# Java 入門レベル

Java プログラミング春学期分は、教科書の前半を片っ端からシラミツブシに学んでいくのではなく、内容を易しい順に３レベルに分け、３回スキャンして全体を学びます。
レベルは、一番簡単なのが「入門レベル」、次が「初級レベル」、最後を「中級レベル」とします。

春学期にマスターすべき項目は、以下の通りです。

入門レベル：
□(1)標準出力
□(2)キーボード入力
□(3)変数（数値）
□(4)条件分岐（if 文）
□(5)繰り返し（for 文）
□(6)配列（1 次元）

初級レベル：
□(7) 変数（文字、文字列）
□(8) 多重 for の繰り返し
□(9)配列（2 次元）
□(10)複雑な条件分岐（if.else.if..と switch 文）
□(11)繰り返し（while 文、while_do 文）

中級レベル：
教科書で落ち穂を拾っていきます。

# １．流れ図（フローチャート）

初めに、何となく流れ図を紹介しながら、“変数”と“繰り返し”のイメージを掴んでもらいます。
計算機の処理手順を図式化したものが流れ図です。流れ図はプログラム以外でも用いられているので親しみやすく、プログラムを直感的に理解することができます。
本章では、簡単な例を使いながら流れ図を説明します。流れ図では数値を格納する“変数”というものを使います、変数はプログラムで重要な役割を果たすので、ここでイメージを掴んでおきましょう。

<例１＞キーボードから２つの数を入力し、小さい方の数値を出力する流れ図を示します。キーボードから入力される２つの数は、計算機の a という記憶場所（変数）と、b という記憶場所（変数）に格納して処理するものとします。

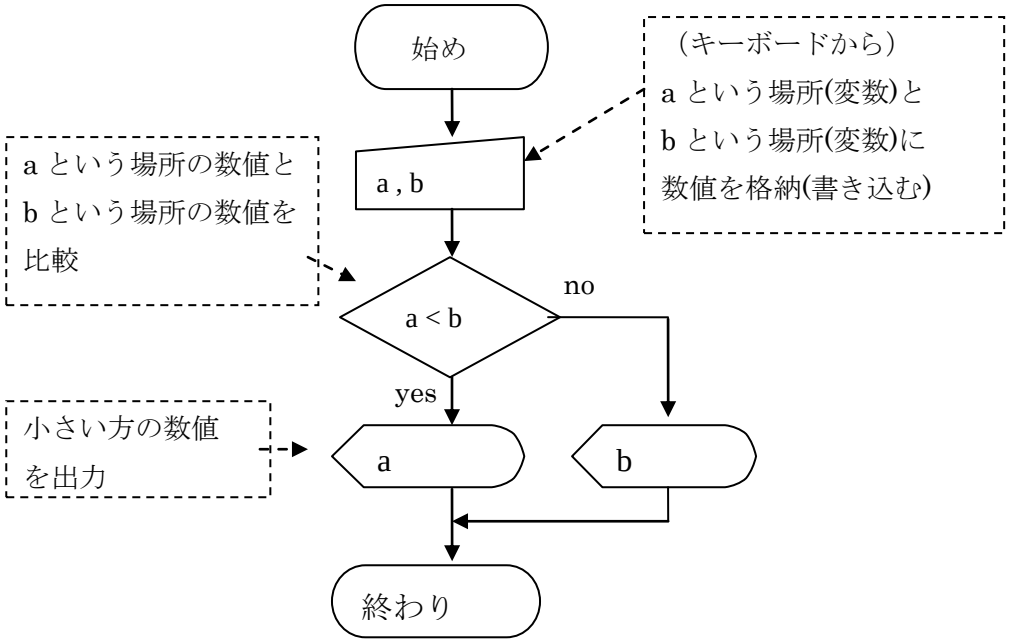



図１．１　簡単な流れ図

| |
|---|
| 変数の特徴：　変数はデータを記憶する場所。黒板のような感じで、一度データを格納する（書き込む）と、次にデータを格納する（書き込む）まで、先に格納したデータは消えない（新しいデータの格納で古いデータが消える)。格納したデータは何度でも読み出せる（読み出しによってデータは消えない)。 |

流れ図で用いられる主な記号は以下の通りです。

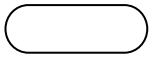
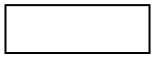
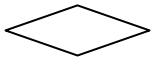
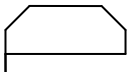
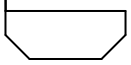
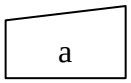
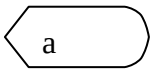
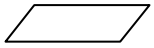
| | | |
|---|---|---|
|  | 端子 | 始めや、終わりを示す |
|  | 処理 | 色々な処理を示す<br>例： c = a + b |
|  | 判断 | 条件により分岐する |
|  | 繰返<br>開始 | 繰り返しの開始 |
|  | 終了 | 繰り返しの終了 |
|  | 入力 | キーボードから変数 a にデータを入力<br>（a という場所に数値を格納） |
|  | 出力 | 変数 a の数値をディスプレーに表示<br>(a という場所に格納されている数値を表示) |
|  | 入出 | 入出力、どちらにも使われる |

図１．２　流れ図で用いる記号

<例２＞　直線的な流れ図
数値を n という場所（変数）に入力し、
$n^3$　を計算して結果を m という場所（変数）に格納し、
その後 m に格納されている数値をディスプレーに表示します。

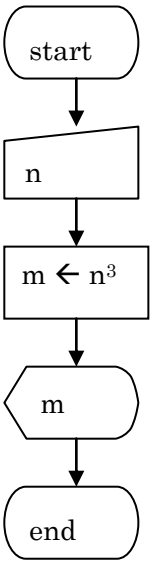



図１．３　３乗を求める流れ図

＜例３＞　繰り返し（ループ）を持つ流れ図

同じ作業を繰り返し行うのは計算機の得意とする所です。極端に言えば“プログラムは、目的とする仕事を‘繰り返し処理’に変換する作業”と言えます。

次の図は“1~10 までの自然数の和を求める”流れ図です。

６角形で挟む繰り返しの表記は、後述プログラムの for 文を意識したもので、プログラムに変換しやすい形といえます。図の右には変数 i, s の遷移を示します。

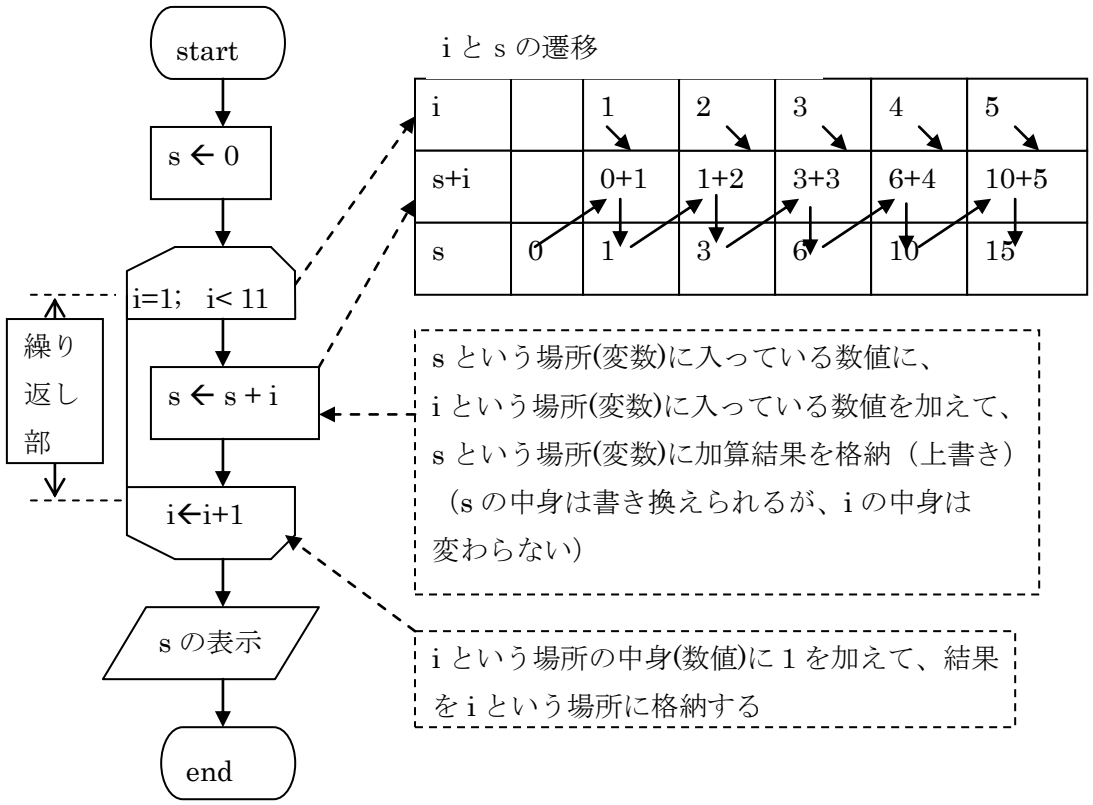
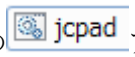



図１．４　繰り返しで、１＋２＋３・・・＋１０を求める

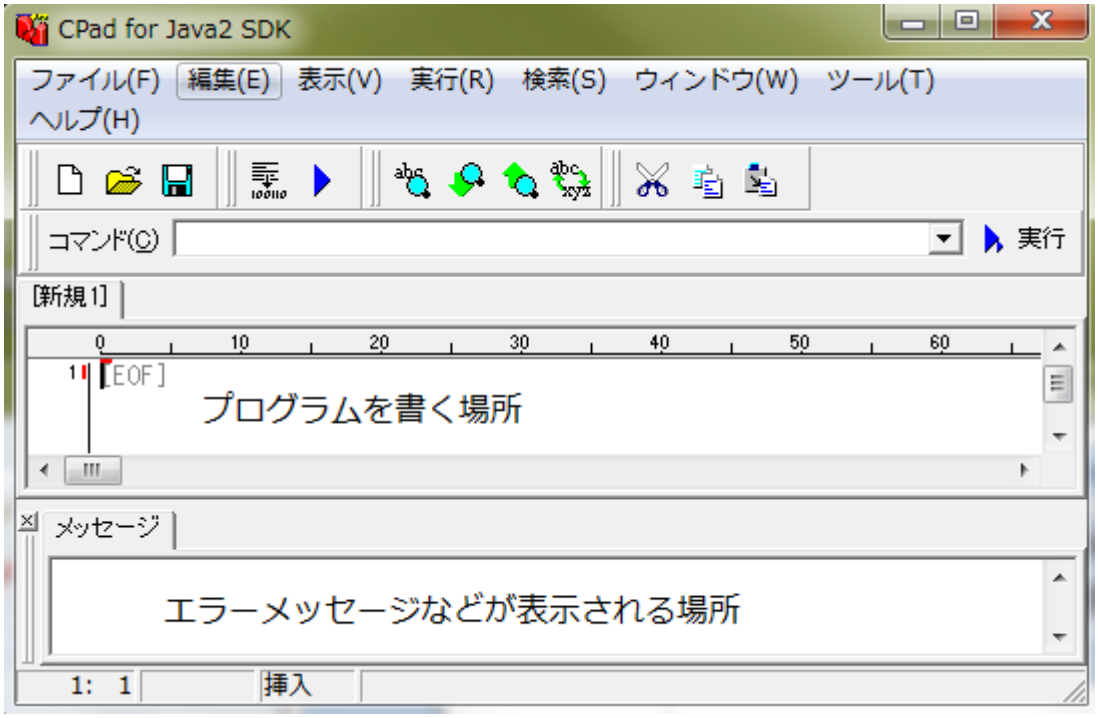
## ２．初めてのプログラミング（JCPAD でプログラミング）

先ずは、「こんにちは」と出力するプログラムを作ってみます。

Step1: USB メモリの 

アイコンをWクリックすると次の画面が現れます。

「プログラムを書く場所」という所にプログラムを書き込みます。




Step2:プログラムを書きます。

１行目の HelloWorld がプログラム名です。

プログラムは｛　｝カッコで囲まれています。

２行目は、「ここからがメイン(main)プログラムですよ」と宣言しています。

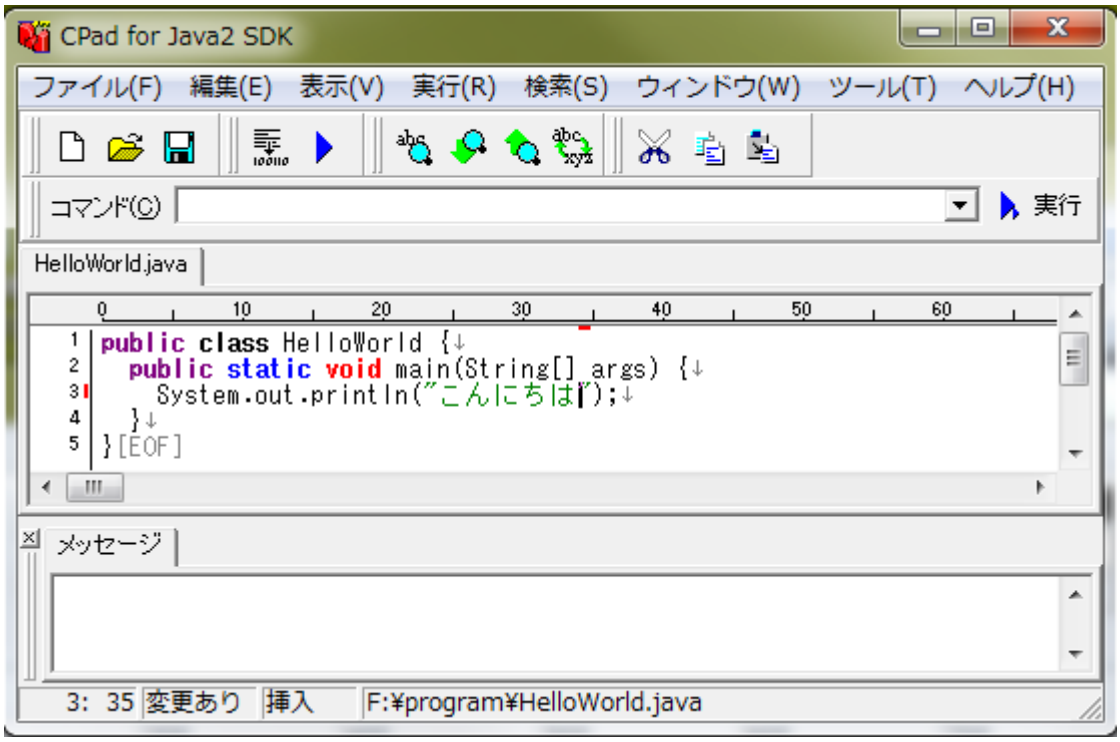
メイン(main)プログラムも｛　｝カッコで囲まれています。

（メインと言うからにはメイン以外もあり得ますが、春学期の内はメインしかないものと思っていて下さい）

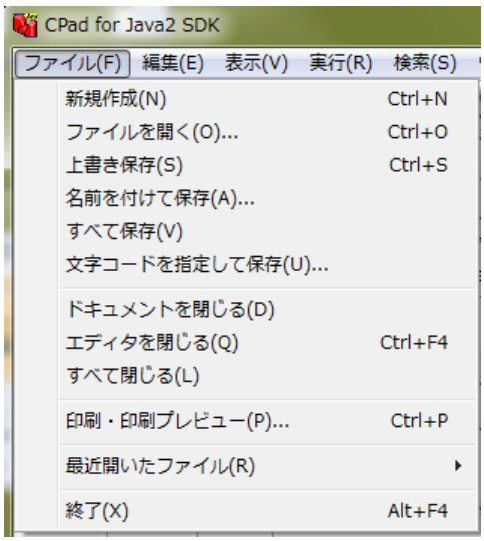
３行目は、プログラムの本体で、このプログラムは１つの命令だけで出来ています。

System.out.println(“こんにちは”);　は、

標準出力に「こんにちは」と出力せよ、という命令です。




Step3: プログラムができたら保存しよう。
「ファイル」ボタンを押すと次のようなものが出ます。

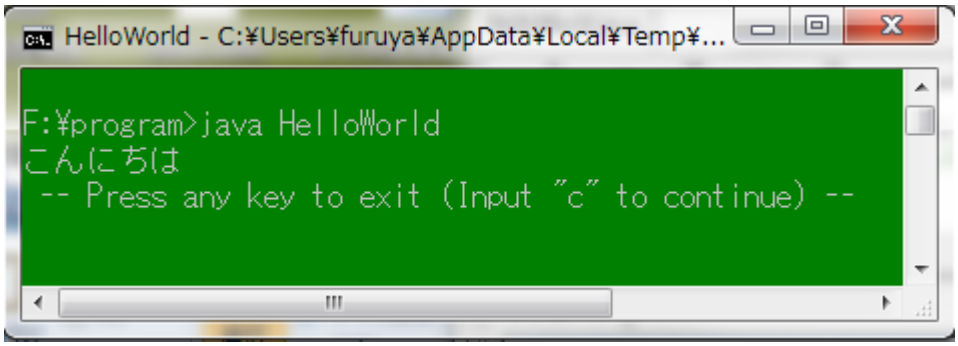



Step4:ここで「名前を付けて保存」を押すと「ファイル名」を書き込む画面が出るので、ファイル名を書きます。
この時、ファイル名はプログラムの１行目に書いたプログラム名（上の場合 HelloWorld）と同じでなくてはいけません。

Step5: 実行

実行ボタン  を押すと、プログラムの実行が始まり、次のように表示されます。

（こんにちは、と出力されています）




Step6:”Esc”ボタンで終了

## ３．　習うより慣れよ

２つの簡単なプログラムを示します。（とりあえず、勘で読もう、プログラム）
プログラムの基本構造は次の通りです。

① （public class プログラム名）：初めにプログラム名を宣言します。
（このプログラム名とファイル名は同じでなくてはいけません）

② （public static void main(String args[])）：プログラム本体の始まりを宣言します。
プログラムの実行は、この main（本体）の後の｛ の後の命令を上から順に実行していきます。

③ main の中では先ず、変数の宣言を行います。
変数は、整数用と実数用は別に宣言します。理由はコンピュータ内で整数と実数は全く違う形をしているためです（秋学期の計算機機構論で習います)。

④ プログラムの実行は、命令部の一番上の命令から順に実行されます

プログラム例１：
整数用変数を２つ、実数用変数を２つ宣言し、簡単な演算を行い、結果を出力します。

```
public class  Start0 {  //---プログラムの開始
                //---（ファイル名とプログラム名(Start0)は同じにすること）

    public static void main(String args[]) {   //---プログラム本体の開始

    //---変数宣言
        int seisu0;  //---seisu0という整数用変数を宣言
        int seisu1=1;   //---seisu1という整数用変数を宣言し初期値0を代入
        double menseki;  //---mensekiという実数用変数を宣言
        double hankei= 3.0; //--hankeiという実数用変数を宣言し初期値3.0を代入

    //----命令部

        System.out.println("seisu1の値は");  //---printlnは改行する
        System.out.println(seisu1);
        System.out.println("です。");

        seisu0= seisu1 + 5;   //---seisu1に５を加えた値をseisu0に代入
        System.out.print("seisu0は");  //---printは改行しない
        System.out.print(seisu0);
        System.out.println("です。");

        menseki= hankei * hankei * 3.14; //-hankei×hankei×3.14をmensekiに代入
        //---次のように続けても書けます
        System.out.println("半径は" + hankei + "面積は" + menseki);
    }   //---mainの終りのカッコ

}  //---Start0の終りのカッコ[EOF]
```

実行結果は、

```
seisu1の値は
1
です。
seisu0は6です。
半径は3.0面積は28.26
```

。

プログラム例２：

キーボードから数値を入力できるようにするにはいくつもの「おまじない」が必要です。

```
import java.io.*;  //キーボード入力（K_B）用おまじない
↓
public class  KeyIn0    {  ↓
↓
    public static void main(String args[]) throws IOException{↓
                                          //上のthrows以下はK_B用おまじない↓
    //---変数宣言↓
        int seisu;↓
        double jissu;↓
        BufferedReader br= new BufferedReader(  //K_B用おまじない↓
            new InputStreamReader(System.in));  //（上と２行で１命令）↓
                ↓
    //----プログラム本体↓
        System.out.println("整数を入力してください");↓
        seisu= Integer.parseInt(br.readLine());  //---キーボードから整数を入力↓
        System.out.println(seisu);↓
        ↓
        System.out.println("実数を入力してください");↓
        jissu= Double.parseDouble(br.readLine()); //---キーボードから実数を入力
        System.out.println(jissu);↓
        ↓
    }   //---mainの終りのカッコ↓
    ↓
}  //---KeyIn0の終りのカッコ[EOF]
```

実行結果は、

```
整数を入力してください
12
12
実数を入力してください
14.5
14.5
```

です。

# ４．　命令の説明

先ず、命令の簡単な説明をし、次に公文風にプログラムをどんどん読んでいきます（読めなきゃ書けない、という発想)。その後で、書いてみます。

## ４．１　プログラムと命令　（命令は、プログラムで“文”と呼ばれる）

プログラムは、いくつかの行からできています。各行は文と呼ばれる（計算機への）命令です。これらの命令は、処理の流れを変える命令がない限り、上の命令から下の命令へと順に実行されます。

| プログラム |
| --- |
| 命令（文）;<br>命令（文）;<br>：<br>命令（文）; |

計算機の命令（文）は、次の５種類です（５種類しかありません)。

| | | |
| --- | --- | --- |
| 入力文 | キーボードからの入力 | |
| 出力文 | 画面への表示 | System.out.println(変数名); |
| 代入文 | 変数へ数値や計算結果を格納 | 変数名=数値；　変数名=計算式； |
| 条件文 | 条件による分岐 | if(条件){条件を満す時の処理}<br>else{条件を満さない時の処理} |
| 繰返し文 | 同じ処理の繰り返し | for(変数初期設定；繰返し条件；変数値変更){<br>繰り返し処理} |

以下に、これらの命令の意味、書き方、使い方を紹介します。

入門編は、数値を扱う変数、for 文での繰り返し、１次元配列、だけを扱います。

## ４．２　基本命令の説明

### （1） 変数の宣言

プログラムで使う変数は、最初に宣言します。
int は整数を表し、double は実数を表します。
こんな感じ、

```
int　seisu0;　//---seisu0 という整数用変数を宣言
int　seisu1=1;　//---seisu1 という整数用変数を宣言し初期値 0 を代入
double　menseki;　//---menseki という実数用変数を宣言
double　hankei= 3.0;　//---hankei という変数を宣言し、初期とを 3.0 にする
int hensu1,hensu2, a, b;　などと並べて宣言することもできます。
```

### 配列の宣言

配列には色々な宣言法がありますが、入門レベルでは最も簡単な、初期値を書く方法だけでいきます。
整数の配列なら、例えば、次のように宣言します。

```
int [ ]　sei= { 3, 5, 4, 8 };
```

（ここで sei は自分で自由に決める配列名）この例は、sei という名の４つの要素からなる配列を定義している。
実数の配列なら、例えば、次のようです。

```
double[ ]　jis={ 2.4, 6.4, 2.2};
```

（ここで jis は自分で自由に決める配列名）この例は、jis という名の３つの要素からなる配列を定義しています。
（知識：int は integer の略、double は倍倍精度の倍を表す）

### （２）出力命令

(1)文字列リテラル”HelloWorld”と出力するには、

```
System.out.println(“HelloWorld”);
```

(2)変数に入っている数値を出力するには、（変数名が hensu だとすると）

```
System.out.println(hensu);
```

(3)文字列と変数を一緒に出力するには、＋で接続する。

```
System.out.println(“答えは” + hensu + “です。”);
```

(4)出力後改行しない場合
前節の println を print に変える。

```
System.out.print(“HelloWorld”);
```

（知識：ln はラインフィード（改行）の略）

（System…という意味不明の長い命令は、我慢して覚えるしかない）
(5)改行だけなら、
System.out.println();
です。

### （３）代入文（代入命令）

代入とは、変数へ数値を格納（書き込む）することです。
**代入には“=”マークを使いますが、これは”等しい”という意味ではなく”←”の意味であることに注意して下さい。**
変数へ数値を代入する形式（フォーマット）は、次の４つです。
(i)　数値を直接代入する：　変数名＝数値；　（例　a=5;）
(ii)　変数の中身を、変数に格納：　変数名＝変数名；　（例　a= b;）
(iii) 数値や変数を使った計算結果を、変数に格納：　変数名＝計算式；(例 a= a+5;)

> プログラムの計算式　：　プログラムでは、数学の“計算式”
> ａ＋ｂ、ａ－ｂ、ａ×ｂ、ａ÷ｂ、ａ÷ｂの余り、を、
> それぞれ、a+b,　a-b,　a*b,　a/b,　a%b,と書きます。
> （計算式の例：　a= b+5;　c= d*e;　f= g%3;　i= i+1;）
> i=i+1;は i++;、　i=i-1;は i--;と省略することができます。

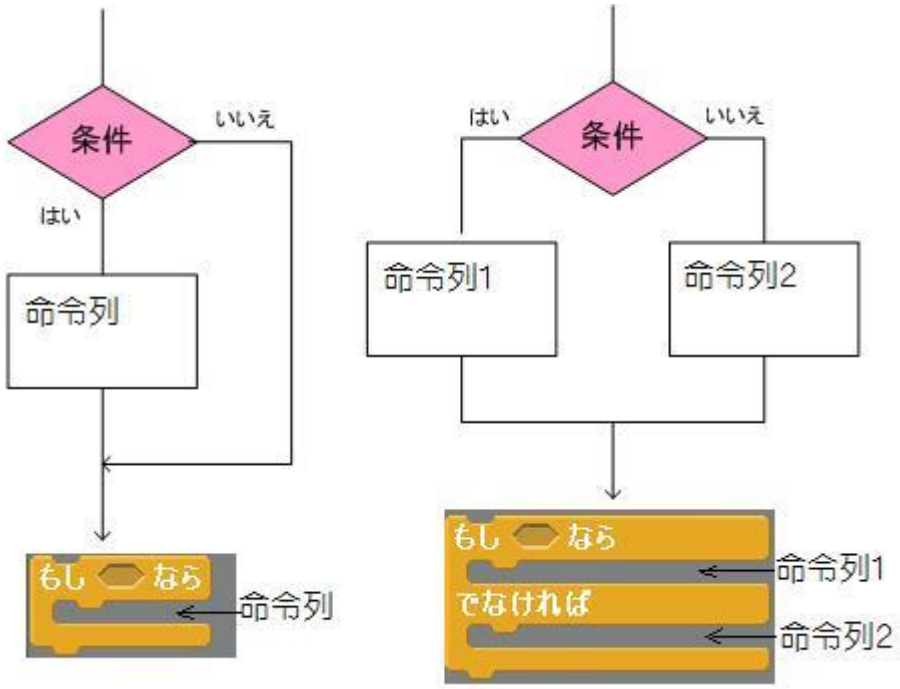
### （４）　キーボード入力命令

この入力命令はユーザにキーボード入力を促す命令で、他の命令と違い、キーボード入力が行われるまで止まって待っています。
入門レベルでは、数値の入力だけを扱いますが、
整数を入力する場合と、実数を入力する場合では命令が違います。
(1) 整数の入力
a という変数に整数を読みこむには、
a= Integer.parseInt(br.readLine());
(2) 実数の入力
b という変数に実数を読みこむには、
b= Double.parseDouble(br.readLine());
（意味不明の長い命令は、我慢して覚えるしかない）

但し、これらの命令を実行可能にするためには、いくつもの「おまじない」が必要になります。前章の「プログラム例２」を参照して下さい。

（５）　条件分岐

２つの条件分岐がある。




(1) if 文

左の図がこのタイプで、「条件判定」で条件を満たした場合だけ命令列を実行。

Java プログラムでは、

if (....){

命令列

}

と書く。

例１：点数を入力し、点数が 60 を超えた時だけ「合格」と出力。

```
import java.io.*; //---キーボード入力用おまじない↓
↓
public class  KeyIn1_1  {↓
↓
    public static void main(String args[]) throws IOException{  ↓
        //---変数宣言↓
        int tensu;↓
        BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
          ↓
        //---プログラム本体↓
        System.out.println("点数を入力して下さい");↓
        tensu= Integer.parseInt(br.readLine());↓
        if (tensu >= 60) {↓
            System.out.println("合格");↓
        }↓
    }   //---mainの終りのカッコ↓
    ↓
}  //---KeyIn1_1の終りのカッコ[EOF]
```

(2)if…else 文

右の図で、条件を満たしたら「命令列１」満たさなかったら「命令列２」を実行。

Java では、

if (….){

命令列１

}

else{

命令列２

}

と書く。

例２：点数が 60 を超えた時「合格」、60 未満なら「不合格」と出力。

```
import java.io.*; //---キーボード入力用おまじない
↓
public class  KeyIn1_2  {↓
↓
    public static void main(String args[]) throws IOException{  ↓
        //---変数宣言↓
        int tensu;↓
        BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
         ↓
        //---プログラム本体↓
        System.out.println("点数を入力して下さい");↓
        tensu= Integer.parseInt(br.readLine());↓
        if (tensu >= 60) {↓
            System.out.println("合格");↓
        }↓
        else{↓
            System.out.println("不合格");↓
        }↓
    }   //---mainの終りのカッコ↓
    ↓
}  //---KeyIn1_2の終りのカッコ[EOF]
```

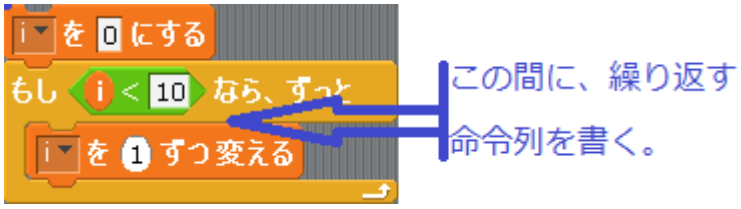
（補足：if と else の後の｛　｝は、｛　｝内の命令が１つの場合省くことができる）

条件は次のように書きます。

| 大小関係 | java |
|---|---|
| a ＝b | a ==b |
| a ≠b | a !=b |
| a ＜b | a <b |
| a ≦b | a <=b |

（６）　繰り返しの for 文

SCRATCH でいうと、次が for 文の形で、矢印の部分に繰り返す命令列を書く。




for 文では、繰り返し数を管理する整数変数が必要になる。この変数には、しばしば、i, j, k といった簡単な名前の変数が用いられる。

例１：３回 Hello と出力するプログラム。

```
public class For0 {
  public static void main(String[] args) {
    for(int i=0; i<3; i++){
        System.out.println("Hello");
    }
  }
}[EOF]
```

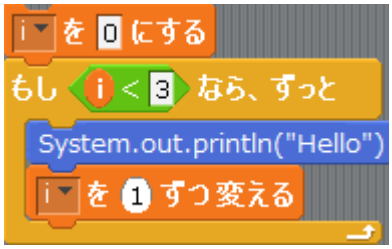
このプログラム例１は、for(…)の（…）内で、

最初に、繰返し数を管理する変数 i を宣言し、初期値０を代入している、

次に、i の繰り返し条件を示し（この例では i<3 なので i は、0,1,2,なら条件を満たす）

最後に、繰り返し命令列の最後に i++で i に１を加える。

SCRATCH なら、こうなります。




もう少しフォーマルに書くと、for 文は次の形式（フォーマット）です。

```
for(繰返し制御変数の初期設定；繰返し条件；繰返し制御変数の変更) {
      繰返し実行するプログラム
}
```

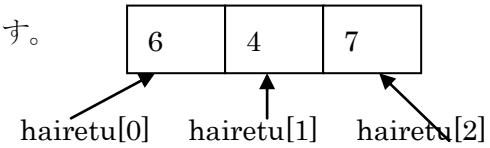
for 文が実行されると先ず、“繰返し制御変数の初期設定”が実行され、次に“繰返し条件”がチェックされ、条件が満たされていれば“繰返し実行するプログラム”が実行されます。その後、“繰返し変数の変更”が行われ、再び“繰返し条件チェック”に戻ります。

（７）　１次元配列

右図は配列名 hairetu（配列名も変数名と同じで自分の覚えやすい名前を付ける)、配列要素（数値の入るマス）数３の１次元配列です。

この配列を宣言して、ついでに右のように初期値を入れる（代入するという）には次のようにします。

int　hairetu[]={6,4,7};




配列要素を全て 0 に初期化するには、次のようにします。

hairetu[0]= 0;

hairetu[1]= 0;

hairetu[2]= 0;

ここで注意が必要なのは、配列の要素を指定する場合、一番左の要素が 0 だということです（SCRATCH では 1 でした)。

配列の操作は、配列の要素を指すインデックス変数を用いると「for 文の繰り返し」を使い簡単に実現できます。

次の例では、変数 i がインデックス変数です。

```
class Array0{
    public static void main(String args[]){
        int a[] = { 1,2,3,4,5 };
        int b[] = { 5,5,5,5,5 };

        for(int i=0; i<5; i++){
            a[i]= 0;
            System.out.println("a"+i+"="+a[i]);
        }

        for(int i=0; i<5; i++){
            System.out.println(b[i]);
        }

        for(int i=0; i<5; i++){
            b[i]= a[i];
        }

        for(int i=0; i<5; i++){
            System.out.println(b[i]);
        }
    }
}
```

このプログラムでは、

先ず、配列 a の全ての要素に 0 を代入し（代入と同時に、代入した数値を出力している)、

次に、配列 b の中身を表示し、

次に、配列 a の中身を配列 b にコピーし、
最後に、配列 b の中身を表示しています。