初心者時のプログラミングの学習

× 教科書を読むだけ

× 頭で[redacted]だけ

× 教え[redacted]…

× 友人のソースをコピー

→ 時間の無駄

○ **書く！**

○ **動かす！**

○ **試してみる！**

○ **エラーメッセージを読む**

○ **動かしながら覚える**

そもそも、コンピュータは思い通りにならないもの

※ ハマるたびに経験値が上がる

望ましいプログラミングの流儀は今後の授業で学びます

(学部生の)プログラミングでできること

- PC上の計算(この授業、基本中の基本)

- GUIプログラミング
- Webプログラミング
- スマートフォンアプリ開発
- その他いろいろ

やってみると、想像してるよりもたぶん簡単
一つマスターすると他も試してみたくなる

プログラミング

- 第1回
  - Java言語の見晴らし台（第0章）
  - WindowsにおけるJavaプログラムの作成
  - Javaでこんにちは（第1章）
  - 計算をやってみよう（第2章）
  - 変数と型（第3章）

- 教科書：Java言語プログラミングレッスン（上）

プログラミング

- 第1回
  - **Java言語の見晴らし台（第0章）**
  - WindowsにおけるJavaプログラムの作成
  - Javaでこんにちは（第1章）
  - 計算をやってみよう（第2章）
  - 変数と型（第3章）

Java言語の見晴らし台（第0章）

- Java言語とは何か
  - プログラミング言語の一つ（C言語も）
  - 機種依存性が少ない言語（WindowsでもLinuxでもMacでも）
  - 誤りをおかしにくい言語（厳しいコンパイルチェック）
  - オブジェクト指向言語
  - マルチスレッドを扱える言語（並行プログラミング）

Java言語の見晴らし台（第0章）

- Java開発環境（JDK）
  - Sunが提供しているJava言語の開発環境
  - Java言語で使用可能な標準的なクラスライブラリを含む

Java言語の見晴らし台（第0章）

- 統合開発環境（IDE）
  - ソースコードの編集、コンパイル、実行、デバグを一つの環境の中で実行
  - Eclipse

  - この授業では、基本的には扱わない
    - (資料) Windowsのエディタを使ってプログラム編集
    - JDKを使ってコンパイル＆実行

    - Eclipseなどを使ってもOK

Java言語の見晴らし台（第0章）

- Javaプログラムの例

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

```
> javac Hello.java        // コンパイル

> java Hello              // 実行
Hello!
```

Java言語の見晴らし台（第0章）

- （参考）C言語の場合

```
#include <stdio.h>


void main() {
    printf(“Hello!¥n”);
}
```

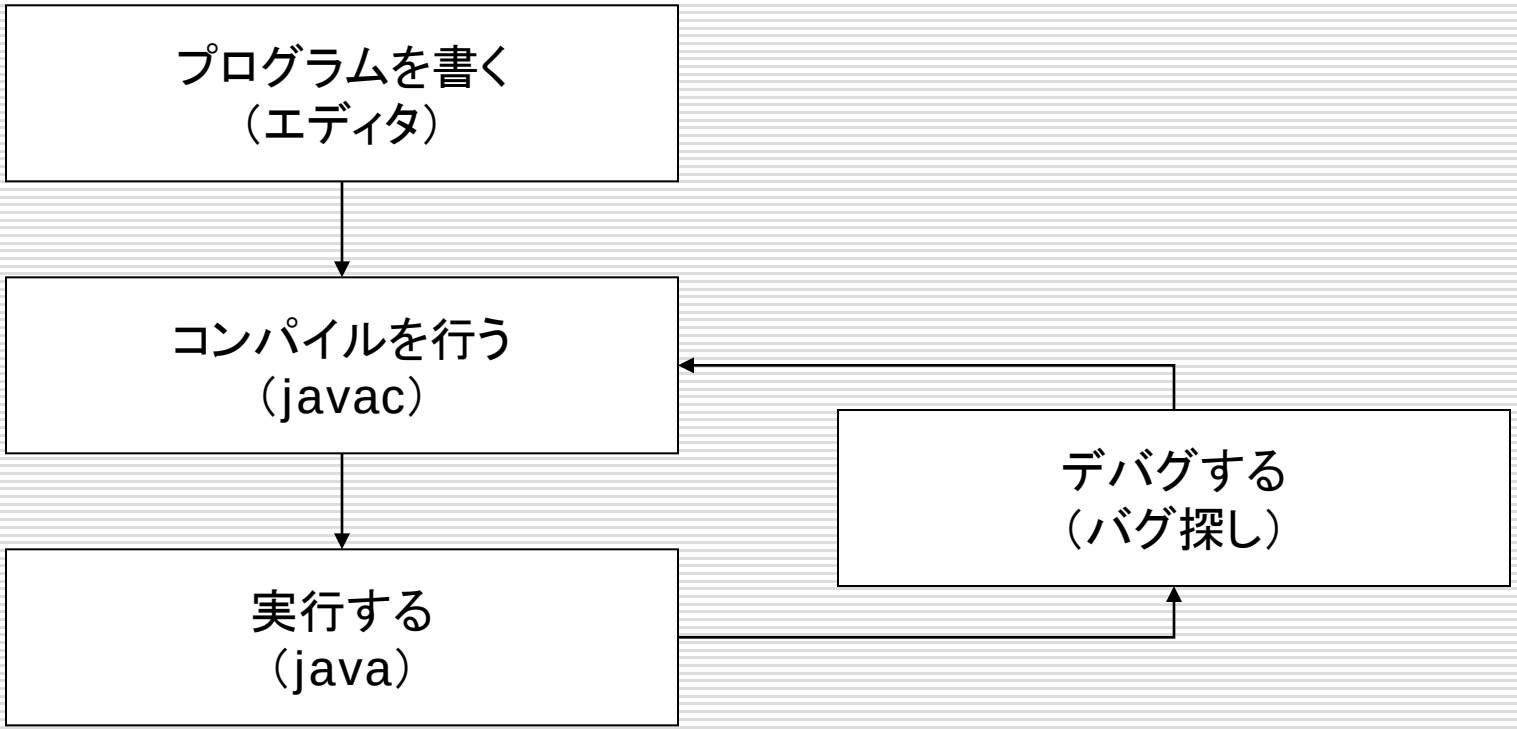
```
> cc Hello.c                    // コンパイル

> a.out                         // 実行
Hello!
```

Java言語の見晴らし台（第0章）

- プログラム開発の流れ
  - プログラムを書く
  - コンパイルを行う
  - 実行する
  - （うまく動かなかったら）デバグする

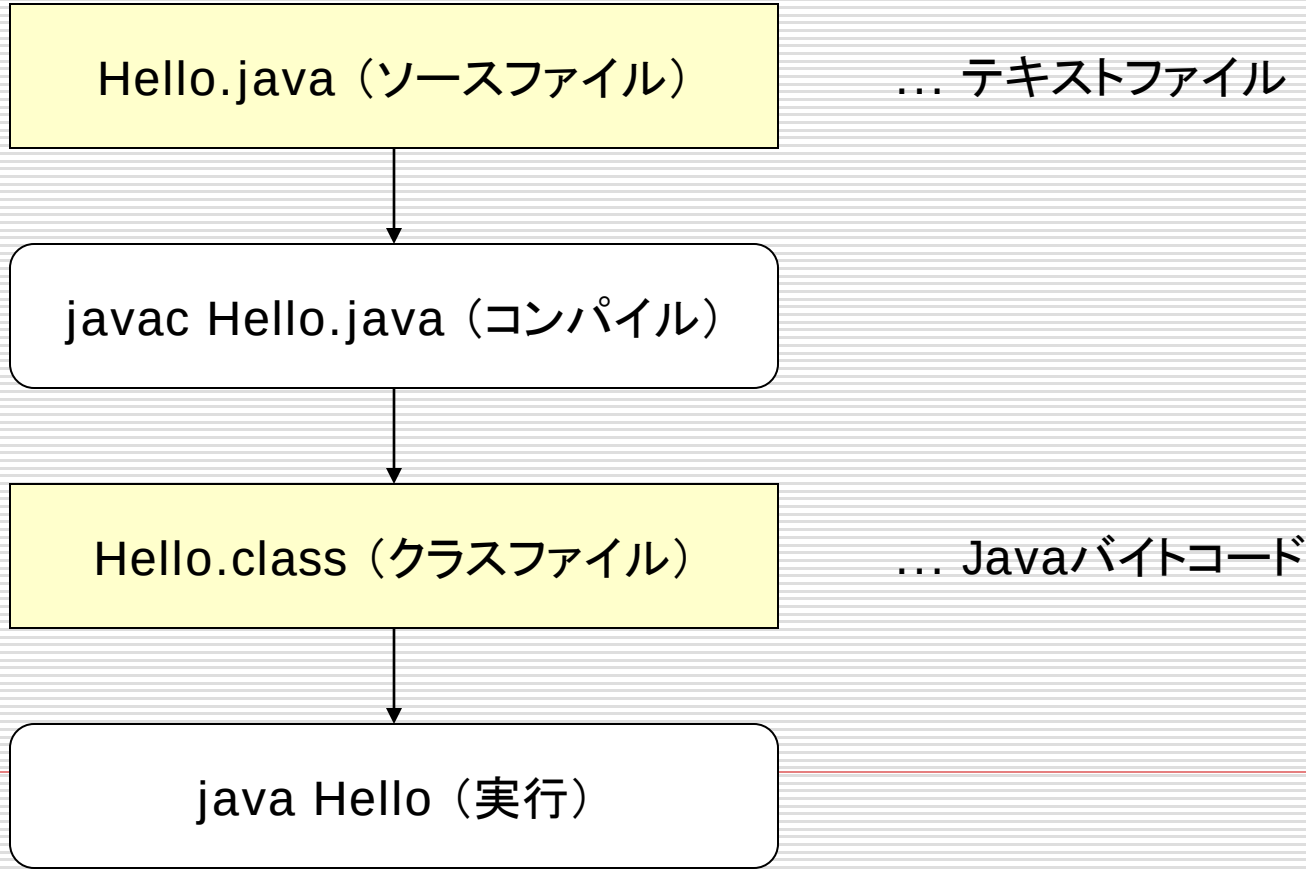
Java言語の見晴らし台（第0章）

- プログラム開発の流れ



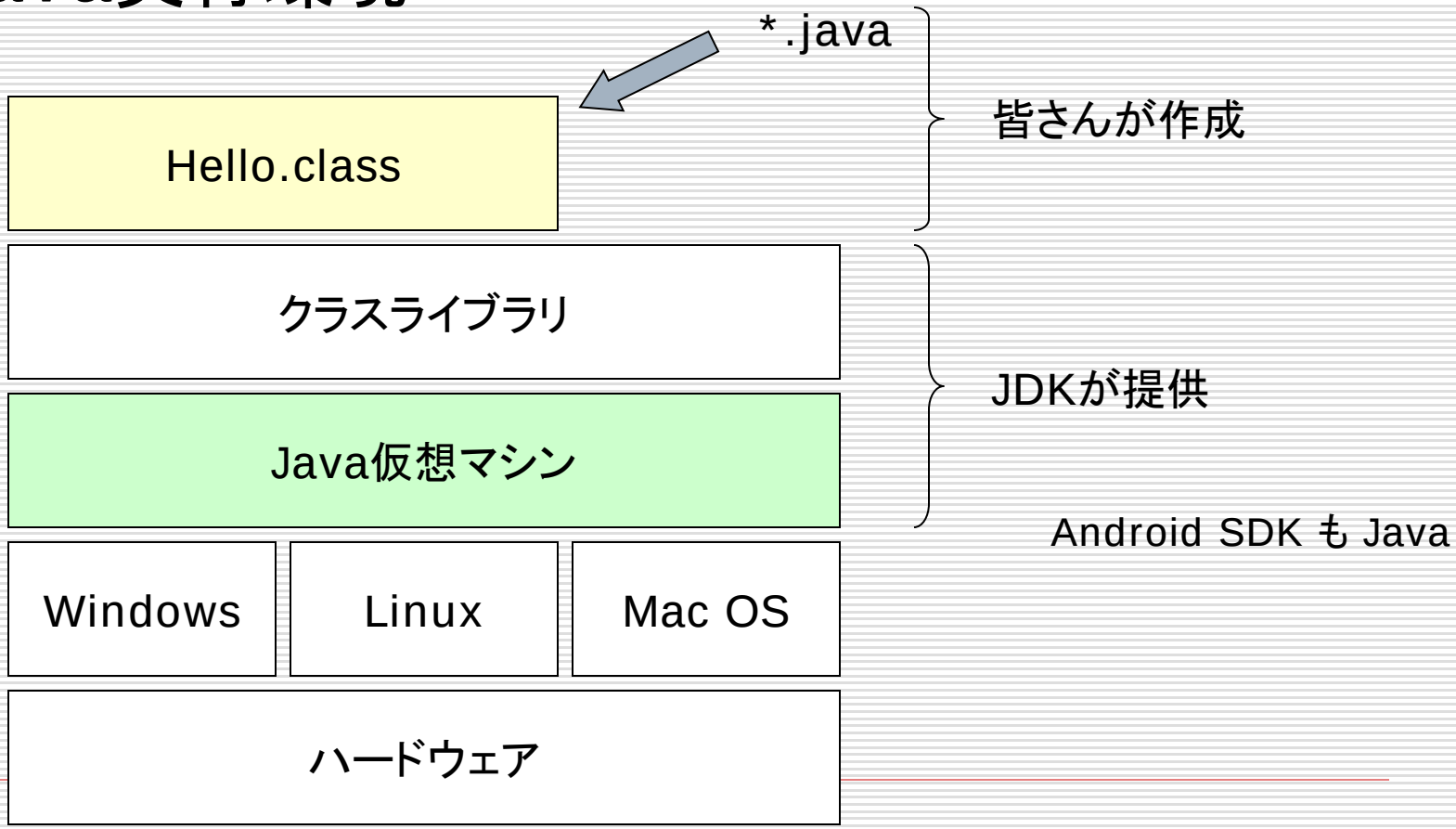
Java言語の見晴らし台（第0章）

プログラム開発の流れ



Java言語の見晴らし台（第0章）

Java実行環境



Java言語の見晴らし台（第0章）

- Java実行環境
  - ハードウェア：コンピュータ本体（CPU、メモリ他）
  - オペレーティングシステム（Windows/Linux/Mac）
  - Java仮想マシン（Java virtual machine）
    - *.classを実行する仮想機械
    - バイトコードのインタプリタ
    - 機種独立性の提供（どんなOSでも動く）
    - ※ Java仮想マシンは機種依存（Windows用など）
  - クラスライブラリ：利用可能な便利なクラスの集合

Java言語の見晴らし台（第0章）

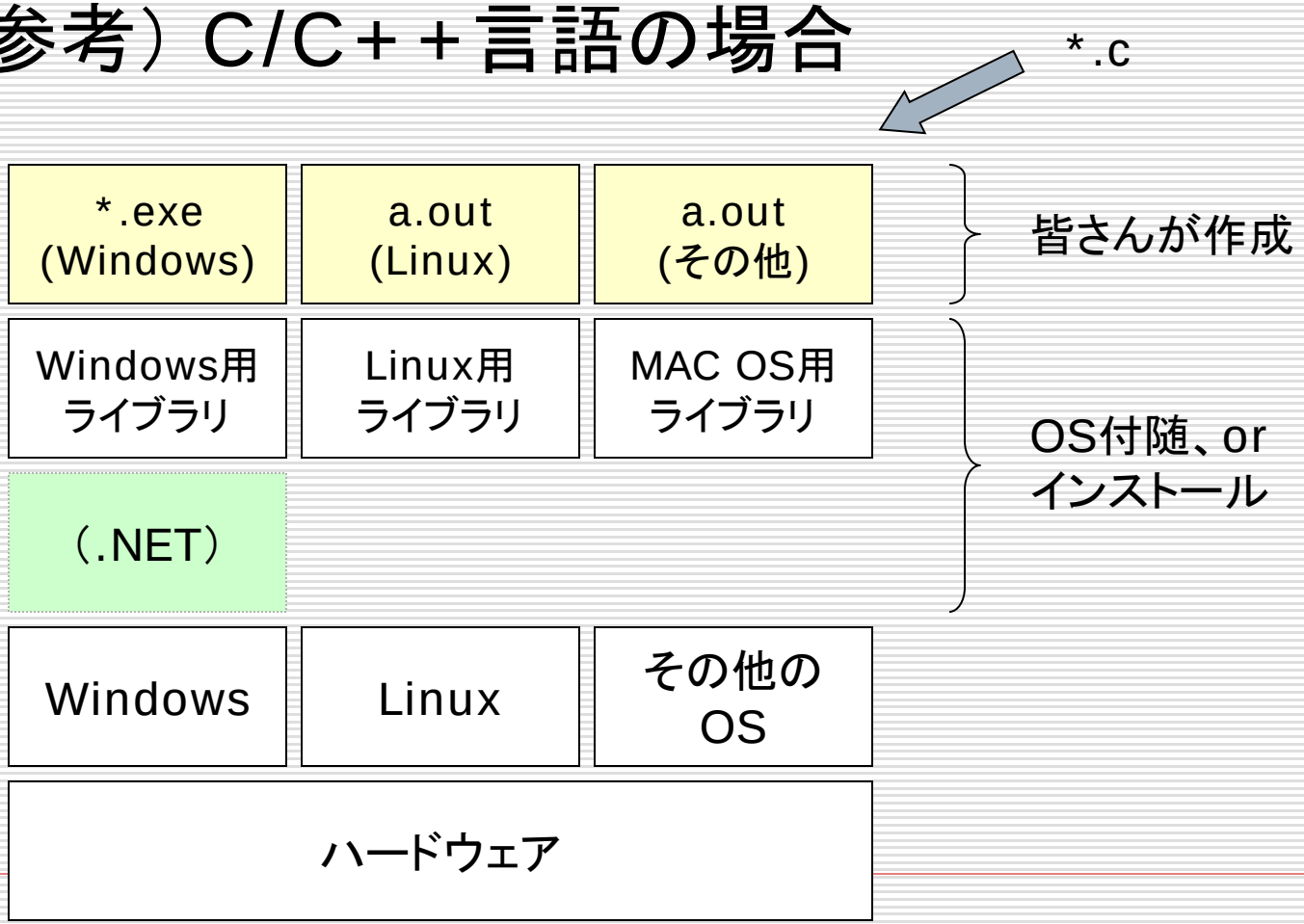
- クラスファイル
  - クラスファイル（バイトコード）は Java 仮想マシンが実行する
  - クラスファイル（バイトコード）はハードウェアやOSに依存しない

  - 機械語：ハードウェアが理解する命令、0/1並び
    - Javaバイトコード ⇒ Java仮想マシン ⇒ OS&HW
    - C/C＋＋実行コード ⇒ OS&HW

最近はC/C＋＋系も仮想マシン上で動作させる流れ（.NETフレームワーク）

Java言語の見晴らし台（第0章）

- （参考）C/C++言語の場合



Java言語の見晴らし台（第0章）

- Java API 仕様

http://docs.oracle.com/javase/jp/6/api/
http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html

Java言語の見晴らし台（第0章）

- Java言語の学び方
  - 準備：Javaに関する情報収集
  - 開発ツール入手：JDK
  - 既存プログラムのコンパイル＆実行
  - 簡単なカスタマイズ ⇒ コンパイル＆実行
  - ソースファイルを読む
  - ゼロから書いてみる、サブセットを作ってみる
  - 考える、読む、聞く、探す
  - …

プログラミング

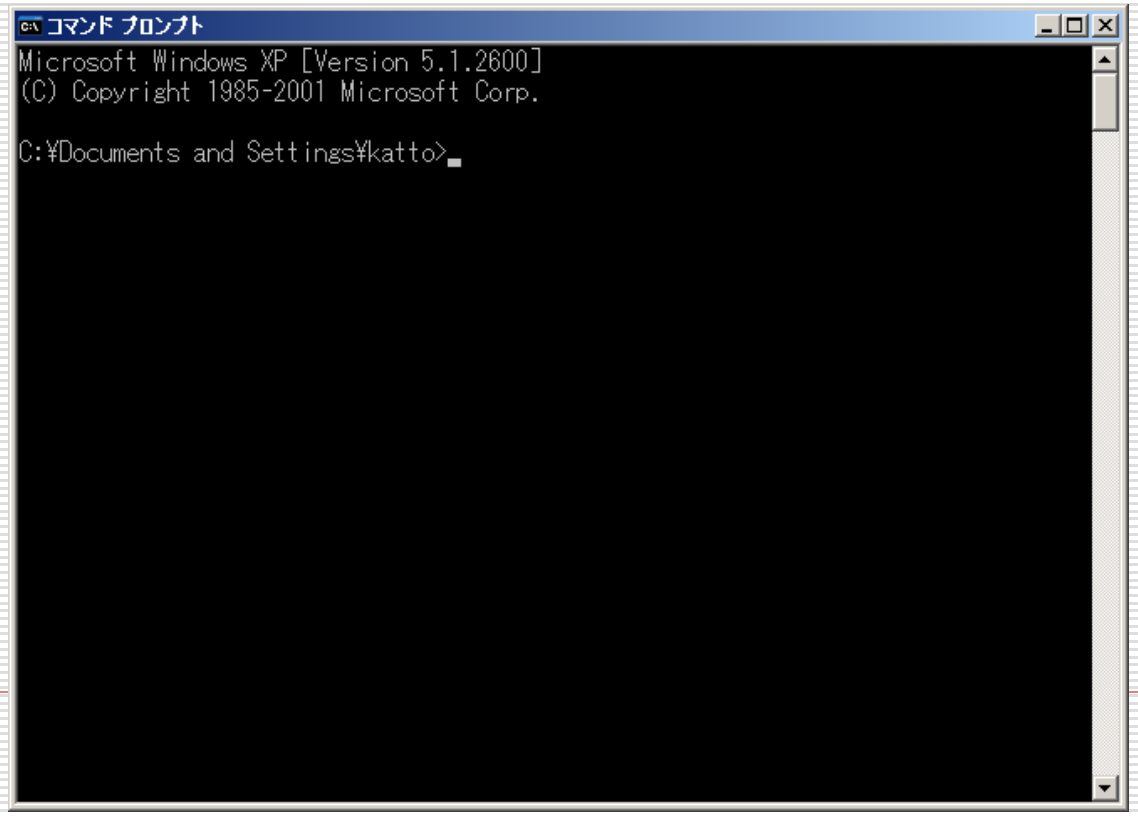
- 第1回
  - Java言語の見晴らし台（第0章）
  - **WindowsにおけるJavaプログラムの作成**
  - Javaでこんにちは（第1章）
  - 計算をやってみよう（第2章）
  - 変数と型（第3章）

WindowsにおけるJavaプログラムの作成

- JDKのインストール
  - （事前案内・別資料）
  - ネット経由のダウンロード＆インストール

WindowsにおけるJavaプログラムの作成

- コマンドプロンプト
  - スタート > プログラム > アクセサリ



WindowsにおけるJavaプログラムの作成

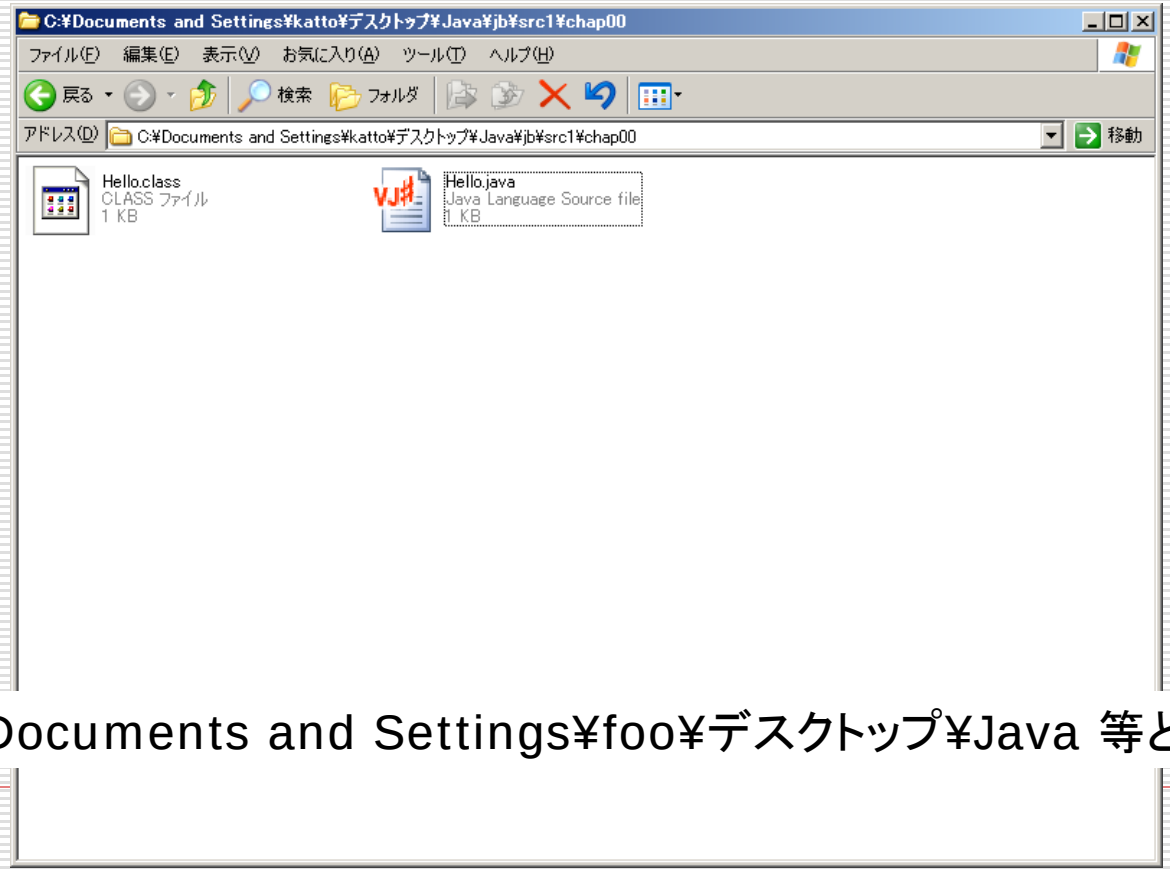
- DOSコマンド
  - cd　　ディレクトリの移動
  - dir　　ディレクトリ表示

> cd C:¥Documents and Settings¥foo¥デスクトップ¥Java

> dir

WindowsにおけるJavaプログラムの作成

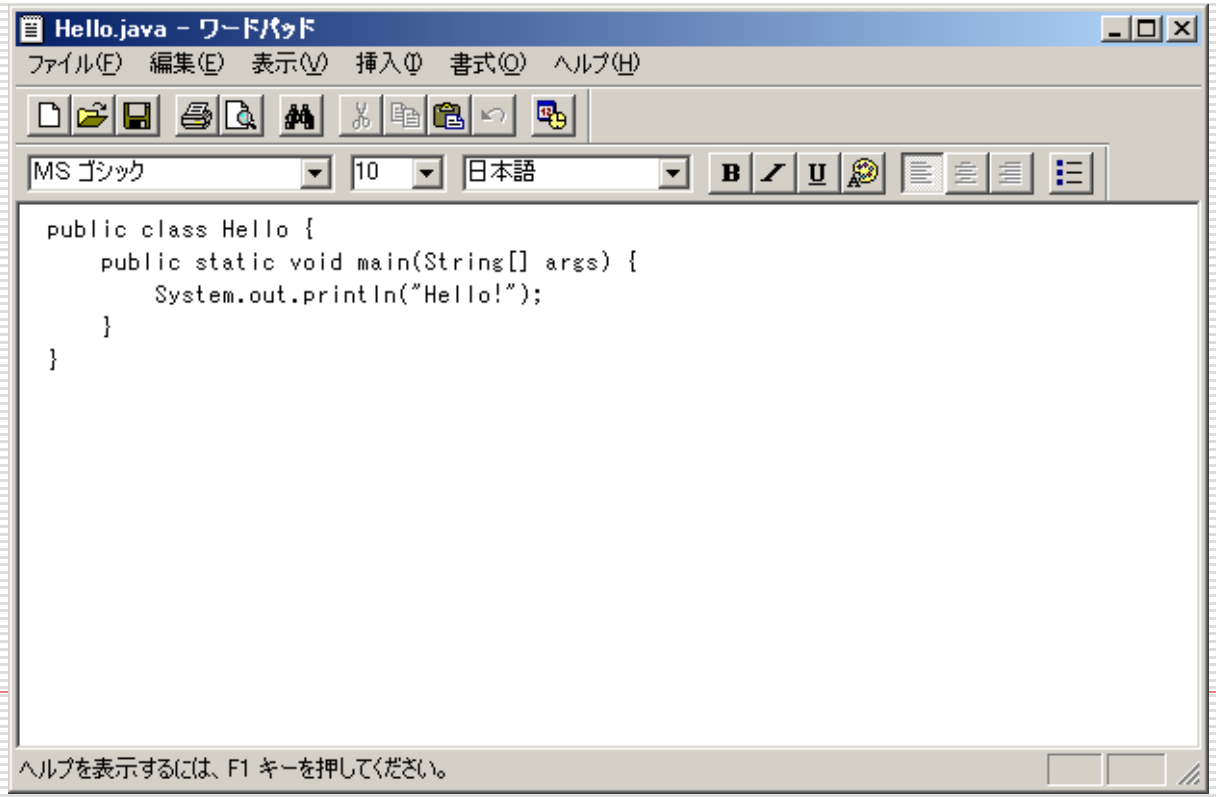
- Explorer



cd C:¥Documents and Settings¥foo¥デスクトップ¥Java 等と同じ場所

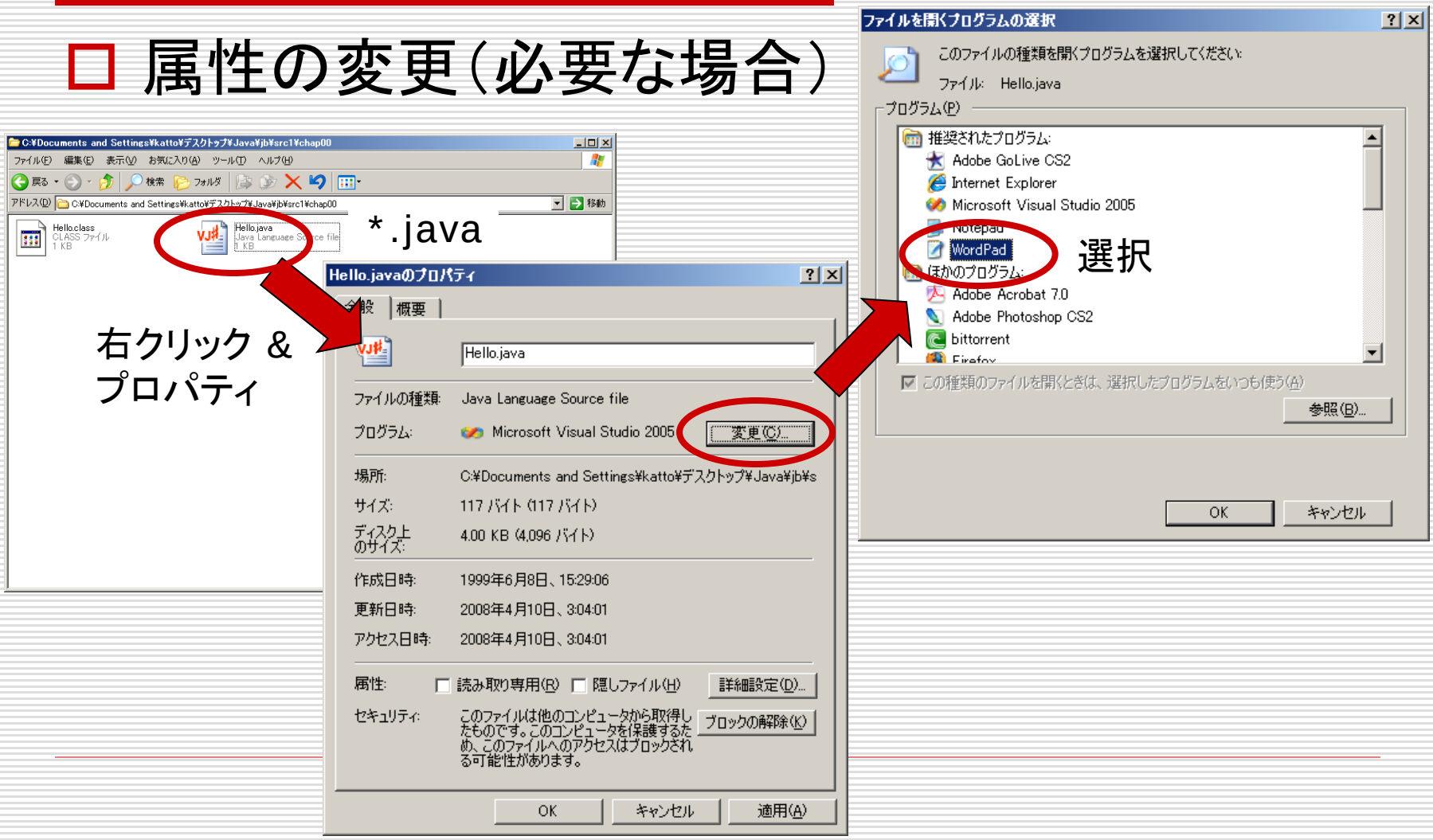
WindowsにおけるJavaプログラムの作成

- エディタ
  - ワードパッド、メモ帳、など



WindowsにおけるJavaプログラムの作成

- 属性の変更（必要な場合）



WindowsにおけるJavaプログラムの作成

- コンパイル＆実行
  - javac　　コンパイル
  - java　　実行

WindowsにおけるJavaプログラムの作成

- まとめ
  - コマンドプロンプト: コンパイル＆実行
    - CUI
  - Explorer: エディタ起動、ファイル管理
    - GUI
  - エディタ: ワードパッド、メモ帳、など

  - 居場所（ディレクトリ）に注意

プログラミング

- 第1回
  - Java言語の見晴らし台（第0章）
  - WindowsにおけるJavaプログラムの作成
  - **Javaでこんにちは（第1章）**
  - 計算をやってみよう（第2章）
  - 変数と型（第3章）

Javaでこんにちは（第1章）

- Hello!を表示するJavaプログラム（再）

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

```
> javac Hello.java        // コンパイル

> java Hello              // 実行
Hello!
```

Javaでこんにちは（第1章）

Hello「クラス」の宣言

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

- Javaでは、クラスの名前は大文字で始めるのが習慣
- Javaでは、大文字と小文字の区別は重要
- クラス名はファイル名のはじめに一致する (Hello.java)
- { … } はクラス宣言の範囲を表す

Javaでこんにちは（第1章）

main「メソッド」の宣言

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

- Javaプログラムは main メソッドから始まる
- {…} はメソッド宣言の範囲を表す

Javaでこんにちは(第1章)

- System.out.println

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

- System.out.println は、文字列をディスプレイに表示する
- “…” は文字列を表す
- ; は処理の区切りを表す

Javaでこんにちは（第1章）

コンパイルと実行

```
> javac Hello.java        // コンパイル

> java Hello              // 実行
Hello!
```

- Javaでは、ファイル名は、クラス名.java とする
- コンパイル時は、javac クラス名.java とする

（クラス名.class というクラスファイルが生成される）

- 実行時は、java クラス名 とする（.class はつけない）

Javaでこんにちは(第1章)

- System.out.println

```
public class Aisatsu {
    public static void main(String[] args) {
        System.out.println("おはよう。");
        System.out.println("こんにちは。");
        System.out.println("こんばんは。");
    }
}
```

```
> javac Aisatsu.java      // コンパイル

> java Aisatsu            // 実行
```

Javaでこんにちは(第1章)

- System.out.println

```
> javac Aisatsu.java      // コンパイル

> java Aisatsu            // 実行
おはよう。
こんにちは。
こんばんは。
```

- System.out.println は、文字列を表示して改行する
- System.out.print は、文字列を表示するが改行しない

Javaでこんにちは(第1章)

エスケープシーケンス

¥b	バックスペース(BS)
¥t	タブ
¥n	改行
¥f	改ページ
¥r	復帰
¥”	二重引用符
¥’	一重引用符
¥¥	バックスラッシュ

～ C言語と同じ

Javaでこんにちは（第1章）

- エラーメッセージ
  - エラーメッセージはちゃんと読む
  - コンパイルエラーは誤った場所の手がかりになる
  - 実行時エラーは足りないクラスライブラリなどの手がかりになる

  - エラーメッセージを読んでいない質問は上達にならない

プログラミング

- 第１回
  - Java言語の見晴らし台（第0章）
  - WindowsにおけるJavaプログラムの作成
  - Javaでこんにちは（第１章）
  - **計算をやってみよう（第2章）**
  - 変数と型（第３章）

計算をやってみよう(第2章)

加減乗除

```
public class Calc1 {
    public static void main(String[] args) {
        System.out.println("加算の結果は" + (3 + 2) + "です。");
        System.out.println("減算の結果は" + (5 - 8) + "です。");
        System.out.println("乗算の結果は" + (3 * 4) + "です。");
        System.out.println("除算の結果は" + (7 / 3) + "です。");
    }
}
```

計算をやってみよう(第2章)

- 算術演算

(3 + 2)	5	加算
(5 - 8)	-3	減算
(3 * 4)	12	乗算
(7 / 3)	2	(整数の)除算

計算をやってみよう(第2章)

□ 文字列の連結(文字列の「加算」)

"加算の結果は" + (3 + 2) + "です。"

"減算の結果は" + (5 - 8) + "です。"

"乗算の結果は" + (3 * 4) + "です。"

"除算の結果は" + (7 / 3) + "です。“

“文字列3” = “文字列1” + (演算結果) + “文字列2”

加算の結果は5です

整数から文字列に型変換

計算をやってみよう（第２章）

- 計算の優先順位
  - ＋、－よりも ＊、/ が優先される
  - 同じ優先順位の場合は左から順番に計算される
  - （　）を使って順序のあいまいさをなくすのが安全

計算をやってみよう(第2章)

- Javaの「型」

	型	取りうる値
boolean型	boolean	1ビット　true または false
整数型	byte	符号あり8ビット整数　-128～127
	short	符号あり16ビット整数　-32768～32767
	int	符号あり32ビット整数　-2147483648～2147483647
	long	符号あり64ビット整数　-9223372036854775808～9223372036854775807
	char	符号なし16ビット整数　0～65535
		1文字
浮動小数点型	float	32ビット実数　単精度浮動小数点数
	double	64ビット実数　倍精度浮動小数点数

符号あり…signed，符号なし…unsigned

計算をやってみよう(第2章)

- 剰余演算 %

(3 + 2)	5	加算
(5 - 8)	-3	減算
(3 * 4)	12	乗算
(7 / 3)	2	(整数の)除算
(10 % 7)	3	(整数の)剰余

計算をやってみよう(第2章)

その他の演算子(付録D)

ビット演算子	& | ^ ~
論理演算子	! && ||
関係演算子	== != < > <= >=
シフト演算子	>> << >>>

～ ほぼC言語と同じ

計算をやってみよう(第2章)

- printf メソッド（付録J）

```
public class Calc1a {
    public static void main(String[] args) {
        System.out.printf("加算の結果は%dです。%n", 3 + 2);
        System.out.printf("減算の結果は%dです。%n", 5 - 8);
        System.out.printf("乗算の結果は%dです。%n", 3 * 4);
        System.out.printf("除算の結果は%dです。%n", 7 / 3);
    }
}
```

¥n でも OK

プログラミング

- 第1回
  - Java言語の見晴らし台（第0章）
  - WindowsにおけるJavaプログラムの作成
  - Javaでこんにちは（第1章）
  - 計算をやってみよう（第2章）
  - 変数と型（第3章）

変数と型（第3章）

- 変数とは何か
  - 変数＝箱のようなもの

  - 変数を作る（変数宣言）
  - 値を入れる（代入）
  - 値を見る（参照）

変数と型(第3章)

- 変数宣言(変数を作る)

```
int  x ;
char  c ;
```

変数と型（第3章）

- 代入（変数に値を入れる）

```
int  x , y ;
char  c ;


x = 4 ;
y = 1 + 2 * 3 ;
```

変数と型（第3章）

□ 参照（変数の値を見る）

```
int  x , y ;
char  c ;


x = 4 ;
y = 1 + 2 * 3 ;


System.out.println(x) ;
System.out.println(2 * y + 3) ;
```

変数と型（第3章）

- 変数の初期化（宣言と代入を同時に行う）

```
int  x = 4 , y = 1 + 2 * 3;
char  c ;


x = 4 ;
y = 1 + 2 * 3 ;


System.out.println(x) ;
System.out.println(2 * y + 3) ;
```

変数と型（第3章）

- 代入演算子（右から左へ）

```
int  x , y ;


x = 4 ;
y = x ;                    // y ← x


y = 1 + 2 * 3 ;
x = y ;                    // x ← y
```

変数と型（第3章）

例1：計算結果の表示

```
public class Var1 {
    public static void main(String[] args) {
        int x;
        x = 4;
        System.out.println(2 * x + 3);
    }
}
```

変数と型（第3章）

例2：二つの変数

```
public class Var2 {
    public static void main(String[] args) {
        int x = 15;
        int y = 32;
        System.out.println((x + y) / 2);
    }
}
```

変数と型(第3章)

- 例3: 浮動小数点 (double)

```
public class Var3 {
    public static void main(String[] args) {
        double x = 15.0;
        double y = 32.0;
        System.out.println((x + y) / 2);
    }
}
```

  - 変数を見たら、その変数の型を確認する

変数と型（第3章）

□ 例4：文字型変数の初期化

```
public class Var4 {
    public static void main(String[] args) {
        char a = 'A';
        char alpha = 'α';
        char hiroshi = '浩';
        System.out.println("変数 a の値は " + a);
        System.out.println("変数 alpha の値は " + alpha);
        System.out.println("変数 hiroshi の値は " + hiroshi);
    }
}
```

' ' … 文字
" " … 文字列

変数と型（第3章）

- Java言語の「型」（再）

	型	取りうる値
boolean型	boolean	1ビット　true または false
整数型	byte	符号あり8ビット整数　-128～127
	short	符号あり16ビット整数　-32768～32767
	int	符号あり32ビット整数　-2147483648～2147483647
	long	符号あり64ビット整数　-9223372036854775808～9223372036854775807
	char	符号なし16ビット整数　0～65535
		1文字 ← Unicode
浮動小数点型	float	32ビット実数　単精度浮動小数点数
	double	64ビット実数　倍精度浮動小数点数

符号あり…signed，符号なし…unsigned

変数と型(第3章)

- Java言語の「型」
  - 直接型 (primitive type)
    - 整数: int
    - 文字: char
    - 浮動小数点: double
    - …
  - 参照型 (reference type)
    - クラス: class
    - 配列: int[]
    - インタフェース: interface
    - 文字列: String
    - …

変数と型（第3章）

キーボードからの入力

```
import java.io.*;

public class HowOldAreYou {
    public static void main(String[] args) {
        System.out.println("あなたの名前を入力してください。");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        try {
            String line = reader.readLine();
            System.out.println(line + "さん、こんにちは。");
            System.out.println("年齢を入力してください。");
            line = reader.readLine();
            int age = Integer.parseInt(line);
            System.out.println("いま " + age + " 歳とすると、10年後は " + (age + 10) + " 歳ですね。");
        } catch (IOException e) {
            System.out.println(e);
        } catch (NumberFormatException e) {
            System.out.println("年齢が正しくありません。");
        }
    }
}
```

変数と型（第3章）

BufferedReader クラス

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String line = reader.readLine();
System.out.println(line + "さん、こんにちは。");
System.out.println("年齢を入力してください。");
line = reader.readLine();
int age = Integer.parseInt(line);
```

- 参照型（クラス）
- データ読み込みを行うためのクラス（第18章）

java.io.BufferedReader

変数と型（第3章）

readLineメソッド

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String line = reader.readLine();
System.out.println(line + "さん、こんにちは。");
System.out.println("年齢を入力してください。");
line = reader.readLine();
int age = Integer.parseInt(line);
```

- 入力待ち＆キーボードからの文字列入力の取得

変数と型(第3章)

- new

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String line = reader.readLine();
System.out.println(line + "さん、こんにちは。");
System.out.println("年齢を入力してください。");
line = reader.readLine();
int age = Integer.parseInt(line);
```

  - オブジェクトの作成
  - (参考) C言語のmalloc

変数と型（第3章）

System.in

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String line = reader.readLine();
System.out.println(line + "さん、こんにちは。");
System.out.println("年齢を入力してください。");
line = reader.readLine();
int age = Integer.parseInt(line);
```

- System.in：キーボードからのデータ入力
- System.out：ディスプレイへのデータ出力
- （参考1）stdin：C言語の標準入力（キーボード）
- （参考2）stdout：C言語の標準出力（ディスプレイ）

変数と型（第3章）

String クラス

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String line = reader.readLine();
System.out.println(line + "さん、こんにちは。");
System.out.println("年齢を入力してください。");
line = reader.readLine();
int age = Integer.parseInt(line);
```

- 「文字列」を表すクラス（参照型）
- String line = “早稲田太郎”;

変数と型（第3章）

- Integer クラス、parseInt メソッド

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String line = reader.readLine();
System.out.println(line + "さん、こんにちは。");
System.out.println("年齢を入力してください。");
line = reader.readLine();
int age = Integer.parseInt(line);
```

  - Integer ： 整数を表すクラス
  - parseInt ： 文字列を整数に変換するメソッド

変数と型（第3章）

- ラッパークラス
  - 基本型を使いやすくしたクラス

基本型	ラッパークラス
boolean	Boolean
byte	Byte
short	Short
int	Integer
long	Long
char	Character
float	Float
double	Double
void	Void

← parseInt メソッド

java.lang.Number

変数と型（第3章）

- 例外処理（第13章）
  - try ... catch

```
import java.io.*;

public class HowOldAreYou {
    public static void main(String[] args) {
        System.out.println("あなたの名前を入力してください。");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        try {
            String line = reader.readLine();
            System.out.println(line + "さん、こんにちは。");
            System.out.println("年齢を入力してください。");
            line = reader.readLine();
            int age = Integer.parseInt(line);
            System.out.println("いま " + age + " 歳とすると、10年後は " + (age + 10) + " 歳ですね。");
        } catch (IOException e) {
            System.out.println(e);
        } catch (NumberFormatException e) {
            System.out.println("年齢が正しくありません。");
        }
    }
}
```

変数と型（第3章）

- import（第17章）
  - クラスライブラリの利用

```
import java.io.*;

public class HowOldAreYou {
    public static void main(String[] args) {
        System.out.println("あなたの名前を入力してください。");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        try {
            String line = reader.readLine();
            System.out.println(line + "さん、こんにちは。");
            System.out.println("年齢を入力してください。");
            line = reader.readLine();
            int age = Integer.parseInt(line);
            System.out.println("いま " + age + " 歳とすると、10年後は " + (age + 10) + " 歳ですね。");
        } catch (IOException e) {
            System.out.println(e);
        } catch (NumberFormatException e) {
            System.out.println("年齢が正しくありません。");
        }
    }
}
```

Kuku.java

```
/**
 * Kuku は、九九の問題をランダムに10個表示して
 * その正解数と正解率を表示するプログラムである。
 *
 * @author 結城浩
 * @copyright 1999 by Hiroshi Yuki.
 */

import java.io.*;

public class Kuku {
    /** 表示する問題の個数 */
    public static final int MAX_QUESTION = 10;
    /**
     * 九九の問題をMAX_QUESTION回繰り返して出題する。
     * 最後に正答率を表示する。
     */
    public static void main(String[] args) {
        int goodAnswers = 0;    // 正答の個数
        System.out.println("これから九九の問題を" + MAX_QUESTION + "問出します。");
        /*
         * 以下、問題を繰り返し表示し、ユーザからの解答を判断する。
         * その後、正答の数を数える。
         */
        for (int i = 0; i < MAX_QUESTION; i++) {
            boolean ok = showQuestion(i + 1);
            if (ok) {
                goodAnswers++;
            }
        }
        double rate = goodAnswers * 100.0 / MAX_QUESTION;
        System.out.println("");
        System.out.println("問題は" + MAX_QUESTION + "問ありました。");
        System.out.println("正しく答えられたのは" + goodAnswers + "問で、");
        System.out.println("間違ってしまったのは" + (MAX_QUESTION - goodAnswers) + "問です。");
        System.out.println("正答率は" + rate + "%です。");
        System.out.println("");
        System.out.println("お疲れさま。");
    }
    /**
     * showQuestionは九九の問題を1問出し、答えを待つ。
     * 正答、誤答の別を表示する。
     * 正答の場合は true を返す。
     */
    public static boolean showQuestion(int questno) {
        int x = (int)(Math.random() * 9) + 1;
        int y = (int)(Math.random() * 9) + 1;
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        try {
            System.out.println("[第" + questno + "問] " + x + " × " + y + " = ?");
            String line = reader.readLine();
            int result = Integer.parseInt(line);
            if (x * y == result) {
                System.out.println("はい、正しいです。");
                return true;
            } else {
                System.out.println("残念、まちがいです。");
                return false;
            }
        } catch (IOException e) {
            System.out.println(e);
        } catch (NumberFormatException e) {
            System.out.println("入力が正しくありません。");
        }
        return false;
    }
}
```

変数と型（第3章）

- 乱数
  - Math クラスの random メソッド

```
int x = (int)(Math.random() * 9) + 1;
int y = (int)(Math.random() * 9) + 1;
```

1～9のランダムな整数値

  - 0.0以上1.0未満のdouble型の乱数を生成

- キャスト演算子（型変換）
  - (int) … double型をint型に変換

java.lang.Math

変数と型（第3章）

- コメント
  - 説明文、注釈文

```
/**
 * Kuku は、九九の問題をランダムに10個表示して
 * その正解数と正解率を表示するプログラムである。
 *
 * @author 結城浩
 * @copyright 1999 by Hiroshi Yuki.
 */


// 正答の個数
```

```
/* コメント1 */

// コメント2

/** ドキュメンテーションコメント */
```

変数と型（第3章）

- その他
  - final：変更不可能な「定数」（代入不可能）
  - boolean：真偽値を表す型 (true / false)

```
public static final int MAX_QUESTION = 10;
...
boolean ok = showQuestion(i + 1);
```