

以下の「コマンドプロンプトでの操作」を自由にできるようにしてください。

- ・任意文字列の表示 (環境変数の記憶内容を含めて表示する方法)
- ・環境変数の変更
- ・カレントディレクトリの変更方法
- ・カレントドライブの変更方法
- ・ディレクトリの作成
- ・ディレクトリの削除
- ・ファイルコピーコマンド ファイル指定で、相対パスと絶対パスの使い分け  
『.』のカレントディレクトリと『..』の親ディレクトリの使い分け
- ・ファイル削除コマンド

そして、コマンドプロンプトの実行ファイルとそこでの環境変数の使われた方を理解ください。

- ・PATH の環境変数の使い方と、指定方法 バッチファイルの意味と実行

**問題** Javaのコンパイラjavac.exe と、クラスファイル指定でJava Virtual Machine 起動するコマンドjava.exe を、ファイル名だけで実行できるバッチファイルj.batを作成せよ。(環境変数 PATH と CLASSPATH 環境に合わせて正しく設定して、コマンドプロンプトを起動するプログラムです。)

**解答例**

以下は、JDKが、『C:\Program Files\Java\jdk1.5.0\_15』にインストールされている場合の例です。ここで、各パス環境変数にセットしている文字列内での**セミコロン『;』**は、検索パスを区切るために使われます。pathは、コマンドラインに入力する実行ファイルを探す順番を示します。以下のcmd.exeで実行されるコマンドプロンプトでは、**『実行ファイル名』**入力に対し、**まず** のパス直下にある実行ファイルを探し**『実行ファイル名』**が見つければ、それを実行します。**見つからなければ次に** のパス直下を探します。そこにも見つからなければ実行できないことになります。

```
set path=C:\Program Files\Java\jdk1.5.0_15\bin;%path%
set classpath=.;C:\Program Files\Java\jdk1.5.0_15
cmd.exe
```

さてJavaで、Java仮想マシンに必要なクラスファイルをロードして実行するのですが、そのクラスファイルを検索するパスが、環境変数のclasspathになります。上記での最初の検索位置はカレントディレクトリを意味する**ドット『.』**になってます。つまり、カレントディレクトリが最初に探すクラスファイルになります。例えば、このバッチファイルj.batをZドライブ直下のjavaの名前のディレクトリに置いてあるとして、このj.batをダブルクリックすると、このファイルがあるディレクトリがカレントディレクトリになります。

次の下線の入力操作は、Java仮想コンピュータで、JTest01.classを実行させようとしたが、そのファイルが見つからないエラーです。

```
Z:¥Java>java JTest01
Exception in thread "main" java.lang.NoClassDefFoundError: JTest01
Caused by: java.lang.ClassNotFoundException: JTest01
```

JTest01.classが存在しないか、環境変数classpathの設定にミスがあるためです。

環境変数classpathにドット『.』があって、カレントディレクトリの Z:¥Java に JTest01.classのファイルがあれば実行できるはずですが。

JTest01.classの作り方は、JTest01.javaのソースファイルをコンパイルすることでできます。以下に簡単な、3入力の合計を表示するプログラム例を示します。JTest01.java

```
import java.util.*;

public class JTest01 //publicのクラス名は、ファイル名と同じになければなりません
{
    public static int v1, v2, v3;

    public static void main(String[] arg)
    {
        Scanner stdin = new Scanner(System.in);
        System.out.print("a=");
        v1 = stdin.nextInt();
        System.out.print("b=");
        v2 = stdin.nextInt();
        System.out.print("c=");
        v3 = stdin.nextInt();

        System.out.print("a+b+c=" + v1 + v2 + v3);//
    }
}
```

このソースファイルのコンパイル例を左下、目標の実行例を右下に示します。

```
Z:¥Java>javac JTest01.java
```

```
Z:¥Java>
```

コンパイルエラーが出なければ  
JTest01.class ができます。

```
Z:¥Java>java JTest01
```

```
a=3
b=5
c=4
a+b+c=12
Z:¥Java>
```

さて、 の行の表現では、加算値でなく、連結の a+b+c=354 の結果表示になるでしょう。+演算子は左から結合しますが、一方が文字列の場合、他方を文字列に変換して、連結した文字列 (String 型) を演算結果にするからです。

つまり、v1 + v2 + v3 を先行して演算できるように、この部分をカッコで囲めば、カッコ内が先行して演算結果を出します。数値だけの演算結果はC言語と同じで、上記演算結果が得られます。

テキスト25ページまでの任意の問題のプログラム作って実行させましょう。

よく使われる文字列操作の練習です。

代入式の形態で入力した文字列から、=の左辺と右辺に分けて表示するプログラムを作成します。式の文字列から左辺の文字列を取得するstaticメソッドをgetNameの名前、右辺の文字列を取得するstaticメソッドをgetValueの名前で、それらメソッドを持つ Assignmentのクラスを作成してみましょう。

右の実行例のように動作する作品を作ります。(4回実行している例です)

これには、以下の文字列操作メソッドを使います。調べて使いましょ

indexOf

substring

trim

```
Z:¥Java>java Assignment
代入式>abc=123
abc の名前に、123 が設定
```

```
Z:¥Java>java Assignment
代入式>xy=5+4
xy の名前に、5+4 が設定
```

```
Z:¥Java>java Assignment
代入式>abcdefg
null の名前に、null が設定
```

```
Z:¥Java>java Assignment
代入式>xy=
xy の名前に、 が設定
```

```
Z:¥Java>
```

これらは、すべてStringクラスのメソッド

です。このようなメソッドの使い方は、『Sun Microsystems社のJava™ 2 Platform Standard Edition 5.0 API 仕様』のリンクで調べるのがよいでしょう。

プログラミングのログインページのメニュー先頭で、このリンクがあります。

(Internet Explorerのお気に入りに追加していつでも使えるようにして、

<http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/>)

**ステップ1**  
この全てのクラスの空いているところをクリックして、Ctrl + F で検索ウィンドウをだして、調べたいクラス名 (String) を入力します。

**ステップ2**  
見つかったクラス名をクリックします。右が、そのクラスの内容に移動します。

**ステップ3**  
希望のメソッドを調べます。

全ての情報は、ここから得られます。

解答例 Assignmentのソースです。完成させましょう。

```
import java.util.*;
public class Assignment
{
    public static String separator = "=";

    public static String getName(String str){
        int idx = str.indexOf(separator); // strの文字列からseparatorの位置を探す
        if( idx == -1 ) return null;
        String name = str.substring(0, idx); // strの文字列で、からidxの部分文字列を取得
        return name.trim(); //
    }
    public static String getValue(String str)
    {
        int idx = str.indexOf(separator);
        if (idx == -1) return null;
        String val = str.substring(idx + 1); // strの文字列のidx+1番目以降部分文字列を取得
        return val.trim();
    }
    public static void main(String[] arg)
    {
        Scanner stdin = new Scanner(System.in);
        System.out.print("代入式>");
        String s1 = stdin.nextLine();
        String n1 = Assignment.getName(s1);
        String v1 = Assignment.getValue(s1);
        System.out.println(n1 + "の名前に、" + v1 + "が設定");
    }
}
```

以上で作成した、Assignmentクラスを再利用して、次ように実行できるプログラムを作成します。(目的は、Assignmentの再利用と、同じ文字列を調べる方法を示しています。)

```
Z:¥Java>java JTest02
代入式>xxx = zzz
xxx と zzz は一致しません

Z:¥Java>
```

```
Z:¥Java>java JTest02
代入式>xxx = xxx
xxx と xxx で一致しました

Z:¥Java>
```

```
import java.util.*;
public class JTest02{
    public static void main(String[] arg){
        Scanner stdin = new Scanner(System.in);
        System.out.print("代入式>");
        String s = stdin.nextLine();
        String a = Assignment.getName(s);
        String b = Assignment.getValue(s);
        if (a.equals(b) == false){ // a と bの文字列が正しくないかの判定
            System.out.println(a + "と" + b + "は一致しません");
        }else{
            System.out.println(a + "と" + b + "で一致しました");
        }
    }
}
```