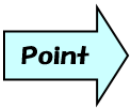
Java
による
オブジェクト指向プログラミング
(Java SE 11対応)

入門編

TAC

Contents

【テキストで使われる記号】



理解を深めるポイントや補足を強調しています。読み飛ばさないように注意しましょう。

Windows OS 向けの説明です

macOS 向けの説明です

Windows/macOS 両方向けの説明です

【システム環境】

以下の環境で動作確認をしています。

- Windows 10（64 ビット版）
- Windows 11
- macOS Big Sur 11

【インストールするアプリケーション】

以下のバージョンのアプリケーションをインストールします。

- OpenJDK Eclipse Temurin 11
- Visual Studio Code 1.65

第2章
Java プログラムの基本構造

この章では、Java 言語を使ってプログラムを
作成する手順と、作成する上でのポイントついて
学習します。

2.3.3　新規ソースファイルを作成する

次に新しく Java ソースファイルを作成します。下図のように[エクスプローラー]エリアにマウスカーソルを移動すると現れる、[新しいファイル]ボタンをクリックします。

新しいファイルボタン

※　フォルダーが折りたたまれている状態だと、[新しいファイル]ボタンは出現しません。そのときはフォルダ名（JAVA）をクリックするとフォルダーが展開され、[新しいファイル]ボタンが出現します。折りたたまれているかどうかはフォルダ名の前のアイコンで判断します。下の図を参考にしてください。

折りたたまれている

展開されている

現われたボックスにファイル名「HelloJava.java」と入力し、Enter キーを押して確定します。ファイル名は**大文字小文字を含め、正確に**入力してください。

ファイル名の入力

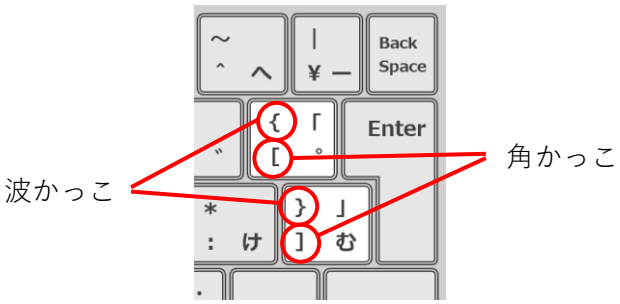
ファイル名を間違えると、動作しません。今回大文字となるのは、Hello の H と Java の J です（拡張子.java はすべて小文字です）。ファイル名を間違えたときは、右クリック⇒[名前の変更]で変更することができます。

「～.java」まで入力すると、Java ソースファイルであると判定して、上の図のようにファイル名の前にアイコンが表示されます。

2.3.4　ソースコードを入力する

以下のソースコードを入力します。Tab の入力は Tab キーで行いますが、今回の VSCode 環境では自動的に入力されます。Java では半角/全角やキーワードの大文字/小文字を厳密に区別しますので、注意しましょう。例えば、class の後の HelloJava は H や J が大文字ですし、String の S や、System の最初の S は大文字です。

波かっこ{}や角かっこ[]は下図のように、Enter キーの隣にあります。もちろん IME をオフにして半角で入力します。

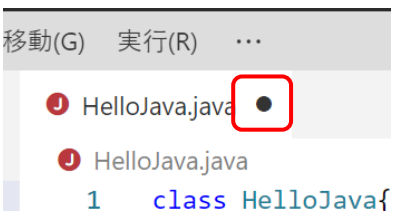


入力するソースコード【ファイル名：HelloJava.java】

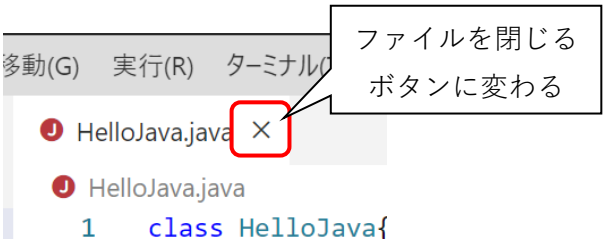
```
1:  class△HelloJava{
2:  ____ public△static△void△main(String[]△args){
3:  ____ ___System.out.println("Hello△Java!!");
4:  ____ ___System.out.println("初めての Java です。");
5:  ____ }
6:  }
```

△：半角スペースを入力　　____：Tab を入力

入力が終わったら、メニューから[ファイル]⇒[保存]で保存します。保存されているかどうかは、下の図のようにエディター画面上部のタブを見ると、●（未保存）→×（保存済み）に変化することで確認できます。



未保存



保存済み

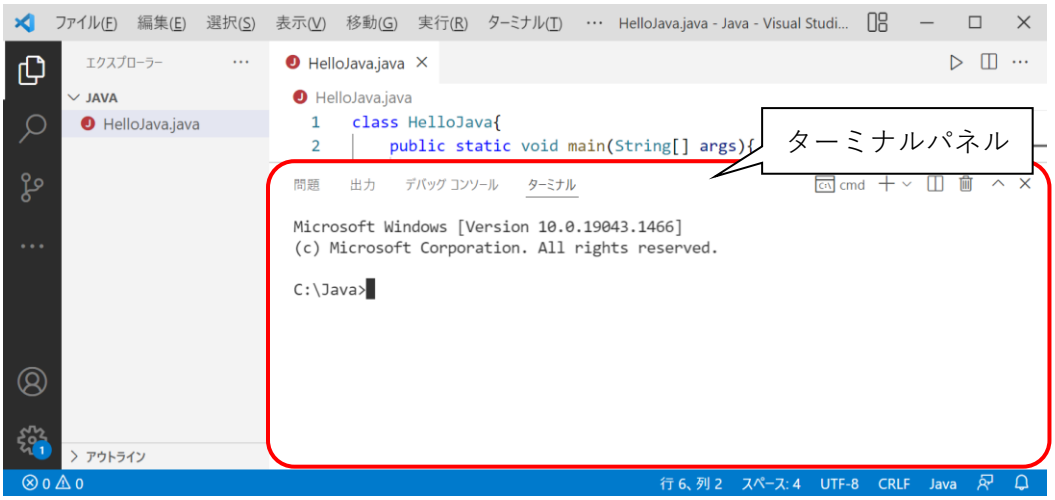
2.3.5　コマンドプロンプトの起動

VSCode を使ってプログラムを作成すると、先述したようにボタンからコンパイル、あるいはプログラムの実行をおこなうことができます。

しかし、今回は初めてプログラムを作成しますので、コマンドを使ってコンパイルとプログラムの実行を行います。

VSCode のメニューから[**ターミナル**]⇒[新しいターミナル]を選択してください。

ソースコードの下部に[ターミナルパネル](Windows ではコマンドプロント、macOS ではシェル）が開きます。



2.3.6　ソースファイルのコンパイル

次に作成したソースファイル[HelloJava.java]をコンパイルします。コンパイルをするには、[ターミナルパネル]に次のコマンドを入力し、Enter キーを押します。

```
javac ファイル名.java
```

「**javac コマンド**」は Java コンパイラを起動するためのコマンドです。その後に半角スペースを入力してからコンパイルしたい「**ファイル名**」を入力します。

Win

今回は「javac HelloJava.java」と入力すればよさそうですが、実は Windows ではエラーとなってしまいます。

```
C:\Java>javac HelloJava.java
HelloJava.java:4: エラー: この文字(0x80)は、エンコーディングwindows-31jにマップ
できません
        System.out.println("蛻昴ａ縺ｦ縺ｄJava縺ｧ縺吶??");
                                                ^
    ======== 以下略 ========
```

コンパイルエラーのメッセージ

これは、VSCode のデフォルト文字コードが UTF-8 で、Windows 標準の文字コードが Shift-JIS（windows-31j）と異なっているため、UTF-8 で書かれたソースコードは、コマンドプロンプトでは（特に日本語文字コードが）あり得ない文字コード番号と誤解釈されてしまうためです。

そこで、UTF-8 で書かれたソースコードであることをコンパイラに教えるために、以下の通りオプションを付けてコマンドを発行します。

```
javac -encoding utf-8 HelloJava.java
```

コンパイルオプション

最後に Enter キーを押して、コマンドを確定します。何も表示されず、次のコマンド待ち「C:¥Java>」が表示されれば成功です。

```
C:\Java>javac -encoding utf-8 HelloJava.java

C:\Java>
```

コンパイル成功

Mac

macOS の場合は、文字コードが UTF-8 で統一されているので、Windows にあるような煩わしさはありません。javac コマンドのあとに何も表示されず、次の入力待ちになれば、コンパイル成功です。

```
$ javac HelloJava.java
$
```

コンパイル成功

Win/Mac

誤っていればなんらかのメッセージが表示されますので、ソースコードを見直し、再度保存してからコンパイルします。保存を忘れがちなので、注意しましょう。コンパイルエラーについては、「エラー！ 参照元が見つかりません。 エラー！ 参照元が見つかりません。」にまとめてあります。

2.3.7　バイトコード（クラスファイル）の生成

ソースファイルのコンパイルに成功すると作業フォルダの中には拡張子「.class」のついたバイトコード（クラスファイル）が生成されます。

ただし、コンパイルエラーが発生した場合は、バイトコードは自動的に生成されません。コンパイルエラーについては次の節で説明します。

Point

バイトコード（クラスファイル）が生成されているかを確認する

本文ではバイトコードが生成される、と説明しましたが、「本当に生成されているの？」と疑問に思うでしょう。そこで、本当にバイトコードが生成されているかどうか確認してみます。
現在開いているコマンドプロンプト画面から次のコマンドを入力して、Enter キーを押してください。

C:¥Java>**dir**
（macOS では **ls** コマンド）
dir コマンドを実行すると現在自分が作業をしているフォルダの中（**カレントディレクトリ**）にあるファイルの一覧が表示されます。
その中に「HelloJava.class」というファイル名があります。それがバイトコード（クラスファイル）です。

```
C:\Java>dir
 ドライブ C のボリューム ラベルは Windows-SSD です
 ボリューム シリアル番号は 52C7-C091 です

 C:\Java のディレクトリ

2022/02/10  19:28    <DIR>          .
2022/02/10  19:28    <DIR>          ..
2022/02/10  19:28               467 HelloJava.class
2022/02/10  18:31               174 HelloJava.java
               2 個のファイル                 641 バイト
               2 個のディレクトリ  791,748,653,056 バイトの空き領域

C:\Java>
```

2.3.8　バイトコードの実行

次に作成されたバイトコード（クラスファイル）を Java VM に実行させます。実行させるには次のコマンドを入力し、Enter キーを押します。

java クラス名（＝拡張子[.class]を除いた、バイトコードのファイル名）

```
C:\Java>javac -encoding utf-8 HelloJava.java

C:\Java>java HelloJava
```

java コマンド

「**java コマンド**」は作成したプログラムを実行するためのコマンドです。その後に半角スペースを入力してから実行したい「**クラス名（バイトコードのファイル名）**」を入力します。ここで、注意しなければいけないのは、**クラス名の後に拡張子「.class」を入力しない**こと、**大文字・小文字も含めて正確に**入力することです。コンパイルとは異なりますので、注意してください。

プログラムを実行すると、画面に「Hello Java!!」と「初めての Java です。」の 2 行が表示されます。

```
C:\Java>javac -encoding utf-8 HelloJava.java

C:\Java>java HelloJava
Hello Java!!
初めてのJavaです。

C:\Java>
```

実行結果

2.3.9　VSCode の機能を用いてワンクリックで実行する

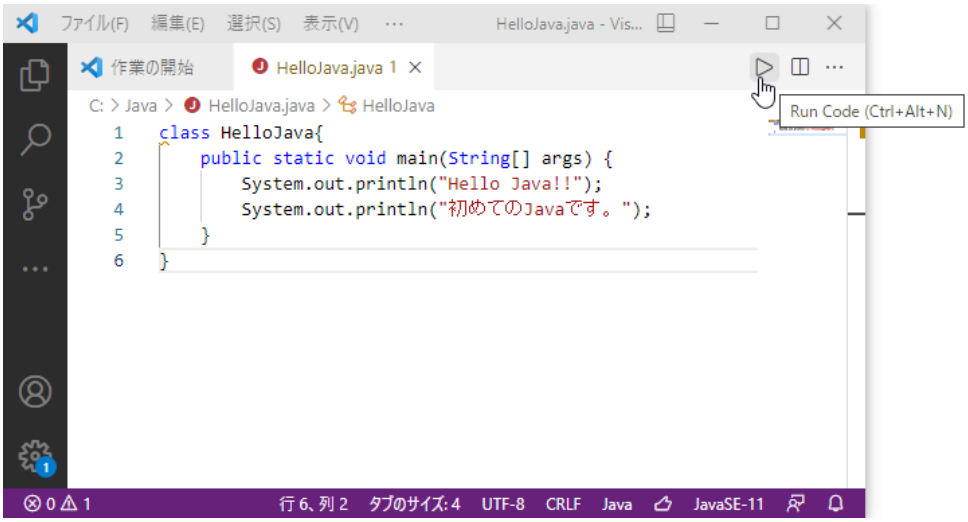
これまで見てきたように、ソースコードの入力が終わったら、①入力したソースコードを保存し、②コンパイルを行い、エラーがなければ③実行しますが、本書付属の DVD に収録されている VSCode では、拡張機能「**Code Runner**」がすでにインストール済みです。Code Runner では、

①入力中のコードを保存する

②コンパイルを行う

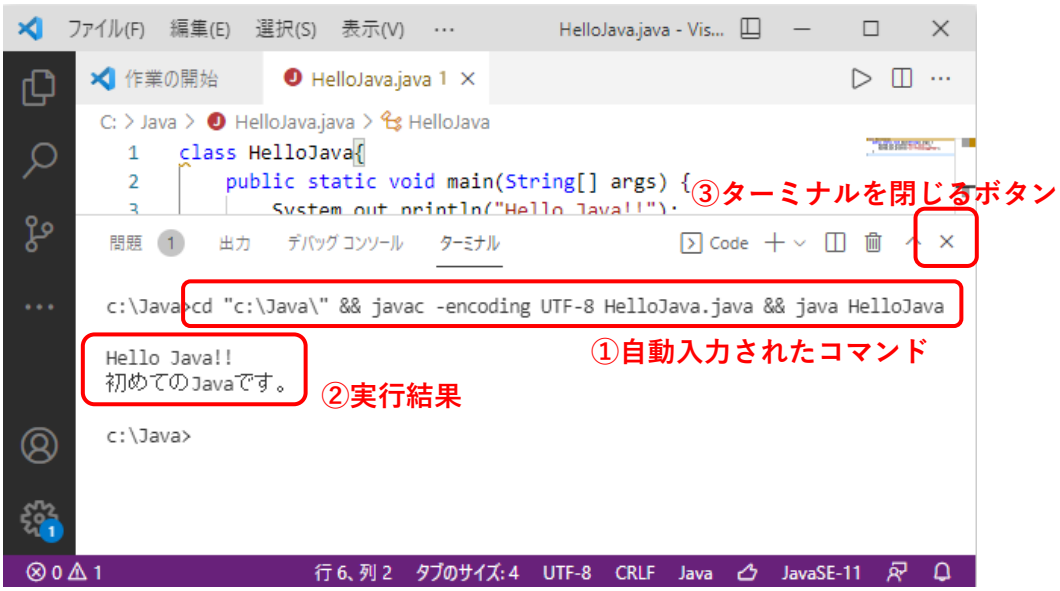
③プログラムを実行する
という 3 つの操作をワンクリックで行えるように設定してあります。

P.18 の「1.5 テキストエディタ（VSCode）の使い方」」にあるように、プログラムを入力したら、下の図のように、**右上の▷ボタン（Run Code）**を押すと、プログラムが実行されます。



Run Code ボタン

実行結果は下の図のように、ターミナルパネルに表示されます。



ターミナルパネルで実行結果を表示

Code Runner はプログラミング言語に応じて、設定したコマンドを自動的に発行することができ、さまざまなプログラミング言語で利用できます。前の図の①を確認してみましょう。&&は複数コマンドの区切りですので、分解してみると、

```
1. cd "c:¥Java¥"
2. javac -encoding UTF-8 HelloJava.java
3. java Hello.java
```

の 3 つのコマンドが発行されています。それぞれ、

1. C ドライブの Java フォルダ（HelloJava.java ファイルの保存場所）に移動
2. javac コマンドで HelloJava.java ファイルをコンパイル
3. java コマンドで HelloJava クラスを実行

という意味で、今まで手入力してきたものを自動化しているのがわかります。コンパイルでエラーが発生したときは、コンパイルエラーメッセージを表示して、プログラムの実行は行いません。

また、③にあるように、ターミナルパネルが邪魔になったら閉じることができます。ソースコード入力中は閉じておくとよいでしょう。

今後は、長いコマンドを入力することなく効率よくプログラミングができますね！

Point

VSCode で¥を入力すると、＼と表示されてしまいます

¥も＼も文字コードとしては変わりありません。実は¥と表示されるのは、日本人向けであって、海外では一般に＼と表示されます。実際、英語キーボードには、¥という文字は刻印されておらず、＼です。
VSCode は海外生まれのソフトウェアなので、¥表示には対応していません。気にせずプログラミングしましょう！

Point

クラスファイルを作成せずに実行する

JDK11 から、java コマンドを用いて、ソースファイル（～.java）を直接実行できるようになりました。例えば、ソースコードを Test.java ファイルに記述した場合、

```
java Test.java
```

とコマンドを入力すると実行できるようになったのです。コンパイルの手間が省けるようになりました。コンパイルされたバイトコードはメモリ上にのみ存在し、クラスファイルは生成されません。

この機能により、コードの一部を簡単にテストしたり、Java の学習を始めたばかりの方のハードルが低くなりました。

ただし、以下のような制限があります。
・ 単一のソースコードのみ実行可能
・ コンパイルオプションは設定できない