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

Javaプログラミングの注意

ファイル拡張子は　.java にする
クラス名はファイル名と同じにする
クラス名は大文字で始める
変数，メソッドは小文字で始める
クラス内にはmain()メソッドが必要 (importされる場合は異なる)
実行したクラスのmain()メソッドから動く
文の行末には；が必要
漢字(全角文字）が使えるのは，文字列内かコメント内
クラス定義，メインメソッド定義は決まり文句なので使い回す

コンパイル　　javac クラス名.java　　(クラス名.classができる)
実行　　java クラス名
エラーは
コンパイルエラー（文法的なもの）と
ランタイムエラー（実行時にこれ以上動作できなくなって強制終了される）がある．エラーが出たらソースコードを修正してやり直す

Javaのプログラム

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，メソッドは型宣言が必要　`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;`

メソッド(クラス内の関数 javaの関数は全てメソッド)

```
public class Ex1 {
    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
    }
}
```

このクラスには，coundtDown(), main()のメソッドがある

通常　クラス.メソッド（引数) で呼ぶが，自分のクラスのメソッドを呼ぶ場合，

メソッド(引数) で呼べる（関数と同じ）

メソッドは戻り値の型を指定する（戻り値がない場合は void)

メソッドの引数と戻り値，オーバーロード

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

オーバーロード　メソッド名が同じで複数の処理を定義できる

引数の並び（型，数）が異なる場合に使える

```
public static void method1(int i){ 定義 }
public static void method1(double i){ 定義 } // OK
public static int  method1(int i){ 定義 }  // NG
```

キーボード入力(P116)

```
import java.util.Scanner;        // Scannerクラスが必要
クラス定義 {
        メソッド定義 {
                Scanner in = new Scanner(System.in);
                        // 読み込む前に1度必要
                int i = in.nextInt();              // 整数として読む
                double d = in.nextDouble();        // 実数として読む
                String s = in.next();              // 文字列として読む
        }
}
```

キーボード(標準)入力には
Scanner型(正確にはクラス)の変数 in を生成(new)する(ファイルのopenに近い)
inは生成されたものを入れる名前なので inでなくても良い
System.in　は標準入力(通常キーボード)を表す
Scanner型で生成されたものに対してnextIntなどのメソッドを呼ぶ
繰り返し入力する場合は，newは1度行えば良く毎回newする必要は無い
pythonと違いプロンプト(入力督促文字)は出ないのでprintln()で表示する

条件分岐(P59～)

if文

if(条件式) { 真の場合 }

if(条件式) { 真の場合 } else { 偽の場合 }

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} ...

条件式

結果がboolean(trueかfalse)となるもの

比較演算子　　== != < <= > >=

論理演算子　　&& || ! ^

変数，定数，メソッド呼出し　を組み合わせられる

pythonとの比較

真理値定数	true, false	vs. True, False
論理演算子	&& || ! ^	vs. and or not (XORビット演算)
elseif	else if( )	vs. elif
条件式の括弧	必要	vs. なくても可

switch case文(場合分け)

- if else if .... でも書ける
- 月の終わりの日を得る(閏年を除く)

```
int m=10, n=0;
switch(m){
    case  1: n=31; break;
    case  2: n=28; break;
    case  3: n=31; break;
    case  4: n=30; break;
    case  5: n=31; break;
    case  6: n=30; break;
    case  7: n=31; break;
    case  8: n=31; break;
    case  9: n=30; break;
    case 10: n=31; break;
    case 11: n=30; break;
    case 12: n=31; break;
    default: n=-1;
}
System.out.println(m+" "+n);
```

同様の処理

```
if( m==2 ){
   n=28;
} else if( m==2 || m==4 || m==5 ||
           m==7 || m==9 || m==11 ){
   n=30;
} else if(n>=1){
   n=31;
} else {
   n=-1;
}
```

switch～case文では，breakを入れ忘れないように注意すること

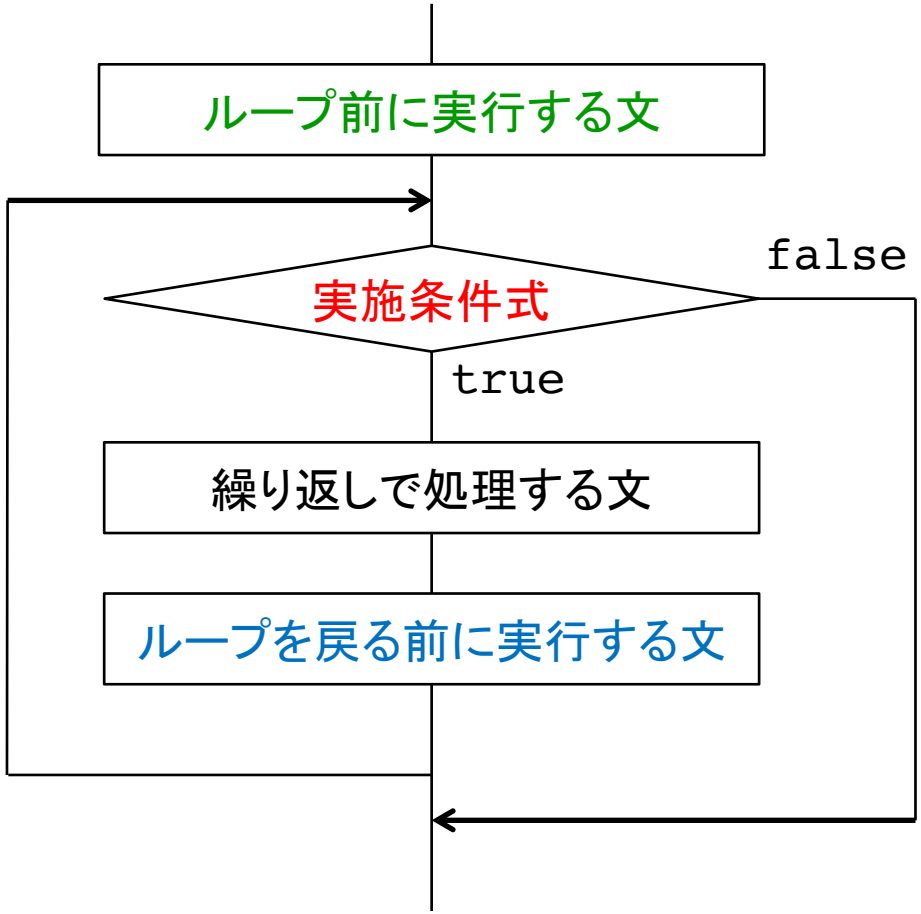
閏年判定　演習

判定方法はいくつかあるが以下をJavaで実装すること
論理式を整理しても良い

西暦年を入力する
年が4で割り切れる
　　年が100で割り切れる
　　　　年が400で割り切れる
　　　　　　閏年
　　　　年が400で割り切れない
　　　　　　閏年でない
　　年が100で割り切れない
　　　　閏年
年が4で割り切れない
　　閏年でない
入力値は，1900,2000, 2018, 2019,2020で確認すること

繰り返し

```
for( ループ前に実行する文 ; 実施条件式; ループを戻る前に実行する文){
        繰り返しで処理する文
}
```

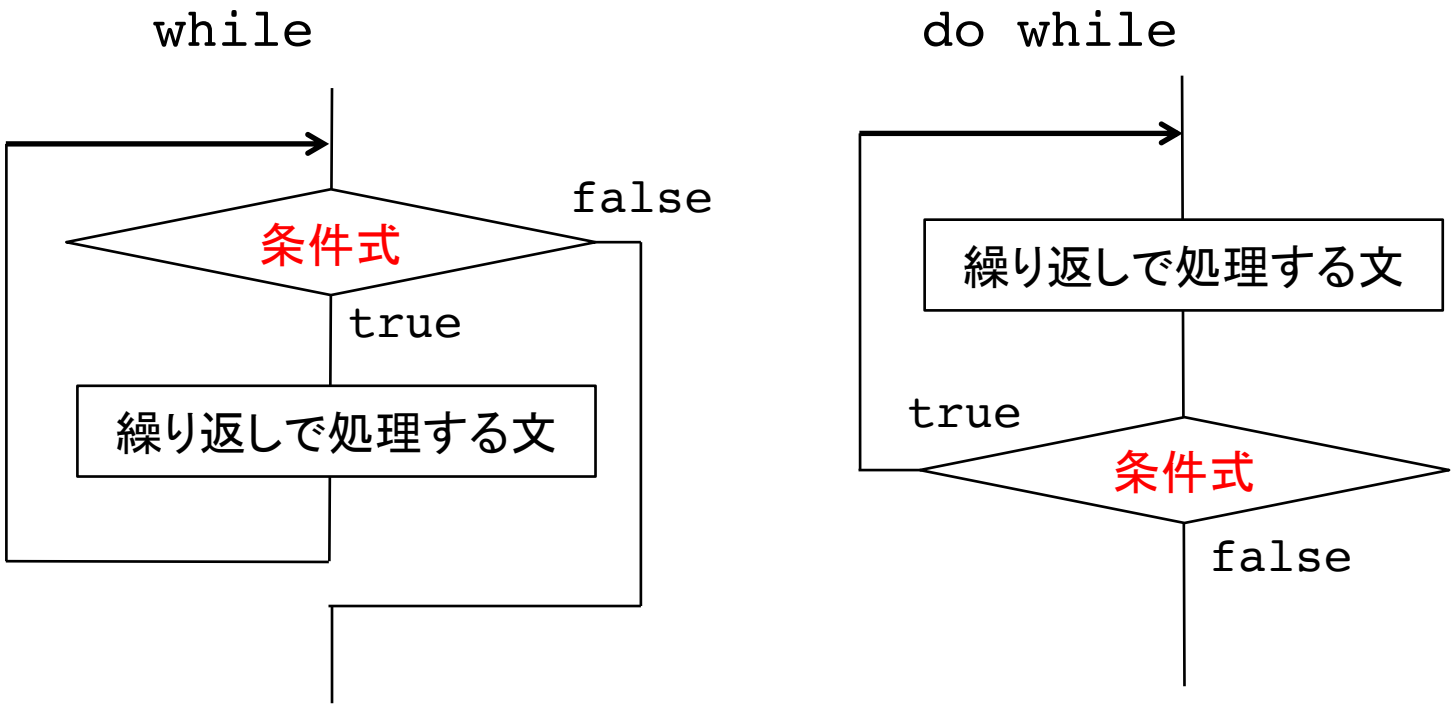


ループ前に実行する文と
ループを戻る前に実行する文は
複数の文が書けるが
区切りは　;ではなく　,

```
for(int i=0, j=0;
     i<10;
     i++, j++){
         繰り返しの処理
}
```

繰り返し　while

```
while( 条件式 ) {　繰り返しで処理する文　}
do{ 繰り返しで処理する文 } while(条件式);
```



変数の初期化が必要ならループの前にやっておく

do whileでは少なくとも1回はループ無いを実行する

繰り返し演習

正数値をキーボードから読む
1から整数値までの総和と総積(階乗)を求める処理を
for文を用いて書け

次に，while文で行うプログラムと，do whileで行うプログラムも作れ

注: 大き過ぎる整数値は総積でオーバーフローを起こす

演習　素数を求めよ

整数値をキーボードから読む

2から整数値までのループ（変数i)

　　2からiまでのループ

　　　　i-1までに割り切れたらループ脱出

　　　　i-1まで割り切れなかったら値を表示

ループを脱出するには以下いずれかを使う

　　break　　一番内側のループを抜ける

　　continue　　以下を飛ばして次のループを行う

　　while(){ }, do{ }while()を使う

配列(dimension, array)

同じ型の値を連続的に管理する変数

型[] 変数名;
変数名　= new 型[要素数];
型[] 変数名　= new 型[要素数]; 上の2行をまとめた形

参照、代入するときは　変数名[添字]　を使う
添字は0からの連続

初期化する場合

型[] 変数名 = {値1, 値2, …}　　要素数は値の並びによる

２次元配列

型[][] 変数名 = new 型[要素数][要素数];

要素数を得る　変数名.length

サンプルプログラムの　man(String[] args) の意味は既にわかるはず

コマンド引数から値を読む

実行時 java プログラム名　引数1 引数2 ...

```
public static void main(String[] args){
        System.out.println( args.length );   // 引数の数
        for(int i=0; i<args.length; i++){
                int j = Integer.parseInt( args[i] );
                System.out.println( j );
        }
}
```

コマンド引数は main()の引数で指定した変数に入る(上記ではargs)
何個指定されたかは　.length でわかる
文字列で入るので，整数化したければ Integer.parseInt() を使う
pythonと異なり，引数はプログラム名は入らない

VSCodeではエクスプローラ(左一番上のアイコン)でファイル名を右クリックし
パスをコピーしターミナルで　cd を打ち　ctrl-Vで貼り付ければ，
ソースを保管した場所に移動できるので，コンパイル，実行をターミナルで行える