発展プログラミング演習II
1. Java言語の基礎 変数，式と演算子

コンピュータ理工学部

玉田春昭　水口充

Java言語の基礎

- 変数
- 文字列
- 式と演算子
- 文字列の比較
- コマンドライン引数

例題1.1 型の基本

- Java言語にはC言語と同じような型が存在する.
- 次のプログラムを入力，実行せよ.

```
public class Example0202{
  public static void main(String[] args){
    int math;
    math = 80;
    int phys = 75;

    System.out.println("数学の点数:" + math + "点");
    System.out.println("物理の点数:" + phys + "点");
  }
}
```

教科書p.14 例2.2

チェックリスト

- Java言語にはC言語と同じような型が存在する.
- 次のプログラムを入力，実行

●ファイル名は分かりますか?
●エラーメッセージは出ていませんか?
◆プログラム中の大文字小文字は合っていますか？
◆javacとjavaコマンドの使い方は間違っていませんか？

```
public class Example0202{
  public static void main(String[] args){
    int math;
    math = 80;
    int phys = 75;

    System.out.println("数学の点数:" + math + "点");
    System.out.println("物理の点数:" + phys + "点");
  }
}
```

教科書p.14 例2.2

変数
変数宣言

- 赤字部分
  - 変数宣言部
  - 型と名前で宣言する.
- 青字部分
  - 式.
  - 変数に代入する値を作成している部分.

- 宣言と代入は同時に行わなくても良い.

```
int math;
math = 80;
```

- 変数に一番最初に代入される値を初期値と呼ぶ.
- 全ての変数は型を持つ.

```
public class Example0202{
  public static void main(String[] args){
    int math;
    math = 80;
    int phys = 75;

    System.out.println("数学の点数:" + math + "点");
    System.out.println("物理の点数:" + phys + "点");
  }
}
```

教科書p.14 例2.2

変数型の種類

- 整数型: `byte`, `short`, `int`, `long`
  - それぞれ, 1バイト, 2バイト, 4バイト, 8バイト
  - すべて符号付き.
    - 符号なし(unsigned)型はない.
- 浮動小数点型: `float`, `double`
  - それぞれ4バイト, 8バイト
- 文字型: `char`
  - 2バイト.
    - Javaでは文字はすべてUnicodeで表されるため. Unicode は1文字1バイトとは限らない.
- 真偽値型: `boolean`
  - `true`, もしくは `false` のどちらかを表す型.

変数

基本データ型まとめ

分類	型名	ビット数（バイト数）	表現できる範囲	例
符号付整数型	byte	8 (1)	$(-2^7, 2^7-1)=(-128, 127)$	`byte v = 60;`
	short	16 (2)	$(-2^{15}, 2^{15}-1)$	`short s = 200;`
	int	32 (4)	$(-2^{31}, 2^{31}-1)$	`int i = 0;`
	long	64 (8)	$(-2^{63}, 2^{63}-1)$	`long v = 100L;` 数字の最後にLを付ける
文字（整数型）	char	16 (2)	Unicode の一文字 (0, 65535)	`char c = '漢';` 漢字や仮名もOK.
浮動小数点数型	float	32 (4)	IEEE 754 規格の 32 ビット浮動小数点数	`float v = 12.5f;` 数字の最後にFを付ける
	double	64 (8)	IEEE 754 規格の 64 ビット浮動小数点数	`double v = 2.5;`
論理値型	boolean	? (?)	true/false	`boolean v = true;`

例題1.2

- 以下のプログラムを入力，実行結果を確認せよ.

```
public class PrimitiveType{
  public static void main(String[] args){
    byte byteValue = 64;
    System.out.println(byteValue);
    short shortValue = 256;
    System.out.println(shortValue);
    int intValue = 1024;
    System.out.println(intValue);
    long longValue = 4092L;
    System.out.println(longValue);
    float floatValue = 10.4f;
    System.out.println(floatValue);
    double doubleValue = 20.6;
    System.out.println(doubleValue);
    char charValue = 'a';
    System.out.println(charValue);
    boolean booleanValue = true;
    System.out.println(booleanValue);
  }
}
```

変数
その他の型

- 先ほどのスライドに示した型は基本データ型，もしくはプリミティブ型と呼ばれる．
- それに対する型の種類は参照型もしくはオブジェクト型と呼ばれる．
  - 基本データ型以外の型は全て参照型となる．
  - クラスが型になる．
  - Java言語での変数の型はオブジェクト型が中心．
    - 標準APIとして標準で多くのクラスが用意されている．
    - 以下のプログラムの赤字部分(`String`)は型であり，`String`クラスの変数`stringInstance`を宣言している．
    - 型は，クラスを作成することで自由に作成できる．

```
String stringInstance;
stringInstance = new String("string");
```

参照型 (1/2)

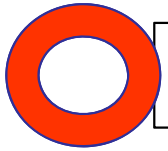
- 参照って．．．C言語でいうところのポインタ？
  - YESであり，NOでもある．
  - 参照型の変数が持つのはオブジェクトの実体へのポインタ（参照）．
    - 実体がどこにあるかは実行環境が決める．
  - C言語のようにポインタに対する演算は不可能．
    - Java言語ではポインタに+1して，次のアドレスにある値を参照することはできない．
- 参照型は数えきれないほど存在する．
  - ユーザが勝手に定義することもできる．
  - 型とオブジェクトは一致していなければならない．

```
String t = new Object();
```

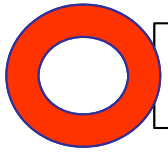
型とオブジェクトが一致していないため，コンパイルエラーが起こる．

参照型 (2/2)

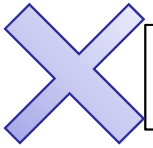
- どんな参照型の変数にも代入できる値がある.
  - null:どのオブジェクトも参照していないことを表すリテラル.



```
String stringInstance = null;
```



```
Integer integerInstance = null;
```



```
stringInstance = integerInstance
```

どちらもnullが代入されているからといって，
違う型同士では，代入できない.

変数まとめ

- 変数（variable）は「型　変数名;」の形式で宣言される.
- 変数に値を代入するには「変数名=式;」のように書かれる.
  - サンプルプログラム中の「`new String();`」についての説明は後日.
- 変数の宣言と代入を同時に行うには「型　変数名 = 式;」のように書く.

文字列

- Java言語では文字列はStringオブジェクトとして表される.
  - Stringクラスは標準で用意されている特別なクラス.
  - Stringという基本データ型があると思ってもあまり支障はない.
- 通常のオブジェクト作成には，new演算子を使うが，Stringはもっと簡単に作れる.
  - `String s = "Hello";`
  - 文字の前後を”(ダブルクォート)で囲むことで，Stringオブジェクトが生成される.
- `String s = "Hello";`
  - "Hello"を文字列リテラルと呼ぶ.
  - 文字列リテラルに”や¥を含めるにはエスケープ文字を使う.
    - `System.out.println("Hello ¥"Java¥"");`

例題1.3, 1.4

- 例題1.3 (StringLiteral)
  - 自分の名字を`String`型変数`s1`に，名前を`s2`に代入し，それぞれを`System.out.println`を使って出力するプログラムを書け.
- 例題1.4 (StringLiteral2)
  - `s1`を出力するメソッドを`System.out.println`から`System.out.print`に変更せよ.

演習1.5 コマンドライン引数

- 以下のプログラムを書き，実行せよ.
  - ただし，実行するとき，引数に何か文字列を与えよ.

```
public class Arguments{
  public static void main(String[] args){
    System.out.println(args[0]);
  }
}
```

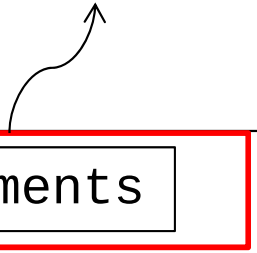
```
$ java Arguments SomeArguments
```

コマンドライン引数

- コマンドラインの引数はmainメソッドの引数の配列に格納される.
  - いくつでも指定可能.

```
public class Arguments{
  public static void main(String[] args){
    System.out.println(args[0]);
  }
}
```

```
$ java Arguments SomeArguments
```



文字列と基本データ型の変換

- 123と”123”の意味は異なる.
  - 123は整数リテラル.
  - “123”は文字列リテラル.

```
int i = 123;       // 正しい
int j = "123";     // 誤り
String p = 123;    // 誤り
String q = "123"; // 正しい
```

- 文字列からint型への変換.
  - `int i = Integer.parseInt("456");`
- 文字列からdouble型への変換
  - `double j = Double.parseDouble("32.7565");`
- int型から文字列への変換
  - `String s = Integer.toString(123);`
- double型から文字列への変換
  - `String s = Double.toString(74.142);`

演習1.6 文字列と基本データ型の相互変換

- 以下の処理を含むプログラムを作成せよ．クラス名は Conversion とする．
  - 文字列"0"をint型に変換し，変数zeroに代入する．
  - 文字列"3.14"をdouble型に変換し，変数piに代入する．
  - 変数zeroを画面に出力する．
  - 変数zeroを文字列に変換して，画面に出力する．
  - 変数piを画面に出力する．
  - 変数piを文字列に変換して，画面に出力する．

式と演算子

- 式
  - 値を計算するための要素.
  - 典型的な式は数値演算，条件判定式など
- 演算子
  - 各種の演算を行うための特別な意味を持った記号のこと.
  - 演算子は大きく以下の5つに分類できる.
    - 数値演算子，比較演算子，論理演算子，ビット演算子，代入演算子
  - 各演算子ごとに優先順序が決められている.

数値演算子

- 四則演算と剰余計算のための演算子.
- 型が異なる値同士は計算できない.
  - Javaの場合，自動的に型変換がなされる.
- 演算結果の型はオペランド（被演算子; 値の型）によって決定する.
  - 計算結果の型の自動変換ルール
    - 一方のオペランドがdoubleの場合，もう一方がdoubleに変換されて，計算結果もdouble型
    - そうではなく，一方のオペランドがfloatの場合，もう一方がfloatに変換されて，計算結果もfloat型
    - そうではなく，一方のオペランドがlongの場合，もう一方がlongに変換されて，計算結果もlong型
    - それ以外ならば，両方のオペランドがintに変換され，計算結果もint型

演算子	意味	例	例の演算結果
+	加算	1 + 4	5
-	減算	3 - 7	-4
*	乗算	3 * 2	6
/	徐算	5 / 3	1
%	剰余	5 % 3	2

比較演算子

- 大きさの比較や同値判定のための演算子.
- これらの演算はすべてboolean型の結果を返す.

演算子	意味	例	
==	等号	x == y	xとyが等しい値であればtrue, そうでなければfalse
!=	不等号	x != y	xとyが異なる値であればtrue, そうでなければfalse
<	未満	x < y	xがy未満であればtrue, そうでなければfalse
>	より大きい	x > y	xがyより大きい値であればtrue, そうでなければfalse
<=	以下	x <= y	xがy以下であればtrue, そうでなければfalse
>=	以上	x >= y	xがy以上であればtrue, そうでなければfalse

論理演算子

- boolean型の式を組み合わせるための演算子.
- AND, OR, NOT, XORが存在する.

式	意味	2つの違い
X && Y	XかつY	&&はXがfalseの場合，Yは評価されない．&はどんな場合でもXとYの両方が評価される.
X & Y	XかつY	
X || Y	XもしくはY	||はXがtrueの場合，Yは評価されない．|の場合は常にXとYの両方が評価される.
X | Y	XもしくはY	
!X	Xではない	
X ^ Y	X XOR Y	

ビット演算

- ビット演算をするための演算子.
  - 整数値を2進数として表し，ビットごとに演算を行うもの.

演算子	意味	例	例の演算結果
&	ビットごとのAND	10&4	0 (1010_2 & 100_2)
|	ビットごとのOR	10|3	11 (1010_2 & 11_2)
^	ビットごとのXOR	10^3	9 (1010_2 & 11_2)
~	ビットごとのNOT	~10	-11
>>	右シフト	10>>3	1
<<	左シフト	4<<1	8
>>>	右シフト(符号無視)	0xffffffff>>>16	0xffff

代入演算子

- 数値演算と代入を同時に行うための演算子.

式	意味
x += y	x = x + y;
x -= y	x = x - y;
x *= y	x = x * y;
x /= y	x = x / y;
x %= y	x = x % y;
x >>= y	x = x >> y;
x <<= y	x = x << y;
x >>>= y	x = x >>> y;
x &= y	x = x & y;
x |= y	x = x | y;
x ^= y	x = x ^ y;

その他の演算子

- インクリメント演算子
  - 「++」で表される演算子．「+1」の意味．
  - オペランドの前にある場合と後ろにある場合で意味が異なる．
    - 前にある場合，参照の前に+1される．後ろにある場合はすべての参照が終わったあと，+1される．
- デクリメント演算子
  - 「--」で表される演算子．「-1」の意味．
  - オペランドの前にある場合と後ろにある場合で意味が異なる．
    - 前にある場合，参照の前に-1される．後ろにある場合はすべての参照が終わったあと，-1される．
- 条件演算子
  - ほとんど使わない．
    - If-elseの方が理解し易い．
  - <条件>? <trueの場合の値>: <falseの場合の値>;

特殊な演算子

- 文字を連結する演算子「+」
  - +演算子は式中にString型があれば，すべてをString型に変換して連結し，一つの文字列を作る．

```
int x = 10;
double y = 20.4;
System.out.println("x = " + x + ", y = " + y);
```

  - 上の例だとx, yがどのような型であっても，文字列に変換され，以下のような出力になる．

```
x = 10, y = 20.4
```

例題1.7 文字列の比較

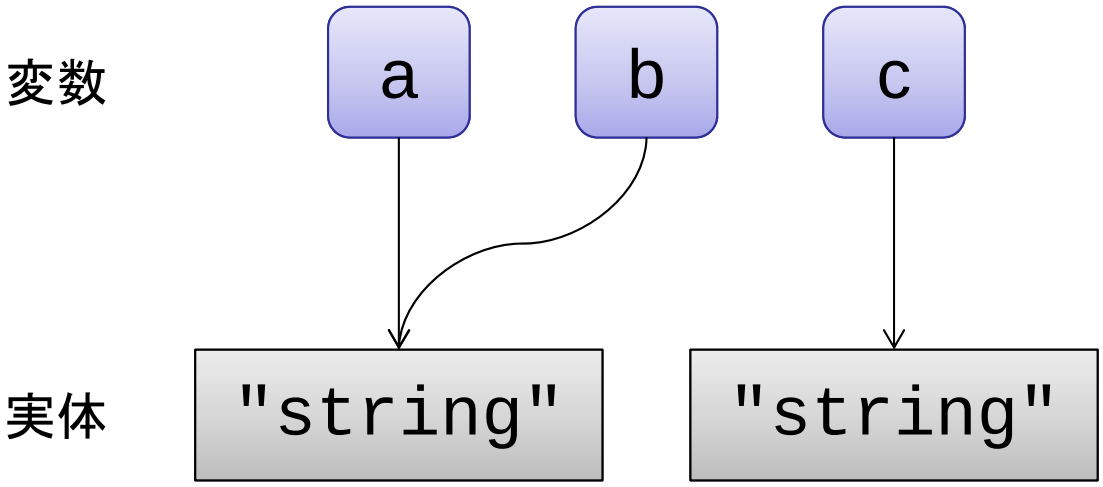
- 以下の処理を行うプログラム(StringCompare)を書き，実行せよ.
  - プログラムの内容
    - 初期値"string"のString型の変数aを宣言する.
    - String型の変数bを宣言し，変数aを代入する.
    - コマンドライン引数args[0]を初期値とするString型の変数cを宣言する.
  - 出力(出力例を右に示す)
    - 変数a，b，cを「a: aの内容, b: bの内容, c: cの内容」となるように出力せよ.
    - 変数を==で比較した結果を出力せよ. なお，比較はaとb, bとc, aとcの3回行うこと.
      - (a == b)
    - 変数をequalsで比較した結果を出力せよ. なお，比較はaとb, bとc, aとcの3回行うこと.
      - a.equals(b)

出力例

```
$ java StringCompare17 string
a: string, b: string, c: string
a == b: true
b == c: false
a == c: false
a equals b: true
b equals c: true
a equals c: true
```

文字列の一致性

- 文字列の内容の一致を確かめるために==は使えない.
- equalsメソッドを使って文字列が同じかどうかを確かめる.



a == b true

b == c false

a == c false

a equals b true

b equals c true

a equals c true

演算式の優先順位

- 各演算子は優先順位が指定されている.
- 右表の上にあるほど優先順位が高い.
  - すなわち，先に評価される.
- 同行にある演算子は式の左から順に処理される.

演算子	説明
., [], ()	. はクラスのフィールド，メソッドで使用. [] は配列で使用,
++, --, !, ~	インクリメント，デクリメント，NOT
instanceof	オブジェクトがあるクラスのインスタンスかどうかを判定する演算子
new	インスタンスを生成するための演算子
*, /, %	乗算，徐算，剰余
+, -	加算，減算.
<<, >>, <<<	シフト
<, >, <=, >=	大小比較
==, !=	同値，非同値
&	AND
^	XOR
|	OR
&&	式のAND
||	式のOR
?:	条件式
代入演算子	各種代入演算子

課題1

- 以下の処理を行うプログラムを作成し，Moodleに提出せよ.
  - コマンドライン引数から数値を2つ受け取り，加算，減算，乗算，除算，剰余を求め，その結果を出力するプログラム.
    - 出力結果には計算式も含めること.
    - 数値はdouble型で計算すること.
- 提出期限
  - 2012/10/01 16:30
- 提出先
  - Moodleの「課題1：変数，式と演算子」
- クラス名
  - Calculator_学生証番号
  - 学生証番号は6桁．頭にgは付けない.