

# 第3章 Javaの基本

## 目次

- プログラムのコンパイルと実行
- コンパイルエラー
- Javaプログラムの作成・実行
- 画面への出力
- コードの内容
- 文字と数値

## Javaのコンパイルの仕組み

コンパイルとは・・・

→プログラム言語をコンピュータが理解できる言語(機械語)に変換すること。

コンパイラとは・・・

→コンパイルのための翻訳ソフトウェア

## Javaのコンパイルの仕組み

コンパイラ言語(C言語など):

→特定のOSに合わせてプログラムを作成する必要がある。

Java:

→Windows、Mac、LinuxなどOSが違ってても、  
プログラムを変更する事なく動かせるという理念  
(Write Once, Run Anywhere)で作られている。

## Javaのコンパイルの仕組み

Javaは特定のコンピュータが理解できる機械語への翻訳作業を2度に分けて行っている。

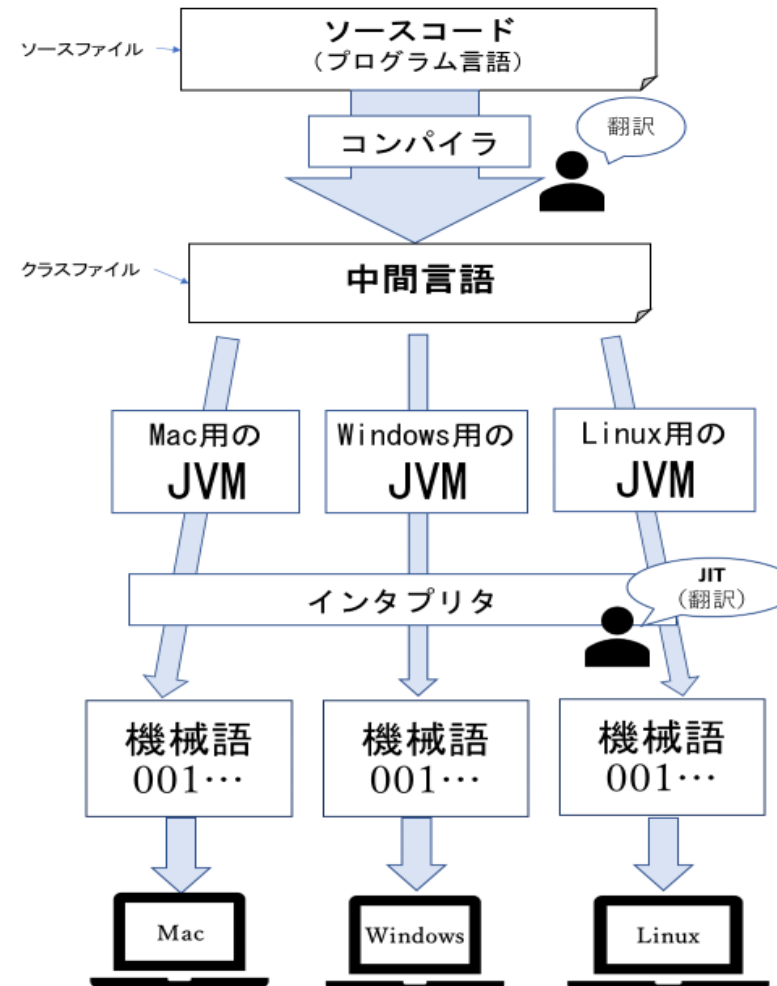
①コンパイラによる中間言語への翻訳

→バイトコードと呼ばれる中間言語に翻訳され、  
クラスファイルが作成される。

②インタプリタによる機械語への翻訳

→バイトコードを0と1で表す機械語に翻訳して実行する。

## Javaのコンパイルの仕組み



## 【Sample0301】を作成しましょう



## コンパイルエラーとは

プログラムをコンパイルした際、コンピュータが分かる形に  
翻訳できなかったときに発生するエラー

```
C:\¥Java_practice>javac Sample0302.java
Sample0302.java:3: エラー: 文字列リテラルが閉じられていません
    System.out.println("ようこそJavaへ!");
                        ^
Sample0302.java:3: エラー: ';'がありません
    System.out.println("ようこそJavaへ!");
                        ^
Sample0302.java:5: エラー: 構文解析中にファイルの終わりに移りました
    }
    ^
エラー3個
C:\¥Java_practice>_
```



# 【Sample0302 コンパイルエラーを発生させる】 を作成しましょう

Let's try!



## Sample0302のポイント

コンパイル結果から、エラーが発生したことが確認できる。  
コマンドプロンプト上にはエラーが発生した場所と  
内容が表示されている。

これらのエラーを正しく改修することで正常に処理が実行できる。

```
C:\Java_practice>javac Sample0302.java
Sample0302.java:3: エラー: 文字列リテラルが閉じられていません
    System.out.println("ようこそJavaへ!");
                        ^
Sample0302.java:3: エラー: ';'がありません
    System.out.println("ようこそJavaへ!");
                        ^
Sample0302.java:5: エラー: 構文解析中にファイルの終わりに移りました
    ]
    ^
エラー3個
C:\Java_practice>_
```

## 【Sample0303】を作成しましょう



## 【Sample0304 コードを書いてみる】を作成しましょう



## Sample0304のポイント

「System.out.println()」

→標準出力に「()」の中の値を表示する処理。

標準出力とは・・・プログラム中の値を表示する出力先のこと。  
指定がない場合、コンソールが標準出力先となる。

```
class クラス名 {  
    public static void main(String[] args) {  
        System.out.println("出力したい文字列");  
    }  
}
```

## 【Sample0305 printを使う】を作成しましょう



## Sample0305のポイント

println: 1回の出力結果ごとに改行される

print: 改行されずに続けて出力される

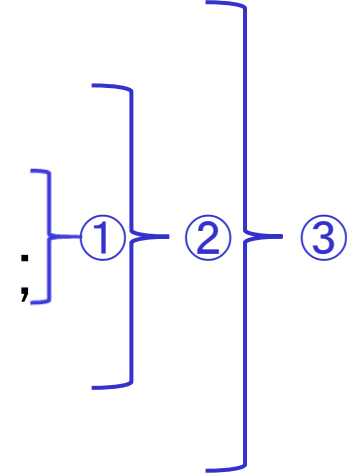
こんにちは！ よろしくお願ひします。

- 文字列を改行したいとき: `System.out.println();`
- 文字列を続けて並べたいとき: `System.out.print();`

## コードの内容:コードの構成

- ①: メソッドの処理
- ②: メソッドの定義
- ③: クラスの定義

```
public class Sample0304 {  
    public static void main(String[] args) {  
        System.out.println ("こんにちは！");  
        System.out.println ("よろしくお願いします。");  
    }  
}
```





## main()メソッド

プログラムを実行したときに、最初に実行されるメソッド

メソッド・・・実行したい処理を記述する

ブロック・・・「{}」のこと、どこからどこまでがその範囲かを表す

```
public static void main(String[] args) {  
    . . .  
}
```

## main()メソッド

mainメソッドが存在しない場合、プログラムが実行されない。

```
public class Sample0304 {  
    public static void start(String[] args) {  
        System.out.println ("こんにちは！");  
        System.out.println ("よろしくお願いします。");  
    }  
}
```

プログラムの処理はmain()メソッドから始まります。

## 1文ずつ処理する

文とは、小さな処理の単位のこと。

1つの文の終わりを示すには「;」をつける。

文は先頭から1文ずつ順番に実行される。

```
System.out.println("こんにちは！");  
System.out.println("よろしくお願いします。");
```

文の終わりには「;」を付ける必要があります。  
文は原則として先頭から順番に処理されます。

## コードを読みやすくする

ブロック内では字下げ(インデント)を行う。  
インデントを行うには、行頭でスペースキー  
またはタブキーを押す。

```
public class Sample0304 {  
→ public static void main (String[] args) {  
→ System.out.println ("こんにちは。");  
}
```

コードを読みやすくするために  
インデントや改行を使いましょう。

## コメントを記述する

コードの中には、プログラムの動作には、  
影響を与えないメモを「コメント」として残すことができる。

- 1行コメント

→「//」を記述した位置から行の終わりまでがコメントになる。

```
// コメント
```

## コメントを記述する

- 複数行コメント

→複数行にわたってコメントを記述することができる。

「/\*」と「\*/」で囲まれた記述がすべてコメントになる。

```
/*  
 * コメント  
 * コメント  
*/
```

## コメントを記述する

- JavaDocコメント  
→コメントが記述されたソースコードにある操作を行うと、  
「そのソースコードについての説明が記載されたファイル」を  
自動で作成することができる。(詳細は別章)

```
/**  
 * コメント  
 * コメント  
 */
```

## コメントを記述する

ショートカットキーを使って、効率的にコメントの入力ができる。

コメントの記述方法	ショートカットキー
行コメント	Ctrl+/ 
ブロックコメント	Ctrl+Shift+/ 
Javadocコメント	Alt+Shift+j 

コメントを使って、コードの内容を分かりやすくしましょう。



## クラス

- Javaのプログラムは、全てクラスの中に記述する。
- 「class」の後に記述される言葉がクラス名となる。
- Javaのプログラムを作成する際には、  
最低1つ以上のクラスを定義する必要がある。

```
public class Sample0304 {  
    . . .  
}
```

Javaのコードは1つ以上のクラスから成り立ちます。

## 【Sample0306 様々な値を出力する】を作成しましょう



## Sample0306のポイント

様々な文字や数値が出力されている。

’J’、”Javaを学ぼう！”、123といった文字や数値の値のことをリテラル(literal)と呼ぶ。

```
J  
Javaを学ぼう！  
123
```

## リテラルとは

文字や数値の値のことをリテラル(literal)という。  
Javaのリテラルには大きく分けて、次の6種類がある。

- ① 文字リテラル
- ② 文字列リテラル
- ③ 整数リテラル
- ④ 浮動小数点リテラル
- ⑤ 論理値リテラル
- ⑥ nullリテラル

## 文字リテラル

「A」や「山」といった1つの文字を表現するリテラル。  
「'」で囲って利用する。

```
'A'  
'a'  
'あ'
```

1つの文字をあらわすときは、「'」で囲みます。

## 整数リテラル

小数点をもたない数値を表現するリテラル。  
「'」や「"」で囲まらずに記述する。

```
1  
10  
100
```

数値は「'」や「"」で囲みません。

## 文字列リテラル

複数の文字を表現するリテラル。  
「 ” 」で囲って記述する。

“Java”  
“プログラミング”  
“楽しい”

複数の文字をあらわすときは、「"」で囲みます。

## その他のリテラル

- ・浮動小数点リテラル

- 小数部をもつ数値を表現するリテラル。  
「'」や「”」で囲まずに記述する。

- ・論理値リテラル

- 真を表す「true」か、偽を表す「false」の値を表現するリテラル。

- ・nullリテラル

- 参照型のデータを扱う際に  
「何も参照していない」を表現するリテラル。



## エスケープシーケンス

「¥」と組み合わせて1文字を表すこと。

エスケープシーケンス	意味している文字
¥'	'
¥"	"
¥¥	¥
¥t	水平タブ
¥n	改行
¥○○○	8進数○○○の文字コードをもっている文字 (○は0から7の数字)
¥uxxxx	16進数¥uxxxxの文字コードを持っている文字 (xは0から9の数字、AからFの英字)

# 【Sample0307 エスケープシーケンスを使う】 を作成しましょう

Let's try!



## Sample0307のポイント

「¥¥」や「¥”」と記述した部分が、「¥」や「”」と出力されている。

円マーク : ¥  
ダブルクォーテーション : ”

エスケープシーケンスを使うことによって、  
特殊な文字を表示させることができます。

## 文字コード

コンピュータ上で文字を表示するために、  
一つ一つの文字に固有に割り当てた番号のこと。

JavaではUnicode(ユニコード)が仕様されている。  
文字コードを指定して、文字を出力することができる。

# 【Sample0308 リテラルとエスケープシーケンス】 を作成しましょう

Let's try!



## 章のまとめ

- `main()`メソッドは、プログラムの開始位置を示しています。
- 文の最後には、「;」を付ける必要があります。
- コメントを使って、コードの内容を分かりやすくできます。
- Javaのプログラムは1つ以上のクラスから成り立ちます。
- 「`'`」で囲まれた記述を、文字リテラルと呼びます。
- 「`"`」で囲まれた記述を、文字列リテラルと呼びます。
- 特殊な文字の出力には、エスケープシーケンスを使います。