

# 第1回：イントロダクション、Java基本文法

## ソフトウェア工学とは

ソフトウェア  
コンピュータシステム（ハードウェア）を動作させるために必要なプログラムコード、及び、ドキュメント（文書）やデータ  
身の回りのソフトウェア  
商用パッケージソフト(COTS)、ゲーム、エンタプライズソフトウェア、組み込みソフトウェア、Webアプリケーション、シミュレーションソフト、OS、クラウド (Software as a Service)  
ソフトウェアの規模 (行数 LOC (Lines Of Code) で数えると、... )

- プログラミング演習：数百LOC
- 研究：千～数万LOC
- Excel：64万 LOC
- DVDレコーダ：100万 LOC
- Firefox 20: 401万 LOC
- 車載ソフト (レクサス)：700万 LOC
- Windows 10：6500万 LOC
- Android OS：2000万 LOC
- Mac OS X Tiger：8500万LOC

ソフトウェア危機  
- こんな大きなソフトウェアをどうやってうまく作るのか？  
- もはや一人の天才だけでは作れない  
- 新しいソフトを1から作るのは非現実  
ソフトウェア工学 (Software Engineering)  
- ソフトウェアの開発、運用（利用）、保守に関する理論（モデル、方法論）を応用して、ソフトウェアの生産性や品質を向上させる技術の総称  
- 「工学」の意味するところ（個人性の保持(習得すれば誰でも同じようにできる）、生産性の向上(良いものを早く）

ソフトウェアのライフサイクル  
1. 要求定義：顧客がソフトウェアに何を求めているのかを洗い出す。  
2. 外部設計（基本設計）：システムがユーザとどのようにインタラクションするかを決定。  
3. 内部設計（詳細設計）：システムをどのようなクラス群で実現するかを決定。  
4. コーディング：プログラミング、コンパイル、デバッグも含む。 **(← プログラミング演習の範囲はここだけ)**  
5. テスト：ソフトウェアが仕様通り動作するか試験する。  
6. 運用：顧客の現場へのインストール、本番運用。  
7. 保守：プログラムメンテナンス、システム補強など。

## オブジェクト指向アプローチ

ソフトウェア開発における2大アプローチ  
- **構造化アプローチ**：ソフトウェアを処理の集合とみなす。対象とする問題を実現する処理を洗い出し、大きな処理を小さな処理へ細分化し、処理を指定した順に呼び出して問題にあたる。各処理は関数を基本単位として実装。データは関数へのパラメータと戻り値としてやり取りされる。  
- **オブジェクト指向アプローチ**：ソフトウェアをモノ（オブジェクト）の集合とみなす。対象とする問題に登場するモノを洗い出し、各モノの仕事の責任範囲を決め、モノ同士が助け合って問題にあたる。データと処理をひとまとまりにしたクラス(オブジェクトのひな形)を基本単位として実装。クラスから生成したオブジェクト（インスタンスという）は、互いに処理を依頼してやってもらう（委譲）。

プログラミング言語  
- 構造化言語：C, Fortran, Pascal, Perl, Basic  
- オブジェクト指向言語：**Java**, C++, C#, Ruby, Python  
本講義がカバーする範囲  
1. 要求定義：今から作るシステムの問題・仕様を自分で作成する。（UML: ユースケース図）  
2. 外部設計（基本設計）：ユーザがシステムとやり取りする画面を自分で設計する。  
3. 内部設計（詳細設計）：システムをどのようなクラス群で実現するかを設計する。（UML: クラス図、シーケンス図）  
4. コーディング：プログラミング、コンパイル、デバッグも含む。（Java言語 on Eclipse）  
5. テスト：ソフトウェアが仕様通り動作するか試験する。（JUnit）

本講義の目標  
オブジェクト指向に基づいた、基本的なJavaプログラミング、および、小規模ソフトウェアが開発可能となる素養を養う。  
第1回から第3回までは、Javaの文法に集中するため、**オブジェクト指向の書き方にはなっていないこと**に注意。

## Hello World in Java

Eclipseの起動  
ログした端末から、Eclipseを起動する（紫色の丸型のアイコン）。EclipseとはJavaで開発するときに必要な開発環境。

プロジェクトの新規作成  
ファイル → 新規 → Javaプロジェクト、プロジェクト名は任意。ここでは、SoftEng01としてみる。次へー完了。

新規クラスHelloの作成  
ファイル → 新規 → Javaクラス、名前をHelloにする。（Hは大文字、elloは小文字）、完了。

ソースのコピー&ペースト  
次のソースコードをマウスでドラッグしてコピー、Eclipse上のHello.javaの上でペーストする。

実行  
左のパッケージエクスプローラ内のHello.javaを右クリック → 実行 → Javaアプリケーション。Hello World！と表示されいOK。

ソースコード：Hello.java

```
/**
 * はじめてのJavaクラス。
 */
// Helloという名前のクラスを自分で宣言している。クラス名は大文字で始める。
public class Hello {
    // Helloクラスの中のmainという名前のメソッド(関数)
    // public, static, args今は気にしない
    public static void main(String[] args) {
        /* ---- 1. 標準出力(画面コンソール)の練習 ---- */
        System.out.println("Hello World!");
    }
}
```

解説

- 上記のコードは、最も基本的なJavaアプリケーションのソースの書き方、覚えてしまう。
- クラス名はHello"main"という処理を持っている。main()部は関数であり、main()部は関数を実行する際に「どのメソッドスタブを作成しますか?」のところで、チェックを入れると自動生成される。
- Javaのコメントは、/ ～ / (C言語風) または、// (1行コメント) で書く。習ったことを忘れないように、コメントを入れる癖をつけよう。
- System.out.println()は、標準出力に書き出す最も基本的な命令。最後に改行が自動的に入る。

## 変数、型、式、標準出力

変数  
データを入れるための箱。C言語と同様、任意の英数字列で名前を付ける。変数を定義(宣言)する際には、前に型を定義しなければならない。変数は、関数の中でも外でも定義可能。関数の中で定義する変数は、**ローカル変数**。関数の外、クラスの内側で定義する変数は、**フィールド変数**と呼ばれる。

型  
変数の中に入れるデータの型を宣言する。基本データ型とクラス型に分かれる。

基本データ型  
Javaは基本データ型として以下の8つを備えている。これらの型の変数は、箱の中身に値がそのまま入っている。

- boolean: true, falseのどちらかを表す。
- byte: 1バイトの文字を1つ入れる変数。あまり使わない。文字列 (String) クラスを使う。
- char: 2バイト文字も含めた文字を1つ入れる変数。あまり使わない。文字列 (String) クラスを使う。
- short: 16ビット長の短い整数型。あまり使わない。
- int: 32ビット整数型。よく使う。
- long: 64ビット整数型。よく使う。
- float: 浮動小数点型。doubleを使う。
- double: 倍精度浮動小数点型。よく使う。

クラス型  
Javaでは、クラス型（ひな形）であり、Javaで初めてから用意されている様々な出来合いのクラス、あるいは、自分で作成したクラスの形をした変数を定義できる。クラス型の変数は、箱の中身にオブジェクト（インスタンス）への参照アドレスが入っている。

- String: 文字列クラス、非常によく使う。文字列の定義は、「あいえお」のように、「”」で囲む。
- Hello: ↑で作成したHelloクラス、宣言と同時にnewして変数を作成する。

式  
Javaでは基本データ型に関しては、C言語にほぼ準拠した式、演算子を使う。

- 代入演算子：x = 10
- 算術演算子：+ (加算)、- (減算)、\* (乗算)、/ (除算)、% (剰余、intのみ適用)
- 比較演算子：== (等しい)、!= (等しくない)、> (より大きい)、< (より小さい)、>= (以上)、<= (以下)
- インクリメント、デクリメント：x++, y--

※一般にクラス型には上記の演算子を適用できない（演算子が定義されていないため）。Stringに関しては少し例外的。  
String str1 = "Nakamura", str2;  
str2 = str1 + " Masahide"; //String型の+演算は文字列の連結。str2= "Nakamura Masahide"になる。  
(str) str1 = str2 + str1; //減算演算子では使えない。  
//比較はできない。  
(数) str1==str2 //str1がstr2と等しいかどうか。間違ひ。  
(区) str1.equals(str2) //str1がstr2と等しいかどうか。

標準出力  
System.out.println("文字列"); //非常によく使う命令。コンソールに1行出力し、最後に改行する。  
- Eclipse 上で、systoutとタイプして、Ctrl+Spaceを押してみる。

Eclipseの便利な機能（インデント）  
Ctrl + Shift + P を押すと、Eclipseがソースコードに正しいインデント（字下げ）を自動でつけてくれる。いつでも気が向いた時に実行して、きれいなソースコードを保とう。

## ソースコード：Hello.javaの拡張

ここから先、コンピュータ内のロボット「ジョーチ」がユーザと対話するイメージを持ってほしい。

Hello World!  
こんにちは！私はジョーチと申します  
:  
標準出力

先ほど作成したHello.javaに追加する形で、下記のソースコードを下記、実行してみよう。

```
/**
 * はじめてのJavaクラス。
 */
// Helloという名前のクラスを自分で宣言している。クラス名は大文字で始める。
public class Hello {
    // Helloクラスの中のmainという名前のメソッド(関数)
    // public, static, args今は気にしない
    public static void main(String[] args) {
        /* ---- 1. 標準出力(画面コンソール)の練習 ---- */
        System.out.println("Hello World!");

        /* ---- 2. 文字列変数の練習 ---- */
        /* Javaでは String型を使う（C言語ではcharの配列だった）。
         * 変数名は小文字で始めること。
         * 2語以上ある時は、つなぎ目を大文字に。call_name等としない。
         */
        String callName = "ジョーチ";
        //文字列同士は、+ で連結できる
        System.out.println("こんにちは！私は" + callName + "と申します。");

        /* ---- 3. 計算の練習 ---- */
        System.out.println("計算をしましょう。");
        System.out.println("■整数四則の場合");
        //(1)内の計算結果を文字列として+で連結する
        System.out.println("123 + 456 = " + (123+456));
        System.out.println("123 - 456 = " + (123-456));
        System.out.println("123 * 456 = " + (123*456));
        System.out.println("123 / 456 = " + (123/456));

        System.out.println("■変数を使ってみよう");
        int x = 123;
        //変数の値を文字列として+で連結する
        System.out.println("x = " + x);
        //以下、定数の場合と同様
        System.out.println("x + 456 = " + (x+456));
        System.out.println("x - 456 = " + (x-456));
        System.out.println("x * 456 = " + (x*456));
        System.out.println("x / 456 = " + (x/456));

        //浮動小数点にキャスト(型変換)
        double z = (double) x / 456;
        System.out.println("z = x / 456 = " + z);

        /* ---- 4. 他OSprint系命令との練習 ---- */
        System.out.println("＊print()は改行しません");
        System.out.print("これは、ソフトウェア工学の");
        System.out.print("授業です。");
        System.out.println("改行文字を入れます。\\n");

        System.out.println("＊printf()はC言語おなじみですね。");
        System.out.printf("x = 456 = %d\\n", (x+456));
        System.out.printf("x - 456 = %d\\n", (x-456));
        System.out.printf("x * 456 = %d\\n", (x*456));
        System.out.printf("x / 456 = %d\\n", (x/456));

        System.out.printf("x / 456 = %4f\\n", ((double)x/456));

        /* ---- 5. 標準エラー出力の練習 ---- */
        System.err.println("＊System.errは標準エラー出力に出します。");
        System.err.println("エラーメッセージのテストです。");
        System.err.println("エラー1：" + x);
        System.err.println("エラー2：テストテスト");
    }
}
```

解説

- System.out.println()の他にも、print()やprintf()が存在する。場合に応じて使い分けること。
- String型の変数は、+で値を連結できる。
- 型変換はキャストで実行する。変換したい変数の前に (型名) 変数名 のように書く。
- System.err.println()は標準エラー出力に文字列を書き出す。エラーメッセージ等は、こちらに出力する。

## 標準入力

キーボードから入力する際のお作法  
以下のようなコードを書く。Javaのお作法が少し面倒くさいが、毎回同じなので覚えてしまおう。

```
//標準入力から読み込んでバッファリングしてくれる。読み取り機
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String line; //読み取った文字列を入れる変数
line = br.readLine(); //1行読み込む。エラーの例外処理が必要
```

読み取られるのはすべて文字列！  
readLine()で読み取られるキーボード入力は、すべて文字列として読み込まれる。たとえば、123.45とキーボードから入力すると、lineには"123.45"という文字列が入る。

文字列の数値に変換したいときは  
int x = Integer.parseInt(line); //line文字列を整数と思って変換。  
double y = Double.parseDouble(line); //lineを倍精度浮動小数点と思って変換。

## ソースコード：KeyboardInput.java

次は、ジョーチの質問にユーザがキーボードで答えるケースを考えよう。

●あなたのお名前？  
花子です  
ジョーチ 標準入力 ユーザ 標準出力

SoftEng01の中に新しくJavaクラスKeyboardInputを作成し、下記のソースコードを貼りつけて、実行を確認してみよう。

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

/**
 * キーボードからの入力を練習するクラス
 */
// KeyboardInputという名前のクラスを宣言。大文字で始める。単語のつなぎ目は大文字に
public class KeyboardInput {

    // KeyboardInputクラスのmainメソッド。
    // throws Exception: 例外 (エラー) が起きたら、外に投げるとの意味。
    public static void main(String[] args) throws Exception {

        /*----- 文字列変数の練習 -----*/
        /*
         * Javaでは String型を使う（C言語ではcharの配列だった）。
         * 変数名は小文字で始める。2語以上の単語で構成するときは、
         * つなぎ目を大文字に。call_name等としない。
         */
        String callName = "ジョーチ";
        System.out.println("こんにちは！私は" + callName + "と申します。");

        /*----- 入力の準備 -----*/
        // キーボード (標準入力) の読み取り機 (リーダー)
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        String line; // 読み取り用の文字列/バッファ

        /*----- 文字列の入力の練習 -----*/
        System.out.print("●あなたのお名前を入力してください：");
        line = br.readLine(); // 1行読み込む
        System.out.println("お名前は " + line + " さんですね。");
        String name = line; // 名前として保持

        /*----- 整数の入力の練習 -----*/
        System.out.print("●年齢を入力してください：");
        line = br.readLine(); // 1行読み込む
        int age = Integer.parseInt(line); // 文字列を整数に変換する
        System.out.println(name + "さんは " + age + " 歳ですね。");

        /*----- 小数の入力の練習 -----*/
        System.out.print("●身長は何cmですか？");
        line = br.readLine(); // 1行読み込む
        double height = Double.parseDouble(line); // 文字列を浮動小数に変換する
        System.out.print("●体重は何kgですか？");
        line = br.readLine(); // 1行読み込む
        double weight = Double.parseDouble(line); // 文字列を浮動小数に変換する

        /*----- 計算と出力の練習 -----*/
        height = height * 0.025; // 身長をmに変換
        double BMI = weight / (height * height); // Body-Mass Index
        System.out.println(name + "さん (" + age + " 歳) のBMI値は。");
        System.out.printf("%.2fです。\\n", BMI);
    }
}
```

解説

- ユーザから、名前、年齢、身長、体重を聞き、BMIを計算して出力する。
- 年齢は、文字列から整数へ変換、身長、体重は、文字列から浮動小数点型に変換している。

## 例外処理

例外を起こしてみる  
上記のKeyboardInput.javaの実行にて、年齢や身長・体重に数値ではなく文字列を入れてみよう。例外(exception)メッセージが出て、プログラムが途中で落ちて止まってしまう。

例外とは  
ある処理を実行中にエラーが起こり、当該エラーについてどう対処すべきかをJavaシステムがプログラマに聞く仕組み。java.lang.NumberFormatExceptionは、入力された数値のフォーマットがおかしいというエラーメッセージである。

なぜ途中で落ちるのか？  
main()関数の横のthrows Exceptionが原因。parseInt()またはparseDouble()からもった例外をそのままmainの呼び出し元スルーに投げているため。

例外処理の重要性  
実際のソフトウェア製品では、ユーザは意図せずに様々な誤入力をしてくる場合がある。そのたびにプログラムがクラッシュしては、信頼性の高いソフトウェアとは言えない。

例外処理の書き方  
例外を吐く可能性のあるコードブロックを、try{} で囲み、その後にcatch(何と+Exception e) {}をつなげる。Javaシステムは、tryブロックをまず実行していき、途中で例外が発生したら、その例外に对应的なcatchブロックの中身が実行される。

Eclipseの便利な機能（例外処理）  
Eclipseでは、当該コードをマウスで選択し、右クリック→囲む→try/catchブロックで自動的に例外処理のひな形が挿入される。

## ソースコード：KeyboardInput.java (例外処理追加版)

ジョーチが異常終了しないように、上のKeyboardInput.javaに例外処理を加えてみよう。

1. main()の横にある throws Exceptionを消す。
2. 例外が処理されない(関数がEclipse上で赤く表示される。readLine()等が赤くなる。
3. 名前、年齢、身長・体重(入力)のそれぞれの部分をマウスで選択し、右クリック→囲む→try/catchブロックを選択する。
4. catchブロックに適当なエラー処理を書いてみよう。
5. 以下のソースコードに合うようにコードを更新（リファクタリングという）してみよう。なお、例外処理はサンプル、もっと凝ったこともある。

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * キーボードからの入力を練習するクラス。 例外処理(try-catch)をするバージョン。
 */
// KeyboardInputという名前のクラスを宣言。大文字で始める。単語のつなぎ目は大文字に
public class KeyboardInput {

    // KeyboardInputクラスのmainメソッド。例外は外に投げる。
    public static void main(String[] args) {
        /*----- 文字列変数の練習 -----*/
        /*
         * Javaでは String型を使う（C言語ではcharの配列だった）。
         * 変数名は小文字で始める。2語以上の単語で構成するときは、
         * つなぎ目を大文字に。call_name等としない。
         */
        String callName = "ジョーチ";
        System.out.println("こんにちは！私は" + callName + "と申します。");

        /*----- 入力の準備 -----*/
        // キーボード (標準入力) の読み取り機 (リーダー)
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        String line; // 読み取り用の文字列/バッファ

        /*----- 文字列の入力の練習 -----*/
        try {
            System.out.print("●あなたのお名前を入力してください：");
            line = br.readLine(); // 1行読み込む。エラーの例外処理が必要
            System.out.println("お名前は " + line + " さんですね。");
            name = line;
        } catch (IOException e) { // readLine()からの入力例外を拾う。
            System.err.println("入力例外：名前を「名無し」で進めます。");
            name = "名無し";
        }

        /*----- 整数の入力の練習 -----*/
        int age = 0;
        try {
            System.out.print("●年齢を入力してください：");
            line = br.readLine(); // 1行読み込む
            age = Integer.parseInt(line); // 文字列を整数に変換する
            System.out.println(name + "さんは " + age + " 歳ですね。");
        } catch (NumberFormatException e) { // 1行読み込む
            System.err.println("フォーマット例外：年齢を0歳で進めます。");
            age = 0;
        } catch (IOException e) { // readLine()からの例外を拾う。
            System.err.println("入力例外：年齢を0歳で進めます。");
            age = 0;
        }

        /*----- 小数の入力の練習 -----*/
        double height = 0;
        double weight = 0;
        try {
            System.out.print("●身長は何cmですか？");
            line = br.readLine(); // 1行読み込む
            height = Double.parseDouble(line); // 文字列を浮動小数に変換する
            System.out.print("●体重は何kgですか？");
            line = br.readLine(); // 1行読み込む
            weight = Double.parseDouble(line); // 文字列を浮動小数に変換する
        } catch (NumberFormatException e) { // 2つのparseDouble()からの例外をまとめて拾う。
            System.err.println("フォーマット例外：BMIが計算できません。終了します。");
            System.exit(1); // 強制終了
        } catch (IOException e) { // 2つのreadLine()からの例外をまとめて拾う。
            System.exit(1); // 強制終了
        }

        /*----- 計算と出力の練習 -----*/
        height = height * 0.025; // 身長をmに変換
        double BMI = weight / (height * height); // Body-Mass Index
        System.out.println(name + "さん (" + age + " 歳) のBMI値は。");
        System.out.printf("%.2fです。\\n", BMI);
    }
}
```

解説

- 名前入力時の例外処理：readLine()の例外をcatchし、例外発生時には、名前を「名無しさん」として進める。
- 年齢入力時の例外処理：readLine()とparseInt()の例外をcatchし、例外発生時には、年齢を0歳として進める。
- 身長・体重の例外処理：readLine()とparseDouble()の例外をcatchし、例外発生時には、システムを正常に終了させる。

## 2種類の例外の取り扱い (throwsとtry/catch)の違いをイメージしてみる

飲食店アルバイト店員であるあなたが、客に料理を出そうとしたところ、誤ってこぼしてしまった (例外発生) としてよう。

throws：  
その場は放置して、とにかく指示元の店長にどうすべきかを請ねる。

try/catch：  
こぼしてしまった時の処理を予め考えておき、万が一こぼした場合には、その手順に従って自分で対処する。



例外は、できるだけtry/catchで拾う方が、より信頼性の高いソフトウェアになる。どうしても自分で決められない (決めない方が良い) 場合には、throwsを書く。

## 演習問題

- [第1回演習問題](#)