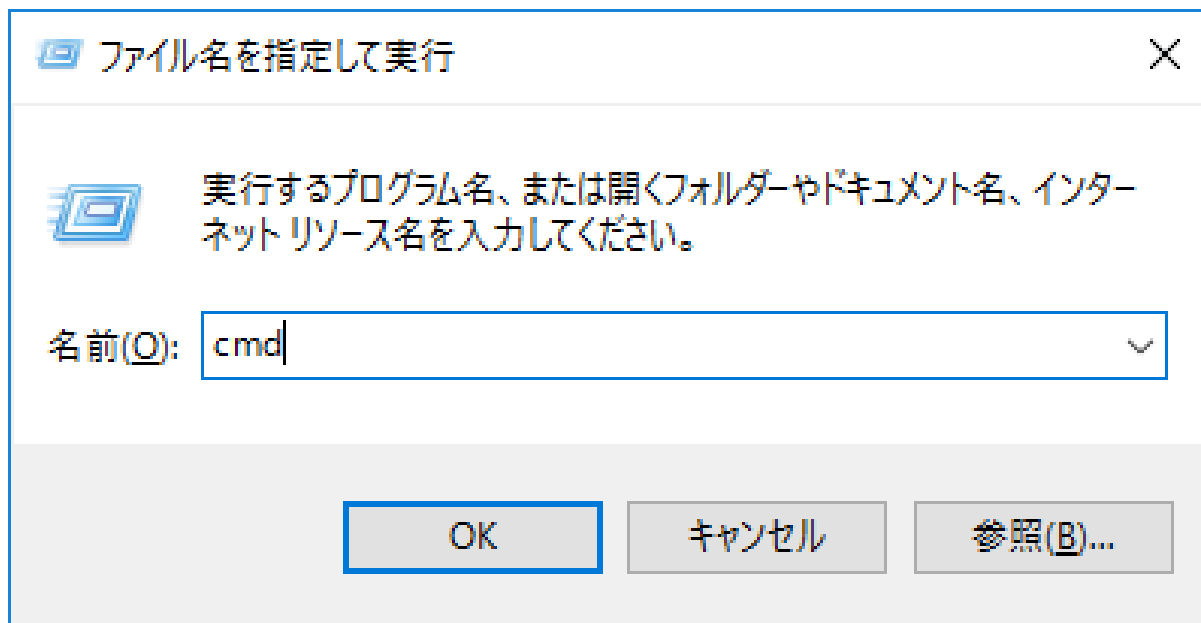


# Java講習第 3 回

# コマンドプロンプトについて

## 起動方法

 Windowsキー + Rキー ⇒ 名前[cmd] ⇒ OK



# コマンド

※ △の記号は空白を表す  
[]で囲まれているものは、  
[]を外し、適当なものを入れる

**dir**

・・・ディレクトリ一覧表示

**terapad△[ファイル名]** ・・・[ファイル名]をterapadで開く

※[ファイル名]が無ければ作成するか聞かれる

**del △[ファイル名]** ・・・ファイル削除

**mkdir △[フォルダ名]** ・・・フォルダ(ディレクトリ)作成

**rmdir △[フォルダ名]** ・・・ディレクトリ削除

例)

**dir**

**terapad△Tes.java**

**del△Tes.java**

**mkdir△ Test**

**mkdir △ Test**

- ・ 現在場所(カレントディレクトリ)移動

**cd△[移動先]**

- ・ ドライブの移動

**[ドライブ名]:**

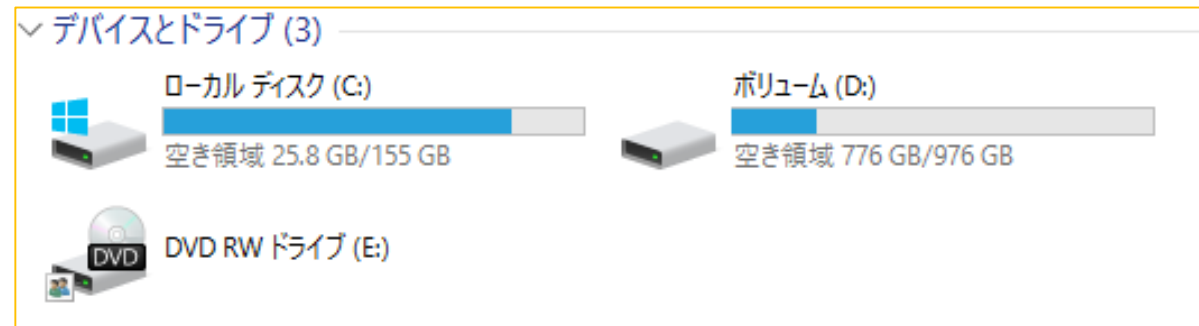
もしくは

**cd△/d△[ドライブ名]:**

※こちらのコマンドでは、  
階層の移動も出来る

**[¥]で下の階層**

**[..]で上の階層    を示す**



例)

**cd△NMC**

**cd△ Test¥Tes**

**C:**

**cd△/d△ Z:**

**cd △/d △Z:¥NMC**

**terapad△Tes.java**

# ファイル名 補完機能

ファイル名の先頭の数文字を入力しただけで、自動的に入力してくれる機能

Tabキーで補完

(候補が多い場合は数回おすと次の候補が表示される)

例) TestHaloWorld.java を実行させる

> java Test

Tabキー

> java TestHaloWorld.java

# cmd 履歴

- 矢印キーの上を押すと過去入力したものが自動入力してくれる  
(下を押すと現在表示されてるコマンドの次に入力したものが表示される)

※cmdを閉じると履歴は初期化

# boolean型について

int型 . . . 整数専用の型 (箱)

boolean型 . . . true / false 専用の型 真偽を表す

true . . . 真 正しい

false . . . 偽 誤り

# Boolean型の使い方

```
boolean flag;
```

//flagという名前のboolean型の変数を(使うと言う事を)宣言

```
flag = true; //変数flagに true を代入
```

```
System.out.println(flag); //変数flagの中身を出力
```

```
flag = false;
```

```
System.out.printf("%b¥n",flag); //変数flagの中身を出力
```

//printfで変数を出力する際は %[種類] が必要

//booleanは%b



# 比較・・・比較結果はbooleanで示される

|            |    |
|------------|----|
| 左右同じか      | == |
| 左右等しくないか   | != |
| 左側の値のより大きい | <  |
| 左の値以上か     | <= |
| 右側の値の方が大きい | >  |
| 右側の値以上か    | >= |

```
boolean flag = false;    //初期値にfalse (宣言と同時に値を入れる)
flag = -3==5;            //-3と5が同じかどうか の結果をflagに代入 (false)
flag = -3!=5;            //-3と5が同じではないか (true)
int a,b;                //変数の比較も同様に行う
a = -3;
b = 5;
flag = a<b;              //a(-3)はb(5)より小さいか (true)
flag = a<=b;             //aはb以下であるか (true)
```

# 演習 1 boolean型

- ・ 変数 aとbを用意し、好きな数をそれぞれ代入する
    - aとbを比較し、
      - ・ aがbより大きいかどうかの結果をtrueかfalseで出力
      - ・ aがb以下であるかどうかの結果をtrueかfalseで出力
      - ・ aとbの中身を出力
- するプログラムを作れ

```
public△class△[ファイル名(拡張子無し)]{  
    public△static△void△main(String[]△args){  
        //処理  
    }  
}
```

# 演習1 解答例

Example.javaという名前のファイルで作成

```
public class Example{
    public static void main(String[] args){
        int a,b;
        boolean answer;
        a = 5;
        b = 10;
        answer = a > b;
        System.out.println(answer);
        System.out.println(a <= b);
        System.out.println("a="+a + "b="+b);
        //+で文字列の連結ができる
        //System.out.printf("a=%d△b=%d¥n",a,b); //printfの場合
    }
}
```

# IF文

if文：条件分岐 （条件:trueかfalseであるか）  
条件に合わせて(trueなら)、処理を分けたい場合に使う  
範囲は『 {』 から『 } 』まで

```
if(boolean型){  
    処理;  
}
```

```
if( true ){  
    処理;  
    //この処理は行われる  
}
```

```
if( false ){  
    処理;  
    //この処理は行われない  
}
```

# 具体例

```
if( boolean[条件が成り立っているか] ){  
    処理;  
}
```

わざわざboolean型の変数を用意せず()内で直接比較することが多い

```
if( 5 > 4 ){  
    //処理される  
}  
  
if( 5 < 4 ){  
    //処理されない  
}
```

```
if( a >= 4 ){  
    //aが4以上なら処理  
}  
if( a*4 == 10 ){  
    //a×4が10なら処理  
}  
  
if( a+b < b*2 ){  
    //a+bがb×2より小さいなら処理  
}
```

# 課題 2

int型の変数 aとbを用意する(a,bには自分で適当な値を代入する)

aの方が大きいなら「aが大きい」と出力

bの方が大きいなら「bが大きい」を出力

aとbの値が同じなら「aとbは等しい」を出力

```
if( boolean ){  
    処理;  
}
```

```
public△class△[ファイル名(拡張子無し)]{  
    public△static△void△main(String[]△args){  
        //処理  
    }  
}
```

# 解答例

```
public class Example{ //Exampleはファイル名
    public static void main(String[] args){
        int a = 10;    //int型の変数aを宣言し、aに10を代入
        int b = 5;     //int型の変数bを宣言し、bに5を代入
        if(a > b){     //aの方が大きいのか
            System.out.print("aが大きい¥n");
            //printはprintlnの自動改行出力なしの動作
        }
        if(a < b){     //bの方が大きいのか
            System.out.printf("bが大きい¥n");
        }
        if(a == b){    //aとbが等しいのか
            System.out.println("aとbは等しい");
        }
    }
}
```

# else

if文がfalseの時に行う処理

⇒ if文の条件が成立しないときに処理

```
if( true ){  
    //ここの処理が行われる  
}else{  
    //ここの処理は行われない  
}
```

```
if( false ){  
    //ここの処理が行われない  
}else{  
    //ここの処理は行われる  
}
```



# 具体例

```
if( boolean[条件が成り立っているか] ){  
    成り立っていた場合(true)の処理;  
}else{ //ifがtrueでない場合  
}
```

```
if( 5 > 4 ){  
    //処理される  
}else{  
    //処理されない  
}
```

```
if( 5 < 4 ){  
    //処理されない  
}else{  
    //処理される  
}
```

```
if( a*4 == 10 ){  
    //a × 4が10なら処理  
}else{  
    //a × 4が10でなければ処理  
}  
  
if( a+b < b*2 ){  
    //a+bがb × 2より小さいなら処理  
}else{  
    // a+bがb × 2以上なら処理  
}
```

# 連続した if - else文

ifの中にifを入れることもできる

```
int a,b,c;           //int型a,b,cがあり
a = 1; b=2; c=3;     //aに1,bに2,cに3を代入
if(a<b){             //aよりbが大きいなら
    if(b<c){         //(a<bで)bよりcが大きいなら
        //処理A
    }else{           //(a<bで)bよりcが大きくないなら → b>=cなら
        //処理B
    }
}else{               //a < bでなければ → b>=a なら
    //処理C
}
```

今回の場合  $a < b$  であり、 $b < c$  なので  $(a < b < c)$  処理Aが行われる

# 演習3 if-else

- int型a,bを用意し、好きな値をそれぞれに代入する
  - 変数bの方が変数aより小さい場合は  
変数aと変数bの値と交換する  
その後変数aと変数bを出力する
- 変数aが変数bの値以上であれば「問題なし」と出力する

ヒント:

2つの変数の中身を交換する場合は別の変数を用意して、一時的に入れておく

# 解答例

```
public class Example{
    public static void main(String[] args){
        int a,b;
        a = 5;    b = 1;
        if(b < a){    //bの値の方がaより大きいなら
            int tmp;    //一時的に数値を置いておく変数
            tmp = a;    //aの値を一時的に置く(複製しておく)
            a = b;    //bの値をaに代入
            b = tmp;    //置いておいたaの値をbに代入
            System.out.println("a="+a + "△b=" + b);
        }else{    //ifが成り立ってなければ→ a ≤ bなら
            System.out.println("問題なし");
        }
    }
}
```

# while文 (繰り返し)

1 ~ 100の値をすべて足した時の合計を求めたいとき

```
int sum = 0;
```

```
sum = sum+1;
```

```
sum = sum+2;
```

```
sum = sum+3;
```

↓

```
sum = sum+100;
```

とこのように書くのは効率が悪い

そこでwhile文やfor文 (次回) を使うことで簡単にプログラムを記述できる

# while文      使い方

```
while( 条件式 ){  
    //条件式がtrueの間繰り返し処理  
}
```

条件式(boolean型)が成り立っている(true)の間  
『{』 から 『}』 に記述された文を実行する

# 具体例

```
while( 条件式 ){  
    //条件式がtrueの間繰り返す  
}
```

```
while(true){  
    //無限ループする  
}  
  
while(false){  
    //一度も実行されない  
}
```

Ctrl + C を押すとプログラム  
強制終了ができる

```
int sum = 0;  
int i=1;  
while( i <= 10 ){  
    //iが10以下なら繰り返す  
    sum = sum + i;  
    i++;        // i= i + 1;と同じ意味  
}  
System.out.println(sum);
```

→ 1 + 2 + 3 + ... + 10 の合計を表示するプログラム

# 演習4

1. Hello World を10回出力するプログラム
2. 1 から 10までの数値を出力するプログラム
3. 1から10までのフィボナッチ数列を出力するプログラム

※フィボナッチ数列    . . . 1,1,2,3,5,

「前の2つの数を加えると次の数になる」という数列

ただし、1番目と2番目の数は両方とも1

$$0+1 = 1$$

$$1 + 1 = 2$$

$$1+2 = 3$$



# 解答例      Hallo    Worldを10回

```
public class Example{  
    public static void main(String[] args){  
        int i = 1;           //何回目かを記録する  
        while ( i <= 10){    //iが10以下なら繰り返す  
            System.out.println("Hallo World");  
        }  
    }  
}
```

# 解答例      1 ~ 10までの数値を出力する

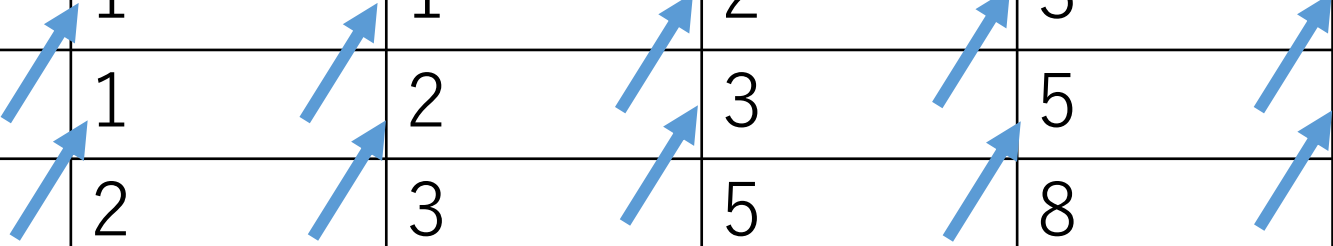
```
public class Example{  
    public static void main(String[] args){  
        int i = 1;           //何回目かを記録する  
        while ( i <= 10){    //iが10以下なら繰り返す  
            System.out.println(i);  
        }  
    }  
}
```

# 解答例      フィボナッチ数列

```
public class Example{
    public static void main(String[] args){
        int first, second, sum, cnt;
        System.out.println(1); //1番目は1である
        //firstには足される数, secondには足す数, sumには計算結果

        first = 0;    second = 1;    sum = first + second;
        cnt = 1; //1をすでに出力したため現在1個出力したと記録
        //フィボナッチ数列[出力する数(sum)] が10以下なら繰り返す
        while (cnt < 10){
            System.out.println(sum);
            first = second;
            second = sum;
            //最初は (0+1)の結果を出力, 足した1をfirst, 結果をsecondに移動
            //その後 1 (足した数)+ 1(sum) の結果をsumに入れる
            sum = first + second;
            cnt++; // 1つ出力したので出力した数を加算
        }
    }
}
```

|        |   |   |   |   |   |    |
|--------|---|---|---|---|---|----|
| First  | 0 | 1 | 1 | 2 | 3 | 5  |
| Second | 1 | 1 | 2 | 3 | 5 | 8  |
| Sum    | 1 | 2 | 3 | 5 | 8 | 13 |



The diagram illustrates the calculation of the sum of two sequences. The 'First' row contains the values 0, 1, 1, 2, 3, 5. The 'Second' row contains the values 1, 1, 2, 3, 5, 8. The 'Sum' row contains the values 1, 2, 3, 5, 8, 13. Blue arrows point from the 'First' and 'Second' rows to the 'Sum' row, showing the addition of corresponding terms: 0+1=1, 1+1=2, 1+2=3, 2+3=5, 3+5=8, and 5+8=13.