初級プログラミング Ⅰ

第2回　プログラミングの初歩

中谷　祐介

第2回の構成

◆ 第1講　ソースファイル

- Javaにおけるソースファイルの形式とプログラムの記述に関する知識を習得する.

◆ 第2講　データの出力

- Javaにおけるデータの出力方法を理解する.

◆ 第3講　変数

- Javaにおける変数の基本を理解する.

◆ 第4講　データの入力・演算

- Javaにおける入力方法と四則演算の方法を理解する.

第2回　プログラミングの初歩

第1講　ソースファイル

第1講の学習目標

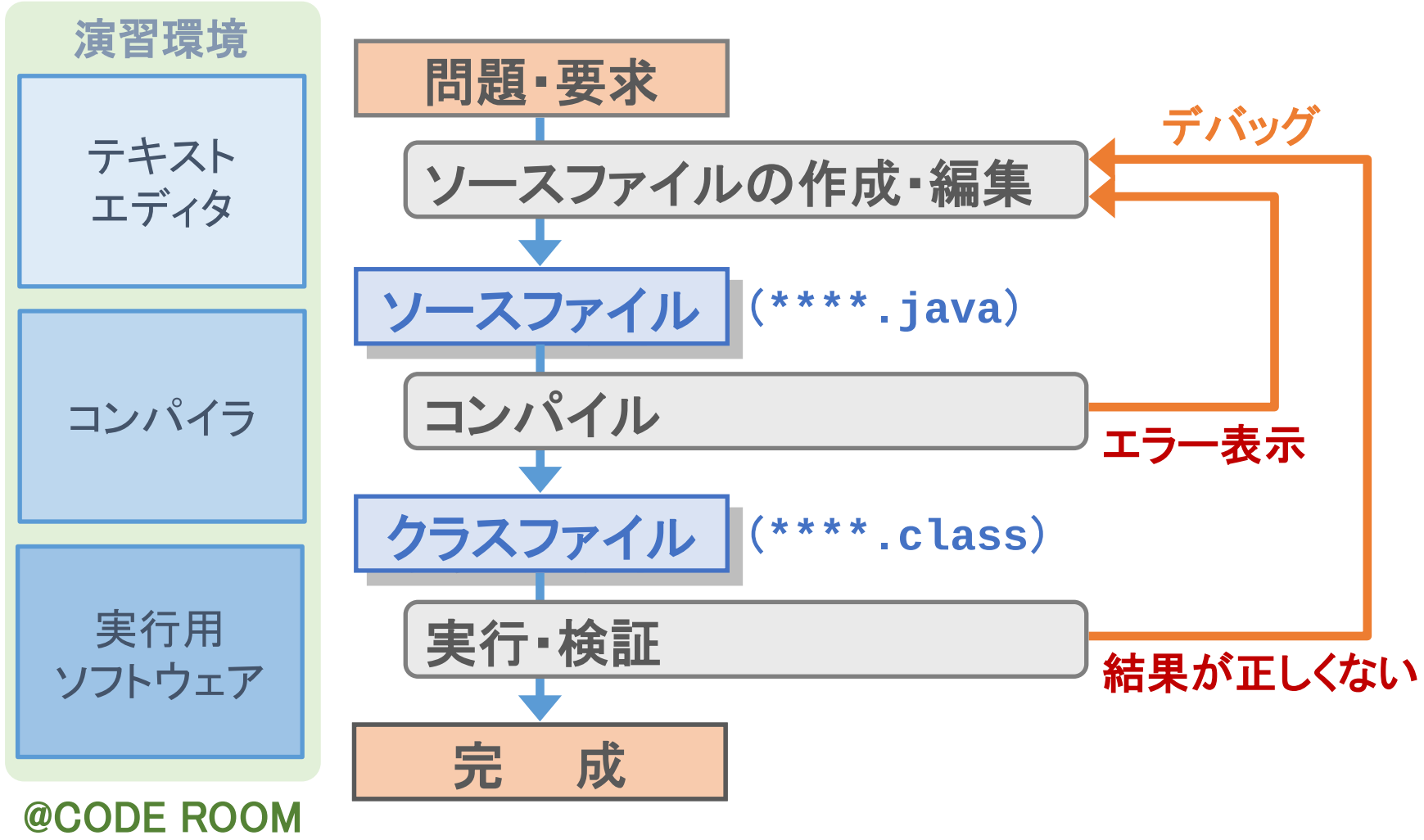
◆ Java におけるソースファイルの形式について理解する：

- ファイル名とクラス名
- クラスの宣言
- メソッドの宣言
- プログラム本体の記述

◆ Java におけるプログラムの記述について，基本事項を理解する：

- 文の終わりは「；」（セミコロン）
- コメントの記述
- 改行，インデント

Javaによるプログラミングの手順



Javaによるプログラミングの手順

ソースファイル（復習）

◆ プログラムを記述したファイルを**ソースファイル**という.

練習問題1-1

標準出力に以下のメッセージを出力するプログラムを作成してください.

```
Hello!
How are you?
```

◆「練習問題1-1」のプログラム例を記述したソースファイルの内容は以下のとおり:

Practice_01_01.java

```
1 /* 「練習問題1-1」のプログラム例 */
2 public class Practice_01_01 {
3     public static void main(String[] args) {
4         System.out.println("Hello!");          // メッセージ1
5         System.out.println("How are you?");    // メッセージ2
6     }
7 }
```

ソースファイルの形式（1） ― ファイル名とimport ―

クラス名.java

```
import ライブラリ名;          // 省略可能

public class クラス名 {
    public static void main(String[] args) {

    /* ここにプログラムの本体を記述 */

    }
}
```

- ソースファイルの拡張子は「java」とする（****.java）.
  - ソースファイル名は，ファイル内で指定するクラス名と一致させる.
- 「import ～」は，プログラム内で必要となるツールを読み込む部分.
  - 例：import java.util.Scanner; ― 標準入力からのデータ入力のツール.
  - 読み込むツールがない場合は，省略可能.

ソースファイルの形式（2） ― クラスの宣言 ―

クラス名.java

```
import ライブラリ名;          // 省略可能

public class クラス名 {
    public static void main(String[] args) {

    /* ここにプログラムの本体を記述 */

    }
}
```

- Java では，ソースファイル内で**クラス**というものを宣言することでプログラムを記述する．
- クラス名を設定する際の注意点：
  - 大文字と小文字は区別される（例：「Abc」と「ABC」は別のもの）．
  - 最初の文字は原則として大文字とする．（例：Abc）．
  - ソースファイル名は「**クラス名.java**」とし，クラス名と一致させる．

ソースファイルの形式（3） ― main メソッドの宣言 ―

クラス名.java

```
import ライブラリ名;          // 省略可能

public class クラス名 {
    public static void main(String[] args) {

    /* ここにプログラムの本体を記述 */

    }
}
```

- Java のクラスは，**メソッド**と呼ばれるもので構成される．
  - 上記のソースファイルでは，main メソッドを宣言している．
- Java のプログラムでは，クラス内に宣言された main メソッドから処理が始まる．

➡ Java のプログラムでは，クラスを宣言し，その中に main メソッドを宣言する．

ソースファイルの形式（4） ― プログラム本体の記述 ―

クラス名.java

```
import ライブラリ名;          // 省略可能

public class クラス名 {
    public static void main(String[] args) {

    /* ここにプログラムの本体を記述 */

    }
}
```

- プログラムは，main メソッドの中に記述する．
- プログラムは，記述した**文**を上から順に実行する：

```
System.out.println("Hello!");
System.out.println("How are you?");
```

  - 最初に「Hello!」を1行で出力する．
  - 次に「How are you?」を1行で出力する．

プログラムの記述（1）

Practice_01_01.java

```
1 /* 「練習問題1-1」のプログラム例 */
2 public class Practice_01_01 {
3     public static void main(String[] args) {
4         System.out.println("Hello!");           // メッセージ1
5         System.out.println("How are you?");     // メッセージ2
6     }
7 }
```

プログラムの記述には半角文字を使用する.

- プログラムの記述に必要なアルファベット，数字，記号，空白はすべて半角文字でなければならない.
- 全角文字を使用すると，コンパイル時にエラーとなる.
- 空白は表示されないので，特に注意が必要.

プログラムの記述（2）

Practice_01_01.java

```
1 /* 「練習問題1-1」のプログラム例 */
2 public class Practice_01_01 {
3     public static void main(String[] args) {
4         System.out.println("Hello!");          // メッセージ1
5         System.out.println("How are you?");    // メッセージ2
6     }
7 }
```

文は基本的に「;」（セミコロン）で終わる.

- 上のプログラム例でも，5行目，6行目は「;」で終わっている.
- 「;」をつけ忘れると，コンパイル時にエラーとなる.

プログラムの記述（3）

Practice_01_01.java

```
1 /* 「練習問題1-1」のプログラム例 */
2 public class Practice_01_01 {
3     public static void main(String[] args) {
4         System.out.println("Hello!");          // メッセージ1
5         System.out.println("How are you?");    // メッセージ2
6     }
7 }
```

```
/* コメント */
// コメント
```
：コメント（注釈）の記入

◆ プログラム中の以下の部分はコメントとして扱われ，コンパイル，実行時には影響しない：

- /* と */ で囲んだ部分．
- // 以降，その行末までの部分．

```
/*
    ここはコメントです．
    ここもコメントです．
*/
```

```
// ここはコメントです．
   ここはコメントではありません．
```

プログラムの記述（4）

Practice_01_01.java

```
1 /* 「練習問題1-1」のプログラム例 */
2 public class Practice_01_01 {
3     public static void main(String[] args) {
4         System.out.println("Hello!");           // メッセージ1
5         System.out.println("How are you?");     // メッセージ2
6     }
7 }
```

改行，インデント（字下げ）を行い，読みやすくする．

- 行頭に空白を入れ字下げを行うことを**インデント**という．
- 上のプログラム例は，次のように記述しても同じである．

```
/* 「練習問題1-1」のプログラム例 */
public class Practice_01_01 {public static void main(String[] args) {
System.out.println("Hello!");// メッセージ1
System.out.println("How are you?");// メッセージ2
}}
```

第1講のまとめ

- Java におけるソースファイルの形式について理解した：
  - ファイル名とクラス名
  - クラスの宣言
  - メソッドの宣言
  - プログラム本体の記述
- Java におけるプログラムの記述について，基本事項を理解した：
  - 文の終わりは「；」（セミコロン）
  - コメントの記述
  - 改行，インデント

第2回　プログラミングの初歩

第1講　ソースファイル

終わり

第2回　プログラミングの初歩

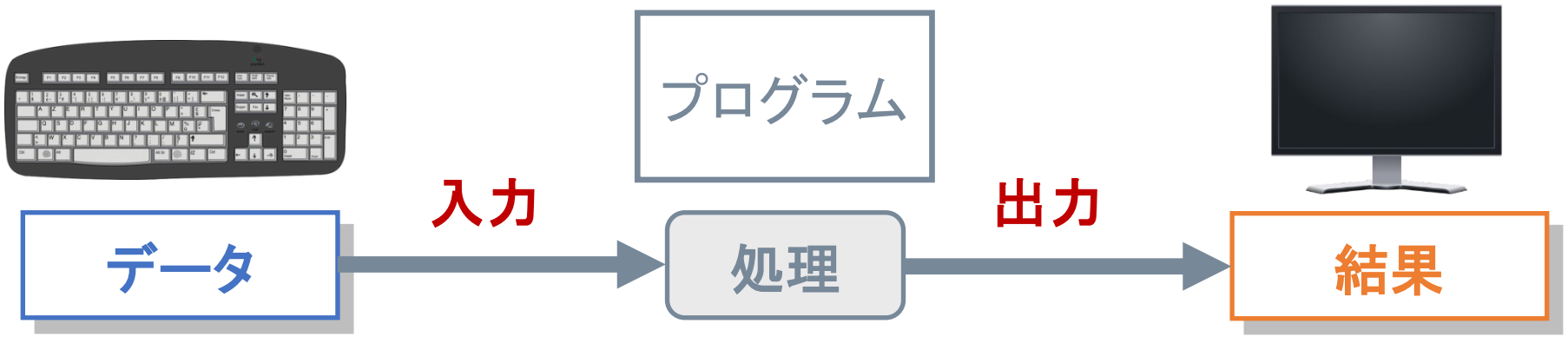
第2講　データの出力

第2講の学習目標

- Java におけるデータの出力方法を理解する:
  - メソッド
  - `println` メソッド, `print` メソッド
  - 改行文字

データの入出力

◆ プログラムに対してデータを入力し，処理を行い，その結果を出力する流れのこと．



データの入出力のイメージ

◆ データの出力 ： プログラム内で処理を行った結果を，書き出す．
出力先として，標準出力（画面）やファイルがある．

➡ 第2講で習得

◆ データの入力 ： プログラム内で使用するデータをプログラム内に読み込む．
入力元として，キーボードやファイルがある．

➡ 第4講で習得

データの出力

- プログラム内で処理を行った結果を，書き出すこと．
- 本科目では，データの出力先は標準出力（画面）とする．
- 第1回の「練習問題1-1」では，「System.out.println」というものを使用してメッセージを出力した：

```
System.out.println("Hello!");
System.out.println("How are you?");
```

```
Hello!
How are you?
```

- 「System.out.println」を使用して，データ（文字，数値，記号，等）を出力する．

メソッド

◆ Java のプログラムにおいて，まとまった処理を行う道具のことを**メソッド**という．

Practice_01_01.java

```
1 /* 「練習問題1-1」のプログラム例 */
2 public class Practice_01_01 {
3     public static void main(String[] args) {
4         System.out.println("Hello!");           // メッセージ1
5         System.out.println("How are you?");     // メッセージ2
6     }
7 }
```

- `main` メソッド：Java のプログラムを動かすためのメソッド．
- `println` メソッド：データを出力するメソッド．

println メソッド

[Sample_02_01.java]

System.out.println(データ)：「データ」を標準出力に1行で出力するメソッド

- 文字列を出力する場合は，文字列を " " で囲む．
- 「データ」の出力後は，改行される．
- 文字列の出力例：

```
System.out.println("Hello!");
System.out.println("How are you?");
```

```
Hello!↵
How are you?↵
```

- 数値の出力例：

```
System.out.println(15);
System.out.println(15 + 12);
```

```
15↵
27↵
```

  - ( ) 内に計算式を記述すると，計算結果を出力する．

print メソッド

[Sample_02_02.java]

System.out.print(データ)：「データ」を標準出力に出力するメソッド

- 使用方法は，println メソッドと同じ．
- 「データ」の出力後は，改行されない．
- print メソッドの使用例1：

```
System.out.print("Hello! ");
System.out.println("How are you?");
```

```
Hello! How are you?
```

- print メソッドの使用例2：

```
System.out.println("Hello!");
System.out.print("How ");
System.out.print("are ");
System.out.println("you?");
```

```
Hello!
How are you?
```

改行文字

[Sample_02_03.java]

¥n ： 出力の際，任意の場所で改行するための文字

- println メソッドや print メソッドで，" " 内に「¥n」を記述すると，その場所で強制的に改行する．
- 改行文字の使用例：

```
System.out.println("Hello!¥nHow are you?");
```

```
Hello!↵
How are you?↵
```

【注意】

演習環境では，「¥」は「\」（バックスラッシュ）で表示される．
演習環境で「¥」と入力しても「\」と表示されることに注意！

文字列の連結

[Sample_02_03.java]

- println メソッドや print メソッドで，文字列を「+」で繋ぐと，文字列を連結する．
- 文字列の連結の例1：

```
System.out.println("Hello!" + "How are you?");
```

```
Hello!How are you?↵
```

  - 2つの文字列「Hello!」，「How are you?」を繋いで出力する．

- 文字列の連結の例2：

```
System.out.println(15 + 12);
System.out.println("15" + "12");
```

```
27↵
1512↵
```

  - "15"，"12" は，文字列．

例題2-1（1）

例題2-1

標準出力に以下を出力するプログラムを作成してください.

```
東京通信大学
情報マネジメント学部
初級プログラミングI
```

Example_02_01.java

```
1 public class Example_02_01 {
2     public static void main(String[] args) {
3         System.out.println("東京通信大学");
4         System.out.println("情報マネジメント学部");
5         System.out.println("初級プログラミングI");
6     }
7 }
```

例題2-1（2）

◆「例題2-1」の別のプログラム例

Example_02_01.java

```
1 public class Example_02_01 {
2     public static void main(String[] args) {
3         System.out.println("東京通信大学¥n情報マネジメント学部¥n初級プログラミングI");
4     }
5 }
```

Example_02_01.java

```
1 public class Example_02_01 {
2     public static void main(String[] args) {
3         System.out.print("東京");
4         System.out.print("通信大学¥n情報");
5         System.out.print("マネジメント");
6         System.out.print("学部¥n初級");
7         System.out.print("プログラミングI¥n");
8     }
9 }
```

- 一つの問題に対して，プログラムの記述は一通りではない．

第2講のまとめ

- Java におけるデータの出力方法を理解した:
  - メソッド
  - `println` メソッド, `print` メソッド
  - 改行文字

第2回　プログラミングの初歩

第2講　データの出力

終わり

第2回　プログラミングの初歩

第3講　変数

第3講の学習目標

- Java における変数の基本を理解する:
  - 変数
    - 変数の型
    - 変数の宣言
    - 変数名
    - 代入
  - 変数の値の出力
  - 変数の計算

変数

- **変数**は，データ（数値，文字，等）を格納する箱のようなもの．

変数のイメージ

- データの種類に応じて，変数の種類（**型**という）がある．
- 変数を利用するためには，事前に箱を準備する必要がある．これを変数の**宣言**という：

```
int abc;          // 整数値を格納する箱 abc を準備
double xyz;       // 実数値を格納する箱 xyz を準備
```

- 変数には，データを格納できる．データを格納する操作を，値の**代入**という：

```
abc = 12;         // 箱 abc に「12」を格納
xyz = 12.34;      // 箱 xyz に「12.34」を格納
```

変数の型

- プログラム内で数値や文字を扱う場合には**変数**を利用し，この中にデータ格納する.
- 変数には**型**というものがあり，変数にどのようなデータを格納するのかでこの型を選択する.

変数の型

	型名	型の内容	値の範囲
	boolean	真偽	true または false
	char	文字	¥u0000 ～ ¥uFFFF
	byte	整数（範囲がとても狭い）	-128 ～ 127
	short	整数（範囲が狭い）	-32,768 ～ 32,767
整数値→	int	整数	-2,147,483,648 ～ 2,147,483,647
	long	整数（範囲が広い）	-9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807
	float	単精度浮動小数点数	±1.401298E-45 ～ ±3.4028235E+38
実数値→	double	倍精度浮動小数点数	4.94065645841246544E-324 ～ ±1.79769313486231570E+308

変数の宣言

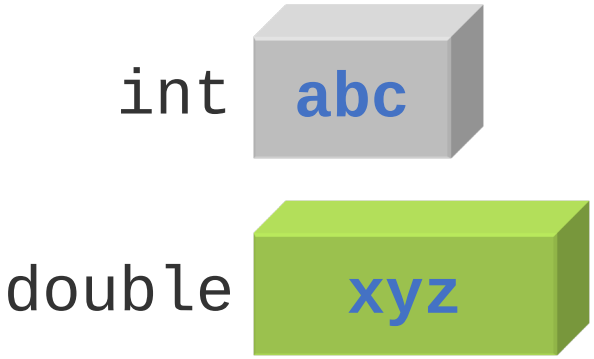
- プログラム内で変数を利用するには，**宣言**をする必要がある．
- 変数の宣言は，次の形で行う：

```
型名　変数名;
```

- 変数宣言の例

```
int abc;
double xyz;
```

  - 型名につづき，変数名を記述する．



変数宣言のイメージ

```
int p, q, r;
```

  - 「,」で区切ることで，複数の変数をまとめて宣言できる．

変数名

- 変数名にはアルファベットおよび「_」が使える.

```
int abc_def;   — ○
int abc-def;   — ×
```

- 2文字目以降は「0」～「9」の数字が使える.

```
int abc5;      — ○
int 5abc;      — ×
```

- 大文字と小文字は区別される.

```
int abc;
int Abc;
```
} 別の変数として扱われる.

- 予約語は使えない.
  - int, double, if, else, for などは，変数名として使えない.

[参考] Java の予約語一覧

Java の予約語一覧

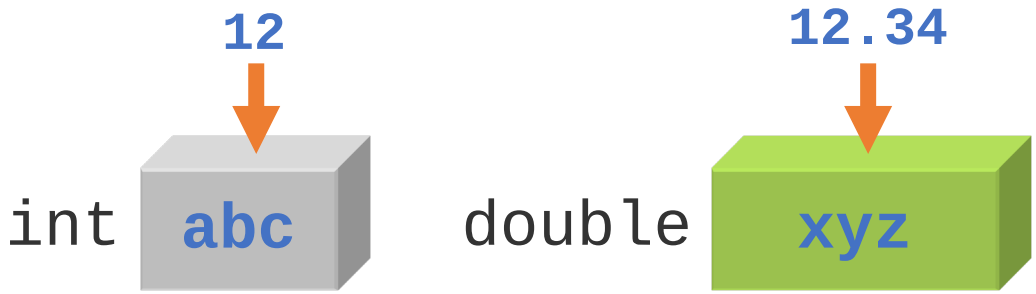
abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	short	static	strictfp	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

値の代入

- 変数に値を格納する操作のことを**代入**という.
- 値の代入には「**=**」(**代入演算子**)を使用する.

```
int abc;
double xyz;

abc = 12;
xyz = 12.34;
```



値の代入のイメージ

  - 「=」の右側の値を，左側の変数に代入する.

- 変数を宣言すると同時に値を代入することができる.

```
int abc = 123;
```

  - これを**初期化**という.

変数の値の出力

[Sample_02_04.java]

◆ 変数に代入されている値を出力するには，println, print メソッドを利用できる：

```
int abc = 12;
double xyz = 12.34;

System.out.println(abc);
System.out.println(xyz);
```

```
12↵
12.34↵
```

- println メソッドの ( ) 内に変数名を記述することで，変数内の値を出力．

```
System.out.println("abc:" + abc);
System.out.println("xyz:" + xyz);
```

```
abc:12↵
xyz:12.34↵
```

- 文字列に変数内の値を連結して出力．

変数の計算

[Sample_02_05.java]

- 変数への代入には，計算式を利用できる：

```
int a;

a = 12 + 5 - 7;
System.out.println(a);
```

```
10
```

  - 計算式「12+5-7」の結果（10）を，変数 a に代入．

- int 型や double 型の変数は，数値と同じように計算できる：

```
int b, c;

b = 12;
c = b + 5;
System.out.println(c);
```

```
17
```

  - 上記の場合，変数 b は「12」として扱われる．．

変数の上書き(1)

[Sample_02_06.java]

◆ 変数へ値を代入するたびに，変数の内容は上書きされる：

```
int a = 12;                              // int型の変数 a を「12」で初期化
double x = 12.34;                        // double型の変数 x を「12.34」で初期化

System.out.println("a:" + a);
System.out.println("x:" + x);

a = 50;                                  // 変数 a に「50」を上書きして代入
x = 100;                                 // 変数 x に「100」を上書きして代入

System.out.println("a:" + a);
System.out.println("x:" + x);
```

```
a:12↵
x:12.34↵
a:50↵
x:100.0↵
```

- プログラムは，上から下に向かって順に実行される．

変数の上書き(2)

[Sample_02_06.java]

◆ 変数の内容を利用して，同じ変数へ代入することができる：

```
int b = 12;

b = b + 10;                          //  変数 b の内容(12)に「10」を加え，変数 b に代入
System.out.println("b:" + b);   //  ここでは，変数 b の内容は「22」

b = b + 20;                          //  変数 b の内容(22)に「20」を加え，変数 b に代入
System.out.println("b:" + b);   //  ここでは，変数 b の内容は「42」
```

```
b:22↵
b:42↵
```

- 代入演算子「=」の機能は，「=」の右側の値を，左側の変数に代入すること．
- 「b = b + 10;」は，「b + 10」の結果を，変数 b にあらためて代入．

例題2-2

例題2-2

以下に示すプログラムを確認してください．このプログラムをコンパイル，実行したときに，「標準出力」に出力される結果を答えてください．

Example_02_02.java

```
1  public class Example_02_02 {
2      public static void main(String[] args) {
3          int a, b, c;
4
5          a = 5;
6          b = 7;
7          c = a + b + 10;    // 「5+7+10」を計算し，その結果(22)を変数 c に代入
8          a = 8;
9          a = c - a;         // 「22-8」を計算し，その結果(14)を変数 a に代入
10
11         System.out.println("a:" + a);
12     }
13 }
```

標準出力

```
a:14↵
```

第3講のまとめ

- Java における変数の基本を理解した：
  - 変数
    - 変数の型
    - 変数の宣言
    - 変数名
    - 代入
  - 変数の値の出力
  - 変数の計算

第2回　プログラミングの初歩

第3講　変数

終わり

第2回　プログラミングの初歩

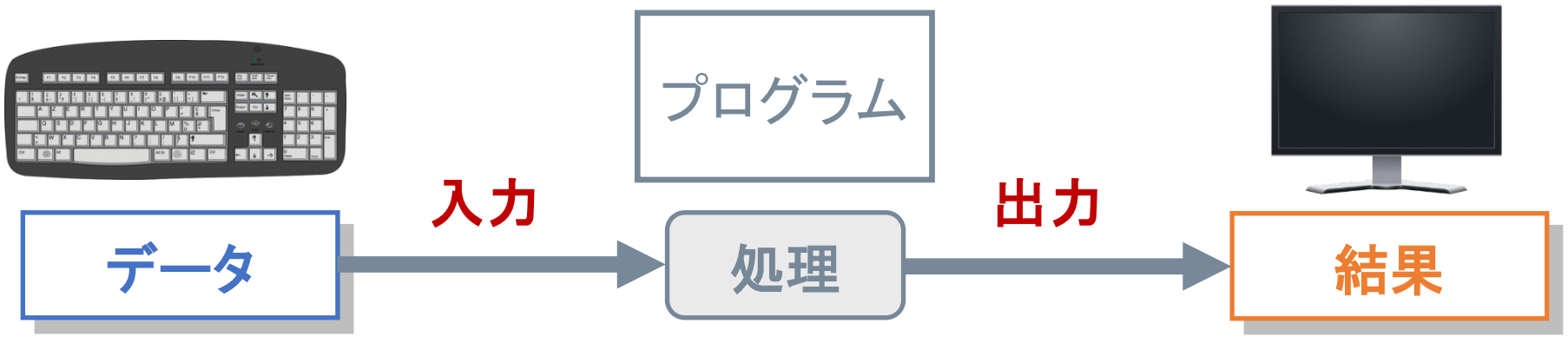
第4講　データの入力・演算

第4講の学習目標

- Java におけるデータの入力方法を理解する:
  - Scanner クラス
  - 整数値，実数値の入力
- Java における四則演算の方法を理解する:
  - 算術演算子
  - 型による計算の違い
  - 演算の優先順位

データの入出力

◆ プログラムに対してデータを入力し，処理を行い，その結果を出力する流れのこと．



データの入出力のイメージ

◆ データの出力 ： `println` メソッド，`print` メソッドを利用

◆ データの入力 ： プログラム内で使用するデータをプログラム内に読み込む．
入力元として，キーボードやファイルがある．
本科目では，入力元を標準入力（キーボード）とする．

データの入力

- プログラム内で使用するデータを読み込むこと.
- 本科目では，データの入力元は標準入力（キーボード）とする.
- 本科目では，データの入力に **Scanner** クラスというツールを利用する.
- Scanner クラスを利用するためには，その準備として，プログラムの先頭に次の1行を記述する必要がある:

```
import java.util.Scanner;

public class クラス名 {
    public static void main(String[] args) {
        /* プログラムの本体 */
    }
}
```

- Scanner クラスというツールを読み込む.

Scanner クラス（1）

- Scanner クラスを利用してデータの入力を行う場合，プログラムの本体の先頭で，次のように記述する：

```
import java.util.Scanner;

public class クラス名 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        /* プログラムの本体 */

    }
}
```

  - 変数の宣言と同様に，Scanner クラスの変数 scan を宣言．
  - この記述以降は，変数 scan を通して標準入力からデータを読み込む．
  - 変数名 scan は自由に設定することができるが，本科目では scan で統一する．

Scanner クラス（2）

- Scanner クラスには，数値をプログラム内に読み込む道具がある：
  - **nextInt メソッド**：整数値（int 型）を読み込むメソッド
  - **nextDouble メソッド**：実数値（double 型）を読み込むメソッド

- 標準入力に入力した整数値をプログラム内に読み込む例：

```
Scanner scan = new Scanner(System.in);

int a = scan.nextInt();
```

  - Scanner クラスの変数 scan を宣言．
  - 変数 scan に対して nextInt メソッドを使用し整数値を読み込む．
  - 読み込んだ整数値を int 型の変数 a に代入．

整数値の入力

[Sample_02_07.java]

◆ 標準入力に入力した2つの整数値をプログラム内に読み込む例：

```
Scanner scan = new Scanner(System.in);

int a = scan.nextInt();          // 変数 a を宣言して，整数値を読み込む
System.out.println("a:" + a);

int b = scan.nextInt();          // 変数 b を宣言して，整数値を読み込む
System.out.println("b:" + b);
```

- 実行時の入力例1

標準入力

```
15
20
```

標準出力

```
a:15↵
b:20↵
```

- 実行時の入力例2

標準入力

```
15 20
```

標準出力

```
a:15↵
b:20↵
```

実数値の入力

[Sample_02_08.java]

◆ 標準入力に入力した2つの実数値をプログラム内に読み込む例：

```
Scanner scan = new Scanner(System.in);
double x;                              // 変数 x を事前に宣言

x = scan.nextDouble();                 // 実数値の読み込み
System.out.println("x:" + x);

x = scan.nextDouble();                 // 実数値の読み込み
System.out.println("x:" + x);
```

- 実行時の入力例1

標準入力

```
123.4
56.7
```

標準出力

```
x:123.4↵
x:56.7↵
```

- 実行時の入力例2

標準入力

```
123.4 56.7
```

標準出力

```
x:123.4↵
x:56.7↵
```

四則演算（1）

[Sample_02_09.java]

- 四則演算（＋，－，×，÷）は，**算術演算子**を使用することで計算できる．
- 四則演算の例（int型同士の計算）

```
int a, b, c;

a = 11;
b = a + 4;    System.out.println(b);   → b は「15」
c = a * b;    System.out.println(c);   → c は「165」
c = c / 10;   System.out.println(c);   → c は「16」
c = c % 10;   System.out.println(c);   → c は「6」
```

```
15
165
16
6
```

算術演算子

演算子	演算
+	加算
-	減算
*	乗算
/	除算
%	剰余算

- **整数同士の演算結果は，整数値となる．**
- 整数同士の割り算の場合は小数点以下が切り捨てられ，整数値として計算される．

四則演算（2）

[Sample_02_10.java]

◆ 四則演算の例（int 型と double 型の計算）

```
int a = 1234;
double x = 123.4, y;

y = a / 10;     System.out.println(y);     → int型 / int型.
y = a / 10.0;   System.out.println(y);     → int型 / double型.
y = x / 10;     System.out.println(y);     → double型 / int型.
y = x / 10.0;   System.out.println(y);     → double型 / double型.
```

```
123.0↵
123.4↵
12.34↵
12.34↵
```

- int 型同士の計算は int 型で，double 型同士の計算は double 型で行われる．
- int 型と double 型を混合して計算する場合，**計算結果は double 型**となる．
- プログラムの中で「10.0」のように「**.0**」をつけると，double 型として解釈される．

四則演算（3）

[Sample_02_11.java]

◆ 四則演算の例（演算の優先順位）

```
int a = 1, b = 2, c;

c = a + b * 3;
c = (a + b) * 3;
```

```
7
9
```

- 演算には優先順位がある.
- 加算(+)，減算(-)より，乗算(*)，除算(/)，剰余算(%)が先に計算される.
- **( )** を使用することで，優先順位を変更することができる.

例題2-3

例題2-3

標準入力に2つの整数値を入力すると，これらの四則演算（+，-，*，/）と剰余算（%）を行い，それぞれの計算結果を標準出力に出力するプログラムを作成してください．

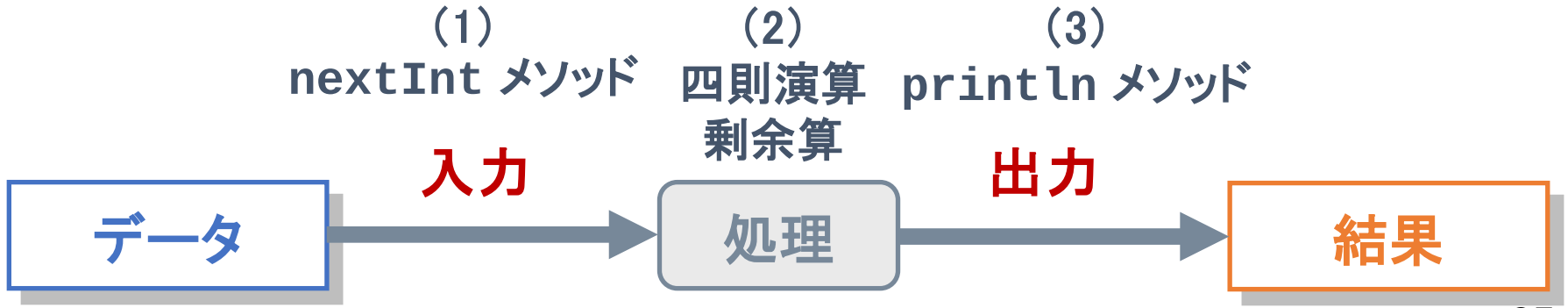
標準入力

```
20
3
```

標準出力

```
20+3=23
20-3=17
20*3=60
20/3=6
20%3=2
```

◆ 作成するプログラムの流れは次のとおり：



「例題2-3」のプログラム例（1）

Example_02_03.c

```
1  import java.util.Scanner;
2
3  public class Example_02_03 {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6
7          int a = scan.nextInt();
8          int b = scan.nextInt();
9
10         int add = a + b;
11         int sub = a - b;
12         int mul = a * b;
13         int div = a / b;
14         int mod = a % b;
15
16         System.out.println(a + "+" + b + "=" + add);
17         System.out.println(a + "-" + b + "=" + sub);
18         System.out.println(a + "*" + b + "=" + mul);
19         System.out.println(a + "/" + b + "=" + div);
20         System.out.println(a + "%" + b + "=" + mod);
21     }
22 }
```

◆ 実行例

標準入力	標準出力
20 3	20+3=23↵ 20-3=17↵ 20*3=60↵ 20/3=6↵ 20%3=2↵

「例題2-3」のプログラム例（2）

Example_02_03_2.c

```
1  import java.util.Scanner;
2
3  public class Example_02_03_2 {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6
7          int a = scan.nextInt();
8          int b = scan.nextInt();
9
10         System.out.println(a + "+" + b + "=" + (a+b));
11         System.out.println(a + "-" + b + "=" + (a-b));
12         System.out.println(a + "*" + b + "=" + (a*b));
13         System.out.println(a + "/" + b + "=" + (a/b));
14         System.out.println(a + "%" + b + "=" + (a%b));
15     }
16 }
```

第4講のまとめ

- Java におけるデータの入力方法を理解した:
  - Scanner クラス
  - 整数値，実数値の入力
- Java における四則演算の方法を理解した:
  - 算術演算子
  - 型による計算の違い
  - 演算の優先順位

第2回　プログラミングの初歩

第4講　データの入力・演算

終わり