うプログラムを記述することができます（図6-01）。



● 図6-01 もし時間があったら

なお、1つのswitch文の中で:と->は混在させることができません。どちらか1つで統一して記述をしてください。

6-2-4 switch式

switchは文だけではなく、式としてswitchの判定に応じた値を返すように使用することもできます。例えば、Bunki5.javaで表示させている文字列を変数messageに代入させるためには、

```
case 1 -> message = "Good morning.";
```

のように、それぞれで変数messageに代入させなければなりません。しかし、switch式を使うと各caseで代入する値だけを決定して、決定した結果を代入することができます。yieldの後に値や式を記述すると、該当するcaseでその値を返しますが、アロー構文を使っている場合は、直接値や式を記述します。

なお、switch式では最後に;を付けるのを忘れないように注意しましょう。

● switch式の書式

```
変数名 = switch(変数) {
    case 定数1 [,定数1-2, …] : yield 値1;
    [ case 定数2 [,定数2-2, …] : yield 値2;
     default : yield 値n; ]
};  ←;を忘れないように(式なので;が必要)
```

● switch式の書式(アロー構文)

```
変数名 = switch(変数) {
    case 定数1 [,定数1-2, …] -> 値1;
    [ case 定数2 [,定数2-2, …] -> 値2;
     default -> yield 値n; ]
};
```

switch式を利用したプログラムがリスト6-12です。

▼リスト6-12　switch式を利用したプログラム(Bunki6.java)

```
1. class Bunki6 {
2.     public static void main(String[] args) {
3.         int a = 1;
4.         String message = switch(a) {
5.             case 1: yield "Good morning.";
6.             case 2: yield "Good afternoon.";
7.             case 3: yield "Good evening.";
8.             default: yield "Good night.";
```

解答・解説集

▶ この解答・解説集は、3～8章の各章末の練習問題の解答です。
▶ 薄く糊付けしてありますが、本書より取り外して使用することもできます。

CHAPTER 3　練習問題　P.67～68

問題 1

答え　④

解説　①はクラス名定義の制約をすべて満たしており、クラス名として使用できます。②はJava予約語の1つであるdoを含んでいますが、クラス名の一部として使用しているので問題はありません。③はJava予約語のnewとJava予約語のinterfaceを組み合わせたものです。それぞれが予約語であっても組み合わせて使用しているのであれば問題ありません。④はJava予約語です。

クラス名に関しては3-1-2をご覧ください。

問題 2

答え　③

解説　①は「'」を意味するエスケープシーケンス、②は「¥」マークを意味するエスケープシーケンス、③はタブを意味するエスケープシーケンス、④は改行を意味するエスケープシーケンスとなります。

エスケープシーケンスに関しては3-3-2をご覧ください。

問題 3

答え

```
1. class Renshu33 {
2.     public static void main(String[] args){
3.         System.out.println("Renshu!!!");
4.     }
5. }
```

解説　4行目に閉じ括弧を1つ加えました。この括弧は2行目の開き括弧と対応しています。2行目から4行目までででmainメソッドの範囲、1行目から5行目までででクラスRenshu33の範囲を示しているのです。

クラスやメソッドの範囲（ブロック）に関しては3-1-6をご覧ください。

問題 4

答え

```
1. class Renshu34 {
2.     public static void main(String[] args) {
```