> **Java基礎文法最速マスター** ★★☆
> Java

Javaの文法一覧です。他の言語をある程度知っている人はこれを読めばJavaの基礎をマスターしてJavaを書くことができるようになっています。簡易リファレンスとしても利用できると思いますので、これは足りないと思うものがあれば教えてください。

1. 基礎

classの作成

プログラムはclassに記述します。たとえばSampleという名前のclassを作る場合、Sample.javaファイル内に次のように書きます。

```
public class Sample {

}
```

mainメソッドの作成

プログラムはclass内のmainメソッドの先頭から実行されます。mainメソッドは次のように書きます。

```
public class Sample {

    public static void main( String[] args ) {
         // 処理を書く
     }

}
```

System.out.printlnメソッド

文字列を表字するメソッドです。

```
System.out.println( "Hello world" );
```

コメント

コメントです。

```
// 一行コメント

/*
   複数行コメント
 */

/**
   JavaDocコメント
 */
```

変数の宣言

変数の宣言です。変数の宣言時にはデータ型を指定します。

```
// 変数
int num;
```

データ型

データ型です。Javaのデータ型にはプリミティブ型と参照型があります。以下はプリミティ

ブ型のデータ型です。

```
// int(整数)型
int num;
// char(文字)型
char c;
// float(単精度浮動小数点)型
float value;
// double(倍精度浮動小数点)型
double value;
// boolean(論理)型
boolean flag;
```

以下は参照型のデータ型です。

```
// String型
String s;
// Date型
Date d;
// 配列型
String[] array;
```

プログラムのコンパイル

プログラムをコンパイルするには、コマンドラインで以下のようにします。

```
javac Sample.java
```

プログラムの実行

プログラムを実行するには、コマンドラインで以下のようにします。

```
java Sample
```

2. 数値

数値の表現

int、float、double型の変数に数値を代入できます。int型には整数だけ代入できます。
float、double型には整数でも小数でも代入できます。

```
int i = 2;
int i = 100000000;

float num = 1.234f;

double num = 1.234;
```

四則演算

四則演算です。

```
num = 1 + 1;
num = 1 - 1;
num = 1 * 2;
num = 1 / 2;
```

商の求め方です。割る数と割られる数が両方とも整数の場合、計算結果の小数点以下が切り捨てられます。

```
num = 1 / 2;  // 0
```

割る数と割られる数のどちらかが小数の場合、計算結果の小数点以下が切り捨てられません。

```
num = 1.0 / 2;    // 0.5
num = 1 / 2.0;    // 0.5
num = 1.0 / 2.0;  // 0.5
```

余りの求め方です。

```
// 余り
mod = 4 % 2
```

インクリメントとデクリメント

インクリメントとデクリメントです。

```
// インクリメント
 ++i;

// デクリメント
 --i;
```

3. 文字列

文字列の表現

文字列はダブルクォートで囲みます。

```
String str = "abc";
```

文字列操作

各種文字列操作です。

```
// 結合
String join = "aaa" + "bbb";

// 分割
String[] record = "aaa,bbb,ccc".split( "," );

// 長さ
int length = "abcdef".length();

// 切り出し
"abcd".substring( 0, 2 )   // abc

// 検索
int result = "abcd".indexOf( "cd" ) // 見つかった場合はその位置、見つからなかっ
```

4. 配列

配列変数の宣言

配列です。

```
// 配列の宣言
int[] array;
```

配列の生成

配列の生成です。配列の生成時には要素数を指定するか、初期データを指定します。

```
int [] array;

// 要素数を指定して配列を生成
array = new int[5];

// 初期データを指定して配列を生成
array = new int[] { 1, 2, 3 };

// 宣言と同時に配列を生成
int[] array2 = new int[5];
```

配列の要素の参照と代入

配列の要素の参照と代入です。

```
// 要素の参照
array[0]
array[1]

// 要素の代入
array[0] = 1;
array[1] = 2;
```

配列の要素数

配列の要素数を取得するには以下のようにします。

```
array_num = array.length;
```

配列のコピー

配列の要素を別の配列にコピーするには以下のようにします。

```
int[] from = new int[] { 1, 2, 3 };
int[] to = new int[5];

System.arraycopy( from, 0, to, 0, from.length );
```

5. 制御文

if文

if文です。

```
if ( 条件 ) {

}
```

if 〜 else文

if 〜 else文です。

```
if ( 条件 ) {

} else {

}
```

if 〜 else if 文

if 〜 else if文です。

```
if ( 条件 ) {

} else if ( 条件 ) {

}
```

while文

while文です。

```
int i = 0;
while ( i < 5 ) {

    // 処理

    ++i;
}
```

for文

for文です。

```
for ( int i = 0; i < 5; ++i ) {

}
```

for-each文

for-each文です。配列の各要素を処理できます。

```
int[] fields = new int[] { 1, 2, 3 };

for ( int field: fields ) {

}
```

6. メソッド

Javaでは関数をメソッドと言います。メソッドを作るには次のようにします。戻り値を返却するにはreturn文を使います。

```
static int sum( int num1, int num2 ) {
    int total;

    total = num1 + num2;

    return total;
}
```

9. ファイル入出力

ファイル入出力です。ファイル入出力を行うには、プログラムの先頭に以下を記述します。

```
import java.io.*;
```

以下がファイル入力の雛形になります。ファイルのオープンや読み込みに失敗した場合、catch節に処理が移ります。

```
BufferedReader reader = null;

try {
    reader = new BufferedReader( new FileReader( filename ) );

    String line;
    while ( ( line = reader.readLine() ) != null ) {

    }

} catch ( IOException e ) {
    // エラー処理:

} finally {
    if ( reader != null ) {
        try {
         reader.close();
        } catch ( IOException e ) {}s
    }
}
```

以下がファイル出力の雛形になります。ファイルのオープンや書き込みに失敗した場合、catch節に処理が移ります。

```
PrintWriter writer = null;

try {
    writer = new PrintWriter( new BufferedWriter( new FileWriter( filename ) ) )

    writer.println( "abc" );
    writer.println( "def" );
    writer.println( "fgh" );

} catch ( IOException e ) {
    // エラー処理:

} finally {
    if ( writer != null ) {
        writer.close();
    }
}
```

知っておいたほうがよい文法

Javaでよく出てくる知っておいたほうがよい文法の一覧です。

繰り返し文の途中で抜ける

繰り返し文の途中で抜けるにはbreak文を使用します。

```
for ( i = 0; i < 5; ++i ) {

    if ( 条件 ) {
        break;    // 条件を満たす場合、for文を抜ける。
    }

}
```

繰り返しの残り部分の処理をスキップする

残りの部分処理をスキップし、次の繰り返しに進むにはcontinue文を使用します。

```
for ( i = 0; i < 5; ++i ) {

    if ( 条件 ) {
        continue;    // 条件を満たす場合、残りの部分処理をスキップし、次の繰り返
    }

}
```

例外処理

例外を投げるにはthrow文を使用します。

```
throw new Exception( "Error messsage" );
```

例外処理をするにはtry 〜 catch文を使用します。

```
try {

    // 例外が発生する可能性のある処理

} catch ( Exception e ) {

    // 例外発生時の処理

}
```

Java参考資料

- Javaを使うなら必ず覚えておきたいデータ構造 - 配列・リスト・マップ
- Javaを使うなら理解しておきたいアルゴリズム - 抽出・ソート・結合・集計 (リスト&マップ編)
- Javaを使うなら理解しておきたいアルゴリズム - 抽出・ソート・結合・集計 (リスト&ビーン編)

基礎文法最速マスターリンク集

この記事は最近の基礎文法最速マスターの流れに便乗して作成したものです。

以下、各種基礎文法最速マスターへのリンクです。

- Perl基礎文法最速マスター - Perl入門〜サンプルコードによるPerl入門〜
- Route 477 - Ruby基礎文法最速マスター
- PHP基礎文法最速マスター | Shin x blog
- Python基礎文法最速マスター - D++のはまり日誌
- Java基礎文法最速マスター - 何かしらの言語による記述を解析する日記
- VBA基礎文法最速マスター - 何かしらの言語による記述を解析する日記
- Brainf*ck基礎文法最速マスター - 医者を志す妻を応援する夫の日記
- Haskell基礎文法最速マスター - think and error

- Bash基礎文法最速マスター - 何かしらの言語による記述を解析する日記
- VBScript 基礎文法最速マスター - CX's VBScript Diary - VBScript グループ
- JavaScript基礎文法最速マスター - なんとなく日記