オブジェクト指向哲学
～入門編～

第1回　プログラムの基本

～他人が読めるプログラムを書く～

学習目標

- Java プログラムを書き、動かせる
  - コンパイルし、実行できる
- 他人が読めるプログラムが書ける
  - インデントをつけることができる
  - 適切な名前をつけることができる
  - 適切なコメントをつけることができる
- 基本的な Java 文法を説明できる
  - 変数と代入について説明できる
  - 出力する方法について説明できる
  - 条件分岐の仕組みを説明できる

1.1.　Java プログラムを動かしてみましょう

1.1.1.　基本的な流れ

まず、Java 言語でソースコード（いわゆるプログラム）を書きます。その後コンパイラを使ってコンパイルし、バイトコードというコンピュータが実行できる形式(厳密には違う)に変換します。それを「実行」することによってプログラムが動きます。

用語

●ソースコード･･･

●コンパイル･･･

●バイトコード･･･

●実行･･･

1.1.2.　実際にやってみましょう

Windows 上で一番単純な開発環境を構成します。

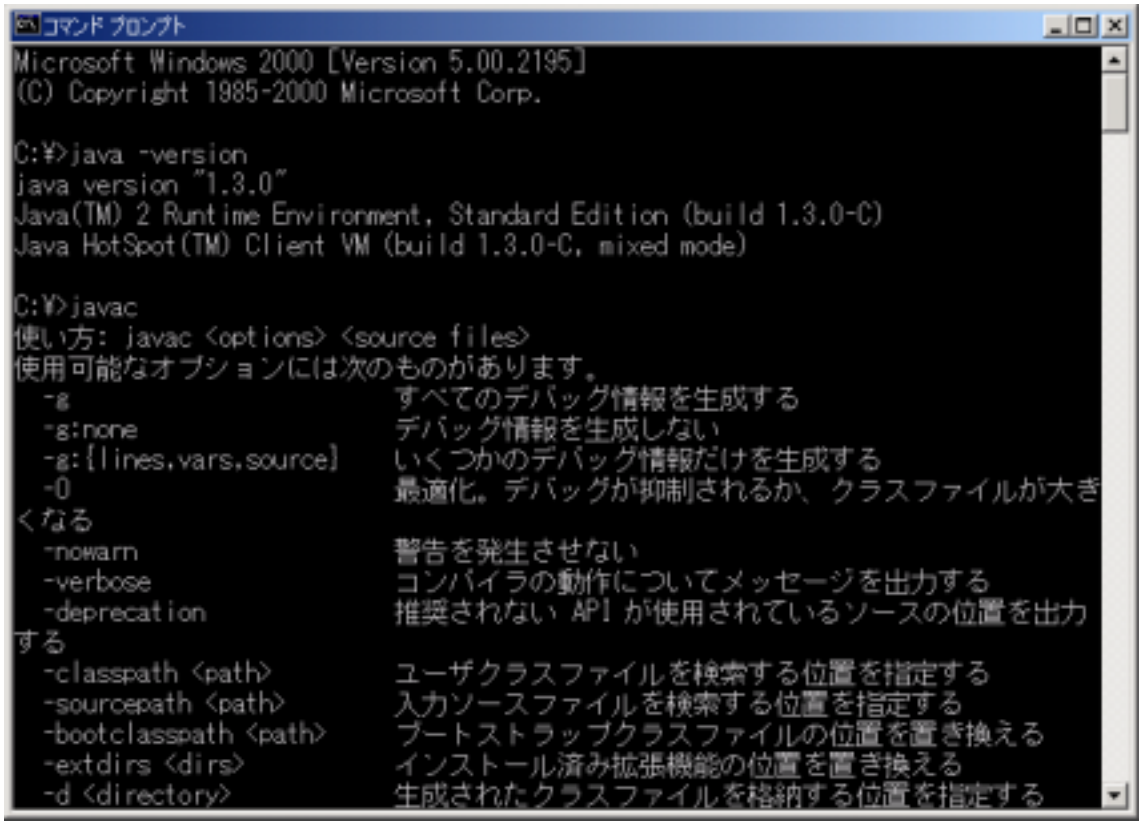
Forte,Jbuilder などの統合開発環境を利用する場合は、ソフトウエアのマニュアルを参照してください。

①.JDK のインストール

Java で書かれたプログラムを動かすには、「コンパイラ」と「実行プログラム」が必要です。これは Java SDK と呼ばれる Sun が無償配布している開発環境に含まれていて、Sun の Web ページ(http://java.sun.com/j2se/)からダウンロード可能です。まだインストールされていない場合は、インストールをして下さい。インストールをしたら、DOS 窓で使えるかどうか確かめてください。環境によっては、環境変数を設定する必要があります。

＜確かめる方法＞

DOS 窓で「java」コマンド（実行プログラム）と「javac」コマンド（コンパイラ）を実行してみます。

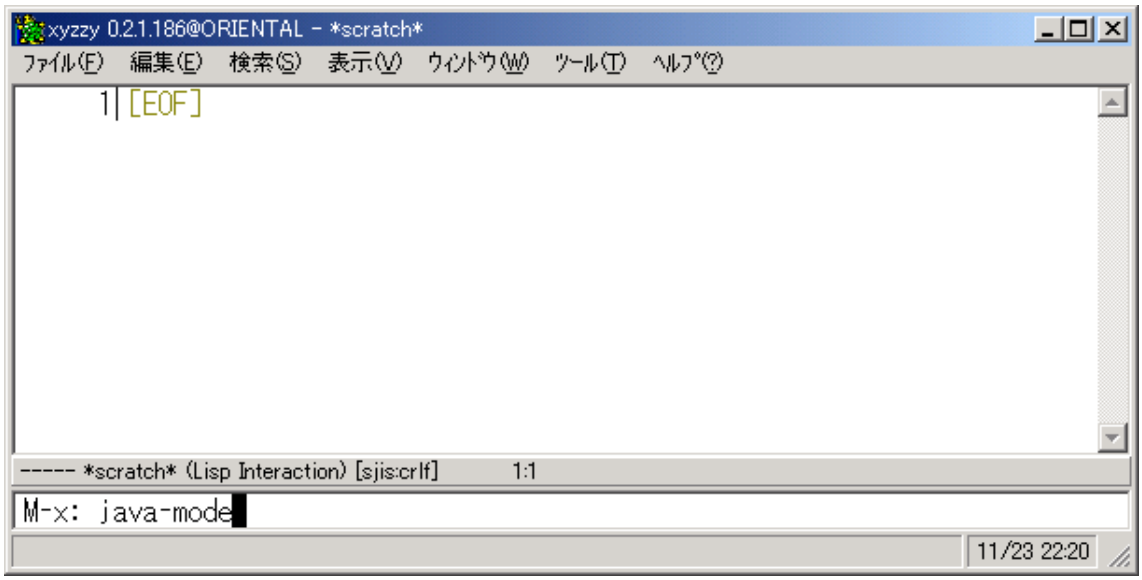


②.エディタを開く

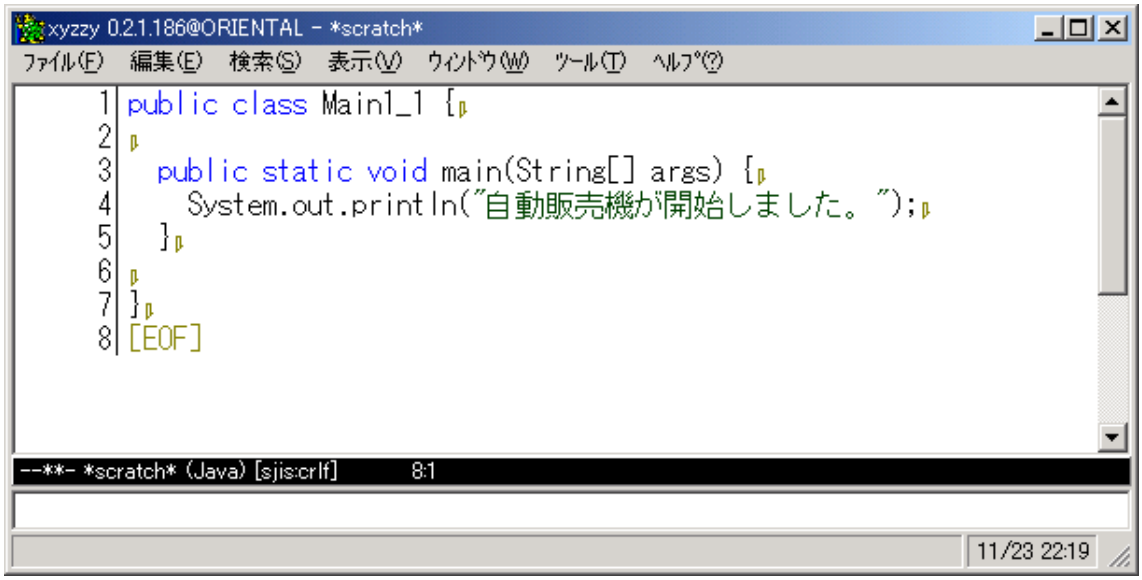
エディタは何でもいいのですが、「xyzzy」がお勧めです。Vector(www.vector.co.jp)からダウンロードできます。メモ帳は拡張子を指定できないのでお勧めしません。

「xyzzy」を起動したら、プログラムを書く前に「Esc」＋「x」を押して、「java-mode」と入力してリターンします。するとインデントや文字の色を変えてくれるので便利です。

①Java－Mode にする



②言葉の種類ごとに色を変えてくれる



③.ソースコードを書く

エディタを使ってソースコードを書きます。まず、一番簡単なプログラムを書いてみましょう。

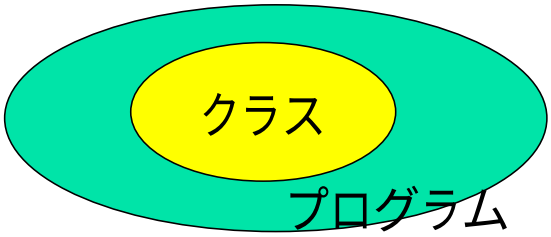
例題 1-1：プログラムを動かしてみよう(Example1_1.java)

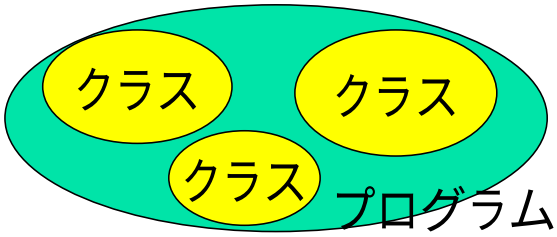
```
/**
* オブジェクト指向哲学　入門編
* 例題 1-1：プログラムを動かしてみよう
* 自動販売機プログラムの開始を知らせるプログラム
*
* メインクラス
*/
public class Example1_1 {

  /**
  * プログラム･メイン（ここからプログラムが始まる）
  * 自動販売機プログラムの開始を知らせるプログラム
  */
  public static void main(String[] args) {

    System.out.println("自動販売機のサービスが開始されました。");//自動販売機
プログラムの開始を知らせる
  }

}
```

Example1_1.java

(1)class って何？

クラスは、Java におけるプログラムの単位です。（詳しくは第４回からやります）Java では、すべてのプログラムはクラスの中に書かれます。class と書いた後にクラス名を書きます。これはプログラムの名前になります。今回は「Example1_1」となります。



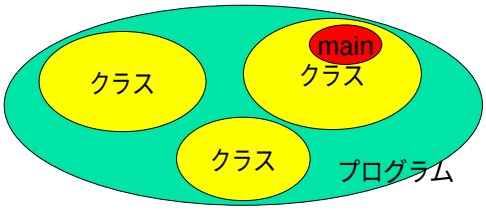


☆注意☆

ファイルの名前とクラスの名前を同じにする必要があります。クラスの名前を「Example1_1」としたら、ファイルの名前は「Example 1_1」に拡張子「.java」を付けた「Example 1_1.java」にしなければ、プログラムは動きません。

Java では大文字と小文字を区別するので注意しましょう。

(2)main **って何？**



main()メソッド（第３回で詳しくやります）は、Java のプログラムのスタート地点です。各プログラムに必ず一つ必要です。

(3)System.out.println(“**自動販売機のサービスが開始されました**”)**って何？**

この行は、""内の文字列を標準出力（コンソール）に表示しなさいという意味です。

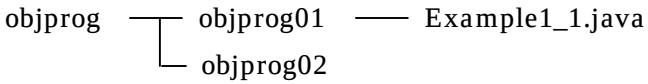
(4)public void String **とかって何？**

この先で徐々に説明します。いまはおまじないですが、本講座が終わる頃にはすべての意味がわかるはずです。

④.保存する

まず、「オブジェクト指向哲学」演習用フォルダを作ります。名前は何でもいいですが、ここでは、「objprog」というフォルダを作ることにして、説明します。

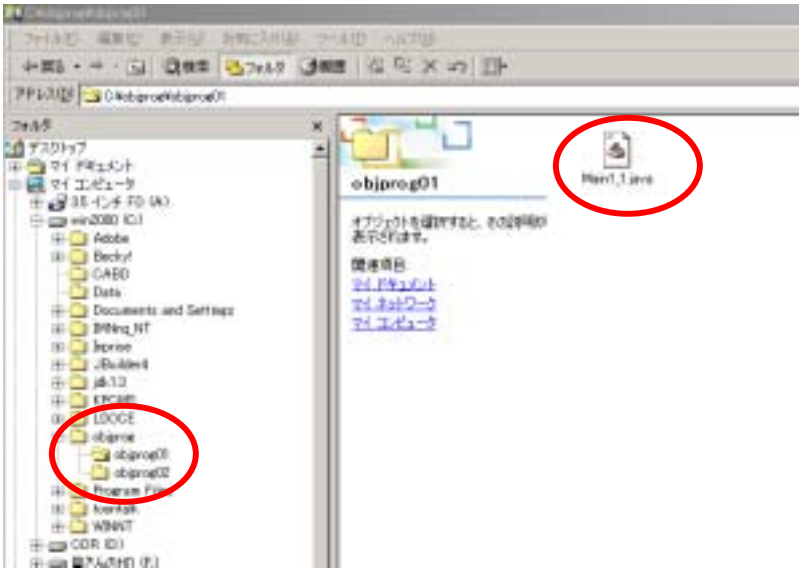
- フォルダ構成



objprog ディレクトリの下に、毎回の作業フォルダを作ります。作業フォルダの名前は、「objprog??」の形式で、名前を付けます。今回の場合は objprog01 になります。
毎回の演習で作るファイルは、すべてその作業フォルダに保存することにします。

- 具体例

この例は、演習用フォルダを「C:¥objprog」にした場合の実際にフォルダを作り、一番初めのソースコード（Example1_1.java）を保存したものです。



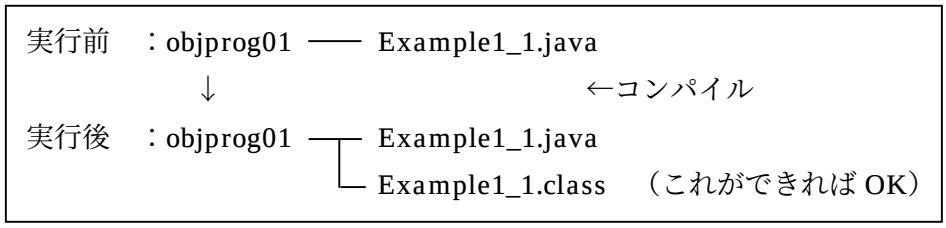
⑤.コンパイルする

ソースコードができたら、コンパイルして、バイトコードに変換します。ソースコードの拡張子は「.java」バイトコードの拡張子は「.class」です。.class ファイルができたら、コンパイルは成功です。

- コンパイルの手順
  1. objprog01 フォルダ（ディレクトリ）に移動
  2. コンパイルするには、javac コマンドを使います。

> ＜javac コマンドの使い方＞
> javac [ソースコードファイル名]
> ex)今回の場合「javac Example1_1.java」と入力する

  3. Example1_1.class　ファイルができれば成功



- 具体例

この例は、演習用フォルダを「C:¥objprog」にした場合のコマンド例です。

```
C:\> cd objprog
C:\objprog> cd objprog01
C:\objprog\objprog01> dir
Example1_1.java

C:\objprog\objprog01> javac Example1_1.java

C:\objprog\objprog01> dir
Example1_1.java
Example1_1.class
```

⑥.実行する

コンパイルが成功したら、実行してみましょう。「.class」ファイルがあることを確認したら、実行コマンドを入力してみましょう。プログラムが動き、「自動販売機のサービスが開始されました。」のメッセージがでたら、実行成功です。

- 手順
  4. バイトコードがあることを確認します。
     dir コマンドを使い、.class ファイルがあることを確認しましょう。
  5. 実行コマンド（java）を入力します。

> ＜java コマンドの使い方＞
> java [クラス名]
> ex)今回の場合「java Example1_1」と入力する
>
> ※注意！　java の後はクラス名です。ファイル名ではありません。
> ファイル名から「.class」を除いたものがクラス名です。

  6. 「自動販売機のサービスが開始されました。」とコンソールに表示されるのを確認します。

- 具体例

この例は、先ほどコンパイルした環境でそのまま実行する時のものです。

```
C:\objprog\objprog01> dir
Example1_1.java
Example1_1.class

C:\objprog\objprog01> java Example1_1
自動販売機のサービスが開始されました。

C:\objprog\objprog01>
```

1.2.　自動販売機プログラミング

本講座は、自動販売機を作りあげる過程を通して、オブジェクト指向の考え方を学んでいくものとなっています。最初は、ちっぽけなプログラムですが、徐々に本格的なソフトウエアになっていき、最後には、下図のような GUI を備えた、自動販売機が完成します。

1.2.1.　買うことのできる商品を提示する

これからいよいよ自動販売機をプログラミングしていきます。自動販売機の使命は商品を売ることです。そのためにはまず、利用者に買うことのできる商品種類のリストを提示しなければいけません。しばらくは、商品のリストを提示するプログラムを書いていきます。

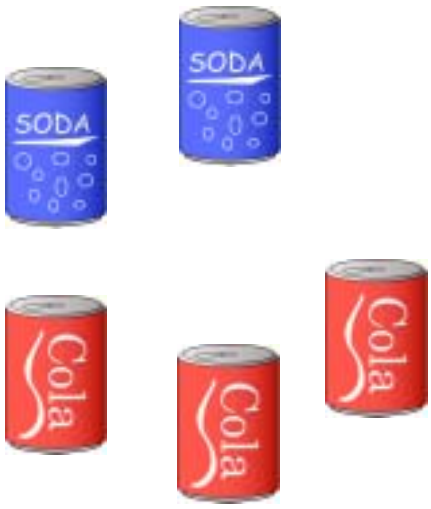
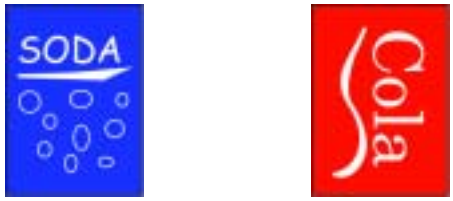
そのための、いくつかの注意点を挙げておきます。

①.商品と種類

本講座のプログラムでは、最後まで「商品」と「種類」を区別してプログラミングしていきます。

一つ理由を挙げると、自動販売機の利用者は商品を種類（コーラ、ソーダ）で選びます。コーラなら何でもいいわけです。実際の商品(このコーラ、あのコーラ)では選ばないし、選べないのです。

本講座では、実際の商品(このコーラ、あのコーラ)のことを「商品」、種類（コーラ、ソーダ）のことを「商品種類」と呼びます。ですから、買うことのできる商品を提示するということは、商品種類のリストを提示するということになります。

	イメージ	本講座での呼び名
実際の商品 (このコーラ、あのコーラ)		「商品」
種類 （コーラ、ソーダ）		「商品種類」

②.商品種類を番号で扱う

商品種類リストを表示するためには、「商品」ではなく、「商品種類」を表示すべきだということは分かりました。ということは、「コーラ」「ソーダ」を表示すればいいのですが、文字列をプログラムで扱うのは難しいので、第３回までは、番号で扱います。これを「商品番号」と呼ぶことにします。

商品番号の例

コーラ	1001
ソーダ	1002
お茶	1003
DD レモン	1004

1.3.　他人が読めるプログラムを書く

他人が読めるプログラムを書くのは基本中の基本です。例えば、次の例題 1-2 を見てみましょう。何をするプログラムだかわかりますか？「プログラムなんて動けばいい！」と思っている人も、本講座を通して、他人が読めるプログラムの重要性に気づいていただければ幸いです。

例題 1-2(Example1_2.java)

```
public class Example1_2 {
public static void main(String[] args) {
System.out.println("自動販売機のサービスが開始されました。");
int x;
x = 1001;
System.out.println(x+"は販売中です");
}
}
```

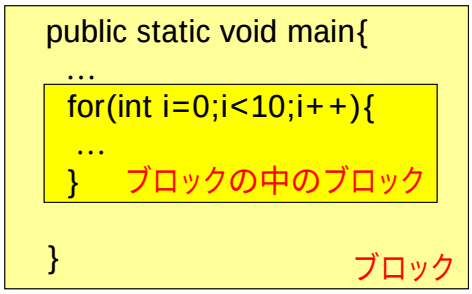
1.3.1.　人が読めるプログラムとは？

①.インデントが付けられている

例題 1-2 改①(Example1_2.java)

```
public class Example1_2 {
  public static void main(String[] args) {
    System.out.println("自動販売機のサービスが開始されました。");
    int x;
    x = 1001;
    System.out.println(x+"は販売中です");
  }
}
```

(1)ブロックとは？



②.変数に適切な名前が付けられている

例題 1-2 改②(Example1_2.java)

```
public class Example1_2 {
  public static void main(String[] args) {
    System.out.println("自動販売機のサービスが開始されました。");
    int itemType;
    itemType = 1001;
    System.out.println(itemType+"は販売中です");
  }
}
```

(1)Java **での名前付け規則**

	規則	例
クラス名	● 最初の文字は大文字 ● 複数の単語のときは各単語の最初の文字を大文字に	Name MyName ItemType
メソッド名、変数名	● 最初の文字は小文字 ● 複数の単語のときは各単語の最初の文字を大文字に	name myName itemType

③.コメントがかかれている

コメントを付けてコンテキストが分からない人にも読めるソースコードにしましょう。

コメントは内容が重要です。内容については次節で詳しく議論します。

(1)Java でのコメントの記述方法

コメントの記述方法を間違えると、コンパイルエラーになるので注意。

範囲指定コメント

プログラム中、「/*」から「*/」までのすべての文字は、コメントとして扱われます。複数行でもかまいません。

Java の慣習として（Javadoc というドキュメントを自動生成するため）始まりは「/**」とし、行の始めに「*」を付けます。

Java において範囲指定コメントが使われる場面は、

- クラス･コメント（後述の見出しコメントに相当）
- メソッド･コメント（後述のブロックコメントに相当）

```
/*　この中がコメント　*/
```

```
/**
* Java での標準形式
*
*/
```

行コメント

プログラム中、「//」からその行の最後までのすべての文字は、コメントとして扱われます。次の行までの効果はありません。

```
int x;//この行のここから先はすべてコメント
```

1.3.2.　コメントの書法

①.コメントとプログラムの目的

(1)コメントを付けてみよう

例題 1-2 改③：コメントを付けてみよう①(Example1_2.java)

```
public class Example1_2 {
  public static void main(String[] args) {
    //
    System.out.println("自動販売機のサービスが開始されました。");
    //
    int itemType;
    //
    itemType = 1001;
    //
    System.out.println(itemType+"は販売中です");
  }
}
```

(2)コメントをつけるときのポイント

-
-
-
-

(3)改めてコメントを考えてみよう

例題 1-2 改④：コメントを付けてみよう②(Example1_2.java)

```
public class Example1_2 {
  public static void main(String[] args) {
    //
    System.out.println("自動販売機のサービスが開始されました。");
    //
    int itemType;
    //
    itemType = 1001;
    //
    System.out.println(itemType+"は販売中です");
  }
}
```

②.目的の階層構造

「目的－手段」の関係に注意しましょう。

- 自動販売機のプログラム
  - 商品種類を提示する
    - 自動販売機が開始したことを知らせる
    - 取扱う商品種類を保存する変数を定義する
    - 取扱う商品種類を追加する
    - 保存してある取扱う商品種類を提示する

③.コメントの種類

(1)見出し･コメント

- プログラムの一番初めに書くコメント（Java では、class 宣言の前に書く）
- 大目的を書く（そもそもこのプログラムは何のために書いたのか）
- 日付や、作成者なども入れると良い

(2)ブロック･コメント

- ブロックの中は何を目的としてかかれているのかを書くコメント（Java では、ブロックが始まる前に書く）
- 中目的を書く

(3)行･コメント

- その行は何を目的としてかかれているのかを書くコメント（Java では、プログラムの後に書く
- 行だけに通用する小目的を書く

例題 1-1：プログラムを動かしてみよう(Example1_1.java)

①

```
/**
* オブジェクト指向哲学　入門編
* 例題 1-1：プログラムを動かしてみよう
* 自動販売機プログラムの開始を知らせるプログラム
*
* メインクラス
*/
public class Example1_1 {

  /**
  * プログラム･メイン（ここからプログラムが始まる）
  * 自動販売機プログラムの開始を知らせるプログラム
  */
  public static void main(String[] args) {

    System.out.println("自動販売機のサービスが開始されました。");//自動販売機プログラムの開始を知らせる
  }

}
```

②

③

＜考えよう！＞次のソースプログラムにコメントを加えてみよう

練習問題　コメントを書いてみよう

```
public class Practice1_1 {




  public static void main(String args[]){


    int a = 49;
    int b = 73;
    int c = 100;
    int d = 45;
    int e = 25;


    double x = a + b + c + d + e;
    double y = x / 5.0;


    y = y * 10;
    if((y % 10) >= 5){
      y = y + 10;
    }
    int z = (int)(y / 10);


    System.out.println(z);

  }

}
```

1.4.　たくさんの商品種類を扱う

1.4.1.　　変数を用意する

一番プリミティブな方法は、簡単に１０個の変数として扱う方法です。

例題 1-3 は、単に変数を１０個に増やしただけです。

例題 1-3：商品種類を複数取り扱う(Example1_3.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 1-3：商品種類を複数取り扱う
4:      * 取扱う商品種類を追加して表示するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example1_3 {
9:
10:       /**
11:       * プログラム･メイン（ここからプログラムが始まる）
12:       * 取扱う商品種類を追加して表示するプログラム
13:       */
14:       public static void main(String[] args) {
15:
16:         //自動販売機プログラムの開始を知らせる
17:         System.out.println("自動販売機のサービスが開始されました。");
18:
19:         //取扱う商品種類を保存するための変数を定義する
20:         int itemType01;
21:         int itemType02;
22:         int itemType03;
23:         int itemType04;
24:         int itemType05;
25:         int itemType06;
26:         int itemType07;
27:         int itemType08;
28:         int itemType09;
29:         int itemType10;
30:
31:         //取扱う商品種類を保存するための変数を初期化する
32:         //何も入っていないことを-1 で表す。
33:         itemType01 = -1;
34:         itemType02 = -1;
35:         itemType03 = -1;
36:         itemType04 = -1;
37:         itemType05 = -1;
38:         itemType06 = -1;
39:         itemType07 = -1;
40:         itemType08 = -1;
```

```
41:         itemType09 = -1;
42:         itemType10 = -1;
43:
44:         //取扱う商品種類を追加する
45:         itemType01 = 1001;//コーラ
46:         itemType02 = 1002;//ソーダ
47:         itemType03 = 1003;//お茶
48:
49:         //取扱う商品種類を表示する
50:         System.out.println(itemType01+"は販売中です");
51:         System.out.println(itemType02+"は販売中です");
52:         System.out.println(itemType03+"は販売中です");
53:         System.out.println(itemType04+"は販売中です");
54:         System.out.println(itemType05+"は販売中です");
55:         System.out.println(itemType06+"は販売中です");
56:         System.out.println(itemType07+"は販売中です");
57:         System.out.println(itemType08+"は販売中です");
58:         System.out.println(itemType09+"は販売中です");
59:         System.out.println(itemType10+"は販売中です");
60:       }
61:     }
62:
```

Java Tips　―変数と代入―

- 変数の宣言

変数の宣言は、「型　変数名」で行います。

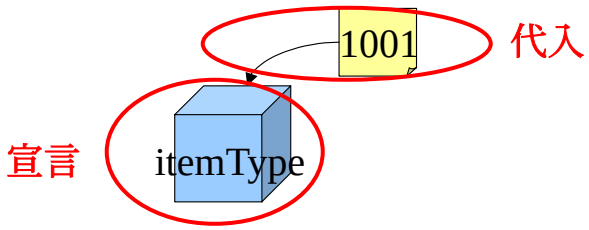
「int itemType;」と書けば、整数を入れることのできる、itemType という名前のついた変数ができます。

- 変数への代入

変数への代入は「＝」演算子を使います。

注意することは、数学の＝の意味と違って、右から左に代入するという意味になるところです。

「itemType = 1001;」と書けば、1001 という整数を itemType 変数に代入することができます。（もちろん、変数を用意していなければエラーです）

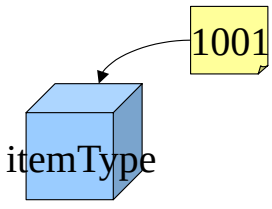


- 宣言と同時に代入

Java では、宣言と代入を一行で書くことができます。

「同時に」と書いてありますが、実際には同時には行われません。変数を作ってから代入されます。

「int itemType = 1001;」と書けば、まず itemType 変数を作り、そこに 1001 を代入します。



Java Tips　―標準出力―

- 文字列をコンソールに出力する

Java で文字列をコンソールに出力するには、System.out.println()メソッドを使います。（メソッド自体については３章で詳しくやります。）

「System.out.println」はきまり文句で、その後の「()」の中に、表示したい文字列を書きます。表示したい文字列は、ダブルクォーテーション「"」で囲む必要があります。このダブルクォーテーションは、表示したい文字列が決まっていて、直接プログラム中に書きたいときに使います。

「System.out.println("おはよう");」と書けば、コンソールに「おはよう」と表示されます。

- 変数をコンソールに出力する

表示したい文字列が分かっているときは直接書けばよいですが、変数の中身を表示したいときがあります。これはプログラムが動くまで分からないので、ダブルクォーテーションを使った方法ではできません。Java の「System.out.println()」メソッドは便利で、「()」の中に変数をいれると、変数の中身が表示されます。この場合はダブルクォーテーションで囲む必要はなく、変数名を書きます。

「System.out.println(itemType);」と書けば、itemType 変数に入っている整数が表示されます。

- 文字列と変数をつなげたものを出力する

今回は、「1001 は販売中です」と変数に入っている整数と決まった文字列を表示する必要があります。その場合は、「()」の中に書く変数や文字列を「＋」記号を使って連結することができます。もちろん、文字列はダブルクォーテーションで囲み、変数はそのまま変数名を書いてください。

「System.out.println(i+"人目のお客さん")」と書けば、i 変数に 187 が入っていたとすると「187 人目のお客さん」と表示されます。

1.4.2. 条件分岐を使う

次の例題 1-4 は、プログラムを修正して、商品が入っているときだけ出力するようにしたものです。

例題 1-4：条件分岐を使う(Example1_4.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 1-4：条件分岐を使う
4:      * 取扱う商品種類を追加して表示するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example1_4 {
9:
10:       /**
11:       * プログラム･メイン（ここからプログラムが始まる）
12:       * 取扱う商品種類を追加して表示するプログラム
13:       * 商品種類が入っていない変数の商品種類は提示しない
14:       */
15:       public static void main(String[] args) {
16:
17:         //自動販売機プログラムの開始を知らせる
18:         System.out.println("自動販売機のサービスが開始されました。");
19:
20:         //取扱う商品種類を保存するための変数を定義する
21:         int itemType01;
22:         int itemType02;
23:         int itemType03;
24:         int itemType04;
25:         int itemType05;
26:         int itemType06;
27:         int itemType07;
28:         int itemType08;
29:         int itemType09;
30:         int itemType10;
31:
32:         //取扱う商品種類を保存するための変数を初期化する
33:         //何も入っていないことを-1 で表す。
34:         itemType01 = -1;
35:         itemType02 = -1;
36:         itemType03 = -1;
37:         itemType04 = -1;
38:         itemType05 = -1;
39:         itemType06 = -1;
40:         itemType07 = -1;
41:         itemType08 = -1;
42:         itemType09 = -1;
43:         itemType10 = -1;
```

```
44:
45:         //取扱う商品種類を追加する
46:         itemType01 = 1001;//コーラ
47:         itemType02 = 1002;//ソーダ
48:         itemType03 = 1003;//お茶
49:
50:         //取扱う商品種類を表示する
51:         if(itemType01 == -1){//入ってない
52:           //何もしない
53:         }else{
54:           System.out.println(itemType01+"は販売中です");
55:         }
56:         if(itemType02 == -1){//入ってない
57:           //何もしない
58:         }else{
59:           System.out.println(itemType02+"は販売中です");
60:         }
61:         if(itemType03 == -1){//入ってない
62:           //何もしない
63:         }else{
64:           System.out.println(itemType03+"は販売中です");
65:         }
66:         if(itemType04 == -1){//入ってない
67:           //何もしない
68:         }else{
69:           System.out.println(itemType04+"は販売中です");
70:         }
71:         if(itemType05 == -1){//入ってない
72:           //何もしない
73:         }else{
74:           System.out.println(itemType05+"は販売中です");
75:         }
76:         if(itemType06 == -1){//入ってない
77:           //何もしない
78:         }else{
79:           System.out.println(itemType06+"は販売中です");
80:         }
81:         if(itemType07 == -1){//入ってない
82:           //何もしない
83:         }else{
84:           System.out.println(itemType07+"は販売中です");
85:         }
86:         if(itemType08 == -1){//入ってない
87:           //何もしない
88:         }else{
89:           System.out.println(itemType08+"は販売中です");
90:         }
91:         if(itemType09 == -1){//入ってない
92:           //何もしない
93:         }else{
94:           System.out.println(itemType09+"は販売中です");
95:         }
96:         if(itemType10 == -1){//入ってない
97:           //何もしない
```

```
98:         }else{
99:           System.out.println(itemType10+"は販売中です");
100:        }
101:      }
102:    }
```

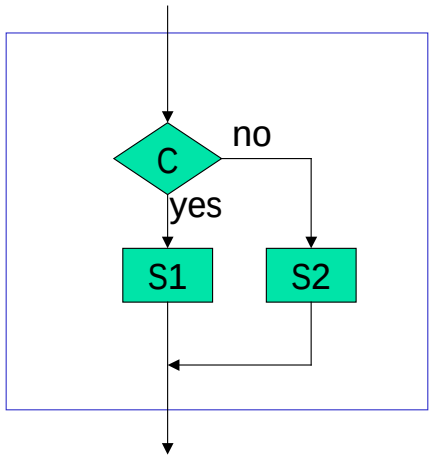
Java Tips　―条件分岐―

条件分岐は if 文を使います。条件式が成立すればその後の文を実行し、不成立の時は、else 節を実行します。else 節はなくてもよく、その場合不成立のときには何も実行しません。

```
if(itemInfo01 == -1){
        成立(true)のときはここを実行
}else{
        不成立(false)のときはここを実行
}
```

条件式

```
if(C){
  S1;
}else{
  S2;
}
```



- else if 文

3 つ以上の場合分けをしたい場合、else if 文が使えます。下記の例で行きますと、変数 i に 8 が入っている場合は S1 を実行します。その時に、２つ目の条件も満たしますが、１つ目の条件を先に満たしているので S2 は実行されません。S2 が実行されるのは、i が 1,2,3,4,5 の場合のみです。

```
if(i > 5){
        S1;
}else if(i > 0){
        S2;
}else{
        S3;
}
```

1.4.3.　さまざまな条件式

次の例題 1-5 は条件式を修正して、「ではないとき」を直接表現することによって、より他人に分かりやすくしたプログラムです。

例題 1-5:「ではないとき」条件分岐(Example1_5.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 1-5：「ではないとき」条件分岐
4:      * 取扱う商品種類を追加して表示するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example1_5 {
9:
10:       /**
11:       * プログラム･メイン（ここからプログラムが始まる）
12:       * 取扱う商品種類を追加して表示するプログラム
13:       * 取扱う商品種類が入っていない変数の商品種類は提示しない
14:       */
15:       public static void main(String[] args) {
16:
17:         //自動販売機プログラムの開始を知らせる
18:         System.out.println("自動販売機のサービスが開始されました。");
19:
20:         //取扱う商品種類を保存するための変数を定義する
21:         int itemType01;
22:         int itemType02;
23:         int itemType03;
24:         int itemType04;
25:         int itemType05;
26:         int itemType06;
27:         int itemType07;
28:         int itemType08;
29:         int itemType09;
30:         int itemType10;
31:
32:         //取扱う商品種類を保存するための変数を初期化する
33:         //何も入っていないことを-1 で表す。
34:         itemType01 = -1;
35:         itemType02 = -1;
36:         itemType03 = -1;
37:         itemType04 = -1;
38:         itemType05 = -1;
39:         itemType06 = -1;
40:         itemType07 = -1;
41:         itemType08 = -1;
42:         itemType09 = -1;
43:         itemType10 = -1;
```

```
44:
45:         //取扱う商品種類を追加する
46:         itemType01 = 1001;//コーラ
47:         itemType02 = 1002;//ソーダ
48:         itemType03 = 1003;//お茶
49:
50:         //取扱う商品種類を表示する
51:         if(itemType01 != -1){
52:           System.out.println(itemType01+"は販売中です");
53:         }
54:         if(itemType02 != -1){
55:           System.out.println(itemType02+"は販売中です");
56:         }
57:         if(itemType03 != -1){
58:           System.out.println(itemType03+"は販売中です");
59:         }
60:         if(itemType04 != -1){
61:           System.out.println(itemType04+"は販売中です");
62:         }
63:         if(itemType05 != -1){
64:           System.out.println(itemType05+"は販売中です");
65:         }
66:         if(itemType06 != -1){
67:           System.out.println(itemType06+"は販売中です");
68:         }
69:         if(itemType07 != -1){
70:           System.out.println(itemType07+"は販売中です");
71:         }
72:         if(itemType08 != -1){
73:           System.out.println(itemType08+"は販売中です");
74:         }
75:         if(itemType09 != -1){
76:           System.out.println(itemType09+"は販売中です");
77:         }
78:         if(itemType10 != -1){
79:           System.out.println(itemType10+"は販売中です");
80:         }
81:       }
82:
83:     }
```

Java Tips　―条件式―

Java での条件式は、すべて真偽値型(boolean)に直して判定します。真偽値型とは true か false のどちらかが入るデータ型です。

- Java における比較演算子(A,B は同じ型の変数とする)

==	A == B	A と B が同じ値の時 true
!=	A != B	A と B が同じ値でない時 true
>	A > B	A が B より大きい時 true
<	A < B	A が B より小さい時 true
>=	A >= B	A が B 以上の時 true
<=	A <= B	A が B 以下の時 true

＜練習問題＞

int i=100; if(i == 100)	true	boolean b=true; if(b == true)	
int i=50; if(i == 100)	false	boolean b=true; if(b == false)	
int i=100; if(i != 100)		boolean b=true; if(b != false)	
int i=100; if(i > 100)		boolean b=true; if(b)	
int i=100; if(i >= 100)		boolean b=false; if(b)	

- Java における論理演算子(A,B は真偽値型)

&&	A && B	A と B 両方とも true の時 true
||	A || B	A もしくは B が true の時 true

＜練習問題＞

boolean b1=true;　boolean b2 = false; if(b1 == true && b2 == false)	
boolean b1=true;　boolean b2 = false; if(b1 == true || b2 == true)	
boolean b1=true;　boolean b2 = false; if(b1 && b2)	
boolean b1=true;　boolean b2 = false; if(b1 || b2)	
boolean b1=false;　boolean b2 = false; if(b1 || b2)	

練習問題

＜記述問題＞

☆記述問題 1-1

他人が読めるプログラムを書くための 3 つのポイントを列挙せよ。

＜プログラム問題＞

☆プログラム問題 1-1

次のプログラムを他人が読めるプログラムにせよ。

```
public class Exercise1_1{
public static void main( String args[] ){
int weight = 55;
System.out.println( “体重は” + weight + “ｋｇです。”);
int height = 168;
System.out.println( “身長は” + height + “ｃｍです。”);
int rohrer = weight * 10000000 / height / height / height;
System.out.println( “ローレル指数は” + rohrer + “です。”);
if( rohrer < 100 ){
System.out.println(“やせすぎですね。”);
}
if( rohrer >= 100 && rohrer < 115 ){
System.out.println(“やせていますね。”);
}
if( rohrer >= 115 && rohrer < 145 ){
System.out.println(“普通ですね。”);
}
if( rohrer >= 145 && rohrer < 160 ){
System.out.println(“太っていますね。”);
}
if( rohrer >= 160 ){
System.out.println(“太りすぎですね。”);
}
}
}
```