Javaプログラミング実習

05. 演算と変数

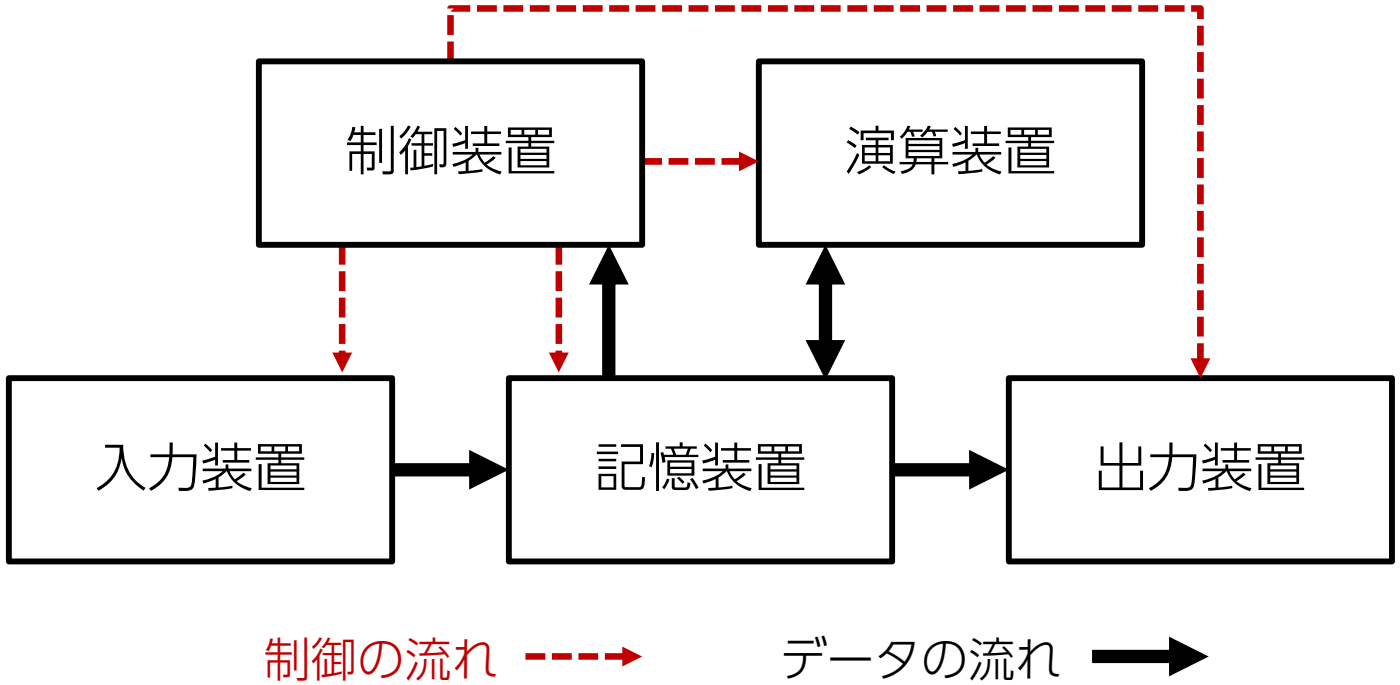
株式会社ジードライブ

今回学ぶこと

- コンピュータの構成とプログラムの流れ
- 演算と演算子
- 変数と型
- 入力値の取得

コンピュータの構成

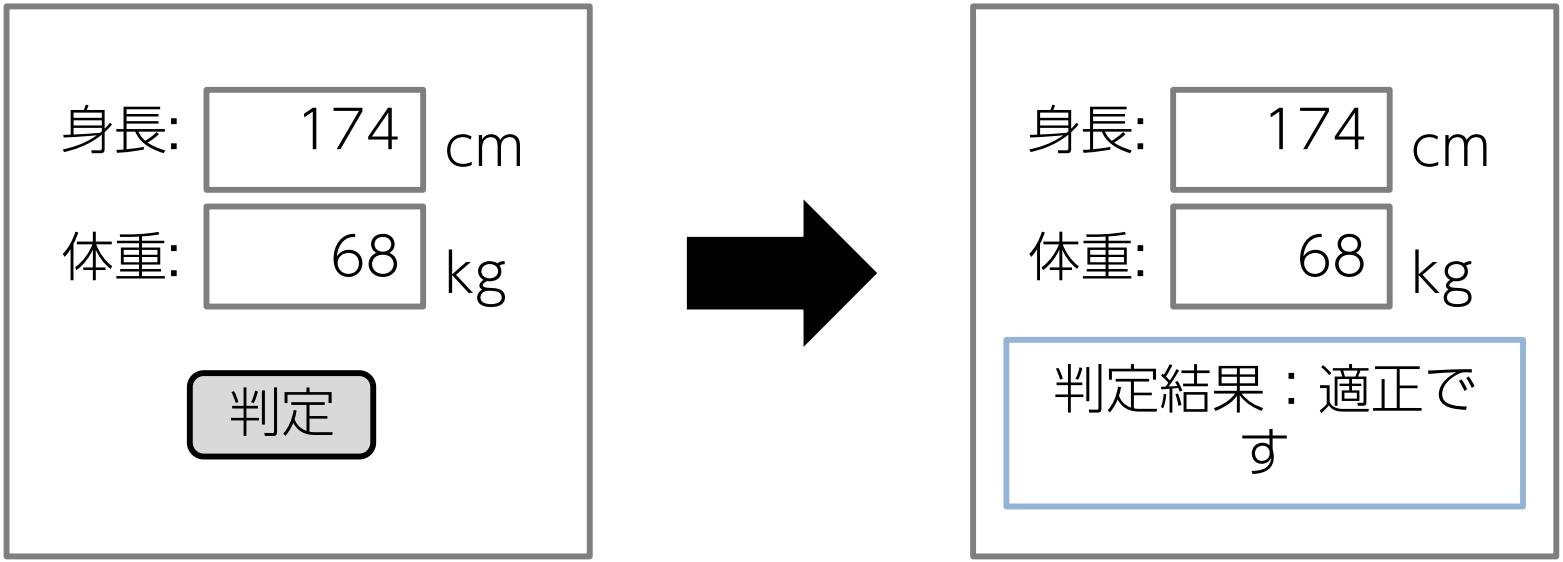
- コンピュータは、**入力**、**記憶**、**演算**、**出力**、制御機能を有する装置で構成される(五大機能・五大装置)



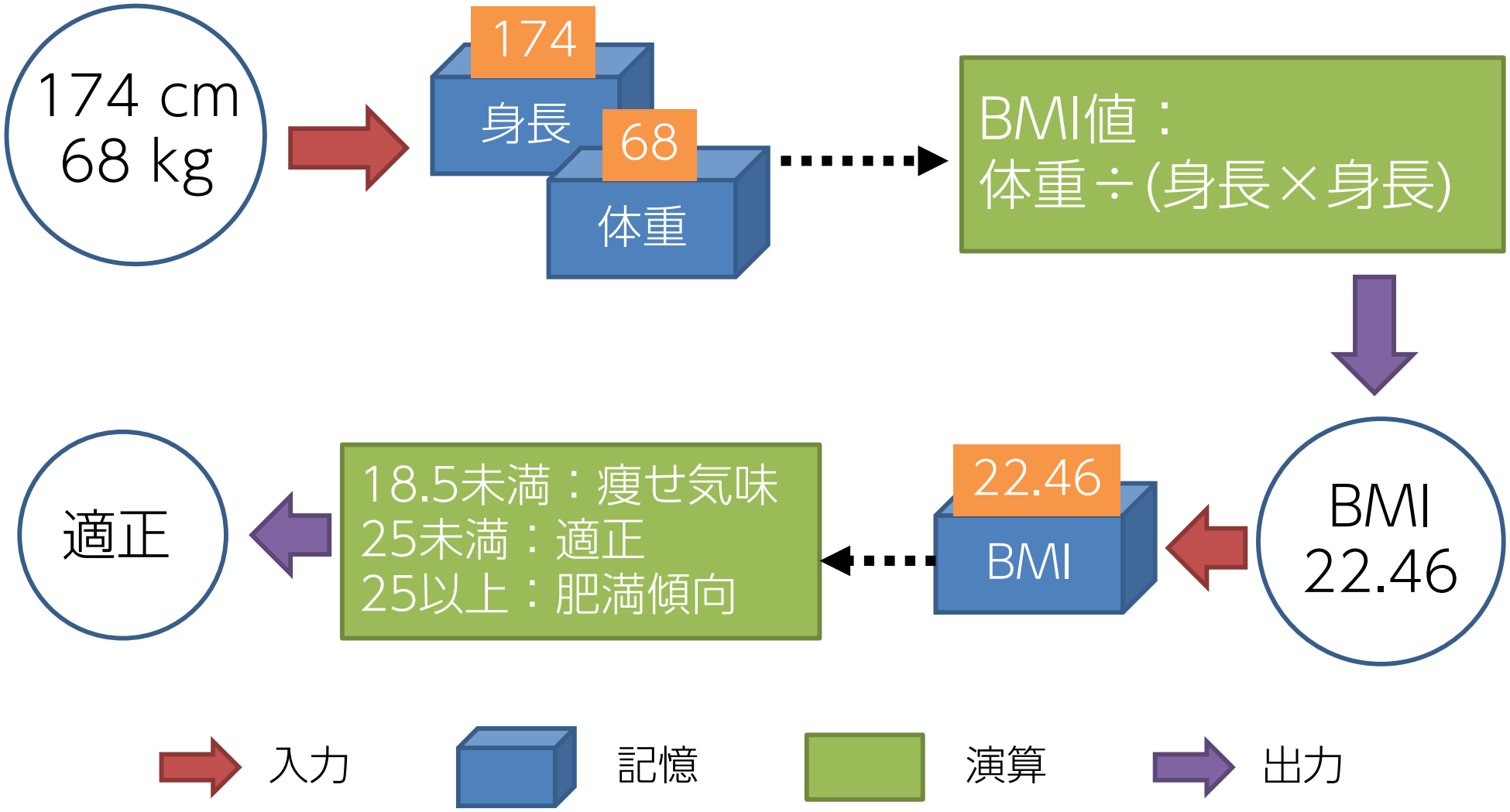
プログラムの流れ

- コンピュータ全体が**入力、記憶、演算、出力**という流れを有しているのと同様に、コンピュータの中で動くプログラムもこの流れを有する
  - この流れを意識できるとプログラムを作成しやすい

例：身長と体重から適正な体重か診断をするプログラム



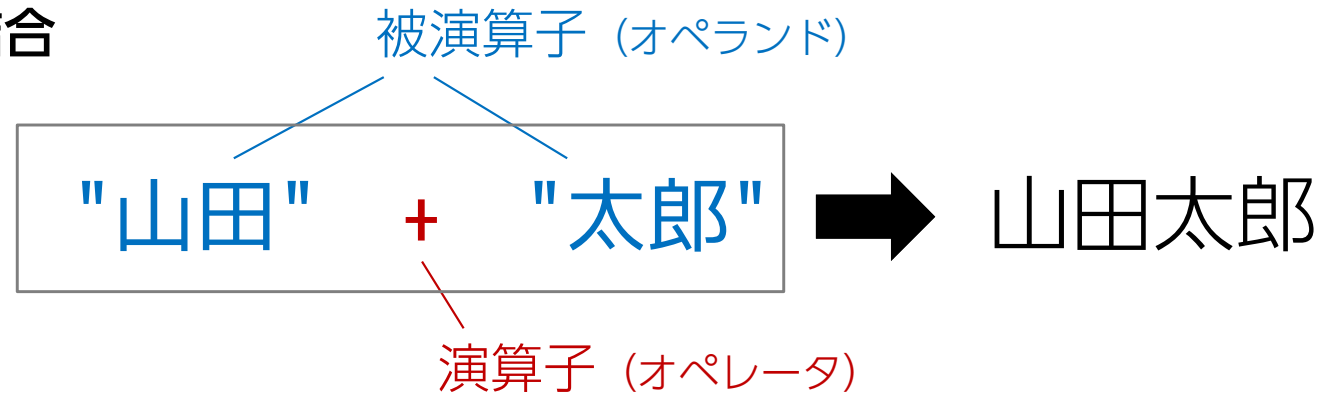
プログラムの流れ



演算と演算子

- プログラムでは数値同士の計算(足し算や引き算)だけでなく、文字同士の足し算(文字列の結合)や「得点が80点以上か否か」といった比較判定を行う
  - これらの計算や判定を総じて演算という
  - 演算で用いる記号を演算子という

文字列の結合



演算の例

- 四則演算の例

```
System.out.println(5 + 3); // 加算
System.out.println(5 - 3); // 減算
System.out.println(5 * 3); // 乗算
System.out.println(5.0 / 3); // 除算
```

5 / 3 と記述すると整数同士の割り算とみなされ、結果は 1 になる

```
8
2
15
1.6666666666666667
```

コンソール出力

演算子の例

- 算術演算子

+	加算
-	減算
*	乗算
/	除算
%	剰余算 (割った余り)

例

```
System.out.println(8 % 3);
```

```
2
```

コンソール出力

演算子の例

- 文字列結合演算子

+	文字列と文字列を結合する

例

```
System.out.println("私は" + "山田です");
```

```
私は山田です
```

コンソール出力

演算子の種類

- 主な演算子の例

演算子	意味
+	加算、文字連結
-	減算
*	乗算
/	除算
%	剰余算
++	インクリメント
--	デクリメント
new	オブジェクトの生成

演算子	意味
==	等しい
!=	等しくない
>	より大きい
>=	以上
<	より小さい
<=	以下
&&	論理積
||	論理和
!	論理否定

演算子の優先順位

- 数学のルールと同じく、乗算や除算は、加算や減算よりも先に計算が行われる
  - 丸カッコ ( ) で優先順位のコントロールが可能

順位	演算子
1	expr++ expr--
2	++expr --expr +expr -expr ~ !
3	* / %
4	+ -
5	<< >> >>>
6	< > <= >= instanceof
7	== !=
8	&
9	^
10	|
11	&&
12	||
13	? :
14	= += -= *= /= %= &= ^= |= <<= >>= >>>=

式

- 値や演算子を組み合わせた記述を式という

```
package com.example.hellojava;

public class Hello {
  public static void main(String[] args) {
    System.out.println(2 + 3);
  }
}
```

式の評価

- 式は**評価する**ことで結果の値を導き出す
  - 「評価する」は「計算する」とほぼ同義
  - プログラム内に式があると自動的に評価される

練習

- 練習05-1

データ型

- Javaでは様々な種類の値を扱うことができる
  - 整数 (1050 など)
  - 実数 (25.3 など)
  - 文字や文字列 ('A' , "こんにちは" など)
  - 真偽値 (true / false)
  - ヌル値 (null)： 何もデータが無いことを表す
    …等々
- 値の種類のことをデータ型や型と呼ぶ

データ型

- Javaで扱うことのできる以下の８種類のデータ型は、基本型 または プリミティブ型と呼ばれる

型名	説明
byte	1バイト整数 (-128 ～ 127)
short	2バイト整数 (-32768 ～ 32767)
int	4バイト整数 (-2^{31} ～ $2^{31}-1$)
long	8バイト整数 (-2^{63} ～ $2^{63}-1$)
float	4バイト単精度浮動小数点数
double	8バイト倍精度浮動小数点数
char	文字（日本語などの全角文字を含む）
boolean	true / false

基本型の名前はすべて小文字で始まる

データ型

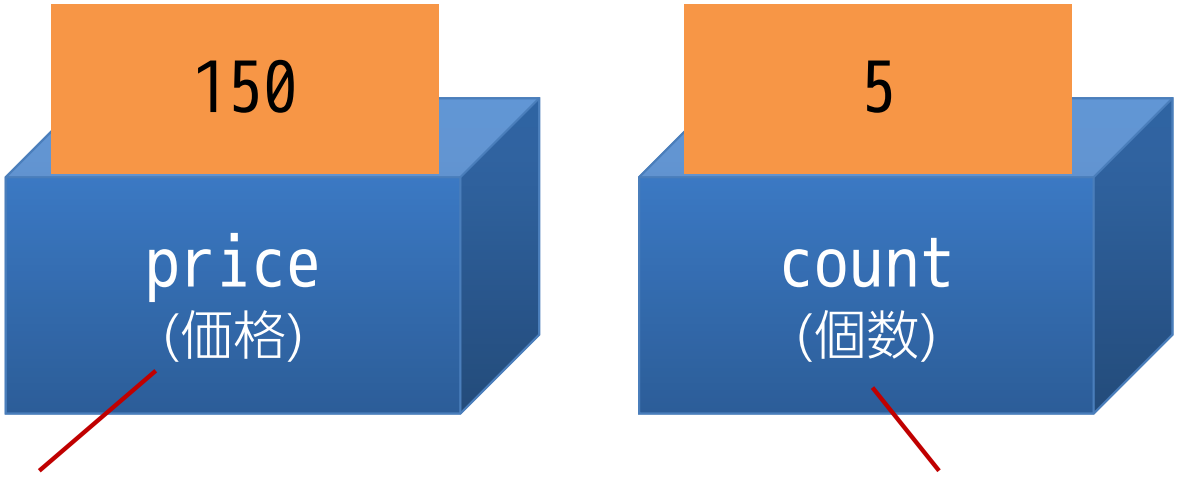
- 基本型以外にも、オブジェクト型、配列型、列挙型といったデータ型が存在する
  - 例： 文字列は`String`型という型で、基本型ではなくオブジェクト型に属する
  - 配列型は複数のデータをまとめて扱う場合に利用する

変数

- プログラムを記述していると、ユーザが入力したデータや演算の結果などを一時的に記憶しておきたい場面に遭遇する
  - 会員登録処理のためにユーザが入力した名前とメールアドレスを記憶しておきたい
  - ショッピングサイトで合計金額を計算／表示するために、小計金額を記憶しておきたい
  - 対戦ゲームで最終的な勝者を決定するために、各プレイヤーのスコアや勝敗数を記憶しておきたい

変数

- 一時的に記憶しておきたい値は変数に格納しておく(変数は箱／入れ物のようなイメージ)
  - 変数には名前を付けて管理する
  - 変数名はプログラマが自由につけてよい



150という値を格納している
priceという名前の変数

5という値を格納している
countという名前の変数

変数の宣言

- Javaでは、変数を使用する前に**変数を宣言する**(箱を作るイメージ)必要がある
- 変数の宣言では、変数名と格納するデータの型を指定する
  - 指定した型以外のデータを格納することはできない

書式

```
型 変数名;
```

例

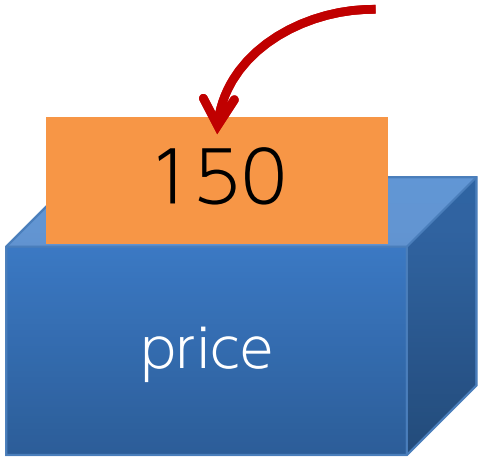
```
int price;    // int型の price という変数を宣言
String itemName; // String型の itemName という変数を宣言
```

変数への値の代入

- 宣言された変数には、代入演算子(=)を使って値を**代入する**(格納する)ことができる

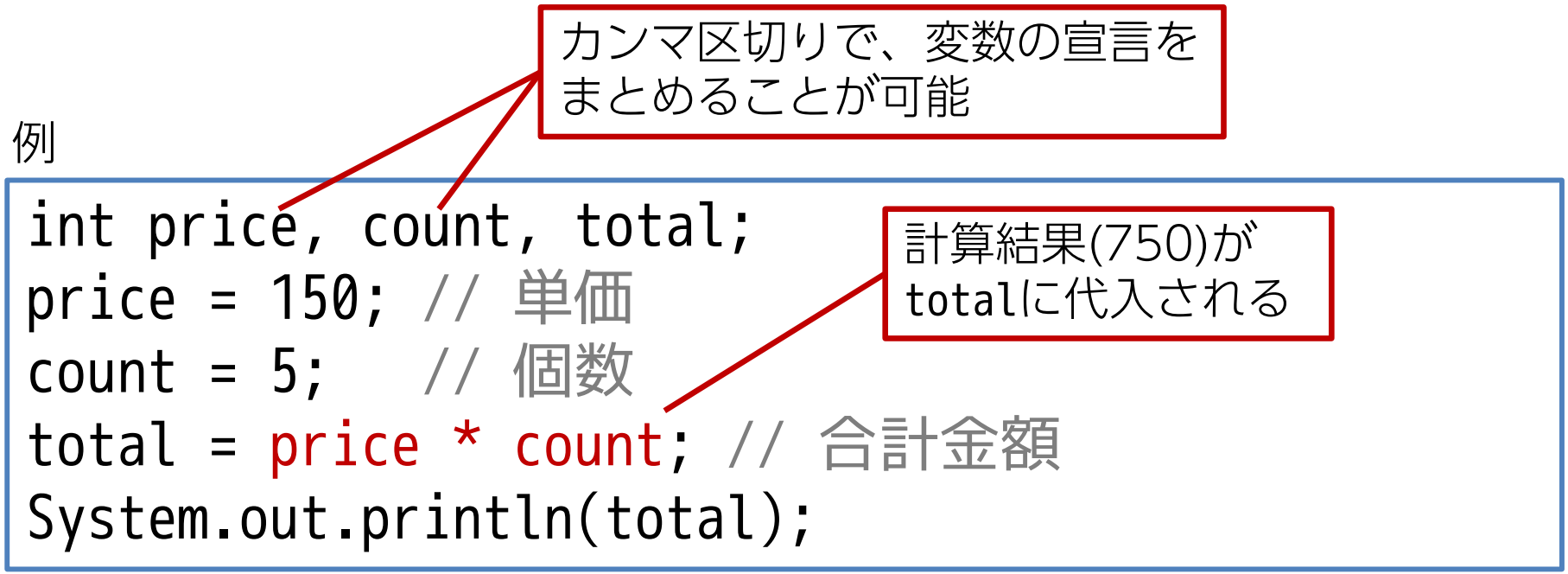
例

```
int price;     // int型の price という変数を宣言(箱を作成)
price = 150;   // 変数price に整数150を代入(箱に値を格納)
```



変数を使った式

- 変数を使って式を作ることができる




```
750
```

コンソール出力

変数への値の代入

- 変数宣言と同時に値を代入することもできる

書式

```
型 変数名 = 値;
```

例

```
int price = 150;
String item = "りんご";
String message = item + "は" + price + "円です";

double height = 1.72;
double weight = 68.4;
double bmi = weight / (height * height);
```

変数宣言の位置

- 変数の宣言は変数の使用前に記述する

正しい例

```
int price;   // 変数priceの宣言
int unit;
String itemName;
price = 150;  // 変数priceの使用
```

間違っている例

```
price = 150; // 変数priceの使用
int unit;
int price;   // 変数priceの宣言
```

× 宣言よりも先に使用してしまっている

練習

- 練習05-2

値の上書き

- 値が格納された変数に、改めて値を代入すると前にあった値は上書きされる
  - 同時に複数の値を格納することはできない

例

```
int price;
price = 150; // priceには150が格納されている
price = 200; // ⇒150はなくなり、200が格納された
System.out.println(price);
```

```
200
```

コンソール出力

値の上書き

例

```
int count = 1;
count = count + 1; // countにcount+1の結果を代入
System.out.println(count); // 2と表示される
count++;  // count = count + 1; と同じ意味
System.out.println(count); // 3と表示される
count += 3; // count = count + 3; と同じ
System.out.println(count); // 6と表示される
```

```
2
3
6
```

コンソール出力

練習

- 練習05-3

変数名について

- 変数名に使える文字は、半角英数字、アンダースコア(_)、ドル記号、日本語
  - 実際の開発では日本語の変数名は使われない
- 先頭の文字に半角数字は使えない
- Javaの**予約語**(キーワード)は変数名として使えない
  - 言語仕様として、予め使用目的が定められている
- `true, false, null` は変数名として使えない

変数名について

- 大文字と小文字は区別される
  - 例： **name** と **Name** は別の変数
- 変数名の長さは無制限
- 同じスコープ内では、同じ名前の変数を複数回宣言できない
  - スコープ = 変数の存在範囲 ≒ ブロック

Javaのキーワード

- 現段階で、すべて覚える必要はない

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

型変換

- ある型の値を、異なる型に変換することを**型変換(キャスト)**という
- 型変換によって、異なる型の変数に代入することができる
- 型変換には明示的なものと、暗黙的(自動的)なものがある

暗黙的な型変換の例

```
int a = 100;
long b = a; // int型の値をlong型の変数に代入している
```

自動的にint型からlong型への型変換が行われている

型変換

- べた書きされた整数(数値リテラル)はint型に、小数点を含む数はdouble型となる
  - 整数同士の計算結果はint型になる
  - 整数を実数を組み合わせて計算する場合、暗黙的な型変換が行われdouble型になる

```
System.out.println(25 / 8);   // 3
System.out.println(25 / 8.0); // 3.125
```

double型

int型 ⇒ double型に型変換されて計算が行われる

型変換

- データサイズの大きな型から、小さな型に変換する場合などには、明示的な型変換が必要

キャストの例（long型からint型への変換）

```
long b = 100;
int a = (int) b;
```

int型へのキャスト

- boolean型はどの型とも変換できない
  - 明示的なキャストも不可

文字列から数値への変換

- 文字列を数値に型変換するには、Integer.parseInt() や Double.parseDouble() を使用する

```
String strNum1 = "3776";
String strNum2 = "47.195";

// 文字列を数値に変換
int num1 = Integer.parseInt(strNum1);
double num2 = Double.parseDouble(strNum2);
```

入力値の取得

- Scanner を利用することで、コンソール画面からの入力を受け付けることができる

```
package com.example.sample;
import java.util.Scanner;

public class Sample {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("名前を入力してください：");
    String name = scanner.nextLine();
    System.out.println("こんにちは" + name + "さん");
    scanner.close();
  }
}
```

Scanner を利用するために必要な記述

Scanner の利用開始

文字列入力の受け取り

Scanner の利用終了

入力値の取得

- 必要に応じて、受け取った文字列を数値に変換する
  - 変換できない場合は、例外（エラー）が発生する

```
System.out.print("名前を入力してください：");
String name = scanner.nextLine();  // 文字列の入力を受け付ける

System.out.print("年齢を入力してください：");
String strAge = scanner.nextLine(); // 文字列の入力を受け付ける
int age = Integer.parseInt(strAge); // int型に変換する

System.out.print("身長を入力してください：");
String strHeight = scanner.nextLine(); // 文字列の入力を受け付ける
double height = Double.parseDouble(strHeight); // double型に変換する
```

練習

- 練習05-4