プログラミングII

黒瀬　浩

kurose@neptune.kanazawa-it.ac.jp

OH: 講義の前後，Eメール問合せ，月3限21-405

居室　67・121

環境変数Pathの設定でうまくいかない場合

- Windowsでプログラムがどこにあるか環境変数PATHを見て探す
- コマンドプロンプトで　echo %PATH%↵　で確認できる
- 環境変数の設定

ユーザの環境変数 Path　に設定してもうまくいかない場合

　　スペルミスを疑う

　　コマンドプロンプト再起動したか確認する

それでもだめなら

ファイルエクスプローラ上部を押すと完全進路名が出るのでコピーして貼り付け

それでもだめなら

システム環境変数の設定　を起動して　システム側のPathに追加して，位置を上にする

ユーザーの環境変数よりシステムの環境変数が優先される

位置は上の方が優先される

Java関連のアプリケーションが邪魔している場合があり，それより優先して起動できるようにすれば良い

日本語表示が文字化けする場合

ソースファイルを utf-8　で作成していることを前提とする

まずコマンドプロンプトで確認

chcp 65001↵　の後に実行してみる

VSCodeの場合

設定の検索で terminal.integrated.sh で検索
Terminal > Integrated > Shell > Windowsを
C:¥Windows¥System32¥cmd.exe にする

その下の方の　Terminal > Integrated > Shell Args: Windowsで
setting.jsonで編集を押す　左画面の鉛筆アイコンを押す
右画面の　[ ]の中に　"/k", "chcp 65001" を追加して，再起動

それでも直らなければ　文字列に英字を使う

直したければ　IntelliJ IDEA CEを入れる(次ページ)

IntelliJ IDEA CE

- ダウンロードしてインストール
- Pleiades 日本語化プラグイン　で検索してダウンロード
  - zipファイルを「すべて展開」して，中のsetup.exeを実行
  - 日本語化するアプリを聞かれるので，IntelliJ IDEAの場所を指定
    C:¥Program Files¥JetBrains¥Intellij IDEA....¥
- IntelliJ IDEA起動後の設定
  - 設定で，　proxy　を検索し，自動プロキシー構成で
    http://www.kanazawa-it.ac.jp/proxy.pac を指定
  - あとは，キーマップ変更，プラグイン追加，カラーテーマ変更等お好きに

- IntelliJ IDEAでのプログラム作成
  - ファイル -> 新規 -> プロジェクト -> Javaモジュール　で
    プロジェクト名を入れる
  - できたプロジェクトの　src を右クリックし　新規->Javaクラス　で
    ファイル名(=クラス名)を入れる
    エディタが開くのでソースを入れて保存
    初回の実行だけ，実行構成が必要（ソース左の実行ボタンを押す）
    一度実行構成を設定すれば，実行ボタンかSHIFT-F10でコンパイル・実行ができる

Java復習

ファイル名は主クラス名と合わせる
ソースの拡張子は　.java
コンパイル　javac クラス名.java↵　エラーがなければ実行　java クラス名↵
クラスの中にデータ(フィールド)や関数(メソッド)を登録する

```
public class Hello { // クラス
    public static void main(String[] args){ // メソッド
        System.out.println("Hello");      // メソッドの中身
    }
}
```

変数は型宣言が必要　`int a;`
文末は ; が必要
主な型 `byte, short, int, long, double, float, boolean, char`
関数も型宣言が必要　値を戻さなければ `void`型
文字（シングルクォート）と文字列（ダブルクォート）は別物
演算は容量が大きい型に合わせられる　文字列+数値　は文字列の結合
型変換はキャストを使う　　`double a = (double)1;`

教科書の例　02-03

```
public class PrintExample3 {
     public static void main(String[] args){
         System.out.print("こんにちは");
         System.out.print("今日も良い天気です");
     }
}
```

- PrintExample　　こんにちは↵ を出力
- PrintExample2　　もう一行追加
- PrintExample3　　println() を print()に変更　print()は最後に↵をつけない

ここでは，ソースコードが少ししか違わないので名前を付けて保存で
新しいファイルを作る（クラス名も合わせることに注意する）

クラス名が同じで内容が異なる場合はディレクトリ（フォルダ）を分ける
（IDEでは，別のプロジェクトを作る）

教科書　02-05

```
public class VariableExample2 {
    public static void main(String[] args){
        int i;
        i = 5;
        System.out.print(i);          // 5↵を出力
        i = 10;
        System.out.print(i);          // 10↵を出力
    }
}
```

変数は使う前に型を宣言する必要がある
型宣言と初期値代入は同時にできる　　`int i = 5;`

教科書 02-09

```
public class CalcExample3 {
    public static void main(String[] args){
        int i;              // iの値は未定
        i = 11;             // iは11
        i++;                // i = i + 1 で12    i += 1 でも同じ
        i /= 2;             // 12 / 2 で 6
        System.out.println("iの値は" + i);  // 文字列連結

        int j;              // 変数は宣言した後で使える
        j = i * i;          // 6 * 6
        System.out.println("jの値は" + j);
    }
}
```

pythonのprint関数の引数はいくつでもよかったがprintln()は違う

教科書 02-12

```
public class CastExample {
    public static void main(String[] args){
        int a = 5;
        int b = 2;
        double c = (double)a / (double)b;
        System.out.println("cの値は" + c);
    }
}
```

pythonには型変換関数 int(), str()などがあった
javaでは型変換にキャストを使う　変換後変数 = (型)返還前の変数や定数

型の異なる演算は容量が大きい方に変換される
整数同士の除算は型変換が起きないため，小数点以下は切り捨てられる
このサンプルはそれを避けるためにキャストで倍精度実数演算を行っている

使う側と作る側

プログラムの出力は字の羅列を人間が判断する
10/2　という印字があったら
　　10÷2の数式を意味するのか
　　10月2日なのか　(欧米では日/月と書く国があるので2月10日)

プログラマは使う側と作る側の両方の立場が必要
　　ソースコードでは　man は英単語ではなく　変数
　　manという字を持つには　　"man" のように文字列定数にする
　　人間が読めるように出力するには　　System.out.println("man");

変数は値を覚えておくもの（後で使いたいから）

関数内で値を印字せずreturnさせるのは，印字形式は呼び出し側にまかせ
決まった機能を提供することに徹するため

条件分岐，制御構文については後でまた学習しますが
pythonと似ている部分もあるので
とりあえず概観します.

条件分岐(conditional branch)

```
// 真の場合のみ
if( 条件式 ){ 真の場合 }
// 真と偽の場合
if( 条件式 ){ 真の場合 } else { 偽の場合 }
// 複数の条件の場合
if( 条件式1 ){ 式1が真の場合 }
else if( 条件式2 ){ 式2が真の場合 }
...
else if( 条件式n ){ 式nが真の場合 }
else { いずれも偽の場合 }
```

条件式は最終的にtrueかfalseの真偽値(boolean)の結果となる

{ }を使わなくても1文書けるが慣れないうちはやめた方が良い

入れ子構造とインデントのレベルは合わせること(pythonと違い，あっていなくてもエラーとならない)

pythonと異なり条件は ( )が必須

pythonのelifはないので else if(条件) で書く

場合分け　switch case

```
switch (式){
        case 値1:
                        式の結果が値1の時に実行する文
                        break;
        case 値2:
                        式の結果が値2の時に実行する文
                        break;
        case 値n:
                        式の結果が値nの時に実行する文
                        break;
        default:
                        いずれでもないときに実行する文
}
```

breakを入れ忘れると次の場合の文も実行してしまう

pythonにはswitch文はないので if ... elif ... else で書く

繰返し(loop)

```
for (ループ実行前に行う文; 繰返し判定条件式;  ループに戻る時に行う文){
        ループ内で実行する文
}

while(ループ実行判定条件){
        ループ内で行う文
}

do {
        ループ内で行う文
} while(繰り返し判定条件);
```

pythonのfor文に相当する拡張for文があるが省略

whileループは1回も実施しない場合があるが
do whileでは無い(判定をループの終わりに行うため)

pythonではdo whileを無限ループで書く

continue　　ループの最初に戻る(continueの後ろを飛ばす)
break　　ループを脱出する

無限ループ　　while(true)　　ループ内でbreakさせれば抜けられる

変数のscope 同じ変数名を複数箇所で宣言する場合の注意

ブロック内で宣言した変数はその内部で有効

```
while (式) {
        int a=10;
}
// ここでは変数aは参照・代入できない
```

同じ変数名の場合は内側が有効

```
int a=2;
while (式){
        int a=3;
        …          //   ここではaは3
}
// ここでaを参照すると？
```

4章　メソッド（クラス内の関数）

関数

何かを渡す（引数），特定のことをやる，何か返す（戻り値)
渡すものや返すものはなくても良い
pythonの関数 len()
文字列，リスト，タプルなど様々なものを渡せた
長さ（要素数）を整数値で返した

メソッド

特定のものに作用する関数
pythonの ",".join(["1", "2","3"]) や　"abc".upper() は文字列に作用するので，文字列定数か文字列変数が前に来る

戻り値 = 対象.メソッド名(メソッドの引数)
対象は，オブジェクト指向言語でオブジェクトと呼ばれる
クラスもオブジェクトのひとつ

メソッド

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

クラスHello内に 戻り値なしmain()関数がある(定義されている)
main関数内に System.out.println()を呼んでいる

java Hello↵ を実行すると　Helloクラスのmain()を呼び出す
そのため，ファイル名とクラス名は合わせておく必要がある

クラス内には別のメソッドも定義できる

教科書 04-01

```
public class CallMethodExamle {
    public static void countdown(){
        System.out.println("カウントを出します");
        for(int i = 5; i>=0; i--){   // iが5,4,3,2,1,0となる
            System.out.println(i);
        }
    }
    public static void main(String[] args){
        countdown();     // メソッド呼び出し
    }
}
```

まずmainが動く，countdown()を呼ぶ

countdownは引数なし，戻り値なし(void)

for文は　for(ループ前の文; 実施条件; ループに戻る前実行する文){
　　ループ内で実行する文
}

メソッドの記述位置

pythonでは上から順に実行していたので最初にやりたい関数を後に書くには main関数を作って，関数定義の後にmain()を呼び出した

Javaはコンパイラ言語なので，メソッドの順番はスクリプト言語の注意はない

実行時にメインクラス=ファイル名のmain()から実行する

```
public class Ex1 {
	public static void methd1( ) { 定義 }
	public static void methd2( ) { 定義 }
	public static void main(String[] args){
		method1() // 呼び出し
		method2() // 呼び出し
	}
}
```

でmethod1, methd2, mainはどの順に定義しても良い

メソッドの引数と戻り値

関数と同様に引数を渡せる

引数は，型を意識する必要がある

```
public static void method1(int i){   定義 }
```

method1を呼び出すには整数型の定数，変数，式が必要

関数と同様に戻り値が返せる

戻り値は，型を意識する必要がある

```
public static int method2(){ 定義; return 整数値 }
```

method1の結果は整数型の変数で受ける必要がある

演習　以下のメソッドを作り動作を確認せよ

method1	整数型の値の2倍を返す
method2	整数型の値2つを指定すると合計を返す
method3	実数型の値2つを指定すると大きい方を返す
method4	文字列と数を指定すると文字列を繰り返して返す

演習

前ページができた人は

　引数1 整数，引数2 整数　引数1の引数2乗を返すメソッドを作れ

　ヒント：　初期値を1にした値に，for文で引数2回分の乗算を行う

　上記ができたら引数1を実数にしたメソッドも追加せよ

教科書4章の例題を確認する