Java講座

～第1回～

情報科学部コンピュータ科学科
2年　竹中　優

今回の内容

- プログラムを書く上で…
- Hello world
- 基礎事項
- 演算子
- 構文

プログラムを書く上で…

- コメントアウト(//, /*　*/, /**　*/)をしよう！
- インデントをしよう！
- 変数などにはわかりやすい名前をつけよう！

要するに、
他人が見て理解しやすいコードを書こうということです。

一番基本的なプログラム(準備)

1. Eclipseを起動
2. 「ファイル」→「新規」→「javaプロジェクト」
3. プロジェクト名を入力し,「完了」
4. パッケージ・エクスプローラーで確認する
5. 作成したプロジェクト内の「src」フォルダで「新規」→「パッケージ」
6. パッケージ名を入力し,「完了」
   (ここでは,「java_lec01」と入力する)
7. 作成されたパッケージ内で「新規」→「クラス」
8. クラス名を入力し,「完了」
   (ここでは,「First」と入力する)

一番基本的なプログラム(コンソール)

```
package java_lec01;

public class First{

	public static void main(String[] args){
		new First().start();
	}
	void start(){
		System.out.println(“Hello world!”);
	}

}
```

一番基本的なプログラム(ダイアログ)

```
package java_lec01;

import javax.swing.JOptionPane;

public class First{

	public static void main(String[] args){
		new First().start();
	}
	void start(){
		JOptionPane.showMessageDialog(null,
							“Hello world!”);
	}
}
```

mainメソッド

- 実行されたクラスのmainメソッドから一番最初の処理が始まる。
  したがって、前項のようにしなくてもmain内に直接、処理を記述しても良い。
- しかし、慣れるまでは前項のFirstクラスのようにstartメソッドを定義することをオススメする。

基礎事項：入出力

<インポート文>
import javax.swing.JOptionPane;

<コード>
//ダイアログから入力(上記インポートが必要)
String input = JOptionPane.showInputDialog(“入力してください”);

//ダイアログ出力(上記インポートが必要)
JOptionPane.showMessageDialog(null, input);

//コンソール出力(インポートはいらない)
System.out.println(“文字列”);//出力後改行
System.out.print(“文字列”);//出力後の改行なし

配列

```
String[] arrayStr = new String[10];
int[] arrayInt = new int[5];
double arrayDbl = new double[1];
//配列の型[]  配列名 = new 配列の型[配列の長さ];
```

- 配列の長さは0以上を指定しなければならない。
- 最初の[]は、型の後でも配列名の後でも構わない。
  (例)int arrayInt[] = new int[5];
- 配列の長さを取得したい場合は、配列名.lengthを用いる。
  (例)int length = arrayInt.length;//5

配列

長さ10の整数型配列に1から10までの数字を入れたいけど、配列作って代入するのめんどくさい、とかそんな時。

```
int[] arrayInt = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
double[] arrayDbl = {0.0, 0.1, 0.2, };//長さ3になる
String[] arrayStr = {“s1”, “s2”, ””};
配列の型[] 配列名 = {値1(,値2,値3…)}
```

2次元配列

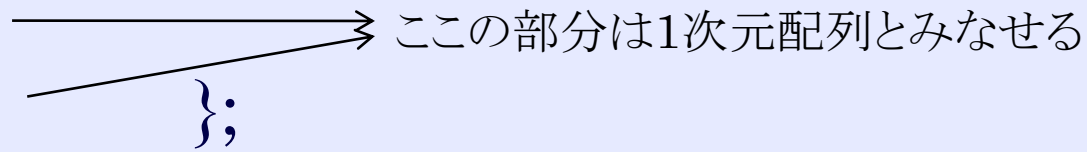
```
int[][] arrayInt2 = new int[2][5];
配列の型[][]  配列名
    = new int[配列の縦の長さ][配列の横の長さ];
```

2次元配列とは、配列の配列。

例えば、int[][] arrayInt2 = {

{1,2,3,4,5},

{6,7,8,9,10} };



は整数型配列の配列となる。

- 上記の場合、arrayInt2.lengthはどうだろうか?

基礎事項：基本型

- int
  - 整数値
- double
  - 小数値
- String
  - 0文字以上の文字列またはnull
- boolean
  - trueかfalseの値(C言語とは異なり数値ではない)

- long
  - 整数値
- float
  - 小数値
- char
  - 1文字分の文字

String型について

C言語とは異なるため、特記する。

- “”で囲んだ部分は文字列として処理される。
- 変数の宣言の仕方は、
  String str = “文字列”;

- 文字数を気にする必要はない。
- 結構、融通が利く。
- 値を変更したい場合、
  - str = “文字列2”;
  - str = str + “文字列を付加”;
  - str = str + 100 + “数値を付加”;

演算子

記号	説明	記号	説明
+	加算(文字列連結)	>	より大きい
-	減算	>=	以上
*	乗算	<	未満
/	除算(商)	<=	以下
%	除算(余り)	==	等価
++	インクリメント(+1)	!=	非等価
--	デクリメント(-1)	!	論理否定(反転)
&&	論理積(かつ)	new	オブジェクト生成
||	論理和(または)	instanceof	型比較

他にもいっぱいあります…

インクリメントとデクリメント

++(--)は前置する場合と後置する場合がある。
例えば、
int a = 0;
int b = a++;//後置インクリメント
と
int a = 0;
int b = ++a;//前置インクリメント
ではbの値は異なる。上ではb=0、下ではb=1となる。
つまり前置は「+1してから代入」、後置は「代入してから+1」という処理になっている。

複合演算子

int a = 10;

a = a + 5;

という処理をしたいとすると、2行目は

a += 5;

のように記述できる。

-, *, /, %でも同様に記述できる。

この「+=」や「-=」などを複合演算子という。

構文

- if
  - 条件分岐(記述しやすい)
- switch
  - 条件分岐(見やすい、速いらしい)
- for
  - 繰り返し
- while
  - 繰り返し

if文

```
if(条件式1){
    処理;
}
else if(条件式2){
    処理;
}
else{
    処理;
}
```

- else if文は0個以上記述できる.
- else文は省略可.
- 条件式はboolean型であれば処理できる.
  極端に言えば,
  if(true){処理;}でも良い.
- {}は処理部分が1つの文の場合,省略可.

サンプルコード:Sample_if.java

switch文

```
switch(整数値){
case 値1:
    処理;
    break;
case 値2:
    処理;
    break;
    ⋮
default:
    処理;
    break;
}
```

- 整数値, 値1, 値2にはint型, char型などが入れられる.
- case, default文は省略可.
- 「break;」は省略可. 省略した場合, その1つ下のcase(またはdefault)文の処理が行われる.

サンプルコード：Sample_switch01.java

switch文：「break;」を省略すると?

break文は、「文の流れを強制的に切る」という役割を持っている。

したがって、switch文ではbreak文が出てくるか、ブロックが終了するまでブロックの中の文が順次処理される。

サンプルコード：Sample_switch02.java

for文

```
for(初期化式;条件式; 1
    ループ毎の処理;){
    処理;
}

わかりにくいので、
for(int i=0;
    i<繰り返し回数;i++){
    処理;
}
```

- 初期化→条件判定がtrue→処理→ループ毎の処理→条件判定→…
- 初期化式,条件式,ループ毎の処理はそれぞれ省略可. 例えば, 全て省略すると
  for(;;){処理;}
  となり処理部分で何かしない限り無限ループすることになる.
- {}は処理部分が1つの文の場合, 省略可.

サンプルコード:Sample_for01.java

サンプルコード:Sample_for02.java

while文

```
while(条件式){
    処理;
}
```

- 基本的な流れはfor文と同様.
- 条件式は省略不可
- 無限ループしないように処理を記述しよう.
- {}は処理部分が1つの文の場合，省略可.

サンプルコード：Sample_while01.java

do-while文

```
do{
    処理;
}while(条件式);
```

- while文と異なるのは必ず1回は処理行われることだけ.
  例えば,
  ```
  while(false){
      処理;
  }
  ```
  では処理は行われないが,
  ```
  do{
      処理;
  }while(false);
  ```
  は1回処理が行われる.
- {}は処理部分が1つの文の場合,省略できるがしないようにしよう.

サンプルコード:Sample_do_while.java

continue

continue文はfor, while文の中で使われるbreak文と対を成すような文である。

ループ内でcontinue文が処理されるとそれ以降の処理をスキップして、次のループに処理を移す。

このときループを続けるかの判定はされるのでforまたはwhile文の条件がfalseの場合はループしない。

サンプルコード：Sample_for03.java

問題1

- ダイアログから整数を入力させ、その値から0までカウントダウンするプログラム。

  例えば、5が入力された時は
  5
  4
  3
  2
  1
  0
  とコンソールに出力する。
  いくつも方法があるので、色々考えてみよう。

問題2

- 任意の2次元整数型配列numbersを宣言し、その中身をfor文を使って、コンソールに表示するプログラム。
  ただし、改行するべき箇所に改行を入れる。

  例えば、int[][] numbers = {
  {1, 2, 3, 4, 5},
  {6, 7, 8, 9, 10},
  {11, 12, 13, 14, 15}
  };
  の場合は、
  1 2 3 4 5
  6 7 8 9 10
  11 12 13 14 15
  という感じ。

終わり