Javaプログラマ育成コース
第二回　Java基礎

ソフトシンク株式会社
代表取締役
周　順彩
zhousc@soft-think.com

目次

- ゲームをやってみよう
- プログラムとは
- Javaとは？
- Javaのデータ型
- データ型の利用
- 配列
- 演算子
- 自動ボクシング
- 制御文－分岐
- 制御文－ループ
- 演習
- 練習課題
- 次回の予習タスク

ゲームをやってみよう

https://blockly-games.appspot.com/maze?lang=ja

プログラムとは？

◆ プログラムとは？

コンピュータに対する命令（処理）を記述したものである。

コンピュータは設計者の意図であるプログラムに従い、忠実に処理を行っている。

結局プログラムはコンピュータで人間の現実の問題を解決するために、いろんな**データ**を処理する**ロジック**のことである。

◆ どんなプログラム言語がある？

◆ プログラムの分類

- コンパイル方式

  ソースプログラムをいったん機械語に翻訳し，その機械語になったプログラムを実行する方式です。

- インタープリタ方式

  プログラミング言語の命令を一つずつ，機械語に解釈しながら実行する方式で，解釈実行方式ともよばれます。

Javaとは？

◆ Javaとは

プログラミング言語JavaおよびJavaプラットフォームは、1990年代前半当時、サン・マイクロシステムズに居たジェームズ・ゴスリン、ビル・ジョイなどの人々によって設計・開発された。2018年現在はサンを買収したOracleによる管理の他、追加提案などはサン時代から続いているJava Community Process (JCP) というプロセスによって進められる。

➢ Javaの実行方式は？

Java は基本的にはコンパイル方式の言語ですが，Java仮想マシンの機械語に翻訳され，仮想マシン上で実行されます。コンパイル方式とインタープリタ方式の中間的な方式といってよいでしょう。

また、この方式によって、Java仮想マシンがインストールされているコンピュータでは、どんなOSでもJavaプログラマを実行することができる。「Write once, run anywhere」

Javaのデータ型

◆ プリミティブ型(primitive type)

Javaプログラミング言語によってあらかじめ定義された型だ。予約されたキーワードによってデータ型ごとに型の名前が付けられている。

◆ 参照型(reference type)

値そのものを保持するのではなく、値が置いてある場所を保持する。

参照型には、クラス型、インターフェイス型、型変数、配列型がある。

プリミティブ型		範囲	参照型 (Wrapperクラス)
整数型	byte(8ビット)	-128～127	Byte
	char(16ビット)	'¥u0000' ～ '¥uffff'	Char
	short(16ビット)	-32768 ～32767	Short
	int(32ビット)	-2147483648 ～2147483647	Integer
	long(64ビット)	-9223372036854775808～9223372036854775807	Long
浮動小数点型	float	32ビット単精度小数点	Float
	double	64ビット倍精度小数点	Double
論理型	boolean	真(true)か偽(false)	Boolean

データ型の利用

◆ 変数定義と代入

```
byte b = 127;
char c = 'A', c1 = '¥u003d';
short s = 61;
int i = 0;
long l = 0l;
float f = 0.1f;
double d = 0.1d;
boolean bl = false;
i = l;      // compile error(type mismatch)
l = i;      // ok
i = bl;     // compile error
l = f;      // compile error
f = l;      // ok
i = (int)l;     // cast(精度が落ちる可能性あり)
l = (long)f;

String  str     =     "abc";
型      変数名  代入  初期値
```

配列

◆ 配列とは

同じ型の複数の値をまとめて一つの変数として扱うことができるもの。

◆ 配列の定義と使用

型名[] 配列変数名;　型名 配列変数名[];

```
int[] intAry = {0, 1, 2};
int[] intAry1 = new int[3];
int intFirst = intAry[0];   // 配列参照、0から
int intLast = intAry1[2];
int len = intAry.length;    // 要素数の取得：3
String[] strAry = {"0", "1", "2"};        // 参照型の配列
String[] strAry1 = new String[3];
System.arraycopy(intAry1, 0, intAry1, 0, 3);    // 配列コピー
int[][] intAry2 = new int[2][3];          // 2次元配列
int intAry3[][] = new int[2][3];
```

演算子

種類	演算子	結合	使用例
一次式	()（優先する式）		(1 + 2) * 3
	new 新規生成	右	new int[3]
	[](配列要素参照)	左	intAry[0]
	（メンバー参照）	左	obj.value
インクリメント・デクリメント	++、--（前置き） ++、--（後置き）	右 左	--j（前置き） i++（後置き）
四則演算	*、/、% +、-	左	1+2*3
ビット演算子（シフト）	>>、<<(符号ビット保持) >>>(符号ビット保持せず、0代入)	左	8 >> 1 // 4
関係演算	>、<、>=、<= ==、!=、	左	1 == 2、3>4
ビット演算子（論理）	&（AND）、|（OR）、^（XOR）、~（NOT）	左	1 | 2 = 3
論理演算	! &&、||	右 左	Integer
代入演算子	=、+=、-=、*=、/=、%= &=、|=、^=、~= <<=、>>=、>>>=	右	i += 1; // i = i+ 1
3項論理演算	?:		x > y ? x :y

自動ボクシング

◆ 自動ボクシングとは？

JDK1.5からの仕様で、プリミティブ型とラッパークラスを互いに直接代入すること。
これを自動ボクシング（オートボクシング：AutoBoxing）と呼ぶ。

```
Integer i = 123;        //オートボクシング（ボクシング変換：boxing conversion）
int j = i;  //オートアンボクシング（アンボクシング変換：unboxing conversion）
//実態はコンパイルで、変換メソッドに置き換えられているだけ
Integer i = Integer.valueOf(123);
int j = i.intValue();

Object obj = 123;       //ok（実体はIntegerのインスタンス）
int i = obj;            //compile error
int i = (Integer)obj;   //ok
```

制御文ー分岐

◆ if/else if/else

```
if（論理演算式）{
    // logic
} else if（論理演算式）{
    // logic
} else {
    // logic
}
```

◆ switch

```
switch(c) {
case 'A':
    // logic
case 'B':
    // logic
    break;
default:
    // logic
    break;
}
```

もう少し考えてみよう
①break句のありなしで、実行はどうなるか？
②Switchのdefault句が一番上に記述した場合、実行はどうなるか？

制御文ーループ

- for

```
for (int i = 0; i < 7; i++) {
    // logic
}
```

- while/do while

```
while (i > 0) {
    i--;
    // logic
}
do {
    --i;
    // logic
} while (i > 0);
```

- for each

```
for (int i : intAry) {
    // logic
}
```

演習

◆ ソート

10個のint要素の配列を昇順(小さいから大きい順)でソートする。

練習課題

◆ 加算器

➢ 任意の加算の文字列を解析し、加算の結果を出力

```
String expression = “123+345”;
// 上記文字列を足し算の式として結果を求める
System.out.println(expression + “=“ + 上記の結果);
// ヒント①：下記の句によって文字列中のすべての文字を1文字ずつ取得できる
   for (char c : expression.toCharArray()) {
// ヒント②：分岐処理によって、一つの文字がどの数字か判断して数値変換を行う。
```

次回の予習タスク

- オブジェクト指向とJava