



発展プログラミング演習II

2. Java言語の基礎 条件分岐, 繰り返し

コンピュータ理工学部
玉田春昭 水口充



Java言語の基礎

- コメント
- 制御構文
 - 条件分岐
 - 繰り返し(ループ)
- 両方ともプログラムの制御構造を形作る重要なものの。

コメント

- Javaのコメントは3種類
 - `//` この記号の後ろ, 改行までがコメントになる.
 - `/*` ここがコメントになる. `*/`
 - `/**` ここがコメントになる. `*/`
 - javadocコメントと呼ばれる.
- Javadoc
 - Javaソースファイルからドキュメントを作成するコマンド.
 - Javadocコメント形式で書かれたコメントを整形し, html形式のドキュメントを出力する.

条件分岐

- プログラムの途中で実行する文を少しだけ変えたい場合に用いる.

```
if (条件式1) {  
    // 条件式1がtrueのとき実行される文  
} else if (条件式2) {  
    // 条件式2がtrueのとき実行される文  
} else {  
    // 上記のどの条件も満たさないとき実行される文  
}
```

else if, else節は必須ではない.
else ifはいくつ書いても良い.
elseはなしか, 1つのみ.

```
switch(式) {  
    case 値1:  
        // 式の結果が値1のときのみ実行される文  
        break;  
    case 値2:  
        // 式の結果が値2のときのみ実行される文  
        break;  
    default:  
        // 上記のどの条件も満たさないとき実行される文  
        break;  
}
```

case節はいくつ書いても良い.
default節はswitch文に多くて1つ. なくても良い.
break文がなければ, 処理が下に落ちていく.

if, else-if, else

- 条件が真(true)のときだけ実行したいとき

```
if(条件) {  
    文  
}
```

- 条件が真(true)のときと偽(false)のときで実行文を変えたいとき

```
if(条件) {  
    文  
}  
else {  
    文  
}
```

- 複数の条件で実行文を変えたいとき

```
if(条件1) {  
    文  
}  
else if(条件2) {  
    文  
}  
:  
else if(条件n) {  
    文  
}  
else {  
    文  
}
```

else-ifは何個でも書くことができる.



条件式

- Java言語の場合は`boolean`型のみを受け付ける.
 - C言語は0が`false`, それ以外が`true`として扱われた.
 - Java言語は条件式が`boolean`型でない場合, コンパイルエラーとなる.

例題2.1

- 以下のプログラムの空白部分を埋めよ.
 - コマンドライン引数で点数を与える.
その点数に対応した成績を出力するプログラムを作成する.
 - 出力内容は右の通り.
 - 点数が59点以下ならば「不可」
- 60点以上, 69点以下なら「可」
- 70点以上, 79点以下なら「良」
- 80点以上, 89点以下なら「優」
- 90点以上, 100点以下なら「秀」
- それ以外なら「不正な成績」

```
public class Evaluator{
    public void print(int score){
        

|  |
|--|
|  |
|  |
|  |
|  |
|  |
|  |


        System.out.println("秀");
        System.out.println("優");
        System.out.println("良");
        System.out.println("可");
        System.out.println("不可");
        System.out.println("不正な成績");
    }
    public static void main(String[] args){
        Evaluator evaluator = new Evaluator();
        int score = 




        evaluator.print(score);
    }
}
```

switch

- 条件の値によって実行文を変えたいときに用いる文.
- case文は何個でも追加できる.
- switchの実行手順
 - 式を評価して値を得る.
 - 値に対応するcase文があれば, その文を実行する.
 - 値に対応するcase文がなければ, default文で指定されている文を実行する.
 - default文がなければ, switchを抜ける.
- 注意事項
 - break文がなければそれ以降の文も実行されてしまう.
 - break文がないとき, 俗に処理が落ちていくという.

```
switch(式) {  
  case 2:  
    文  
    break;  
  case 3:  
    文  
    break;  
  case 5:  
    文  
    break;  
  case 7:  
    文  
    break;  
  default:  
    文  
    break;  
}
```




例題2.2

- 教科書p.24例2.7を変更し, 成績にSを追加すること.
 - 成績と対応する出力文は以下の通り.
 - S: 非常に優秀な成績で合格です.
 - A: 優秀な成績で合格です.
 - B: 良い成績で合格です.
 - C: 合格です.
 - D: 不合格です.
 - その他: 成績が正しくありません.
- 完成すれば, ローカル変数のgradeの値を変えて, 実行すること.

switchとifの使い分け

- 式の評価回数で決める.
 - if文は条件式のたびに分岐が評価される.
 - switch文で式が評価されるのは1回だけ.
- 分岐が単純な数値に分けられる場合はswitch.
- 条件が複雑な場合はif.

```
if (条件式1) {  
    // 条件式1がtrueのとき実行される文  
} else if (条件式2) {  
    // 条件式2がtrueのとき実行される文  
} else {  
    // 上記のどの条件も満たさないとき実行される文  
}
```

← 式が評価される箇所

```
switch(式) {  
    case 値1:  
        // 式の結果が値1のときのみ実行される文  
        break;  
    case 値2:  
        // 式の結果が値2のときのみ実行される文  
        break;  
    default:  
        // 上記のどの条件も満たさないとき実行される文  
        break;  
}
```



繰り返し(ループ) (1/2)

- プログラミング言語の目的はプログラムを短く分かり易く書くこと.
 - その一手段として繰り返しがある.
 - 同じような処理が繰り返される時, その処理を1回分だけ書き, 後はその処理を繰り返すという命令を加える.
 - 結果としてプログラムが短くなる.
- 繰り返しは制御構造の一つ.

繰り返し(ループ) (2/2)

- Java言語の繰り返しはwhile, do-while, forの3種類.
 - C言語とほぼ同じ.
 - ループ継続の条件をbooleanで制御することだけが異なる.

```
while(式)  
  文
```

```
while(式) {  
  文  
  文  
  文  
}
```

```
do  
  文  
while(式);
```

```
do {  
  文  
  文  
} while(式);
```

```
for(初期化式; 条件式; 反復式)  
  文
```

```
for(初期化式; 条件式; 反復式) {  
  文  
  文  
  文  
}
```

whileループ

- whileはループの継続条件をループの最初に置く場合と、最後に置く場合がある.
 - 最初に置く場合: `while (式) { ~ }`
 - 最後に置く場合: `do { ~ } while (式);`
- 両方とも、式の結果が`true`の場合、ループを繰り返す.

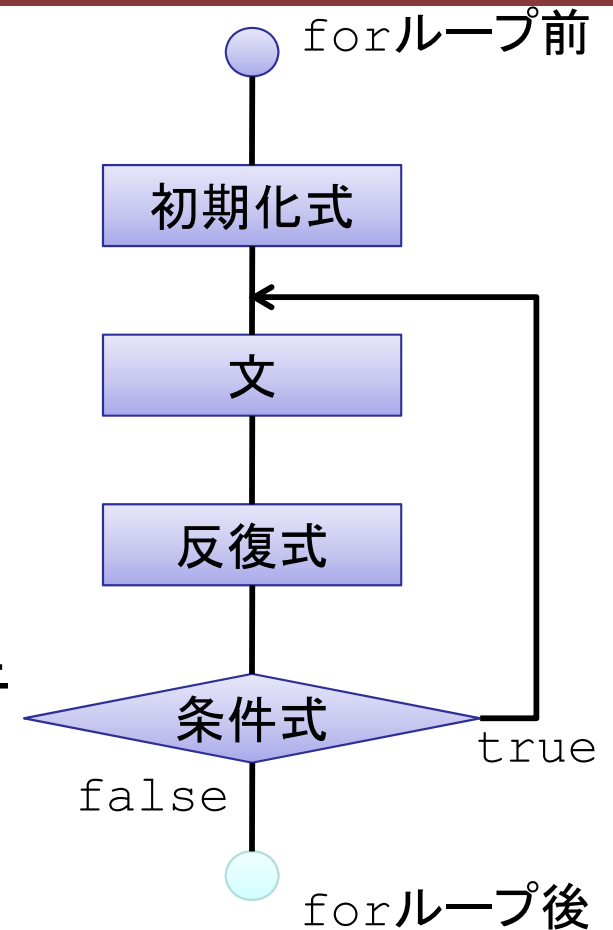
```
while (式) {  
    文  
}
```

```
do {  
    文  
} while (式);
```

forループ

- 条件式がtrueもしくは、空欄の場合に繰り返すループ。
 - 初期化式はforループに入る前に一度だけ実行される。
 - C言語と異なり、初期化式で変数宣言が可能。
 - for(int i = 0; i < 10; i++)
 - 反復式はforループ内の文が全て実行された後に実行される式。

```
for(初期化式; 条件式; 反復式) {  
    文  
}
```



例題2.3, 2.4

- 下のプログラムをループを使って書き直せ.
 - 例題2.3
 - whileループを用いる.
 - LoopAssignment1.java
 - 例題2.4
 - forループを用いる.
 - LoopAssignment2.java

```
public class LoopAssignment{  
    public static void main(String[] args){  
        System.out.println("iの値は0です. ");  
        System.out.println("iの値は1です. ");  
        System.out.println("iの値は2です. ");  
        System.out.println("iの値は3です. ");  
        System.out.println("iの値は4です. ");  
        System.out.println("繰り返しが終わりました. ");  
    }  
}
```

例題2.5: 繰り返しの入れ子

- 条件分岐や繰り返しは入れ子にして使える.
- 以下のプログラムを作成し, 実行せよ.

```
public class LoopNesting{
    public static void main(String[] args){
        for(int i = 0; i < 2; i++){
            for(int j = 0; j < 3; j++){
                System.out.println("i = " + i + ", j = " + j + "です.");
            }
        }
    }
}
```

例2.10(教科書p.30)

例題2.6: 繰り返しの入れ子

- 右のような出力になるようにプログラムを作成せよ.
 - クラス名はLoopNesting2とする.

```
i=3, j=0です.  
i=2, j=0です.  
i=1, j=0です.  
i=3, j=1です.  
i=2, j=1です.  
i=1, j=1です.  
i=3, j=2です.  
i=2, j=2です.  
i=1, j=2です.  
i=3, j=3です.  
i=2, j=3です.  
i=1, j=3です.  
i=3, j=4です.  
i=2, j=4です.  
i=1, j=4です.  
i=3, j=5です.  
i=2, j=5です.  
i=1, j=5です.
```



break文とcontinue文

- break文
 - breakが実行されると即座に一番内側のループ, switchから抜ける.
 - 外側のループを抜けるためにはラベルを用いる.
- continue文
 - continueが実行されると最も内側のループの残りのブロックを無視し, 次の繰り返しに制御が移る.

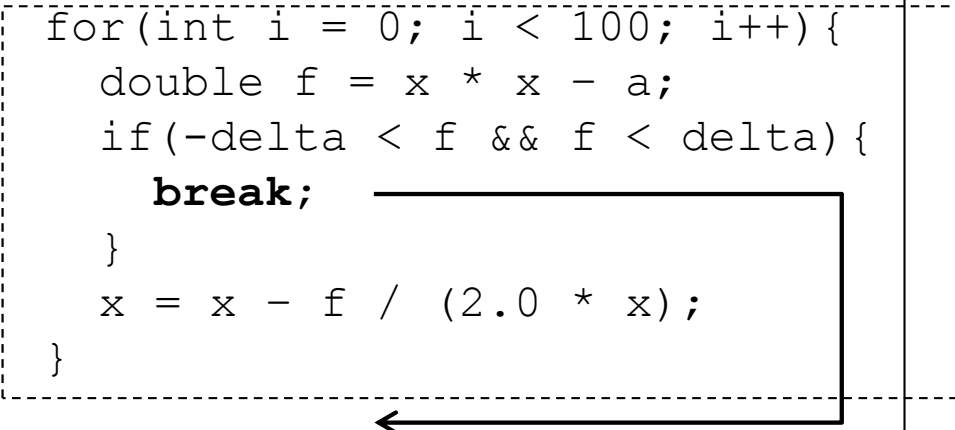
break文

- breakが実行されると
即座に一番内側の
ループ, switchから
抜ける.
 - 繰り返しを打ち切る条
件が複雑な時に使う.

```
public void run(){
    double delta = 0.0001;
    double a = 2.0;
    double x = a;

    for(int i = 0; i < 100; i++){
        double f = x * x - a;
        if(-delta < f && f < delta){
            break;
        }
        x = x - f / (2.0 * x);
    }

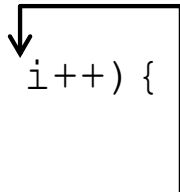
    System.out.println(x);
}
```



continue文

- continueが実行されるとループ内の処理を打ち切り、即座にループの最初に戻る。
 - for文の場合、反復式が実行された後、最初に戻る.
 - ループが途中で終了するわけではなく、残りのループは実行される.

```
public void run(){
    int s = 0;
    for(int i = 1; i <= 100; i++){
        if((i % 3) != 0){
            continue;
        }
        s += i;
    }
    System.out.println(s);
}
```



ブロック

- `if`や`while`, `for`文などで`{ }`で囲まれた部分をブロックという.
 - `{ }`がない場合は最初に現れるセミコロン(`;`)までがブロックとなる.
- `{ }`の中は文をいくつでも書くことができる.
 - `{ }`がない場合, ブロックを勘違いし易いので, できるだけ`{ }`を書く方が良いとされている.
- `if`や`while`, `for`がなくても`{ }`を書いてブロックを作れる.

例題2.1: 入れ子

- 以下の出力結果になるよう, プログラムを作成せよ.
 - クラス名は「PatternPrinter_学生証番号」とする.

```
    0123456789
0:  x           x
1:   x           x
2:    x         x
3:     x       x
4:      xx
5:       xx
6:      x     x
7:     x       x
8:    x         x
9:   x           x
```

課題2.2 ニュートン法

- ニュートン法を使ってコマンドライン引数に与えられたdouble型の数 α の平方根を求めよ。
 - α の平方根は $f(x)=x^2-\alpha$ と置き $f(x)=0$ の解を求めることで計算できる。
 - 以下の漸化式で x を計算する。
 - $x_{n+1}=x_n-f(x_n)/f'(x_n)$
 - 誤差はここでは、0.0001とする。
 - $f(x)$ の値が0.0001以下であれば、その時点の x を α の平方根とする。
 - Javaで絶対値を求めるには`Math.abs(値)`とする。

```
public class NewtonMethod{
    private double delta = 0.0001;
    private double startX = 2d;
    public double calculate(double alpha){
        double x = startX;
        if(alpha < 0){
            return Double.NaN;
        }
        while(Math.abs( ) > delta){
        }
        return x;
    }
    public static void main(String[] args){
        double alpha = ;
        NewtonMethod nm = new NewtonMethod();
        double value = nm.calculate(alpha);
        System.out.println(
            "sqrt(" + alpha + ") = " + value
        );
    }
}
```



課題に関する注意

- 提出期限
 - 2012/10/8(月) 16:30
- 提出先
 - 課題2.1
 - Moodleの「課題2.1: 入れ子」
 - 課題2.2
 - Moodleの「課題2.2: ニュートン法」
- クラス名
 - 課題2.1
 - PatternPrinter_学生証番号
 - 課題2.2
 - NewtonMethod_学生証番号
 - 学生証番号は6桁. 頭にglは付けない.
- 注意
 - ソースコードを提出すること.
 - コンパイルできることを確認してから提出すること.