シャバドゥビ、ジャバ
ー速習Java覚書ー

岐阜経済大学　経営学部 経営情報学科　井戸　伸彦

来歴：

0．0版　2004年5月10日

スライドの構成

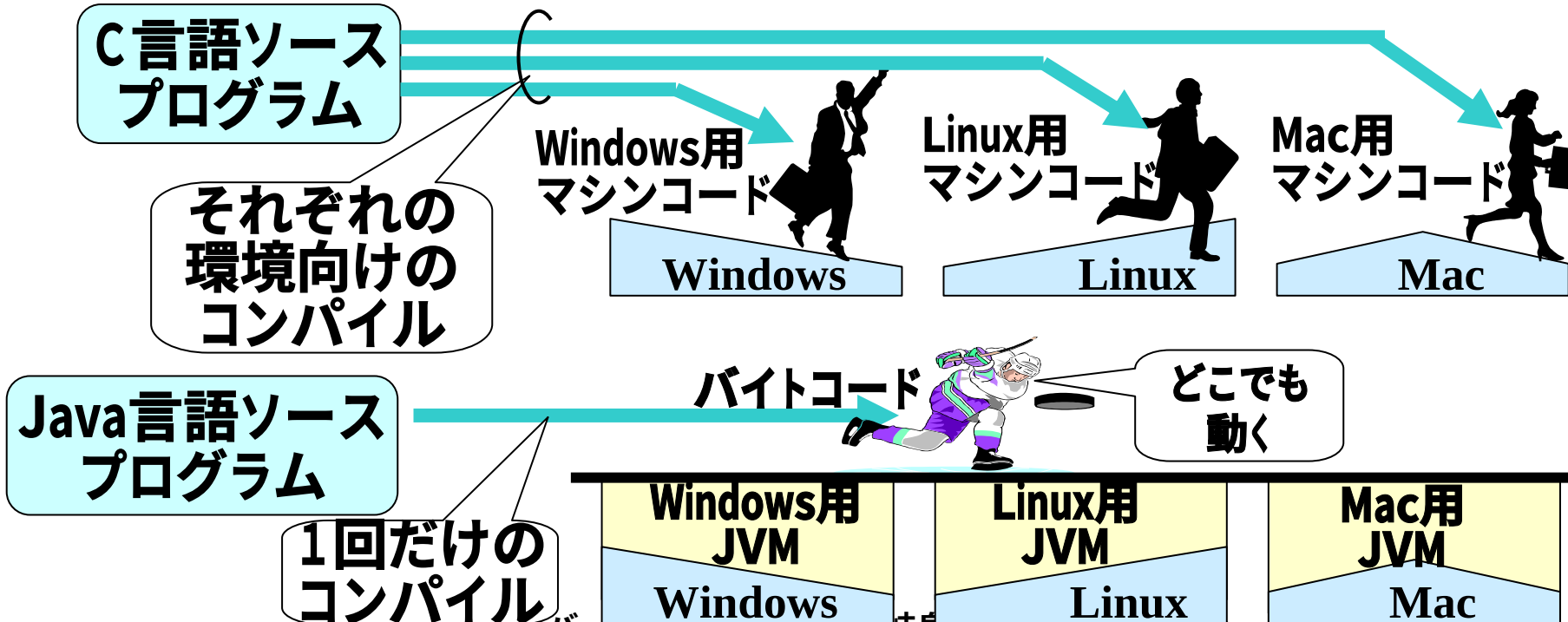
はじめに
(1) Javaの仕組み
(2) データの出力
(3) 変数
(4) 配列
(5) 演算
(6) データの入力
(7) プログラムの実行
(8) 分岐、if文
(9) 繰り返し処理
(10) 段下げ、名前空間
(11) 多次元配列
(12) メソッド
(13) 標準のメソッド
(14) ファイル入力処理
(15) ファイル出力処理
(16) 例外処理
(17) ディレクトリ・ファイルの操作

はじめに

- 本スライドは、Javaによるプログラミングを説明するために使用するスライドです。単独で教科書となることは意図していません。
- 本スライドでは、Javaの限られた機能についてのみ扱っています。また、オブジェクト指向に関する機能については扱っていません。
- オブジェクト指向に関わるJavaの仕様については、次のスライドにて説明します。
  - 「ドゥビドゥバ、ジャバ　ー直感Javaのオブジェクトー」
- Javaについてきちんと勉強したい方は、プログラミング等の講義を受講して頂くことをお願いします。
- 説明は直感的な分かりやすさを重視し、厳密には不正確な言い方もしています。
- Javaの実行・デバッグ環境としては、eclipseを前提としてます。これについては、下記の資料を参考にしてください。
  - 月に吠える　ーeclipseによるJavaアプリケーション作成ー
    (http://www.gifu-keizai.ac.jp/~ido/doc/java/eclipse_application.pdf)

(1．1) Javaのしくみ

■Java言語は、実行環境(Unix,Windows)に依存しないように工夫がされています。(例えば)C言語では、各環境用にコンパイルを行う必要があります。これに対してJavaは、各環境にJVM(Java Virtual Machine)を配備し、コンパイルではJVM上で実行されるバイトコード(＝中間言語のようなものです)を作成する方法を採ることにより、一度のコンパイルでどの環境でも実行可能なファイルが作成されます。



（１．２）３つのＪａｖａ

■Ｊａｖａアプリケーション

- 通常のＰＣ上で動作させる形式のプログラムです。本スライドでは、これのみを扱います。

■Ｊａｖａアプレット

- 端末ＰＣで開いたWｅｂブラウザ上で実行される形式のプログラムです。プログラムはWｅｂサーバからダウンロードされ、ブラウザを起動しているＰＣ上で実行される訳です。

■Ｊａｖａサーブレット

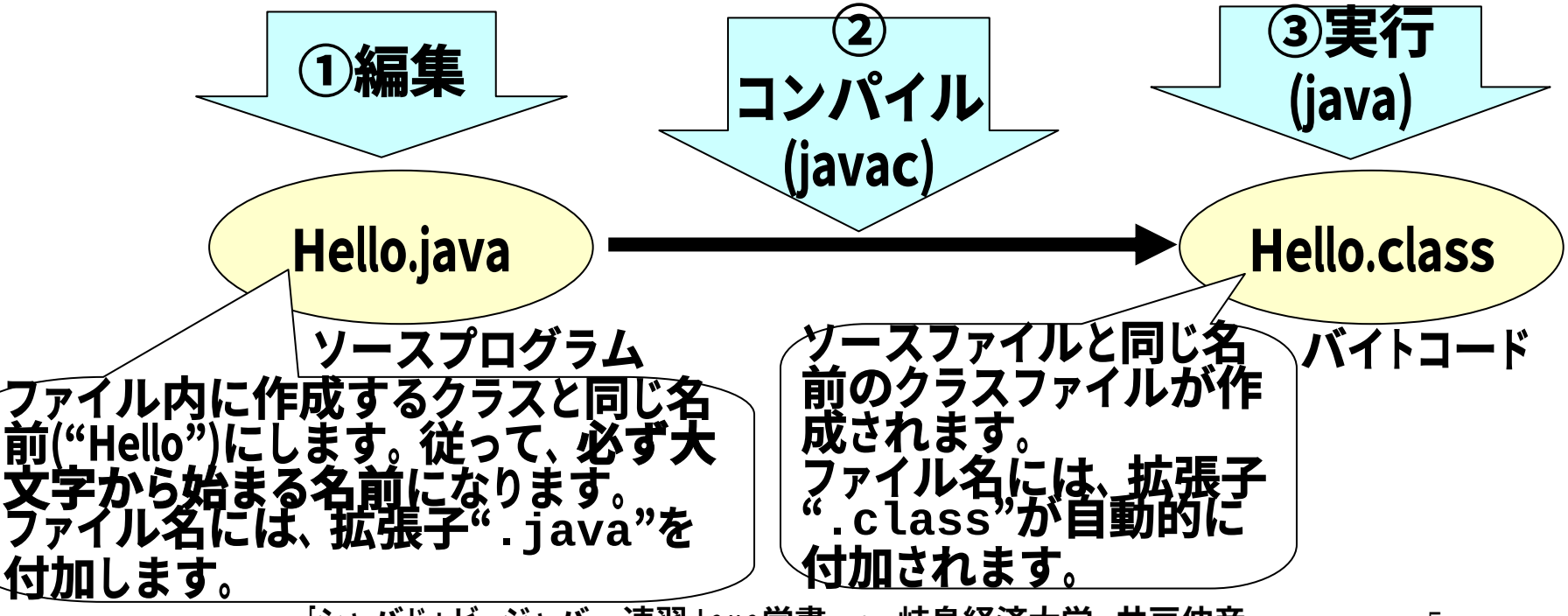
- Wｅｂサーバ上で実行され、その実行の結果作成したＨＴＭＬ等を端末に送る形式のプログラムです。

■本スライドで扱うのは、Javaアプリケーションです。

(1．3) プログラムの作成と実行

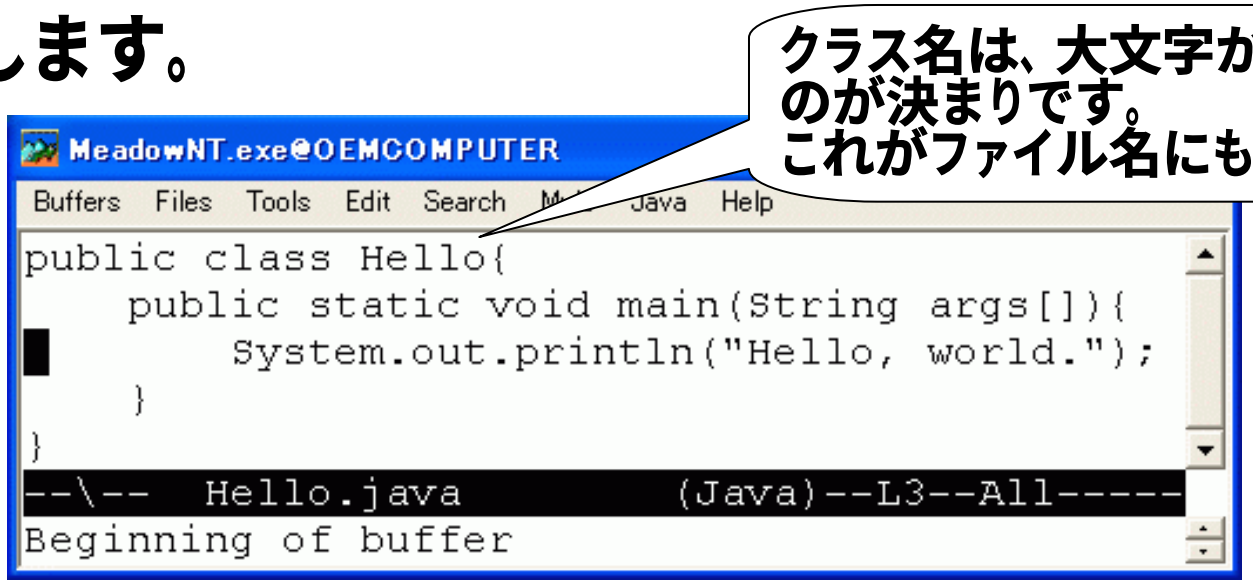
■本スライドでは、次のeclipse上でのプログラム作成・実行を前提としていますが、ここでは簡単にコマンドライン上での実行方法を説明します。

■次のように、①ソースプログラムの編集、②コンパイル、③バイトコードの実行の順で行います。

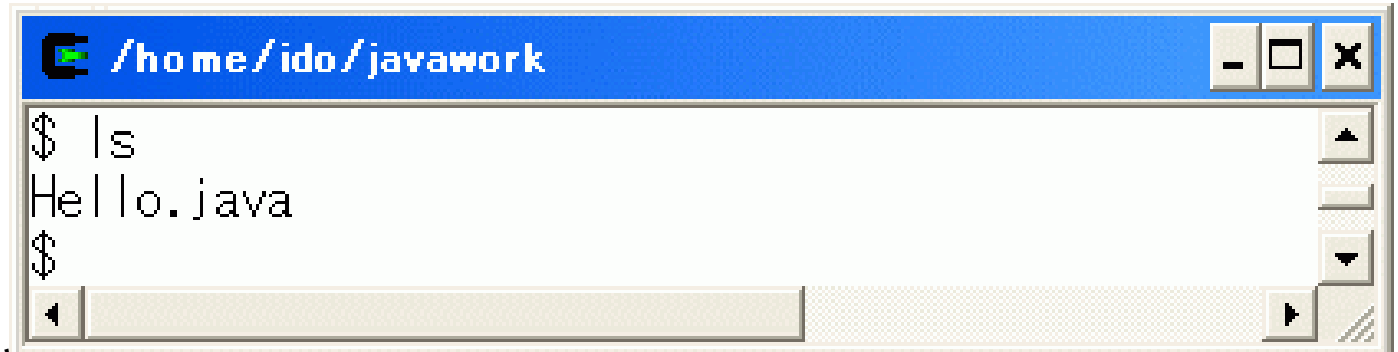


(1．3．1) ソースプログラムの編集

■適当なテキストエディタを用いて、ソースプログラムを編集します。



■編集して作成したプログラムを、コマンドラインから確認しましょう（下図はUnixイメージです）。

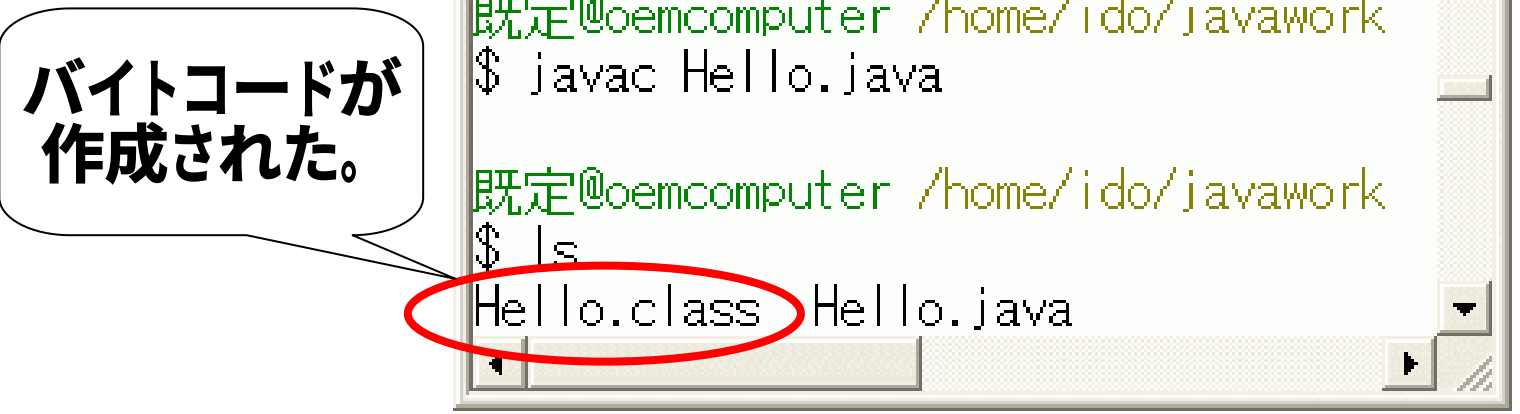


(1．3．2) コンパイル

■コマンドラインにて、次のようにコマンドを投入します。

```
$ javac Hello.java
```

■ファイル一覧を表示させると、Hello.classというファイル(バイトコード)が出来ているのがわかります。



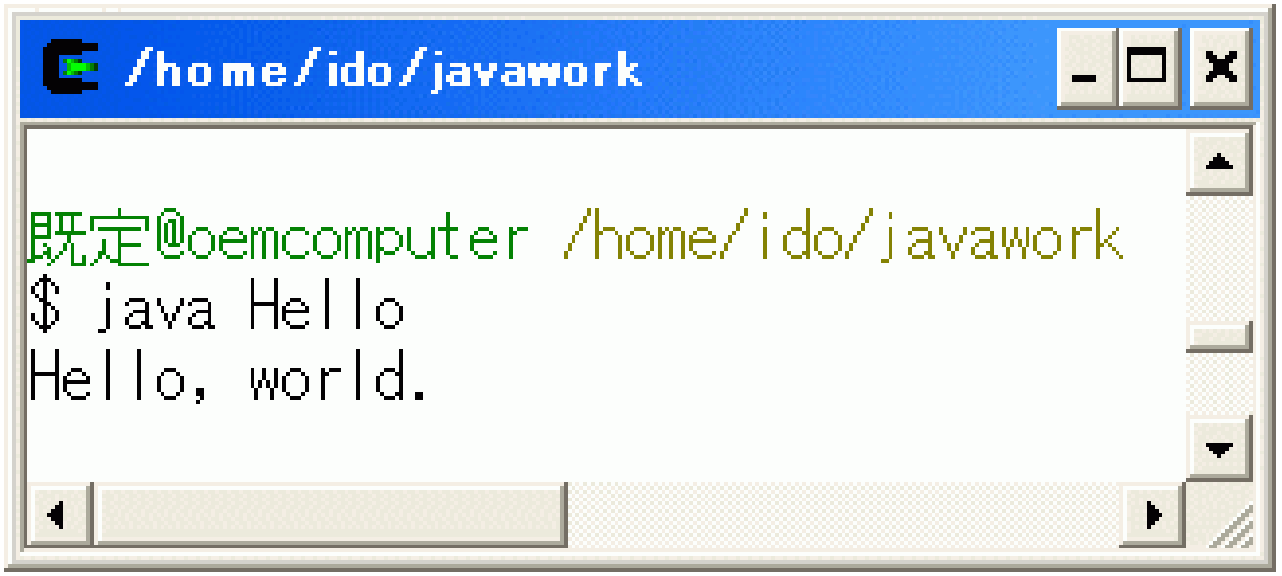
(1．3．3) 実行

■コマンドラインにて、次のようにコマンドを投入します。

```
$ java Hello
```

- 上記のコマンドで“Hello.class”を実行しているのですが、“.class”の部分はタイプしていないことに注意してください。

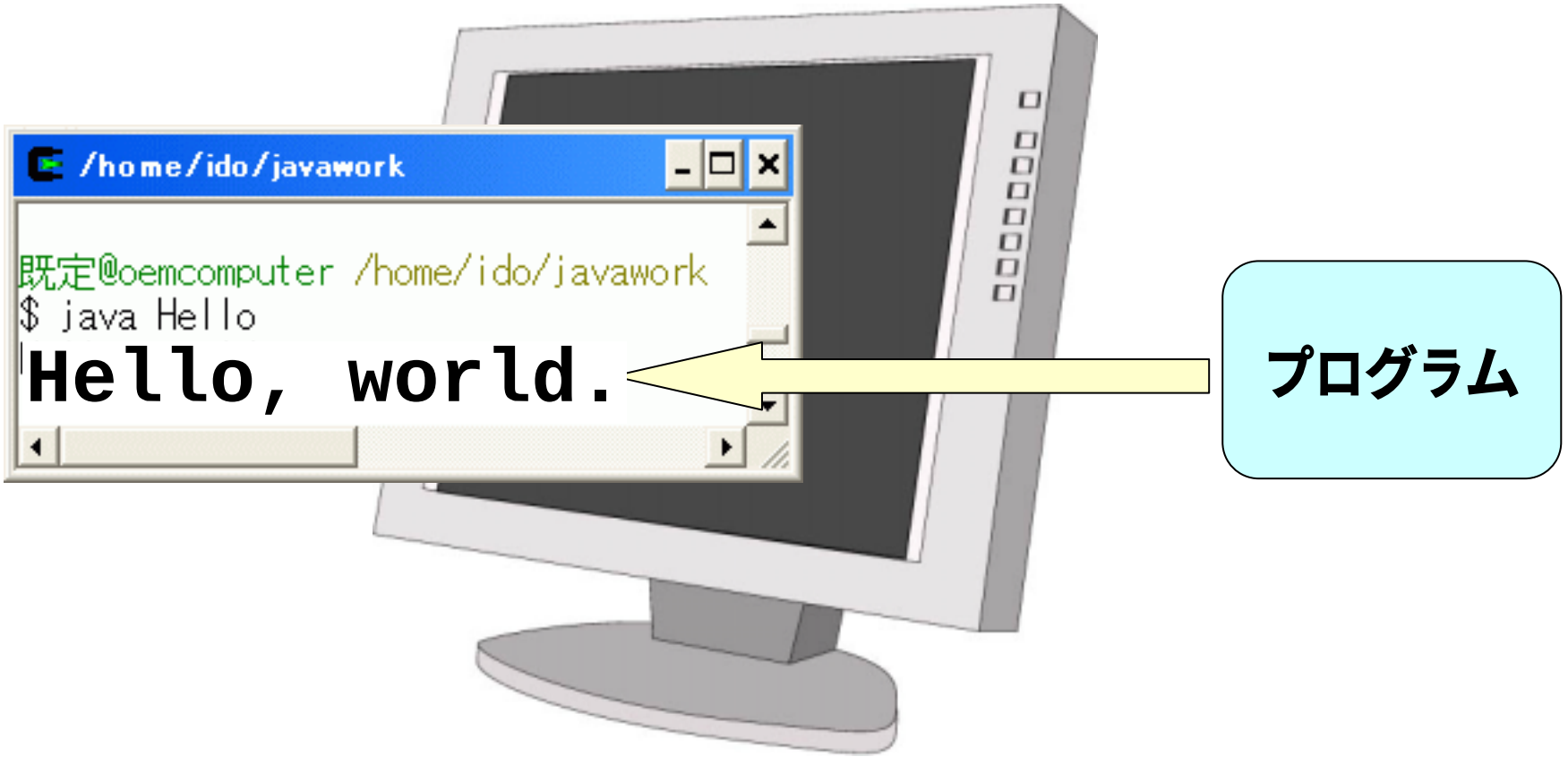
■次のように実行結果が表示されます。



(2) データの出力

■何をしたいのか？

- 作成するプログラムから、使用している端末のコマンドラインに、“Hello, world.”という文字列を出力させます。



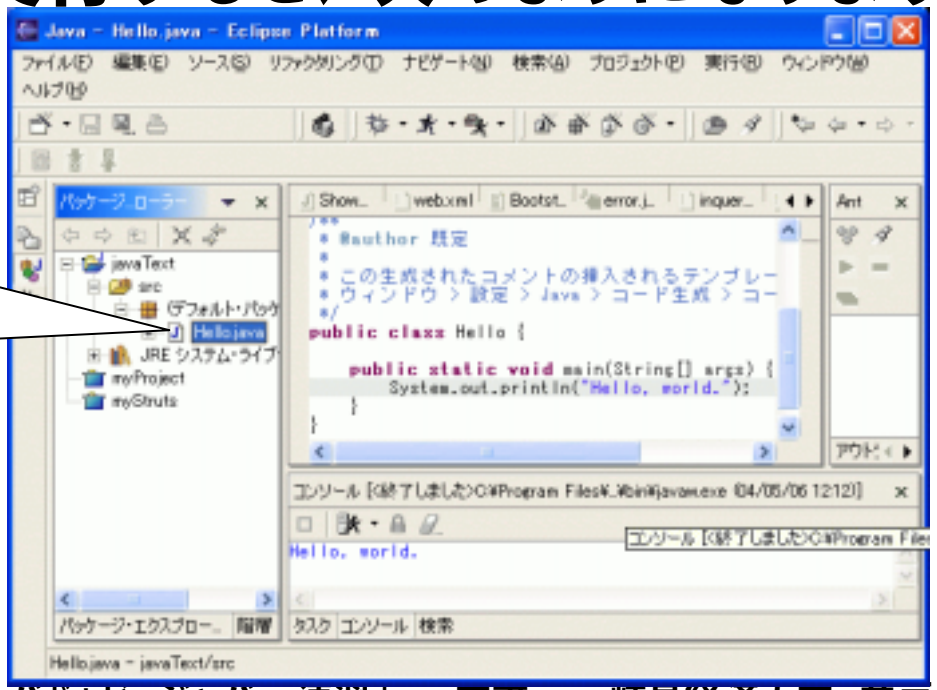
(2．1) プログラム

■次のプログラムを作成します。

前述のとおり、クラス名（“Hello”）は、大文字から始めるのが決まりです。

```
public class Hello{
    public static void main(String args[]){
        System.out.println("Hello, world.");
    }
}
```

■eclipseで実行すると、次のようになります。



前述のとおり、ファイル名は、クラス名と同じ（“Hello”）になります。

(2．2) クラス、メソッド、実行文

■プログラムの中には、クラスがあります。
■クラスの中には、メソッドがあります。
■メソッドの中に、実行文が記述されます。
■“main”という名前の特別なメソッドの中の実行文が実行される場合をしばらく見ていきます。

クラス
メソッド
実行文

```
public class Hello{

    public static void main(String args[]){

        System.out.println("Hello, world.");

    }

}
```

- 括弧の始まり(“{”)と終わり(“}”)とが対応していることに注意してください。これは有効範囲の始まりと終わりを示します。

(2.3)“println”(プリント文)

■形式

“System”の”S”は大文字

- System.out.println(＜文字列＞);

実行文の末尾には、セミコロン“;”が必要。

■概要

- 標準出力(PC画面中のコマンドライン)へ文字を出力し、最後に改行します。

■例

＜プログラム＞

```
System.out.println(”abcd”);
```

文字列は、“”で囲う

```
System.out.println(”abcd”+”efg”);
```

「 + 」は、2つの文字列を連結する。

```
System.out.println(”var=”+514);
```

数値もそのままプリント出来る。

＜出力＞

```
abcd
abcdefg
var=514
```

(2.4) “¥n”(“\n”)

■用法

- 文字列の中に“¥n” (Unixでは“\n”) を挿入すると、そこで改行する意味となります。

■例

＜プログラム＞

```
System.out.println(”ab¥ncd”);

System.out.println(”abc”+”d¥nef”);

System.out.println(”var¥n=”+514);
```

＜出力＞

```
ab
cd

abcd
ef

var
=514
```

(2.5) print

■形式

- System.out.print(＜文字列＞);

■概要

- “println”との違いは、最後に改行しないことです。

■例

＜プログラム＞

```
System.out.println(”abcd”);
System.out.println(”abcd”+”efg”);
System.out.println(”var=”+514);

System.out.print(”abcd”);
System.out.print(”abcd”+”efg”);
System.out.print(”var=”+514);
```

＜出力＞

```
abcd
abcdefg
var=514

abcdabcdef
gvar=514
```

続き、
改行されていない

(2．6) コメント

■形式

- (その1) // これ以降、この行はコメント
- (その2) /* この範囲はコメント、複数行でもＯＫ */

■用法

- プログラム中に、メモを残す場合に使います。プログラムの実行には影響を与えません。

＜プログラム＞

```
/* 本プログラムは、
   データの出力例を示すためのものである。*/
System.out.println(”abcd”); // OK

// 次の行は実行されない (コメントアウトされている)
//System.out.println(”abc”+”def”);

// 次の行は実行される
System.out.println(”var=”+514);
```

＜出力＞

```
abcd

var=514
```

(2．7) 例題2．1：文字による図形

■右のような出力を行うプログラムを作成してください。

```
#####
*####
**###
***##
****#
*****
```

■回答例：

```
public class Reidai21 {
  public static void main(String[] args) {
    System.out.println("#####");
    System.out.println("*####");
    System.out.println("**###");
    System.out.println("***##");
    System.out.println("****#");
    System.out.println("*****");
  }
}
```

(2．8) 課題

■課題2－1

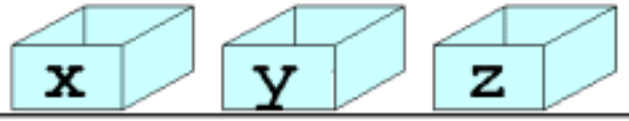
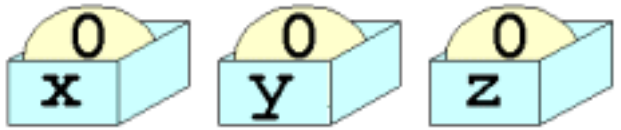
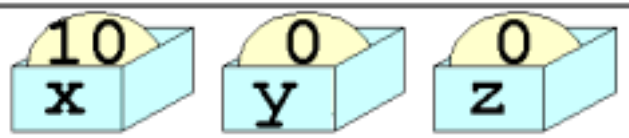
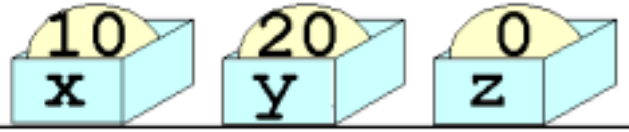
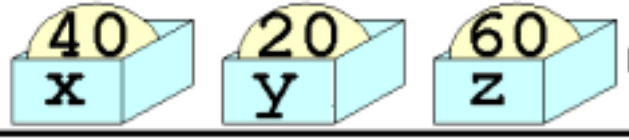
- 次のような出力を行うプログラムを作成してください。

  - ◆for文を使う必要はありません（使ってもＯＫ）。
- ヒント：“＊”は、アスタリスクです。
- ヒント：空白（“　”）も、文字として扱われます。

■この問題は簡単ですが、後に続編があります。

(3) 変数

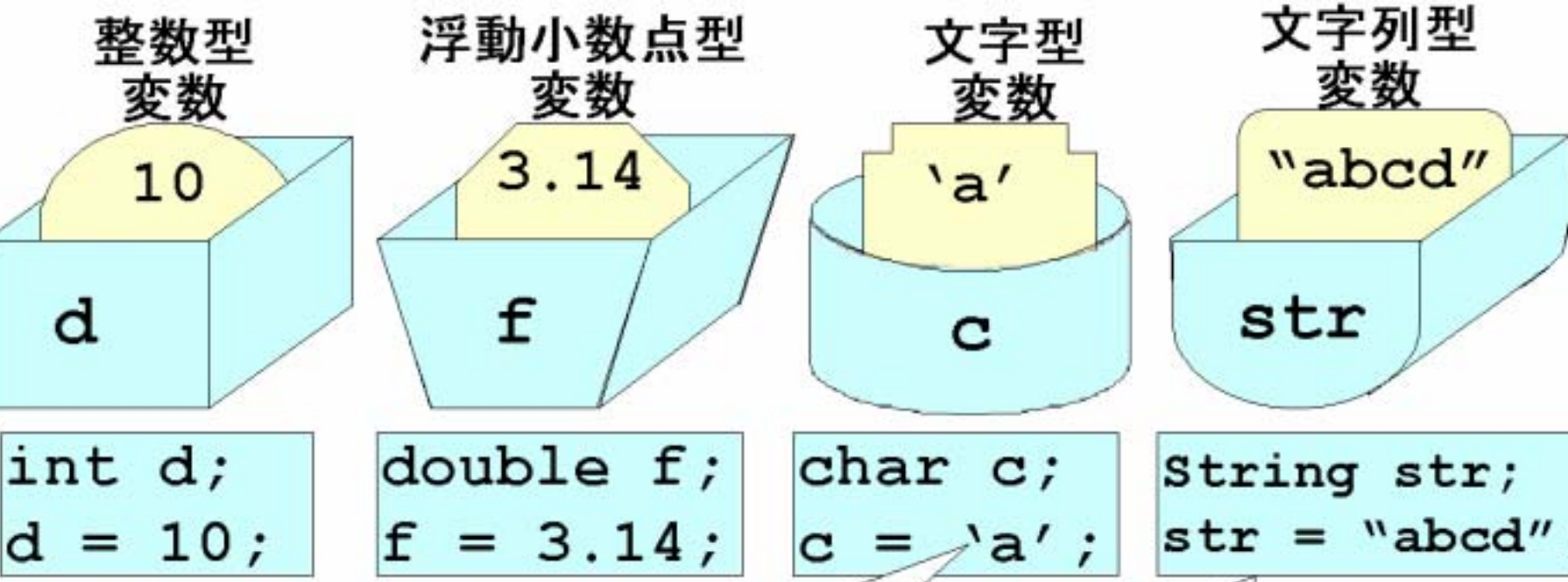
■変数とは、データを入れておく“箱”と考えます。

操作	記述例	操作後の変数の内容
変数x,y,zを用意する	`int x,y,z;`	
x,y,zの内容を0にする	`x=0;y=0;z=0;`	
xに10を入れる	`x=10;`	
yに20を入れる	`y=20;`	
xに30を加える	`x=x+30;`	
xとyを足してzに入れる	`z=x+y;`	
zの内容を出力する	`println(z);`	

(3．1) 変数の型

■どのようなデータを入れるかにより、変数(箱)にも種類があります。これをデータ型と言います。

■変数を用いる際、その前に宣言を行う必要があります。



(3．2) 変数の出力

■スライド(2.3)(2.5)に記した“println”、“print”により、変数をコンソールに出力できます。

■「＋」を使って、出力をつなぎ合わせることが出来ます。

■例

＜プログラム＞

セミコロン“;”で区切って、1行に複数の実行文を書くことが出来る

```
int d; d = 10;
System.out.println(d+”点！”);

double f = 3.14;
System.out.println(”円周率は、”+f);

char c = ‘a’;
System.out.println(c+” b c”);

String str = “Happy”;
System.out.println(str+”!!!”);
```

このような書き方もＯＫ（初期化と言います）

＜出力＞

```
10点!

円周率は、3.14

a b c

Happy!!!
```

(3．3) 課題

■課題3－1

- スライド(3.2)のプログラムを作成して実行してください。

■課題3－2

- (A)のような変数の宣言を行い、これを用いて(B)の出力を得るプログラムを作成してください。

(A)

```
String stars0="    *";
String stars1="   ***";
String stars2="  *****";
String stars3="*******";
```

(B)

```
    *
   ***
  *****
*******
  *****
   ***
    *
```

(3.4) 課題

■課題3－3

- (A)のような変数の宣言を行い、これを用いて (B)の出力を得るプログラムを作成してください。

(A)

```
int kamakura=1192;
int muromachi=1338;
int edo=1603;
String bakufu="幕府:";
String nen="年";
```

(B)

```
鎌倉幕府:1192年
室町幕府:1338年
江戸幕府:1603年
```

（3．5）課題

■課題3－4

- (A)のような変数の宣言を行い、これを用いて (B)の出力を得るプログラムを作成してください。

> セミコロン“；”で区切って、1行に複数の実行文を書くことが出来る

(A)

```
char c0='あ';   char c5='か';
char c1='い';   char c6='き';
char c2='う';   char c7='く';
char c3='え';   char c8='け';
char c4='お';   char c9='こ';
char c10='赤';
char c11='青';
char c12='黄';
char c13='丘';
char c14='顔';
char c15='声';
```

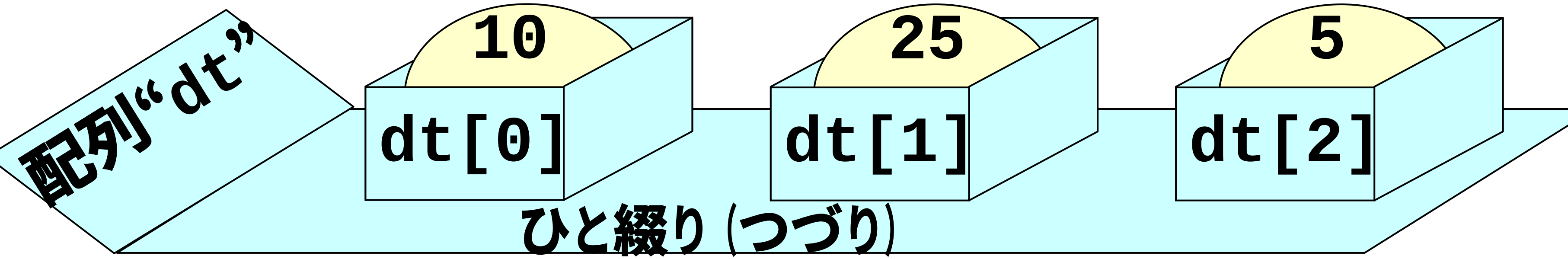
> Javaでは、日本語の1文字も、charとして扱うことが出来る。

(B)

```
赤は、あか
青は、あか
黄は、き
丘は、おか
顔は、かお
声は、こえ
```

(4) 配列

■配列は、番号のついた箱(変数)のひと綴りと考えます。



■配列の各々の要素は、変数と同じように扱うことが出来ます。

＜プログラム＞

```
int dt[] = new int[3];// スライド(4.1)参照
dt[1] = 25;
System.out.println(dt[1]+”点！”);
```

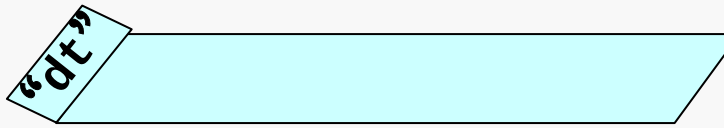
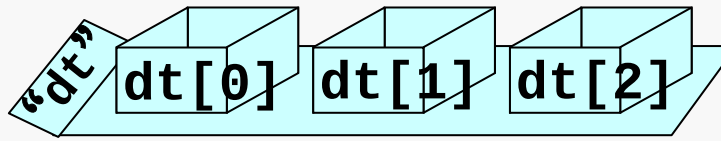
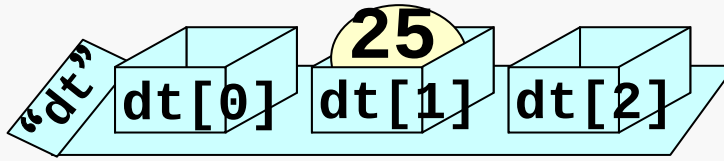
＜出力＞

```
25点
```

■ひとつの配列の各要素は、みな同じデータ型となります。番号は、“0”から始まります。

(4．1) 配列の宣言

■①配列変数の名前の宣言と、②データが格納される実体の確保とは、別になります。

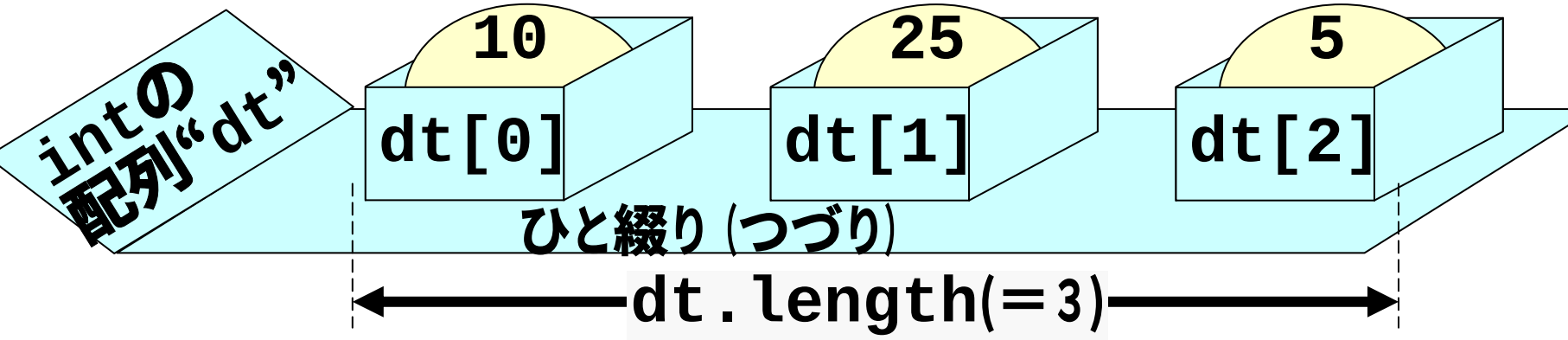
操作	記述例	操作跡の変数の内容
①整数型(int)の配列変数dtを宣言する	`int dt[];` (`int[] dt;`でも同じ)	
②配列dtのintの実体3つを確保する	`dt=new int[3];`	
番号1の要素(dt[1])に25を入れる	`dt[1]=25;`	

■①②を同時に行うことも出来ます。

```
int dt[] = new int[3];
```

(4．2) 配列長

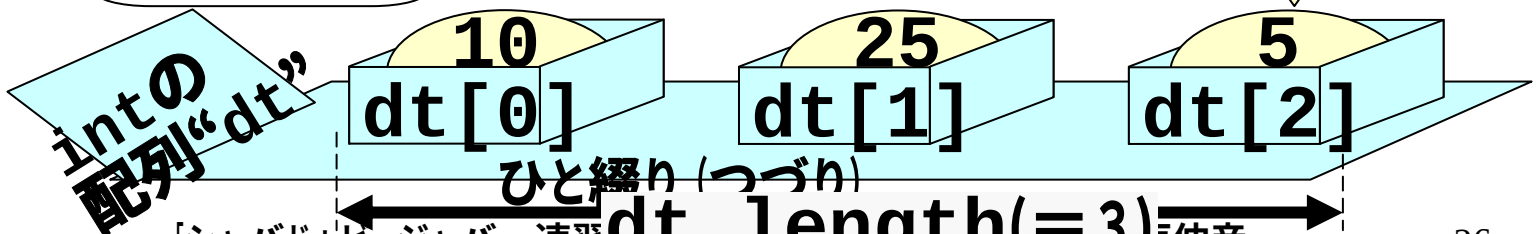
■配列“dt”の長さ(要素の数)は、“dt.length”と表すことが出来ます。



■配列の一番最後の要素に“5”を設定したいとき、次のようにします。

```
int dt[] = new int[3];
dt[dt.length-1] = 5;
```

3 −1 ＝ 2

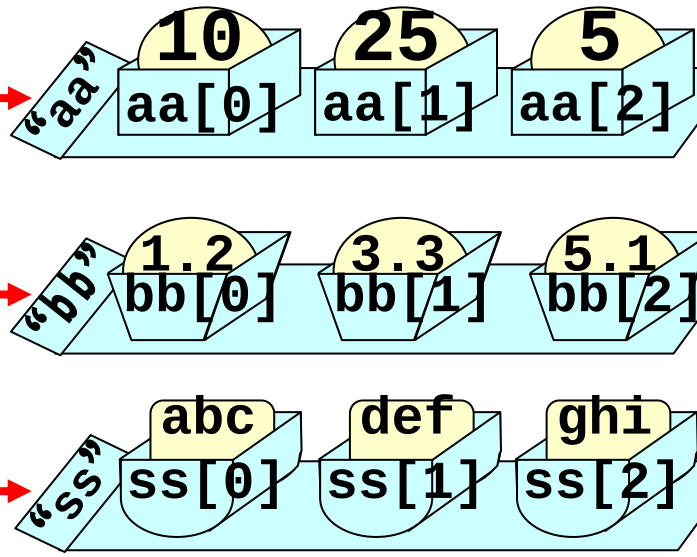


（4．3）配列の初期化

■変数を宣言した際に値を入れておくことを、初期化と言います。（スライド(3.2)：double f = 3.14; ）

■配列も、初期化を行うことが出来ます。その際、配列の長さは、値が入れられた数に設定されます。

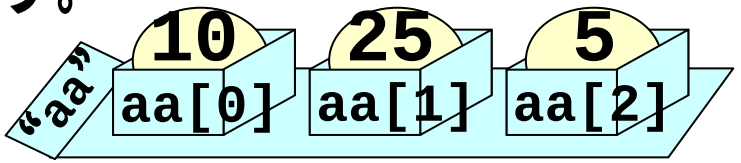
```
int aa[] = {10,25,5};

double bb[] = {1.2, 3.3, 5.1};

String ss[] = {“abc”,”def”,”ghi”};
```



(4．4) 配列のコピー

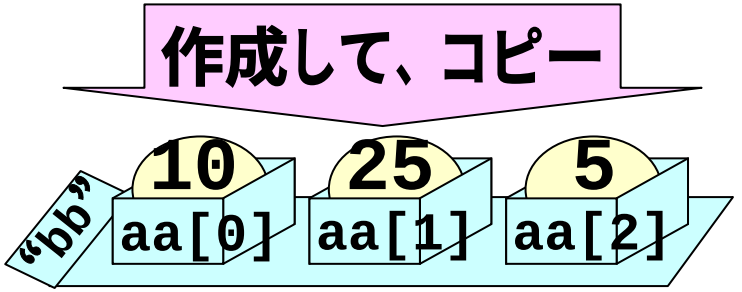
■初期化された配列“aa”があるとします。

```
int aa[] = {10,25,5};
```



■新たな配列“bb”を宣言して、aaの内容をコピーする処理は、次のよになります（後で習うfor文を使えばもっと簡単です）。

```
int bb[] = new int[3];
bb[0]=aa[0];
bb[1]=aa[1];
bb[2]=aa[2];
```



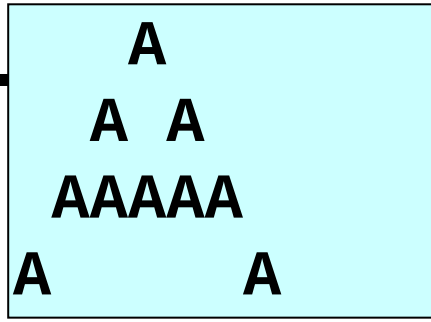
■上記のコピーは、次のように書くことも出来ます。少し高度な内容ですが、覚えておくと便利です。

```
int bb[] = (int[])aa.clone();
```

```
int bb[] = new int[3];
System.arraycopy(aa,0,bb,0,aa.length);
```

(4.5)例題4.1

■各行を配列に入れて、右のような文字を出力してください。



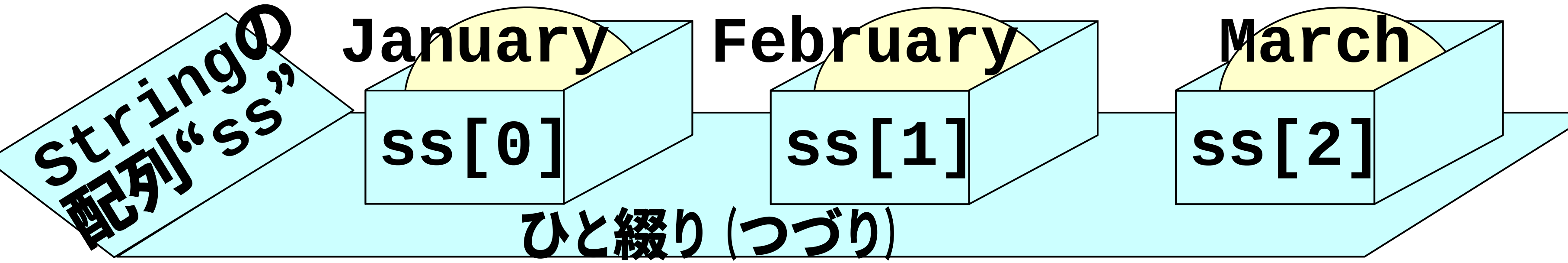
■解答例

```
public class Reidai41 {
  public static void main(String[] args) {
    String aa[]={
           "   A",
           "  A A",
           " AAAAA",
           "A     A",
    };
    System.out.println(aa[0]);
    System.out.println(aa[1]);
    System.out.println(aa[2]);
    System.out.println(aa[3]);
  }
}
```

(4.6)課題

■課題4－1

- 次のような値を持つ配列を持ち、順に出力するプログラムを作成してください。




```
$ java Kadai41
January
February
March
```

（４．７）課題

■課題４－２

- (A)のような変数の宣言を行い、これを用いて (B)の出力を得るプログラムを作成してください。

(A)

```
String stars[]={"   *",
                "  ***",
                " *****",
                "*******"};
```

(B)

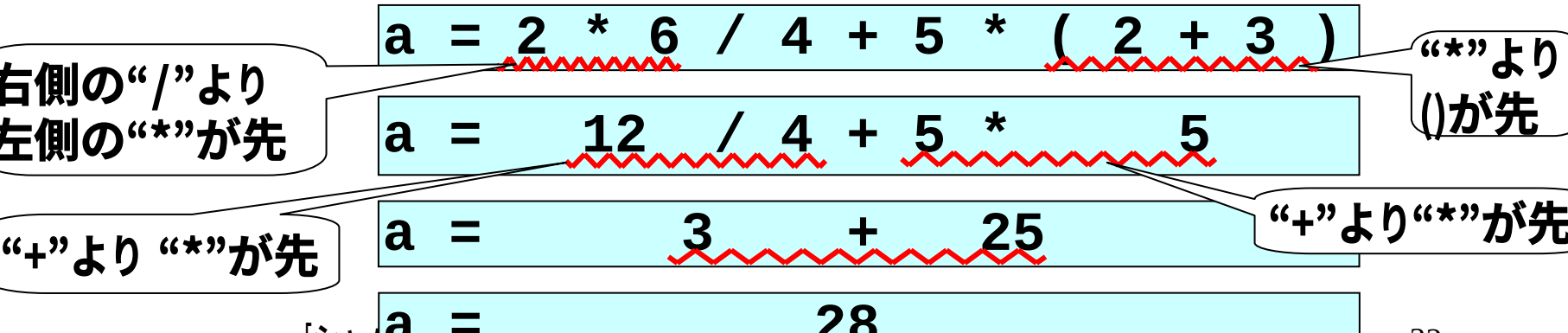
```
   *
  ***
 *****
*******
 *****
  ***
   *
```

(5) 演算

■算術演算子

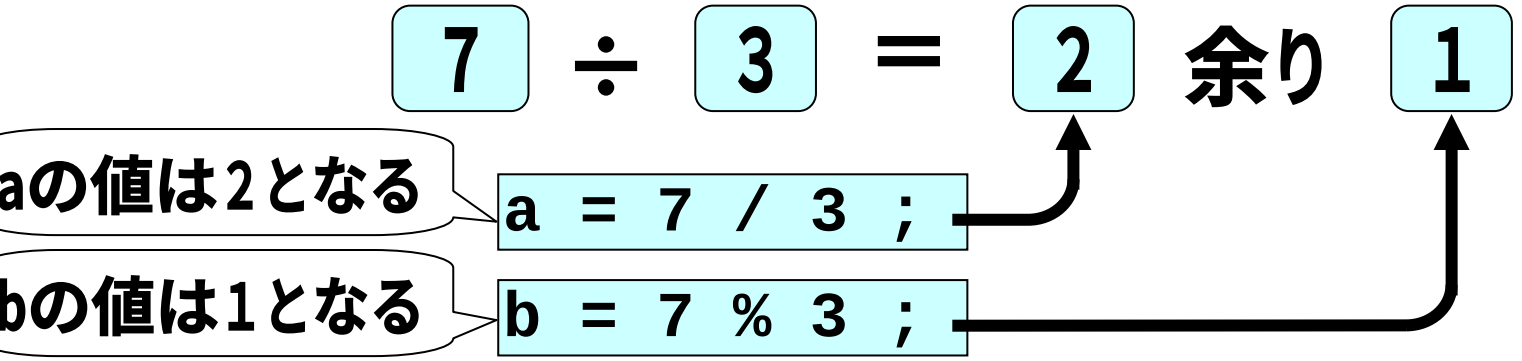
演算子	意味	使用例	優先度
*	掛け算	a = b * c ;	高
/	割り算	a = b / c ;	
%	あまり	a = b % c ;	
+	足し算	a = b + c ;	低
-	引き算	a = b - c ;	

- 優先度が高いものは、低いものより先に計算します。同じ優先度のもの同士は、左から計算します。
- () があれば、その中身を先に計算します。

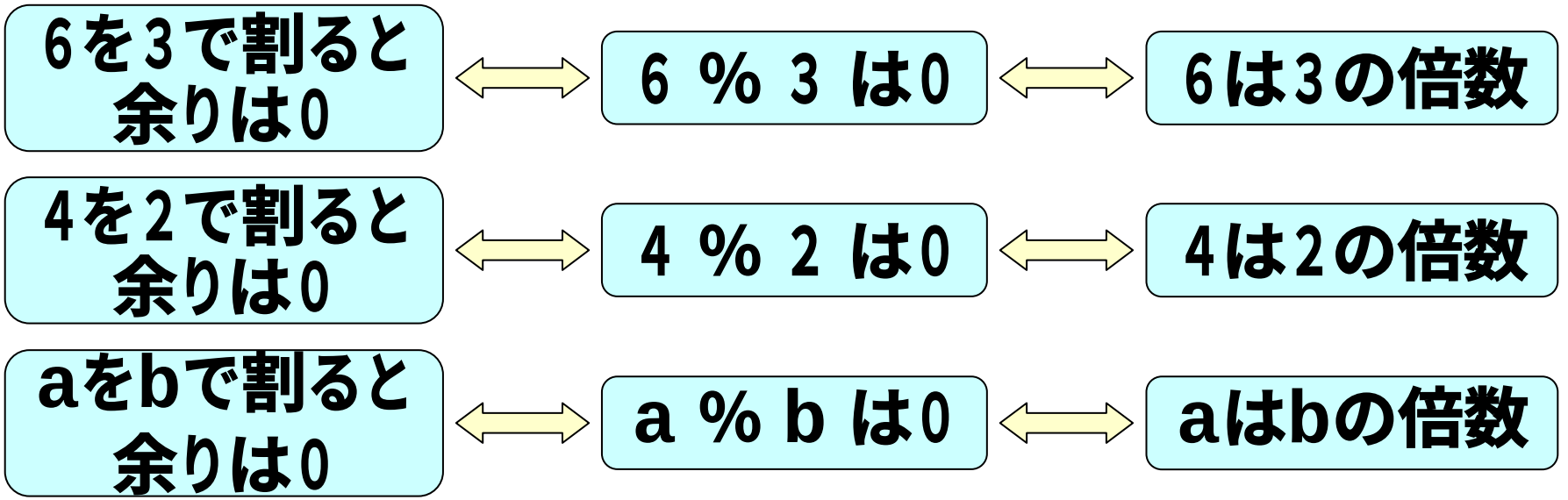


(5．1) “%” (余り、剰余)

■整数の割り算 (／) と余り (%)



■余りと倍数



(5．2) インクリメント演算子、その他

■インクリメント演算子

`++a;` または `a++;` → `a=a+1;` と同じ。

■デクリメント演算子

`--a;` または `a--;` → `a=a-1;` と同じ。

■インクリメント演算子の代入(使用しなくてもOK)

`b = a++;` → `b = a; a = a+1;` と同じ。

`b = ++a;` → `a = a+1; b = a;` と同じ。

■代入演算子

`a += 3;` → `a = a + 3;` と同じ。

`a *= 3;` → `a = a * 3;` と同じ。

- この他、“-=”、“/=”、“%=”なども同じ。

(5.3)例題5.1

■ふたつの変数"a","b"を宣言し、それぞれの値を8と4として、その四則演算(加減乗除:＋、－、＊、／)の結果を次のように出力するプログラムを作成してください。

```
$ java reidai51
a+b = 12
a-b = 4
a*b = 32
a/b = 2
```

■解答例

なぜ括弧"(...)"が必要なのか？

```
public class Reidai51 {
  public static void main(String[] args) {
    int a=8;int b=4;
    System.out.println("a+b="+(a+b));
    System.out.println("a-b="+(a-b));
    System.out.println("a*b="+a*b);
    System.out.println("a/b="+a/b);
  }
}
```

なぜ括弧"(...)"が必要無いのか？

(5．3．1)引き算のNG例

■次のプログラムはコンパイルエラーとなります。

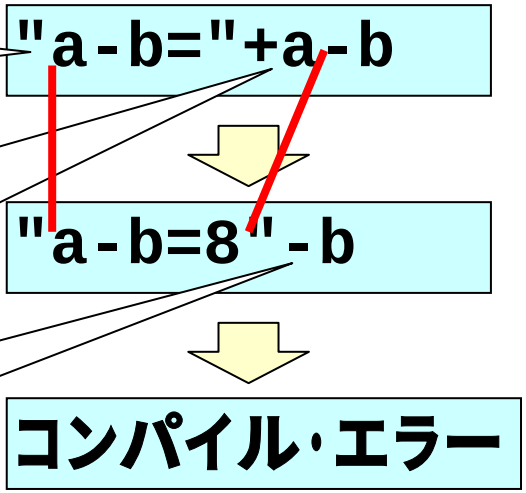
```
        :
    System.out.println("a-b="+a-b);
        :
```

■コンパイラは、次のように解釈してエラーとします。

①System.out.printlnは文字列を引数とするから、まず“a+b=”は文字列。

②文字列のあとに+だから、文字列を連結することにしよう。次は数値のaだけど、数値はその文字列と解釈する約束だから、aは文字の“8”として連結しよう

③文字列“a-b=8”に対して、-というのは、何をしたいのか？エラーだ！



(5．3．2)引き算のOK！

■次のプログラムは、意図通りの動作をします。

```
        :
System.out.println("a-b="+(a-b));
        :
```

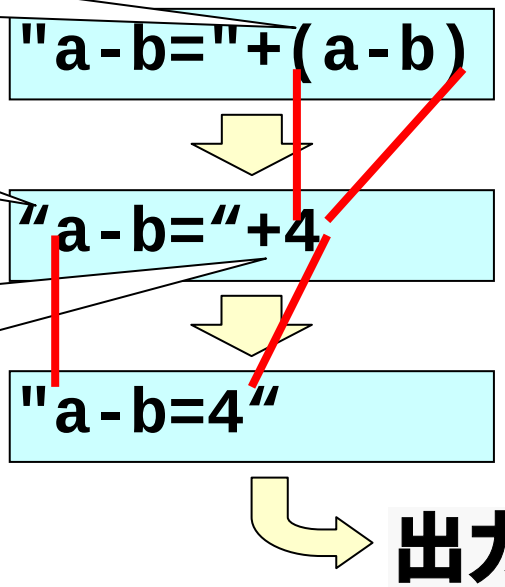
8　4

■コンパイラは、次のように解釈します。

①括弧“(…)”は最初に解釈する、
引き算だから、8－4＝4だ！

②System.out.printlnは文字列を引数とするから、まず“a-b=”は文字列。

③文字列“a-b=”に対して、+というのは、文字列を連結すること。数値は文字列と解釈するから、数値の4は、文字列の“4”として連結しよう！



(5．3．3) 足し算のNG例

■次のプログラムは、意図と違う出力をしてしまいます。

```
        :
System.out.println("a+b="+a+b);
        :
```

8　4

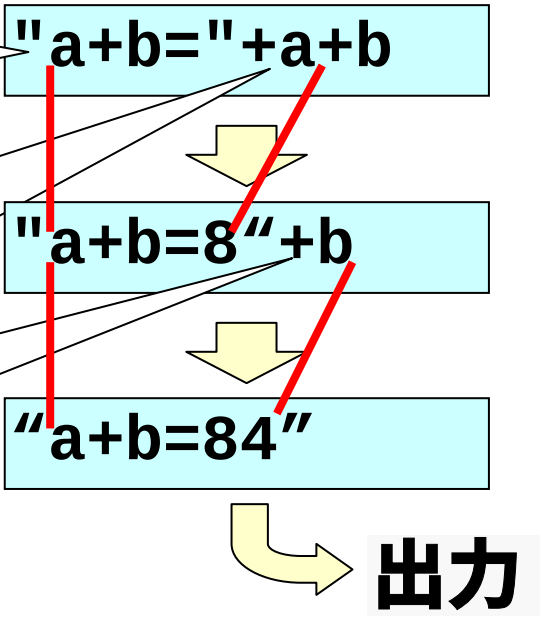
a+b=84

■コンパイラは、次のように解釈します。

①System.out.printlnは文字列を引数とするから、まず“a+b=”は文字列。

②文字列のあとに+だから、文字列を連結することにしよう。次は数値のaだけど、数値はその文字列と解釈する約束だから、aは文字の“8”として連結しよう

③文字列のあとに+だから、文字列を連結することにしよう。次は数値のbだけど、数値はその文字列と解釈する約束だから、bは文字の“4”として連結しよう



(5．3．4) 足し算のOK!

■次のプログラムは、意図通りの動作をします。

```
        :
System.out.println("a+b="+(a+b));
        :
```

8　4

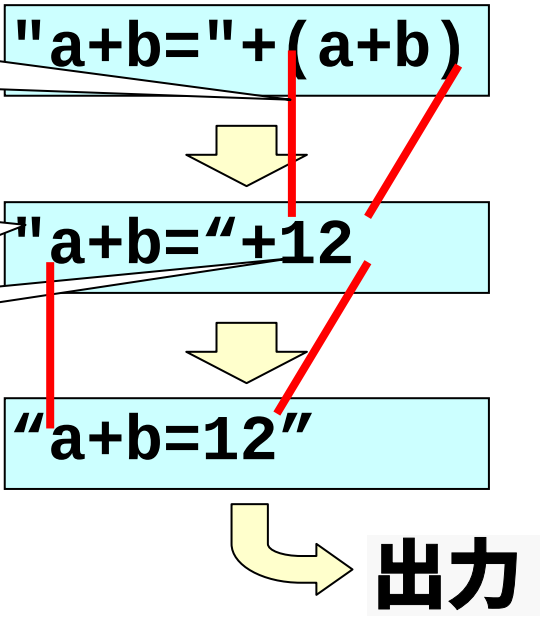
a+b=12

■コンパイラは、次のように解釈します。

①括弧“(…)”は最初に解釈する、引き算だから、8+4＝12だ！

②System.out.printlnは文字列を引数とするから、まず“a+b=”は文字列。

③文字列のあとに+だから、文字列を連結することにしよう。次は数値の12だけど、数値はその文字列と解釈する約束だから、文字列の“12”として連結しよう



(5．3．5) 掛け算はOK！

■次のプログラムは、意図通りの動作をします。

```
        :
System.out.println("a*b="+a*b);
        :
```

8　4

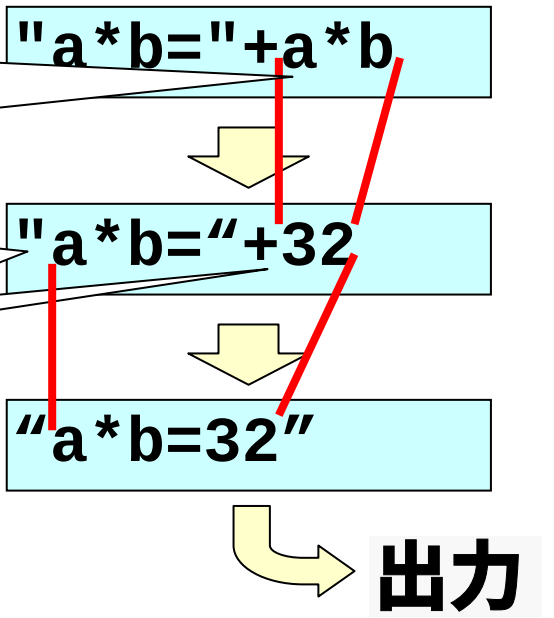
a*b=32

■コンパイラは、次のように解釈します。

①掛け算＊は、連結+よりも、優先して解釈する、
掛け算だから、8*4＝32だ！

②System.out.printlnは文字列を引数とするから、まず“a*b=”は文字列。

③文字列のあとに+だから、文字列を連結することにしよう。次は数値の32だけど、数値はその文字列と解釈する約束だから、文字列の“32”として連結しよう



(5．3．6) 割り算もOK！

■次のプログラムは、意図通りの動作をします。

```
        :
System.out.println("a/b="+a/b);
        :
```

8　4

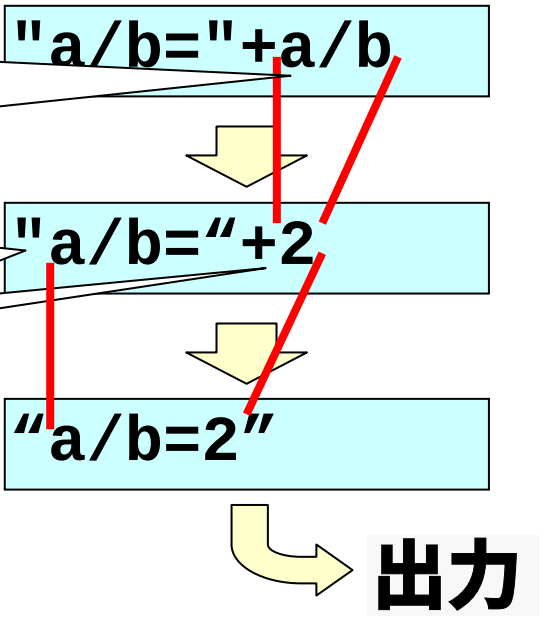
a/b=2

■コンパイラは、次のように解釈します。

①掛け算/は、連結+よりも、
優先して解釈する、
掛け算だから、8/4＝2だ！

②System.out.printlnは文字列を引数とするから、まず“a/b=”は文字列。

③文字列のあとに+だから、文字列を連結することにしよう。次は数値の2だけど、数値はその文字列と解釈する約束だから、文字列の“2”として連結しよう



(5．4) 課題

■課題5－1 (例題を参考にしてください)

- 変数"a","b","c","d" (それぞれの値は、100,200,250,500) を宣言し、次の表示結果を得るように、計算を行うプログラムを作成してください。

```
$ java kadai51
a*b = 20000
a*b+c = 20250
a*b+c*d = 145000
(a+b)*(c+d) = 225000
a+b*c+d = 50600
b/a+d/c = 4
```

（5．5）課題

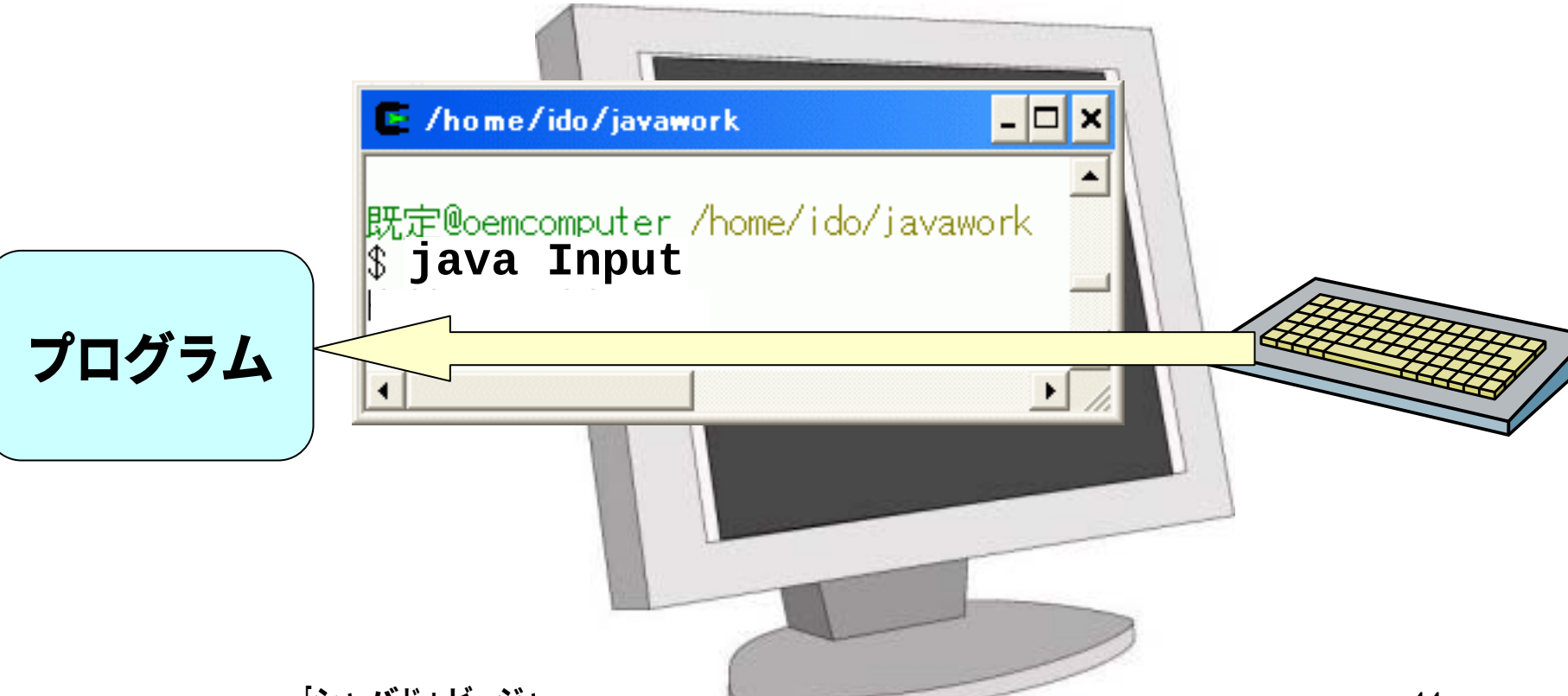
■課題5－2

- 次の事実を確認するためのプログラムを作成してください（すなわち、次の計算結果を出力するプログラムでOKです）。
  - ◆1を9でわると、次のようになります。
    0.1111111111111…
  - ◆12を99で割ると、次のようになります。
    0.12121212121212…
  - ◆123を999で割ると、次のようになります。
    0.123123123123123…
  - ◆1234を9999で割ると、次のようになります。
    0.12341234123412341234…
  - ◆12345を99999で割ると、次のようになります。
    0.1234512345123451234512345…

(6) データの入力

■何をしたいのか？

- 作成するプログラムへ、使用している端末のキーボード(標準入力という言い方もします)から、文字列を入力します。
- キーボードは、プログラムを動かすコマンドラインで入力状態になっています。



(6．1) 入力を行うプログラム

■次のようなプログラムを考えます。

```
import java.io.*;

public class Input {
  public static void main(String[] args) throws IOException{
    BufferedReader kbd =
     new BufferedReader(new InputStreamReader(System.in));
    String ss = kbd.readLine();
    System.out.println(ss);
  }
}
```

前述のとおり、クラス名（“Input”）は、大文字から始めるのが決まりです。

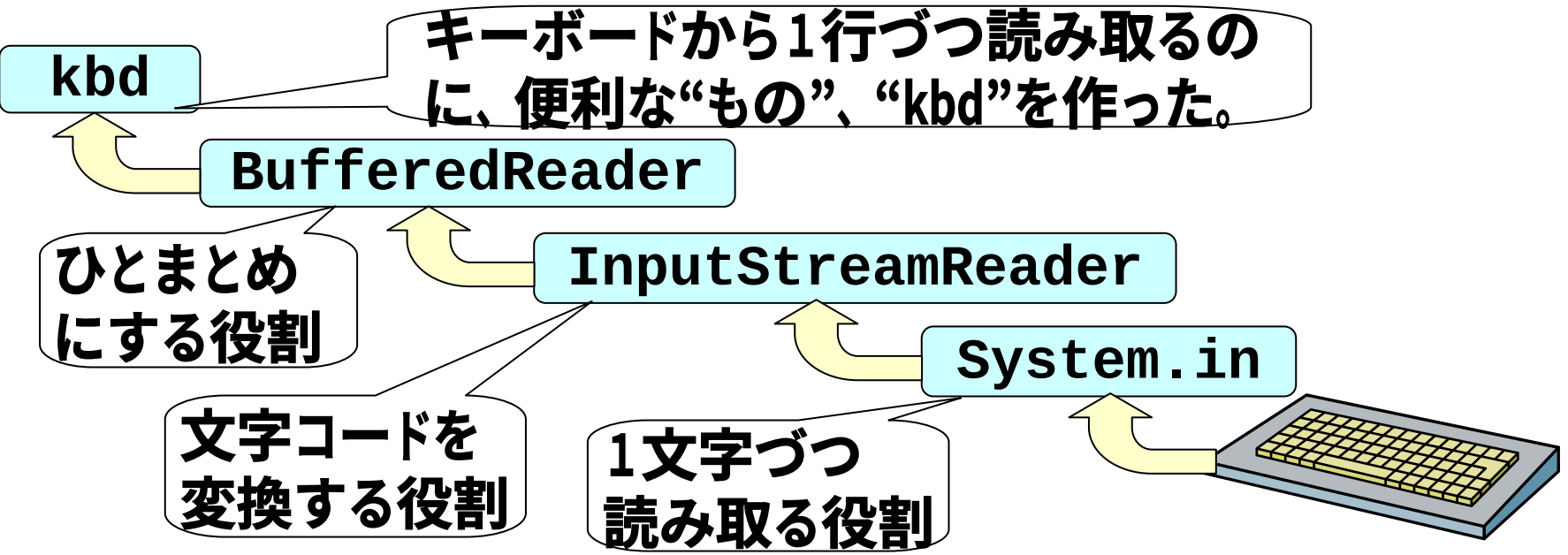
■このプログラムは、入力した文字列1行をそのまま打ち出すプログラムです。

(6．2．1) 入力を行う2行　－1－

■入力を行うプログラムは2行だけです。

■最初の行では、1行分を入力するために便利なものである、“kbd”を作っています（少し荒っぽい言い方ですが）。

```
BufferedReader kbd =
  new BufferedReader(new InputStreamReader(System.in));
```

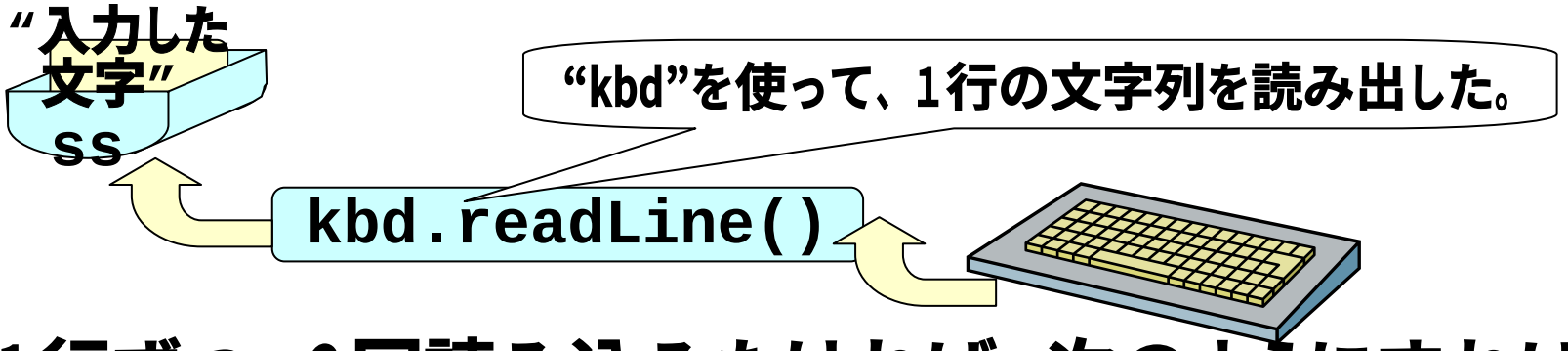


■さし当たって、上記の意味を理解する必要はなく、「このように作成するものなんだ」と理解しておいてＯＫです。

(6．2．2) 入力を行う2行　－2－

■1行分を入力するために便利なもの“kbd”を使えば、1行を読み込む処理は、次のように簡単です。

```
String ss = kbd.readLine();
```



■1行ずつ、2回読み込みたければ、次のようにすればOKです。

```
BufferedReader kbd =
  new BufferedReader(new InputStreamReader(System.in));
String ss1 = kbd.readLine();
String ss2 = kbd.readLine();
```

(6．3) インポート

■スライド(6.1)のプログラムには、次のように書くことも可能です。
すなわち、“BufferedReader”を
“java.io.BufferedReader”というように書いています。

```
public class Input {
  public static void main(String[] args)
                          throws java.io.IOException{
    java.io.BufferedReader kbd = new java.io.BufferedReader(
        new java.io.InputStreamReader(System.in));
    String ss = kbd.readLine();
    System.out.println(ss);
  }
}
```

■このように記すのは煩わしいので、スライド(6.1)では、
“java.io.”の部分を省略できるように、プログラムファイルの
最初に右のインポート文を入れています。

```
import java.io.*;
```

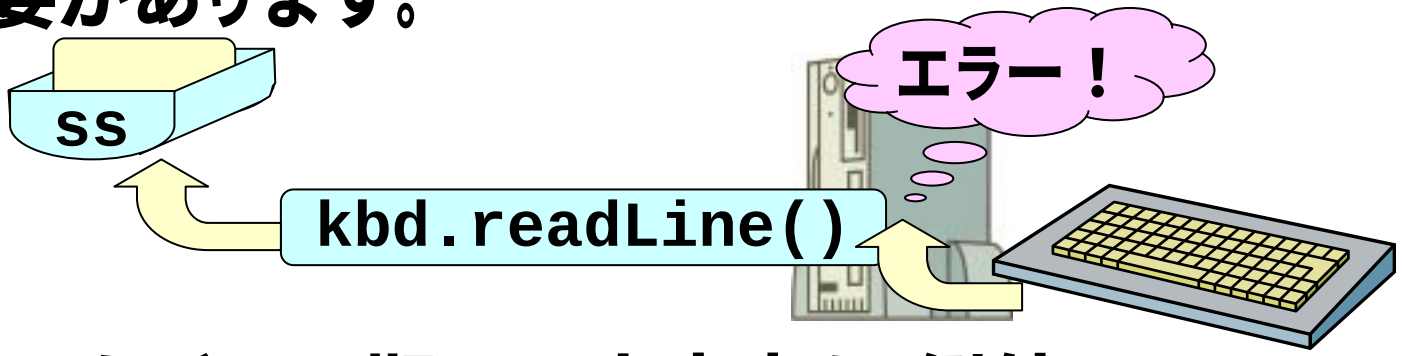
■すなわち、次の2つは、同じことになります。

```
import java.io.*;
          :
  ... BufferedReader ...
          :
```

```
          :
... java.io.BufferedReader ...
          :
```

(6．4) 入力での例外処理

■キーボードからの入力処理では、エラーが発生する可能性があり、Javaではこれに対する対処を記述しておく必要があります。



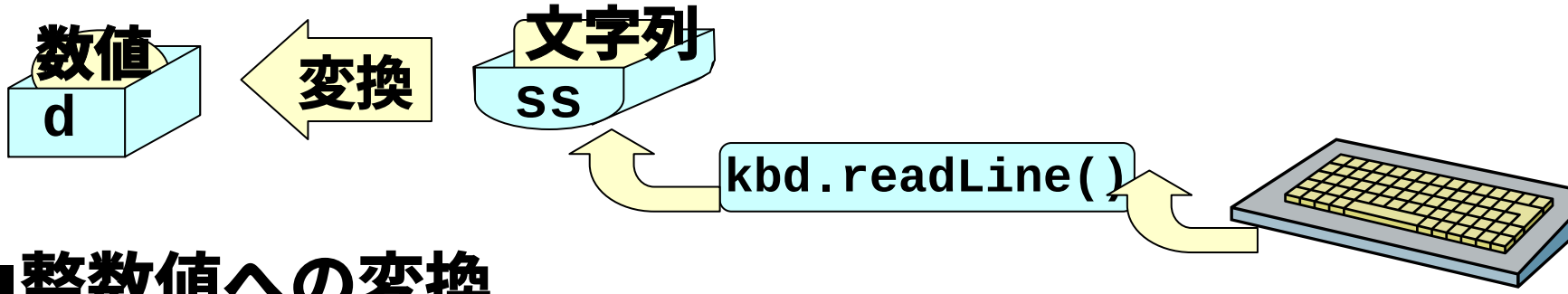
■エラーなどの予期せぬ出来事を“例外”と言い、これに対する処理を、“例外処理”と言います。

■スライド(6.1)では、次のような一番簡単な例外処理を行っています。すなわち、例外“IOException”が発生したら、それを呼び元に投げる“throws”という処理です。例外処理については、後述します(スライド(16))。

```
public static void main(String[] args)
                        throws IOException{
```

(6．5) 数値への変換

■キーボードから数値を入力したい際には、まず、文字列として入力してから、数値へ変換します。



■整数値への変換

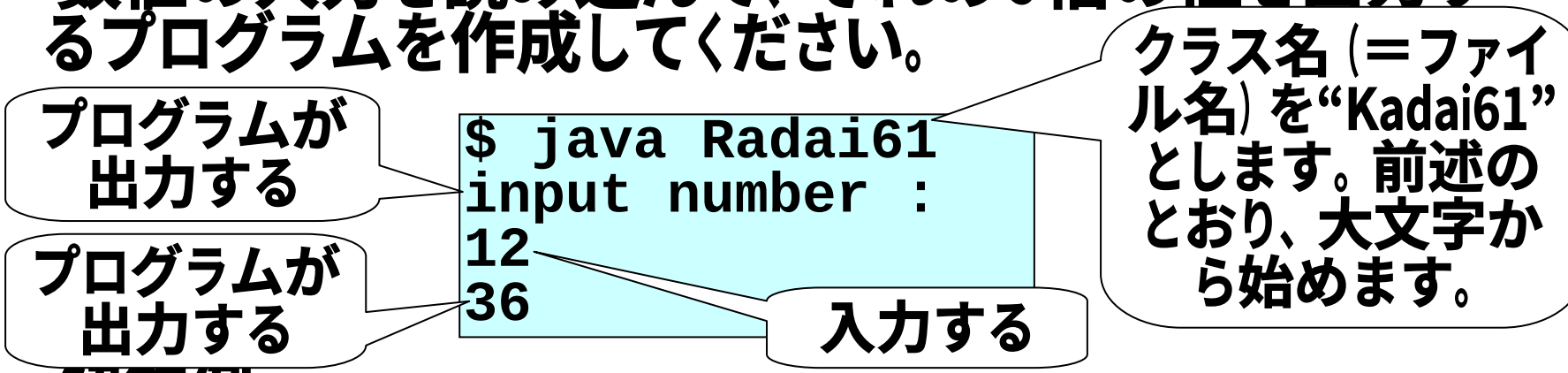
```
String ss = kbd.readLine();
int d = Integer.parseInt(ss);
```

■浮動小数点数値への変換

```
String ss = kbd.readLine();
double f = Double.parseDouble(ss);
```

(6.6) 例題6.1

■数値の入力を読み込んで、それの3倍の値を出力するプログラムを作成してください。



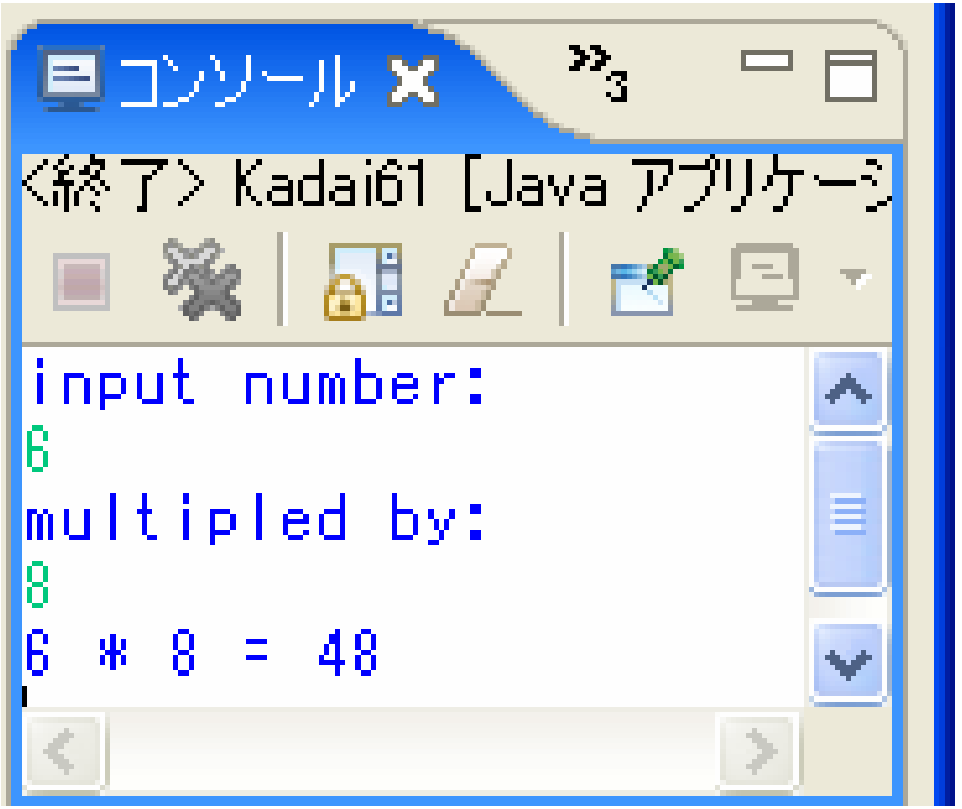
■解答例

```
import java.io.IOException;
public class Reidai61 {
  public static void main(String[] args)
                                    throws IOException{
    String ss;
    java.io.BufferedReader kbd
    = new java.io.BufferedReader(
              new java.io.InputStreamReader(System.in));
    System.out.println("input number:");
    ss = kbd.readLine();
    int d = Integer.parseInt(ss);
    System.out.println(d*3);
  }
}
```

(6．7) 課題

■課題6－1 (例題を参考にしてください)

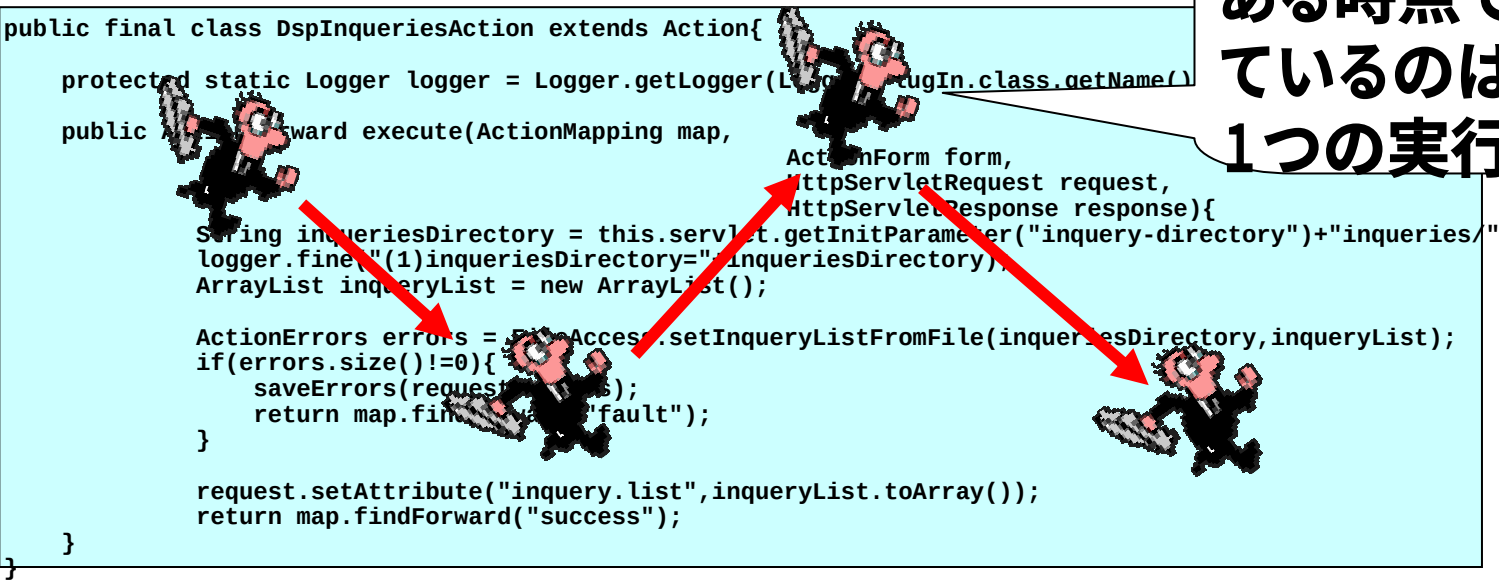
- 2つの数値を入力して、その積を出力するプログラムを作成してください。



(7) プログラムの実行

■プログラムの実行は複雑なようにも見えますが、「ある時点で実行している実行文はひとつだけ」という意味では単純とも言えます。

■すなわち、プログラムの実行は、プログラム中の実行箇所1箇所があちらこちらに走り回っているように考えることが出来ます。



(7．1) 制御文

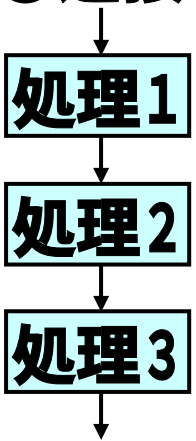
■実行箇所がどのように動くのかを決めるプログラムの要素を、制御文と言います。

■Javaには、次のような制御文があります。

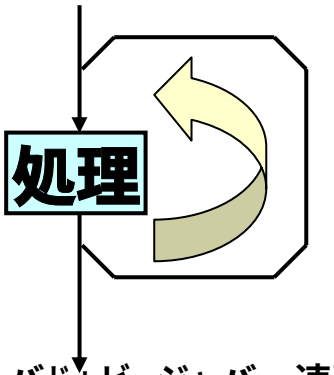
- `if while for do-while switch break continue return`

■プログラムの実行は複雑なようにも見えますが、制御文により実現される動きは、次の3つの組み合わせと考えることが出来ます(“構造化プログラミング”参照)。

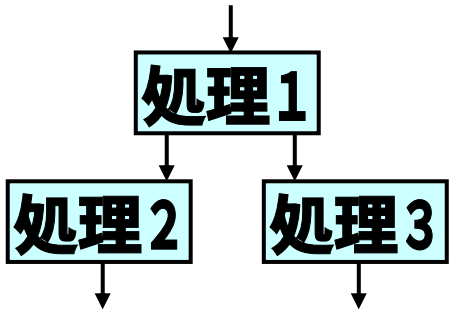
①連接:



②繰り返し:

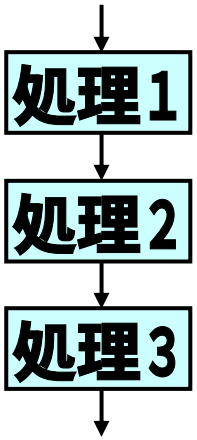
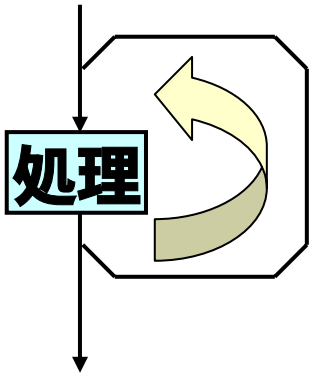
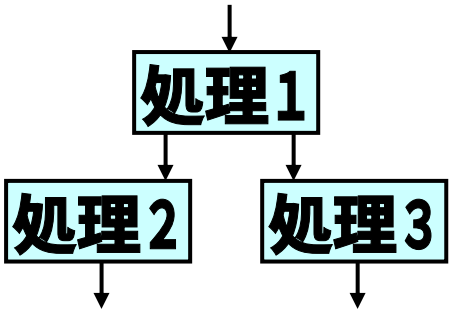
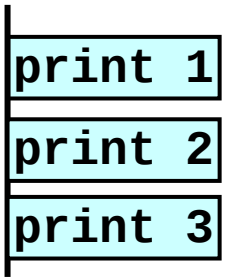
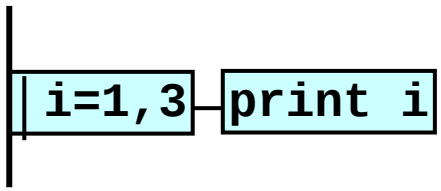
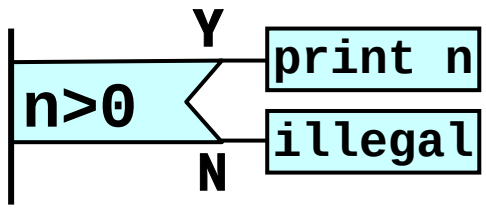


③分岐:



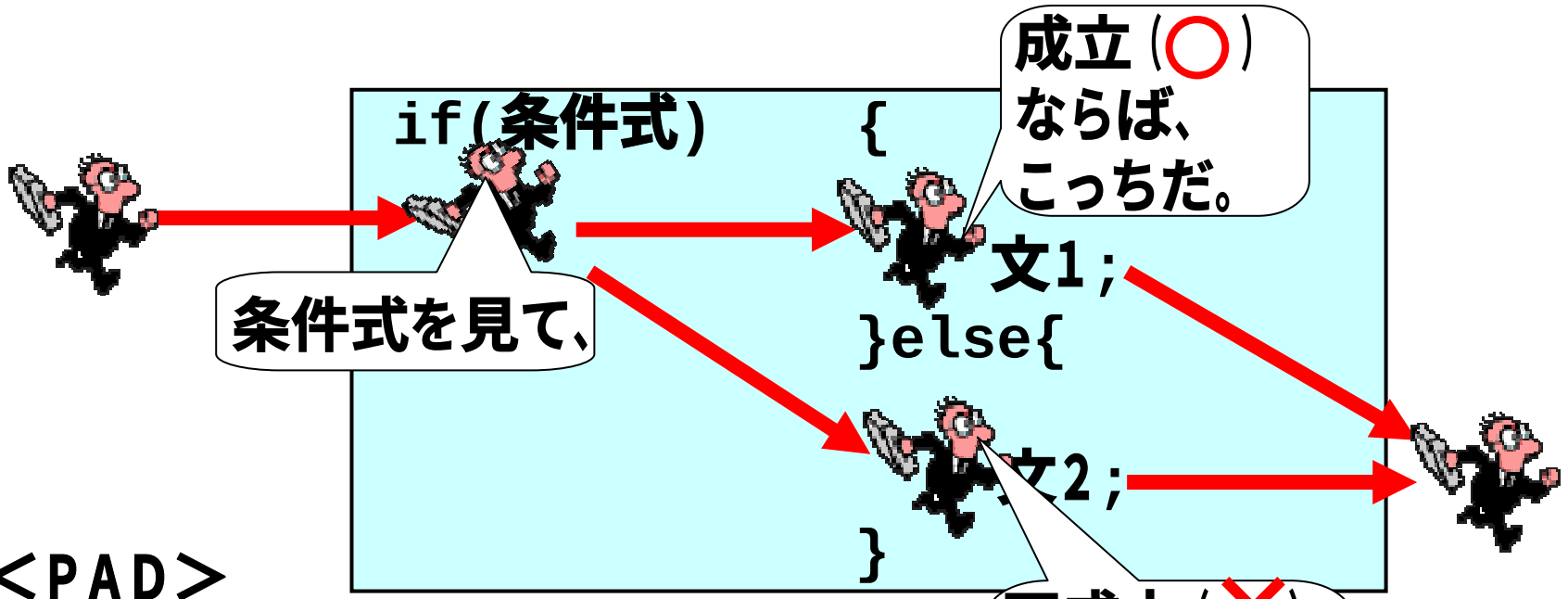
(7．2) PAD

■本資料では、PADと呼ばれる記法を制御文の説明時に併記しています。PADについて勉強されたい方は、別資料「山のあなたの空とおくー構造化プログラミングー」を参照してください。

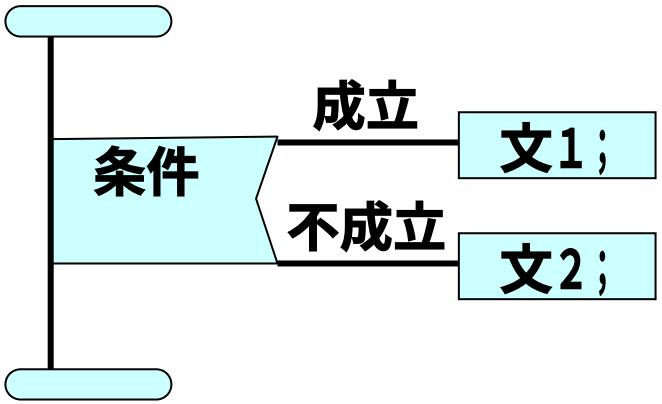
	①連接：	②繰り返し：	③分岐：
			
PAD			

(8) 分岐、if文

■分岐では、”if”文がもっともよく用いられます。



＜PAD＞



(8．1)“if文”

■形式

```
if(条件式){
    文1;
}else{
    文2;
}
```

“{”と”}”に囲まれた範囲（＝ブロック）に、いくつ実行文があっても良い。

■例

＜プログラム＞

```
int a=0;
int b=0;
if (a==0){
   b=1;
}else{
   b=2;
}
System.out.println(“b=“+b);
```

条件式「“a”は“0”である」

条件が成立（○）した場合、こちらを実行

条件が不成立（✕）した場合、こちらを実行

＜出力＞

```
b=1
```

(8.2) 条件

■関係演算子、等値演算子による条件

- 条件式は、関係演算子、等値演算子を用いて記述します。

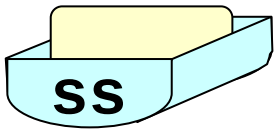
関係演算子	比較	使用例	意味	xの値				
				3	4	5	6	7
<	より小さい	`if(x<5){`	xが5より小さいなら	○	○	×	×	×
<=	より小さいか等しい	`if(x<=5){`	xが5以下なら	○	○	○	×	×
>	より大きい	`if(x>5){`	xが5より大きいなら	×	×	×	○	○
>=	より大きいか等しい	`if(x>=5){`	xが5以上なら	×	×	○	○	○

等値演算子	比較	使用例	意味	xの値				
				3	4	5	6	7
==	等しい	`if(x==5){`	xが5なら	×	×	○	×	×
!=	等しくない	`if(x!=5){`	xが5でないなら	○	○	×	○	○

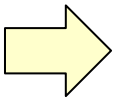
※ ○：成立　×：不成立

(8．3) 文字列の等値判定

■“String”で宣言した変数が同じであるか否かの判定は、前スライドの等値演算子では行えません。

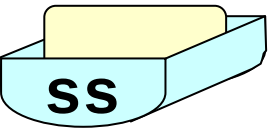


が“abc”と同じか？

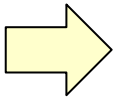


~~if(ss==“abc”){~~

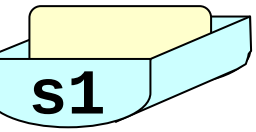
■上記のような比較は、次のように行います
(“String”クラスについては後述します(13.1))。



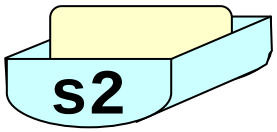
が“abc”と同じか？



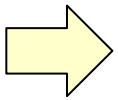
```
if(ss.equals(“abc”)){
```



が



と同じか？



```
if(s1.equals(s2)){
```

(`if(s2.equals(s1)){` でも同じ)

(8．4) 論理型 (boolean)

■条件が成立しているか、不成立であるかを表すデータのことを、論理型データ(boolean：ブーリアン)といいます。

- 成立していること (〇) を、真 (true) といいます。
- 不成立であること (×) を、偽 (false) といいます。

■論理型データを、整数型データと比較してみます。

	論理型(boolean)	整数型
変数の宣言	`boolean b;`	`int d;`
変数の取る値	`true,false`	`…,-3,-2,-1,0,1,2,3, …`
可能な演算	論理演算 && (かつ)、|| (もしくは)	算術演算 ＋ (足す)、－ (引く)

■条件式は、論理型データの値を取るということになります。

`x=3;` ⇒ “x<5”の値は、“true”

(8．5) 論理型 (boolean) 変数の使用例

＜プログラム＞

```
int x = 6;

boolean b = x>5;

System.out.println("b="+b);

if(b){
  System.out.println("真");
}else{
  System.out.println("偽");
}
```

“b”には、真 (true:○) が代入される

“b==true”と考えて良い

“こちらは実行されない

＜出力＞

```
b=true
真
```

(8．6) 論理演算子

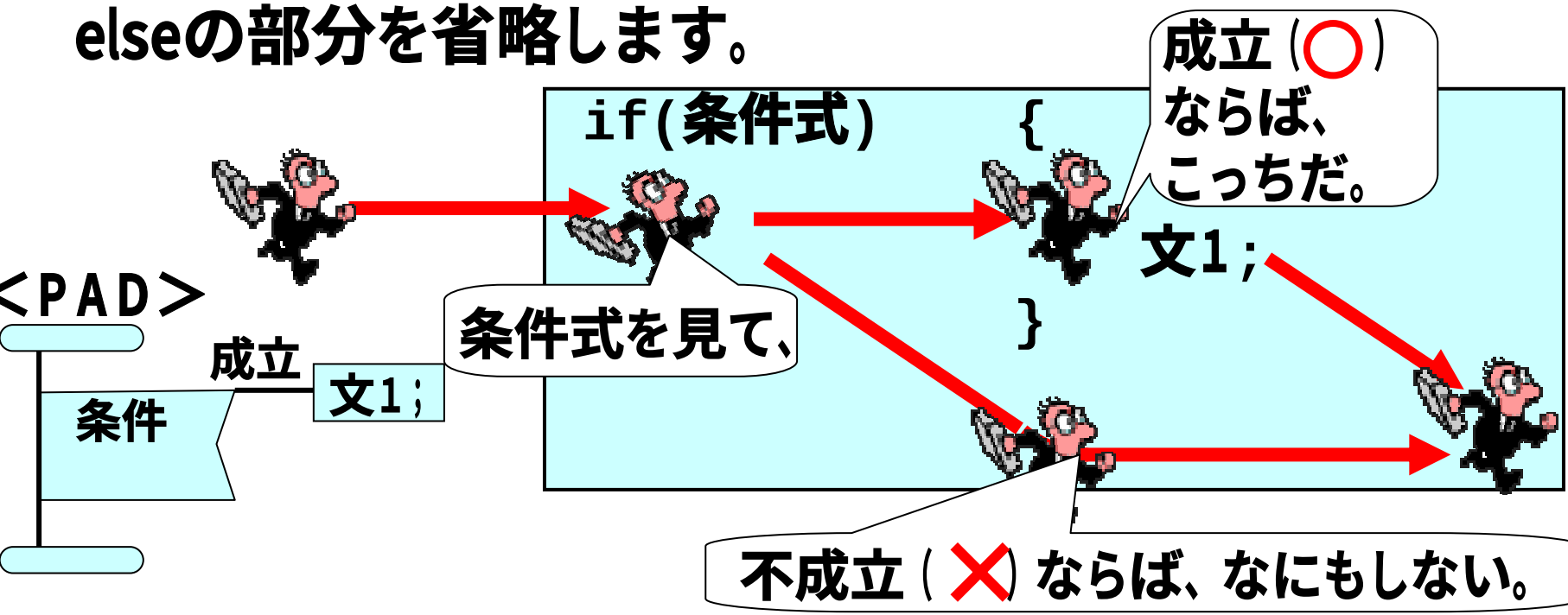
■2つ以上の条件を合わせて記述する場合などに、論理演算子を用います。

- 「今日は、13日の金曜日です」
  ⇒「今日は13日」、かつ(and)、「今日は金曜日」
- 「今日は、休日です。」
  ⇒「今日は日曜日」、または(or)、「今日は祝祭日」

論理演算子	演算	使用例 意味	xの値 3	 4	 5	 6	 7
&&	論理積 (かつ、and)	`if(3<x && x<6){` xが3より大きい、かつ、xが6より小さい	×	○	○	×	×
||	論理和 (または、or)	`if(x<5 || 6<x){` xが5より小さい、または、xが6より大きい	○	○	×	×	○
!	否定 (ではない、not)	`if(!(x<5 || 6<x)){` xが5より小さい、または、xが6より大きいではない	×	×	○	○	×

(8.7)“else”の省略

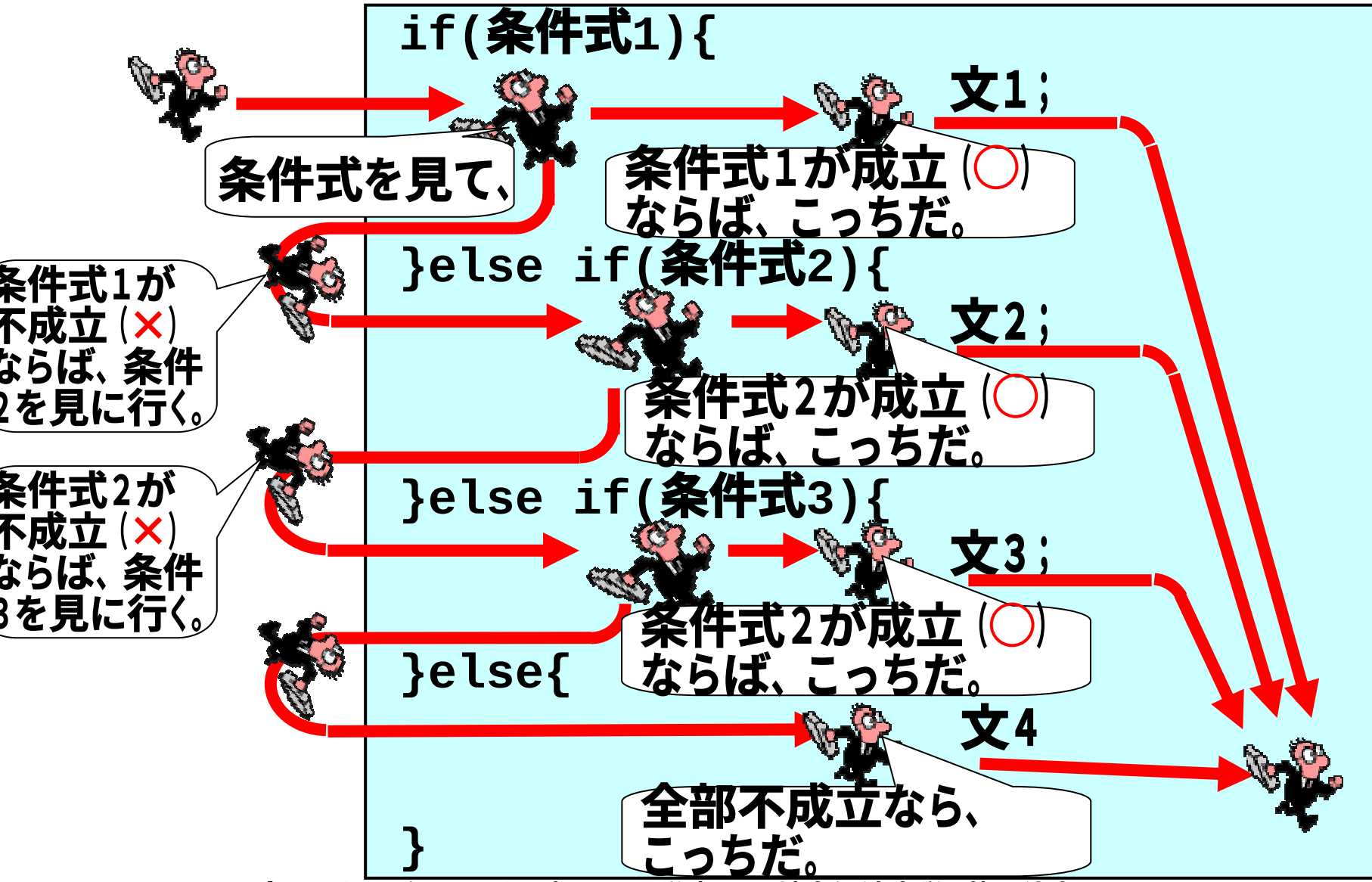
■条件式が成立しないときに何もしないのであれば、elseの部分を省略します。



■例：1000円以上ならば、100円引き

```
if(x>=1000){
   x = x - 100;
}
```

(8.8.1)“else if”


if(条件式1){
文1;
条件式を見て、
条件式1が成立（○）ならば、こっちだ。
}else if(条件式2){
文2;
条件式1が不成立（×）ならば、条件2を見に行く。
条件式2が成立（○）ならば、こっちだ。
}else if(条件式3){
文3;
条件式2が不成立（×）ならば、条件3を見に行く。
条件式2が成立（○）ならば、こっちだ。
}else{
文4
全部不成立なら、こっちだ。
}

(8.8.2) “else if”の例

■例：

- 10,000円以上ならば、1,000円値引き、
- 5,000円以上ならば、400円円値引き、
- 3,000円以上ならば、100円値引き、
- 1,000円以上ならば、10円値引き

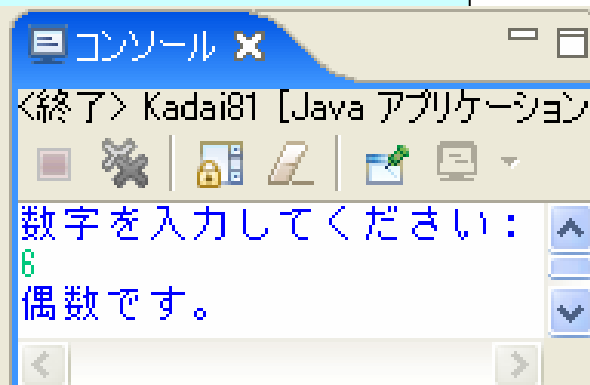
```
if(x>=10000){
  x = x - 1000;
}else if(x>=5000){
  x = x - 400;
}else if(x>=3000){
  x = x - 100;
}else if(x>=1000){
  x = x - 10;
}
```

(8.9) 例題8.1：奇数・偶数

■数字の入力を読み込んで、偶数か奇数かを出力するプログラムを作成してください。

■解答例：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Kadai81 {
  public static void main(String[] args)
                          throws IOException {
    BufferedReader kbd
    = new BufferedReader(
       new InputStreamReader(System.in));
    System.out.println("数字を入力してください：");
    String ss = kbd.readLine();
    int d = Integer.parseInt(ss);
    if(d%2==0){
      System.out.println("偶数です。");
    }else{
      System.out.println("奇数です。");
    }
  }
}
```



(8．10) 例題8．2：うるう年の条件の整理

■うるう (閏) 年である条件

- (1) 4で割れる年は、うるう年
  - 2004年はうるう年。
  - 2005年は平年 (うるう年でない) 。
- (2) 上記 (1) の例外として、100で割り切れる年は平年 (うるう年でない) 。
  - 1900年は平年 (うるう年でない) 。
  - 1904年はうるう年。」
- (3) 上記 (2) の例外として、400で割り切れる年はうるう年。
  - 2000年はうるう年。
  - 1900年は平年 (うるう年でない) 。

■同じ条件の別の言い方

- [1]400で割れる年はうるう年。
- [2]上記[1] ではなくて、100で割れる年は平年 (うるう年でない)
- [3]上記[1][2]でなく、4で割れる年はうるう年。
- [4]上記[1][2][3]でない場合は平年 (うるう年でない) 。

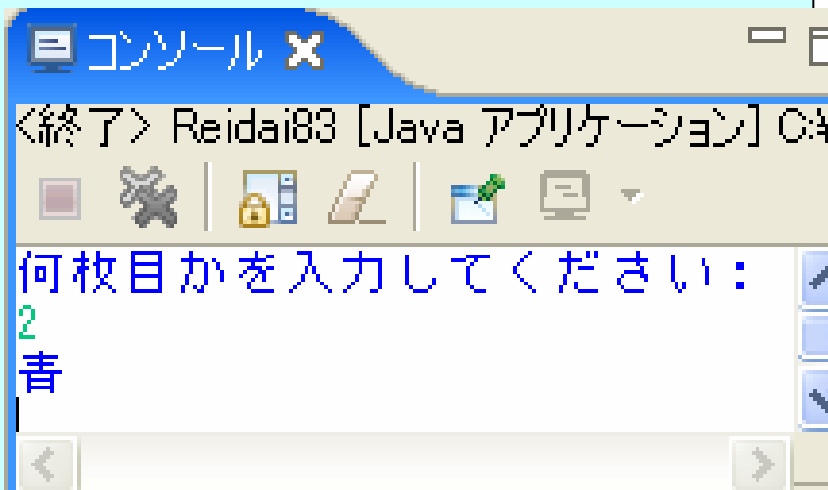
(8．11) 例題8．3：敷石

■次のように並んでいる舗道の敷石について、敷石が何枚目かを入力して、その色を出力するプログラムを作成してください。

■解答例：

1枚め	2	3	4	5	6	7	8	9	・・・
赤	青	黄	赤	青	黄	赤	青	黄	・・・

```
import java.io.IOException;
public class Reidai83 {
  public static void main(String[] args) throws IOException {
    java.io.BufferedReader kbd = new java.io.BufferedReader(
                new java.io.InputStreamReader(System.in));
    System.out.println("何枚目かを入力してください：");
    String ss = kbd.readLine();
    int d = Integer.parseInt(ss);
    String color="";
    if(d%3==1){
      color="赤";
    }else if(d%3==2){
      color="青";
    }else{
      color="黄";
    }
    System.out.println(color);
  }
}
```

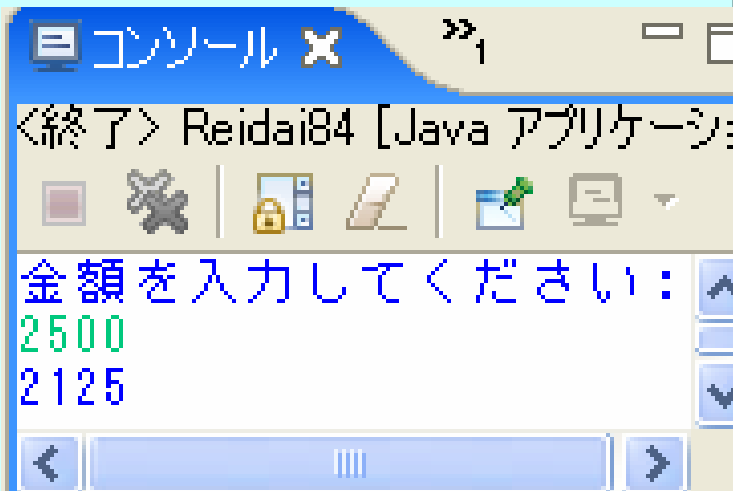


(8．12) 例題8．4：値引き

■金額(円)を入力して、3000円以上なら2割引き、2000円以上なら1割5部引き、1000円以上なら1割引きの額を出力するプログラムを作成してください。

■解答例：

```
import java.io.IOException;
public class Reidai84 {
  public static void main(String[] args) throws IOException {
    java.io.BufferedReader kbd
    = new java.io.BufferedReader(
       new java.io.InputStreamReader(System.in));
    System.out.println("金額を入力してください:");
    String ss = kbd.readLine();
    int d = Integer.parseInt(ss);
    if(d>=3000){
      d *= 0.8;
    }else if(d>=2000){
      d *= 0.85;
    }else if(d>=1000){
      d *= 0.9;
    }
    System.out.println(d);
  }
}
```

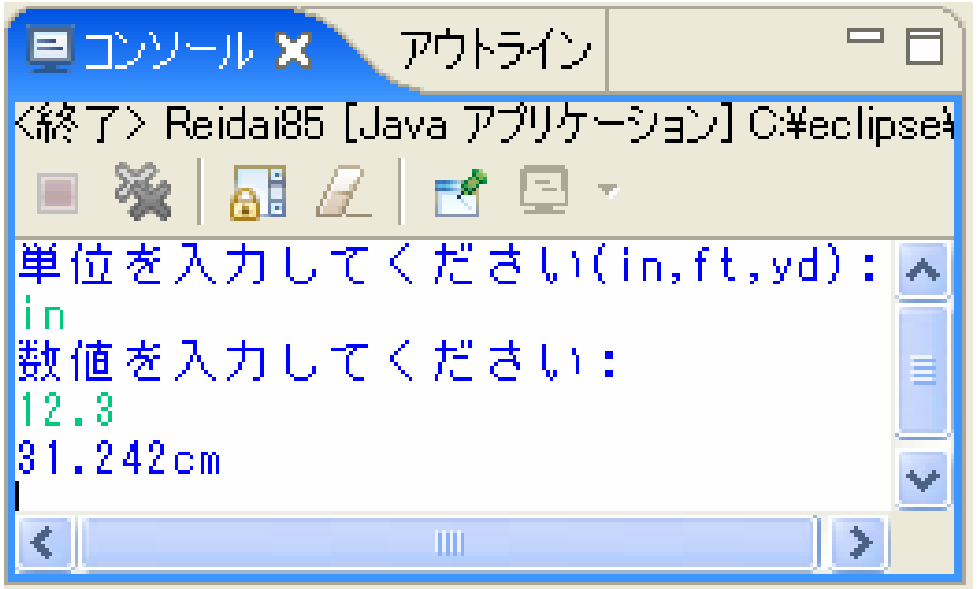


(8．13．1) 例題8．5：単位の換算

■長さの単位、インチ、フィート、ヤードは、cm(センチメール)に換算すると次のとおりとなります。

	1インチ (in)	1フィート(ft)	1ヤード (yd)
センチ (cm) 換算	2．54	30．48	91．44

■単位 (in,ft,yd) と数値とを入力して、そのセンチメートル換算の値を出力するプログラムを作成してください。



(8.13.2)例題8.5:続き

■解答例:

```
import java.io.IOException;
public class Reidai85 {
  public static void main(String[] args) throws IOException {
    java.io.BufferedReader kbd
    = new java.io.BufferedReader(
      new java.io.InputStreamReader(System.in));
    System.out.println("単位を入力してください(in,ft,yd):");
    String unit = kbd.readLine();
    System.out.println("数値を入力してください:");
    String ss = kbd.readLine();
    double d = Double.parseDouble(ss);
    double mul=0;
    if(unit.equals("in")){
      mul=2.54;
    }else if(unit.equals("ft")){
      mul=30.48;
    }else if(unit.equals(“yd")){
      mul=91.44;
    }else{
      System.out.println("in,ft,ydのいずれかを入力してください。");
      return;
    }
    System.out.println(mul*d+"cm");
  }
}
```

(8．12．1) 課題

■課題8－1 (例題6．1、8．1を参考にしてください)

- 数字の入力を読み込んで、次の条件の“いずれか”を満たす場合に、“条件を満たす”と出力するプログラムを作成してください。
  - ◆入力した数字は、3で割り切れる。
  - ◆入力した数字を用いて128を割ると、あまりが0になる。

■課題8－2 (例題8．2を参考にしてください)

- 西暦 (ex. 2004) の数字の入力を読み込んで、うるう年か否かを出力するプログラムを作成してください。

■課題8－3 (例題8．3を参考にしてください)

- 今月の日にち (ex. 3, 15, 27) の数字の入力を読み込んで、曜日を出力するプログラムを作成してください。その際、配列を必ず用いてください。

（8．12．2）課題

■課題8－4（例題8．4を参考にしてください）

- 肥満度の判定の1つに、BMI判定があります。
  - ◆指標＝体重(kg)／（身長(m)×身長（m））
  - ◆指標に対して、次の表にて判定する。

指標	判定
18.5未満	やせ
18.5～25未満	標準
25～30未満	肥満
30以上	高度肥満

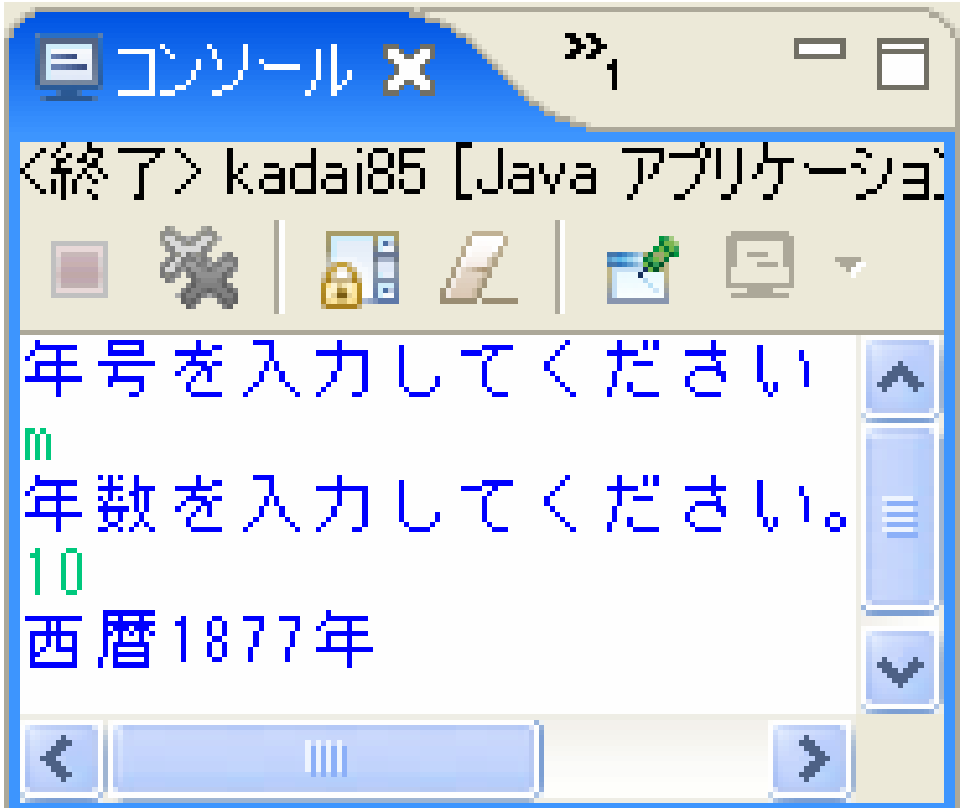
- 次のいずれかの方法で作成してください((1)が簡単です)。
  - (1)体重（kg）と身長(m)とを、小数（”double”）として入力する。
  - (2)体重（kg）と身長(cm)とを、整数（”int”）として入力する。

(8．12．3) 課題

■課題8－5 (例題8．5を参考にしてください)

- 元号 (明治を“m”、大正を“t”、昭和を“s”、平成を“h”) と年 (1、2、…) とを別々に入力して、西暦を出力するプログラムを作成してください。

明治元年	1868年
大正元年	1912年
昭和元年	1926年
平成元年	1989年

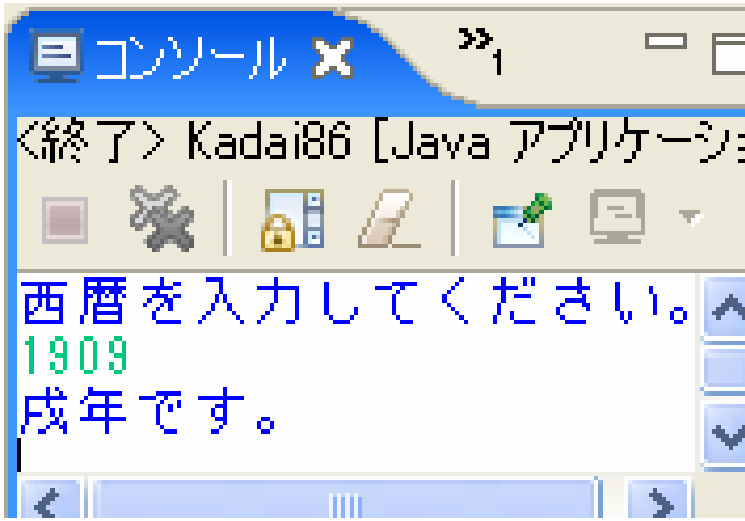


(8．12．3) 課題

■課題8－6

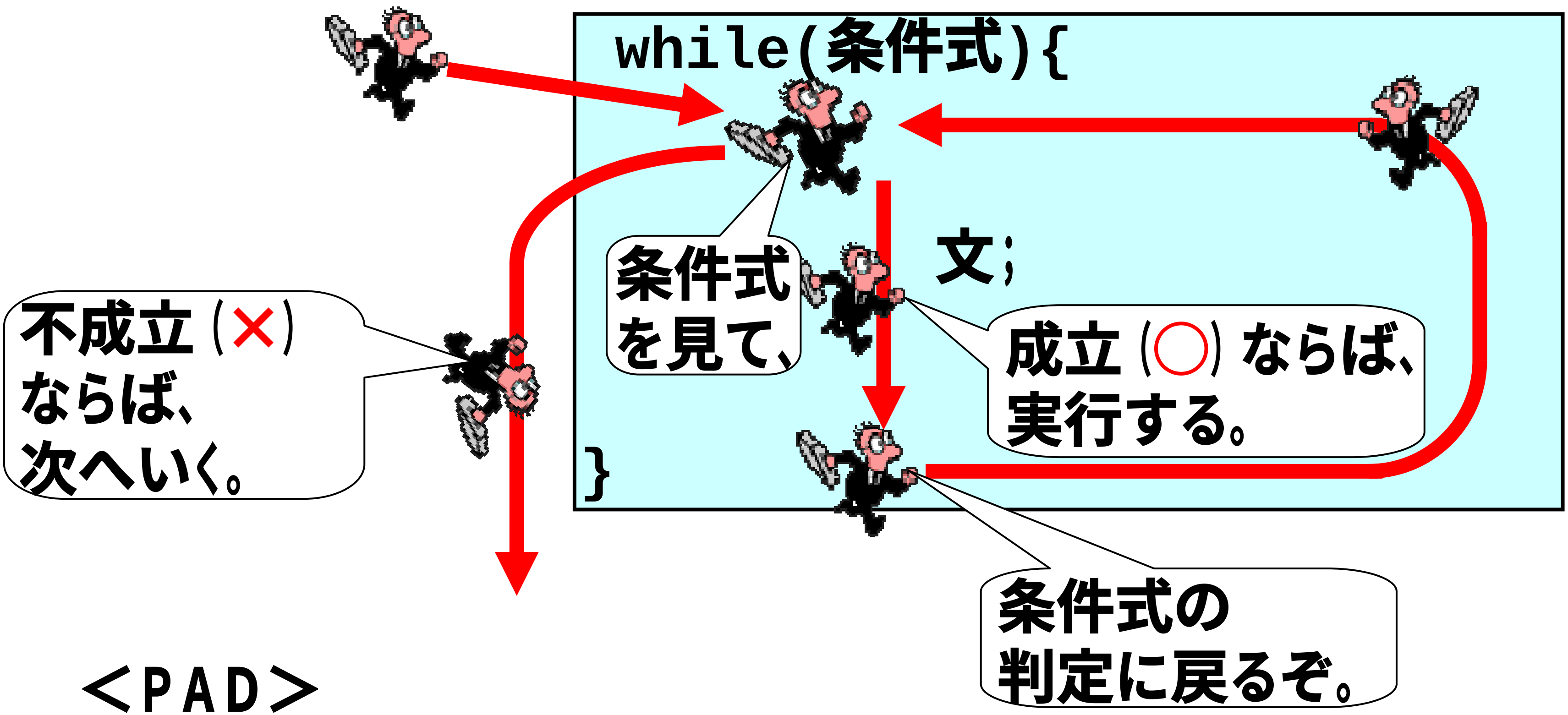
- 近年の干支は下の表のとおり。
- 西暦を表す数字の入力を読み込んで、干支を出力するプログラムを作成してください。
- ちなみに、2004を12で割ると、あまりが0になります。

1995	亥(い)	2001	巳(み)
1996	子(ね)	2002	午(うま)
1997	丑(うし)	2003	未(ひつじ)
1998	寅(とら)	2004	申(さる)
1999	卯(う)	2005	酉(とり)
2000	辰(たつ)	2006	戌(いぬ)

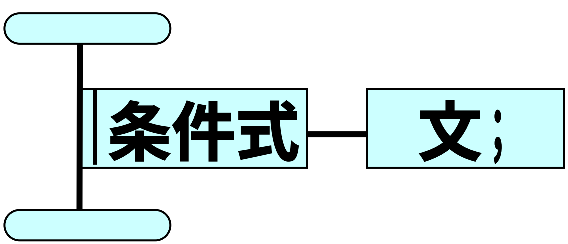


（9．1．1）while文

■最初に基本的な繰り返しwhile文を見ます。



＜PAD＞



(9.1.2) while文の形式・例

■形式

```
while(条件式){
      文;
}
```

■例

＜プログラム＞

```
int a=3;
while(a>0){

   System.out.println(“a=“+a);
   a--;
}
```

“a”が“0”より大きい間、繰り返し

“a=a-1”と同じ。スライド(5.2)参照。

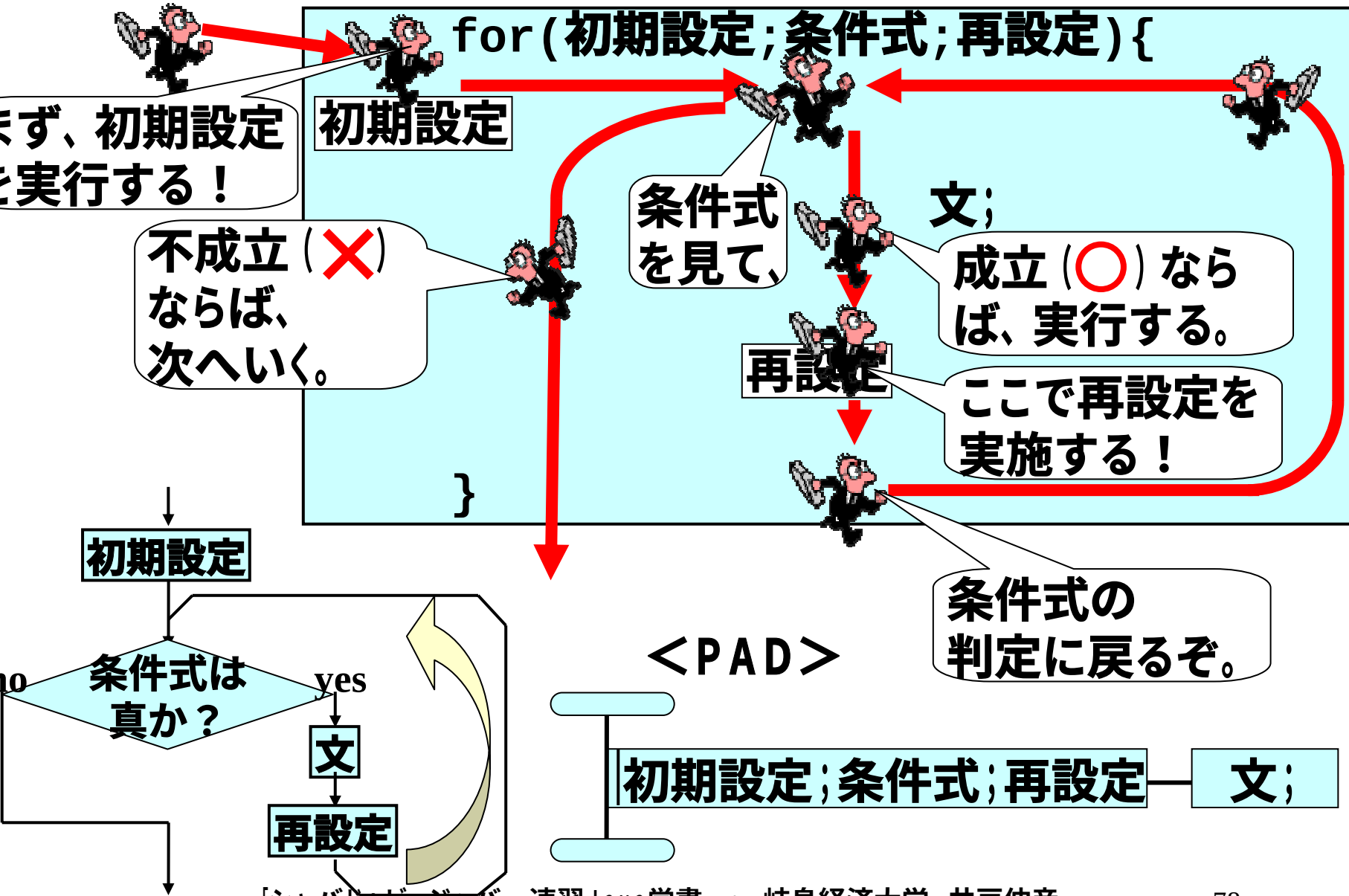
＜出力＞

```
a=3
a=2
a=1
```

■注意

- 繰り返される部分に、条件を変化させていく文（上記の例では、“a--;”）がないといつまでも繰り返しを続けるという事態（無限ループ）になる。

(9.2.1) for文


for(初期設定;条件式;再設定){
初期設定
まず、初期設定を実行する！
条件式を見て、
文;
成立(○)ならば、実行する。
不成立(×)ならば、次へいく。
再設定
ここで再設定を実施する！
}
条件式の判定に戻るぞ。
初期設定
条件式は真か？
no
yes
文
再設定
<PAD>
初期設定;条件式;再設定
文;

(9．2．2) なぜfor文を使うのか？

■while文との比較

- 次の2つのプログラムは、まったく同じ動きとなります。

```
for(文1;条件式;文3){
        文2;
}
```

同じ

```
文1;
while(条件式){
        文2;
        文3;
}
```

※　文1と文3は、複合文ではないこと。
　文2は“continue”を含まないこと。

■なぜfor文を使うのか？

- 上記の例で、文1を「初期化」、文3を「再設定」と捉えることで、繰り返しの条件が明瞭になります。

条件が明瞭

```
for(a=3;a>0;a--){
    文;
}
```

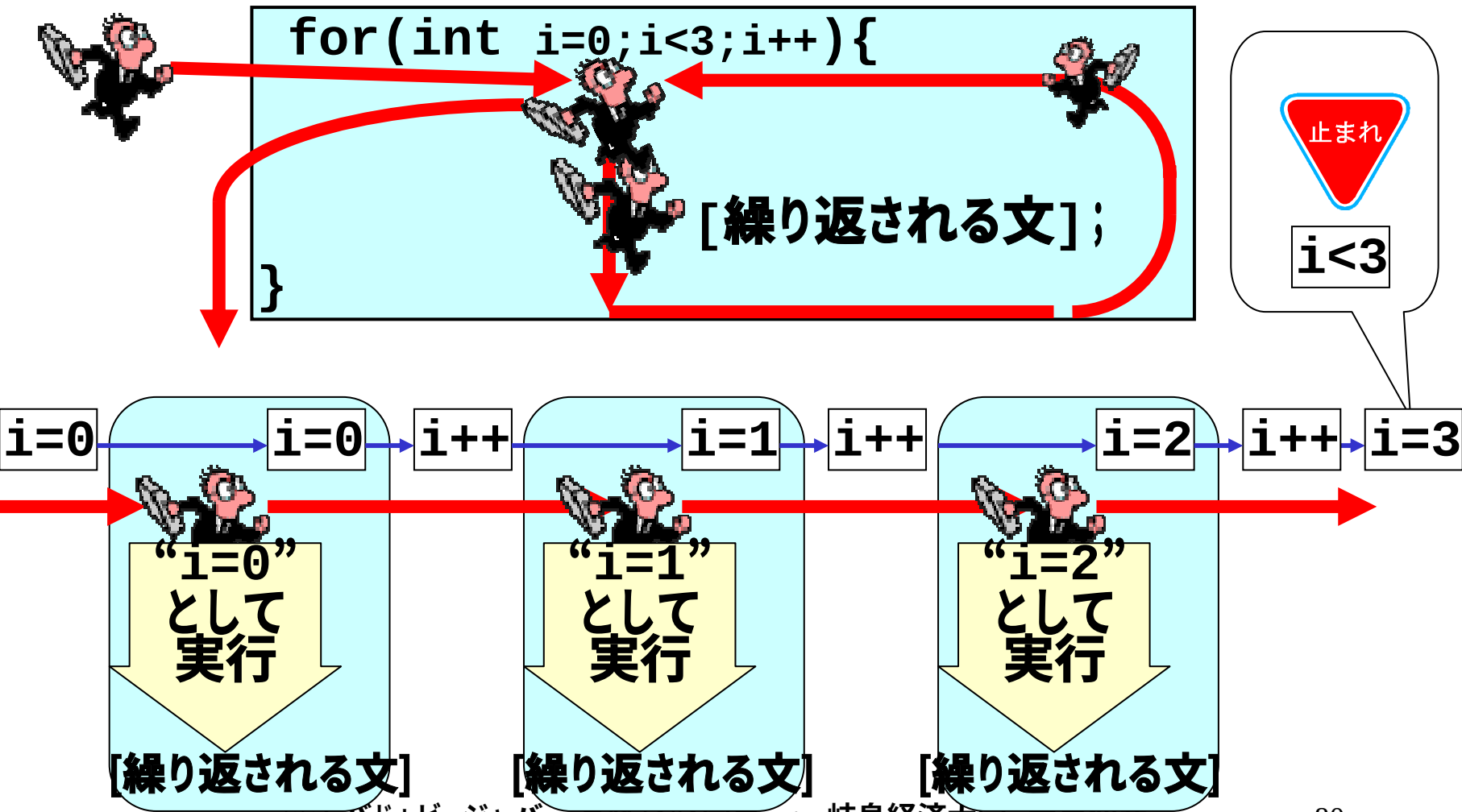
同じ

```
int a=3;
while(a>0){
   文;
   a--;
}
```

ここまでが長くなる。

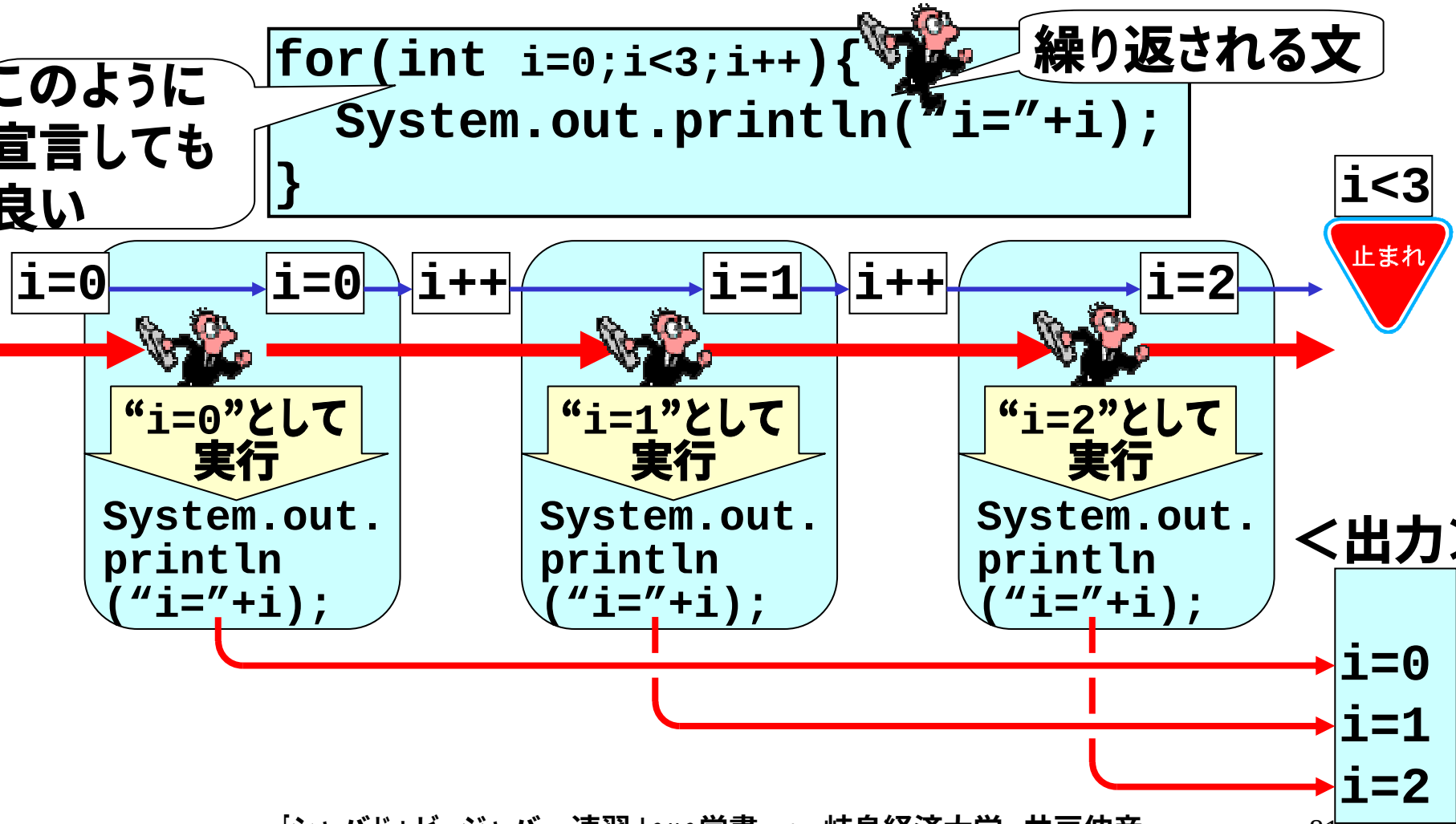
（9．2．3）for文による繰り返し

■for文の中で使われる変数は、繰り返し方を決めることになり、よく“i”（“iteration”の“i”）が使われます。



(9．3．1) for文による繰り返しの例

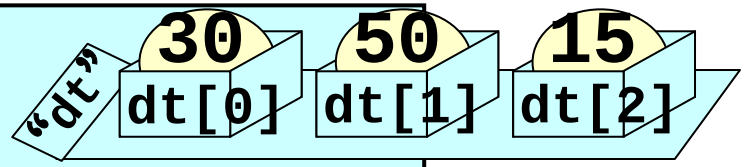
■for文の中で使われる変数は、繰り返し方を決めることになり、よく“i”(“iteration”の“i”)が使われます。

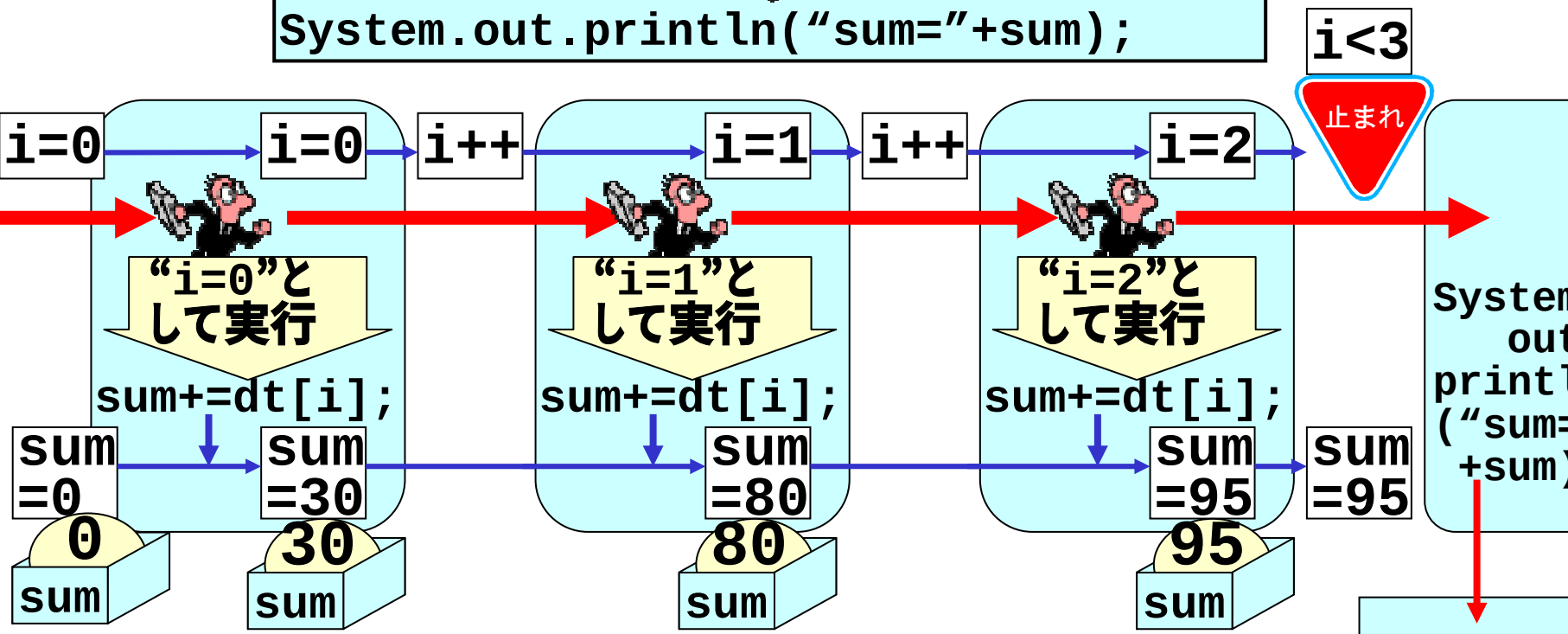


(9．3．2) 配列の合計

```
int dt[] = {30,50,15};
int sum = 0;
for(int i=0;i<3;i++){
  sum += dt[i];
}
System.out.println("sum="+sum);
```

繰り返される文





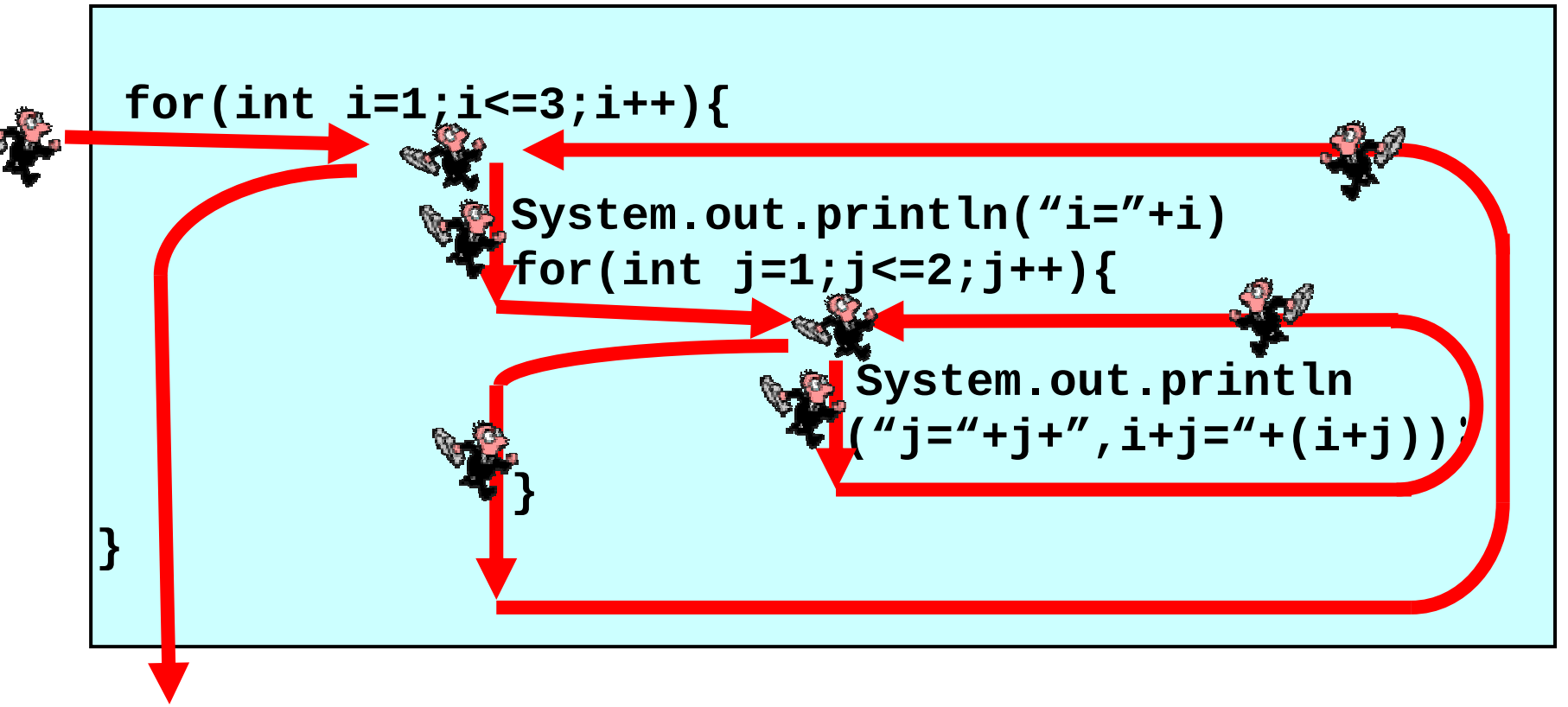
<出力>

sum=95

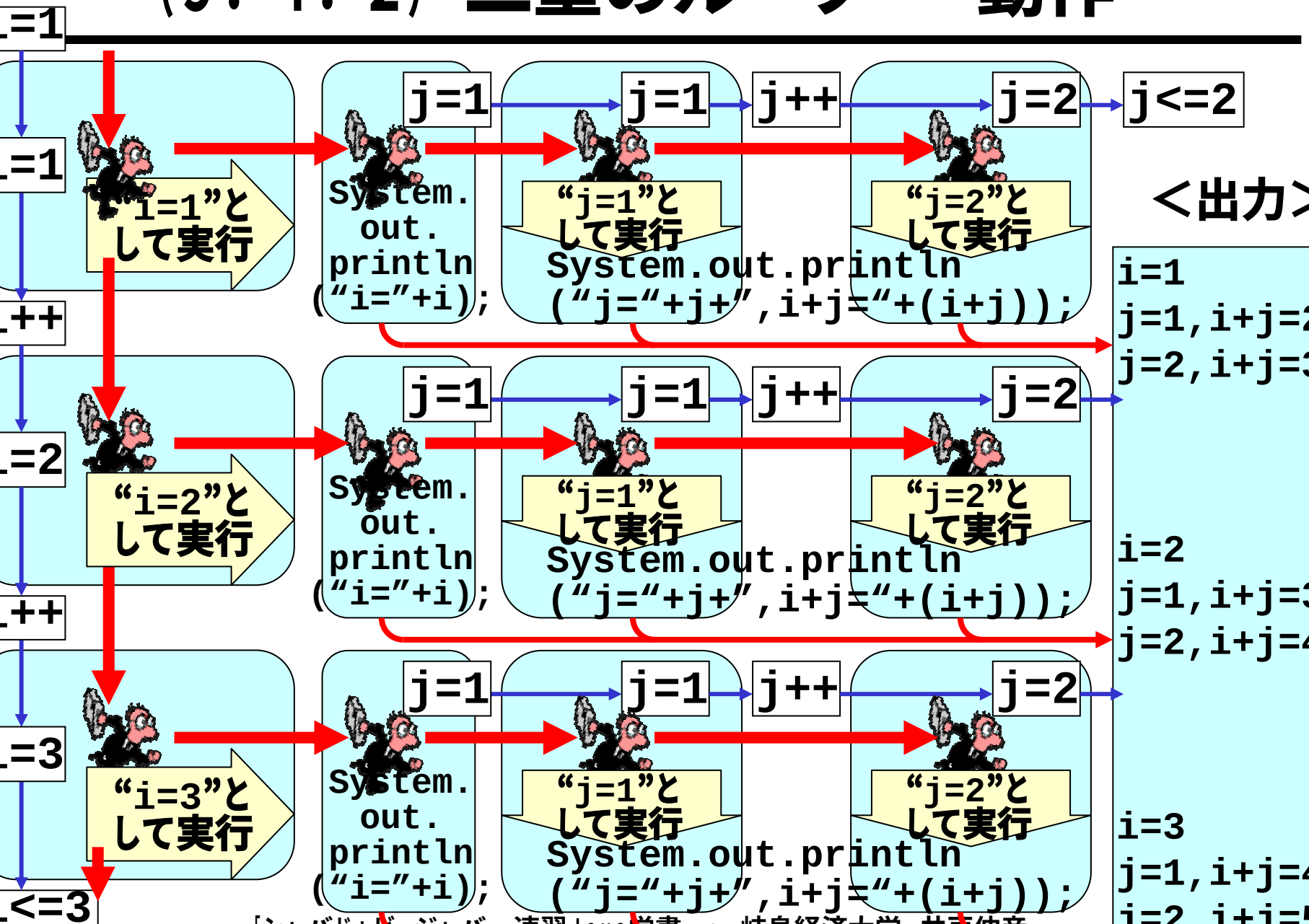
"+=" ⇒代入演算子、スライド(5.2)参照

(9．4．1) 二重のループ

■for文の中にfor文が現れる形も、同じように理解出来ます。



（9．4．2）二重のループ　ー動作ー



(9．5．1) whileによる繰り返し

■繰り返しの条件が、回数として明確な場合には、今まで見てきたとおり、for文を使うことが良いようです。

■while文を使う場合を、次のプログラムを例に見ていきます。

```
import java.io.*;

public class EchoTwice {
  public static void main(String[] args) throws IOException {
    String ss;
    BufferedReader kbd =
      new BufferedReader(new InputStreamReader(System.in));
    while((ss = kbd.readLine()) != null){
      System.out.println(ss+" "+ss);
      if(ss.equals("end")){
        break;
      }
    }
    kbd.close();
  }
}
```

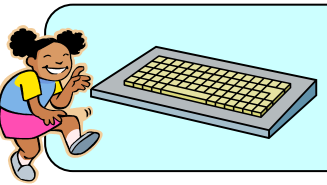
キーボードからの入力。スライド(6)参照。

while文による
繰り返しの部分

(9．5．2) 入力がある限り。。。

■スライド(9.5.1)のプログラムでは、キーボードからの入力がある限り、繰り返しを行っています。

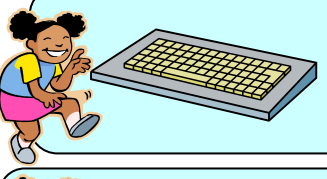
```
while((ss = kbd.readLine()) != null){
```



“asd” 入力 → “asd” ss

繰り返し

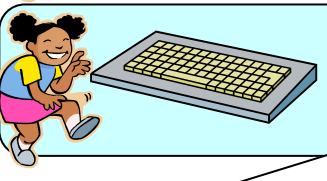
出力



“yz” 入力 → “yz” ss

繰り返し

出力



[Ctrl]+[D] 入力

繰り返し終了

(ss = kbd.readLine()) は、nullとなる

＜出力＞

```
asd asd

yz yz
```

入力の終了マーク
(Windowsでは[Ctrl]+[Z]。eclipseでは効かない場合あり。)

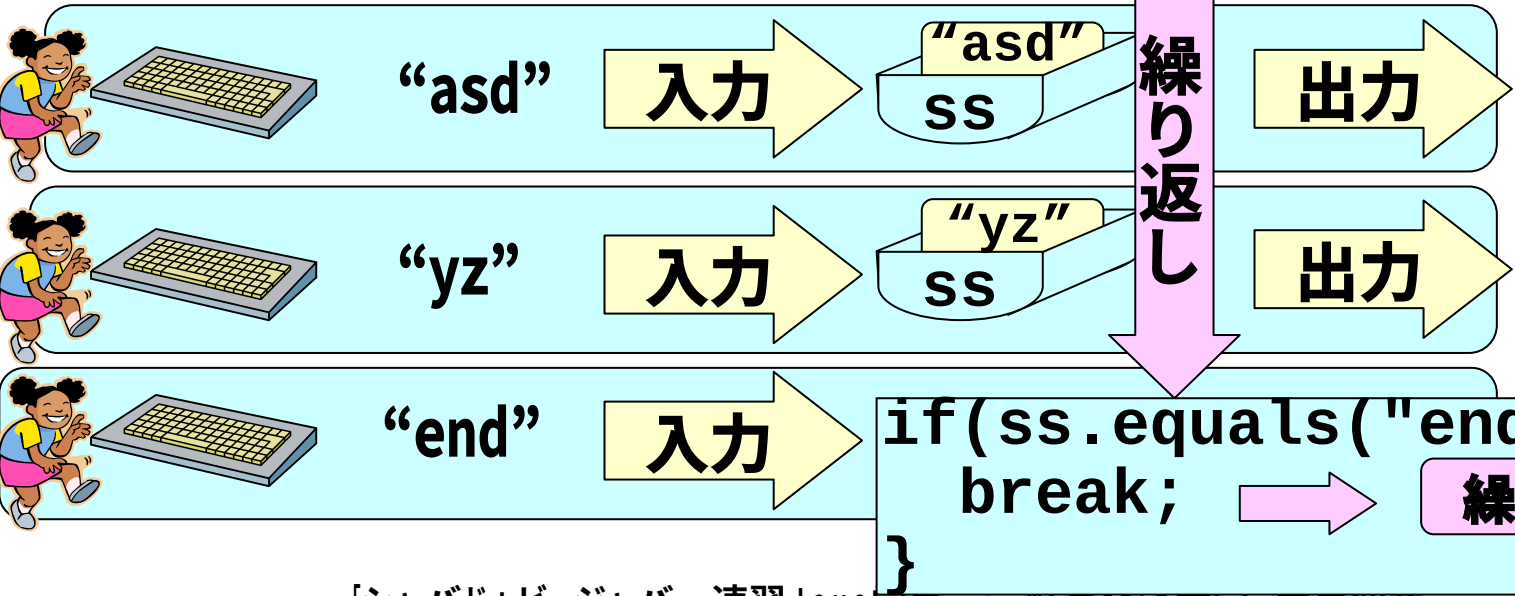
■このように回数ではなく条件で繰り返しが決まる場合は、while文を使うことが多いようです。

(9．5．3) break文

■スライド(9.5.1)のプログラムでは、“end”と入力しても繰り返しが終了します。

■これは、入力が“end”のとき、break文が実行されるからです。break文は、繰り返しを終了させる働きを持ちます。

```
while((ss = kbd.readLine()) != null){
```

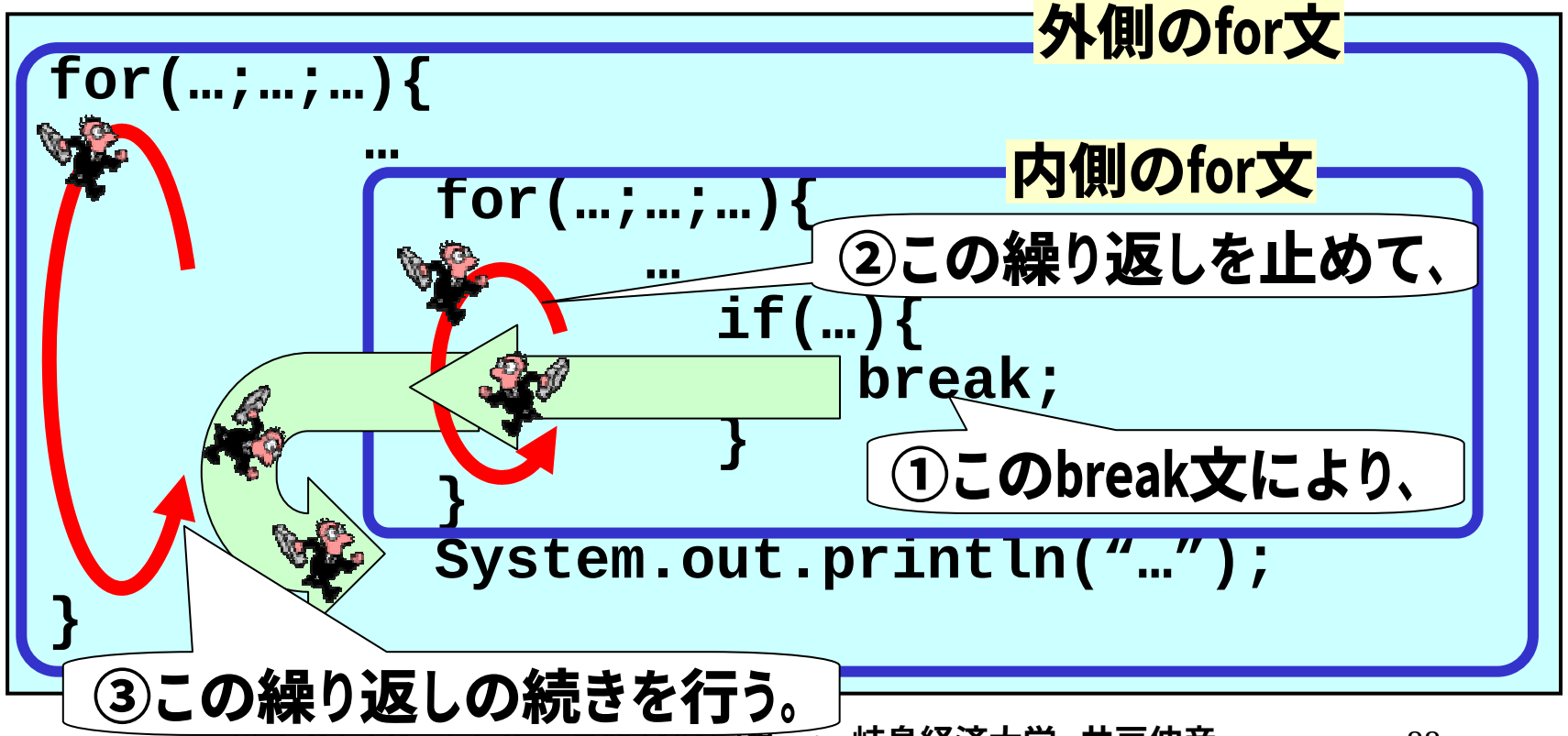


＜出力＞

```
asd asd

yz yz
```

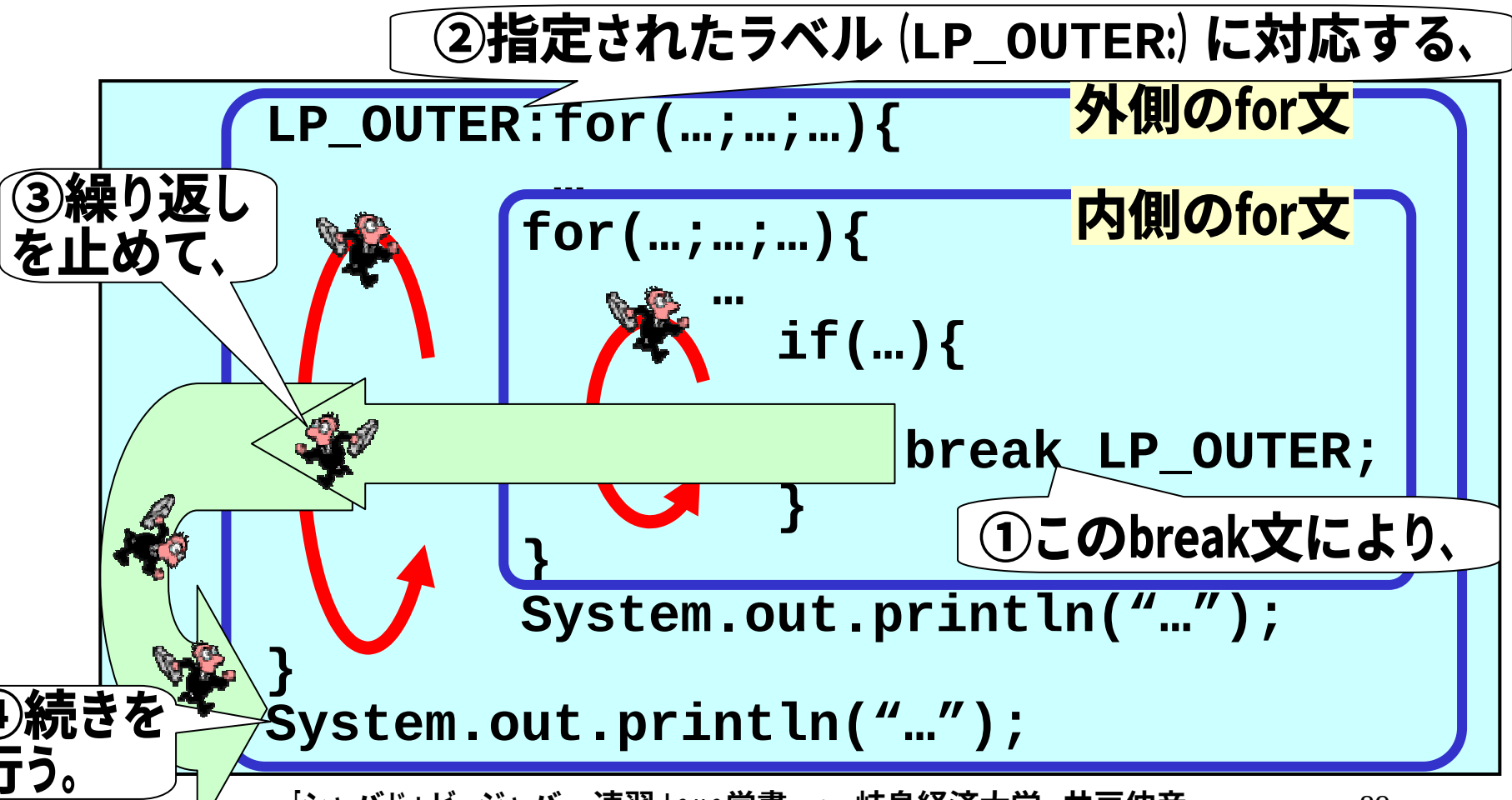
(9．6．1) break文で中止する繰り返し

■break文により終了される繰り返しは、そのbreak文を含む一番小さい繰り返し(＝一番内側のfor文、while文)となります。すなわち、2重ループなどでbreak文を実行すると、内側の繰り返しを止めて、外側の繰り返しを続けることになります。



(9．6．2) ラベルつきbreak

■一番内側の繰り返し以外の繰り返しを止めるには、繰り返しにラベルをつけ、そのラベルを指定したbreak文(ラベルつきbreak文)を用います。



(9.7)例題9.1:最大値

■次のように初期化された配列のそれぞれの値の、最大を求めるプログラムを作成してください。

```
int dt[] = {5,4,12,23,7};
```

■解答例

```
public class Reidai91 {
   public static void main(String[] args) {
      int dt[] = {5,4,12,23,7};
      int max=dt[0];
      for(int i=1;i<dt.length;i++){
         if(max<dt[i]){
            max=dt[i];
         }
      }
      System.out.println("最大値="+max);
   }
}
```

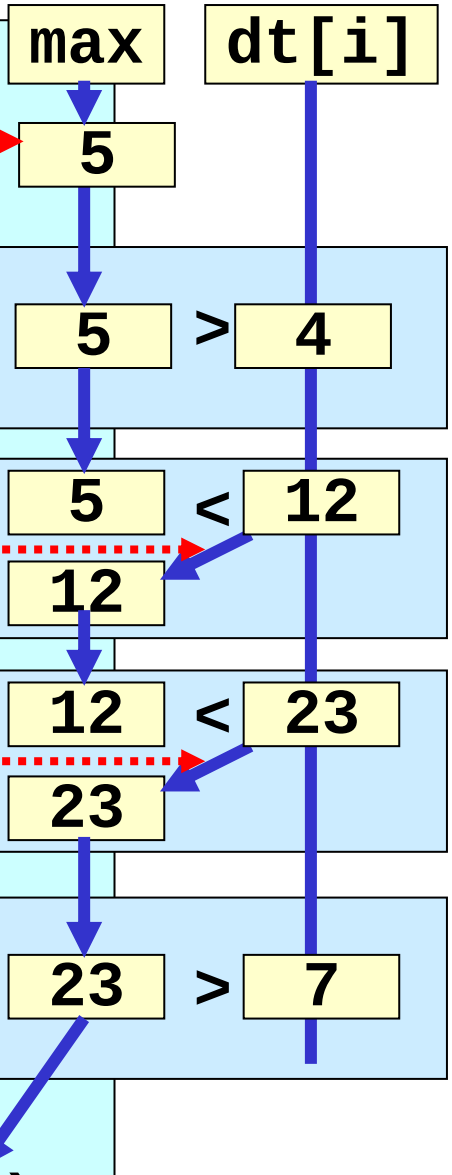
(9.7.1) 解答例のプログラムの見方

```
int dt[] = {5,4,12,23,7};
int max=dt[0];
for(int i=1;i<dt.length;i++){
    if(max<dt[i]){
        max=dt[i];
    }
}
System.out.println("最大値="+max);
```

- `int max=dt[0];` ：ひとまず、最初の要素を最大値とする。
- `for(int i=1;i<dt.length;i++){` ：2つめ以降の要素について、順次調べる
- `if(max<dt[i]){` ：もし、要素が最大値よりも大きければ
- `max=dt[i];` ：その要素を最大値とする

(9.7.2) 動作

```
3.  int dt[] = {5,4,12,23,7};
4.  int max=dt[0];
5.  for(int i=1;i<dt.length;i++){
      6.    if(max<dt[i]){
      7.      max=dt[i];
      8.    }
      6.    if(max<dt[i]){
      7.      max=dt[i];
      8.    }
      6.    if(max<dt[i]){
      7.      max=dt[i];
      8.    }
      6.    if(max<dt[i]){
      7.      max=dt[i];
      8.    }
9.  }
0.  System.out.println("最大値="+max);
```



(9．8) 例題9．2：文字絵

＜出力＞

```
#####
*####
**###
***##
****#
*****
```

■右のような出力を行うプログラムを
作成してください（for文を使ってください）。

■解答例：

```
 1.public class Reidai92 {
 2.  public static void main(String[] args) {
 3.    for(int i=0;i<6;i++){
 4.      for(int j=0;j<i;j++){
 5.        System.out.print("*");
 6.      }
 7.      for(int j=0;j<5-i;j++){
 8.        System.out.print("#");
 9.      }
10.      System.out.println("");
11.    }
12.  }
13.}
```

(9.8.1)“*”の部分

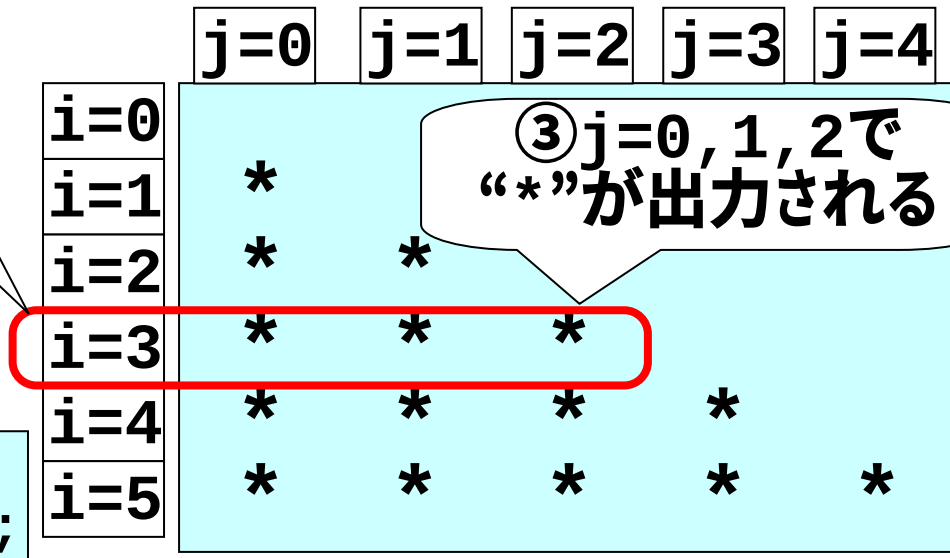
■アスタリスク“*”の部分だけをまず考えます。

```
 3.  for(int i=0;i<6;i++){
 4.    for(int j=0;j<i;j++){
 5.      System.out.print("*");
 6.    }
10.    System.out.println("");
11.  }
```

①例えば
i=3の時は、

②4行目は
このようになるので、

```
4. for(int j=0;j<3;j++){
5.   System.out.print("*");
6. }
```



(9．8．2)“#”の部分

■i行目で出力される“#”の数は、“5-i”と表すことが出来ます。

```
#####
*####
**###
***##
****#
*****
```

行	#の数	iで表すと…
0行目(i=0)	5	= 5-i = 5-0
1行目(i=1)	4	= 5-i = 5-1
2行目(i=2)	3	= 5-i = 5-2
3行目(i=3)	2	= 5-i = 5-3
4行目(i=4)	1	= 5-i = 5-4
5行目(i=5)	0	= 5-i = 5-5

■よって、“#”を出力は、“0”から“5-i”まで繰り返せばよい訳です。

```
 3. for(int i=0;i<6;i++){
 7.   for(int j=0;j< 5-i ;j++){
 8.     System.out.print("#");
 9.   }
10.   System.out.println("");
11. }
```

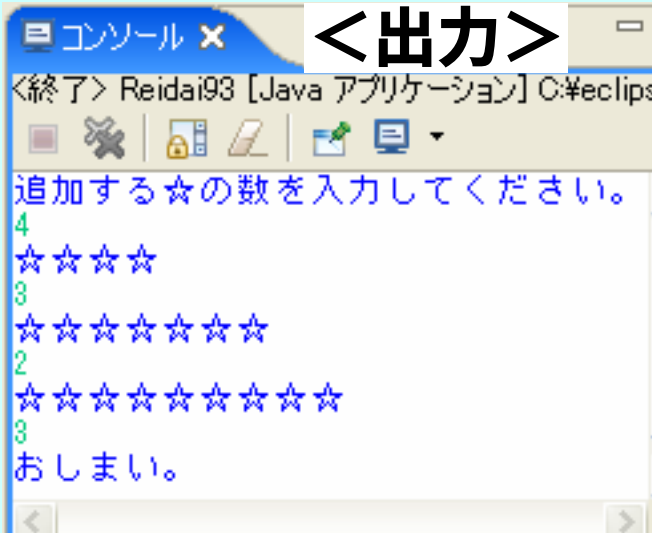
(9．9) 例題9．3：星の増加

■数字を連続して入力し、その数だけ増加させながら☆を印刷するプログラムを作成してください。但し、１０個を超えた場合はプログラムを終了させます。

```
 1:import java.io.IOException;
 2:public class Reidai93 {
 3:  public static void main(String[] args) throws IOException {
 4:    String unit,ss;
 5:    java.io.BufferedReader kbd
 6:    = new java.io.BufferedReader(
 7:                      new java.io.InputStreamReader(System.in));
 8:    String ssout="";
 9:    System.out.println("追加する☆の数を入力してください。");
10:    while((ss = kbd.readLine())!=null){
11:      int d=Integer.parseInt(ss);
12:      for(int i=0;i<d;i++){
13:        ssout+="☆";
14:      }
15:      if(ssout.length()>10){
16:        System.out.println("おしまい。");
17:        break;
18:      }
19:      System.out.println(ssout);
20:    }
21:  }
22:}
```

(13.1.1)参照

＜出力＞

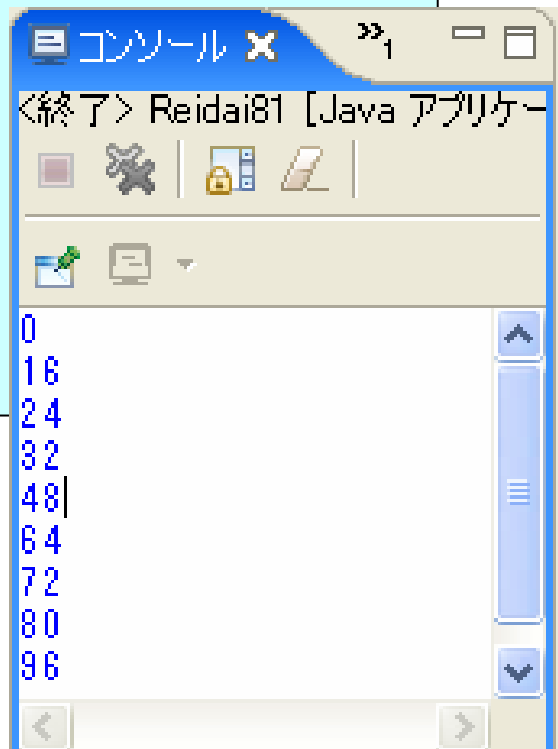


(9.10) 例題9.4:倍数

■0から100までの数の中で、16の倍数であるか、24の倍数である数をすべて打ち出してください。

■解答例:

```
public class Reidai81 {
  publlc static void main(String[] args) {
    for(int i=0;i<=100;i++){
      if((i%16==0)||(i%24==0)){
        System.out.println(i);
      }
    }
  }
}
```



(9.11.1) 課題

■課題9－1（例題1を参考にしてください）

- 次のように初期化された配列のそれぞれの値の、①平均、②最大、③最小を求めるプログラムを作成してください。

```
double dt[] = {172.3,168.5,177.8,181.5,174.8};
```

■課題9－2（例題2を参考にしてください）

- 次のような出力を行うプログラムを作成してください（for文を使ってください）。

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

(9.11.2) 課題

■課題9－3

- 下図のような入出力を行うプログラムを作成してください。

```
$ java Kadai93
abc
abc
ed
abced
ccc
abcedccc
clear

xyz
xyz
end
good bye.
```

入力する

プログラムが出力する

クラス名（＝ファイル名）を“Kadai93”とします。前述のとおり、大文字から始めます。

(9．11．3) 課題

■課題9－4

- 右のような出力を得るプログラムを作成してください (二重ループを用います)。

コンソール

<終了> Kadai123 [J

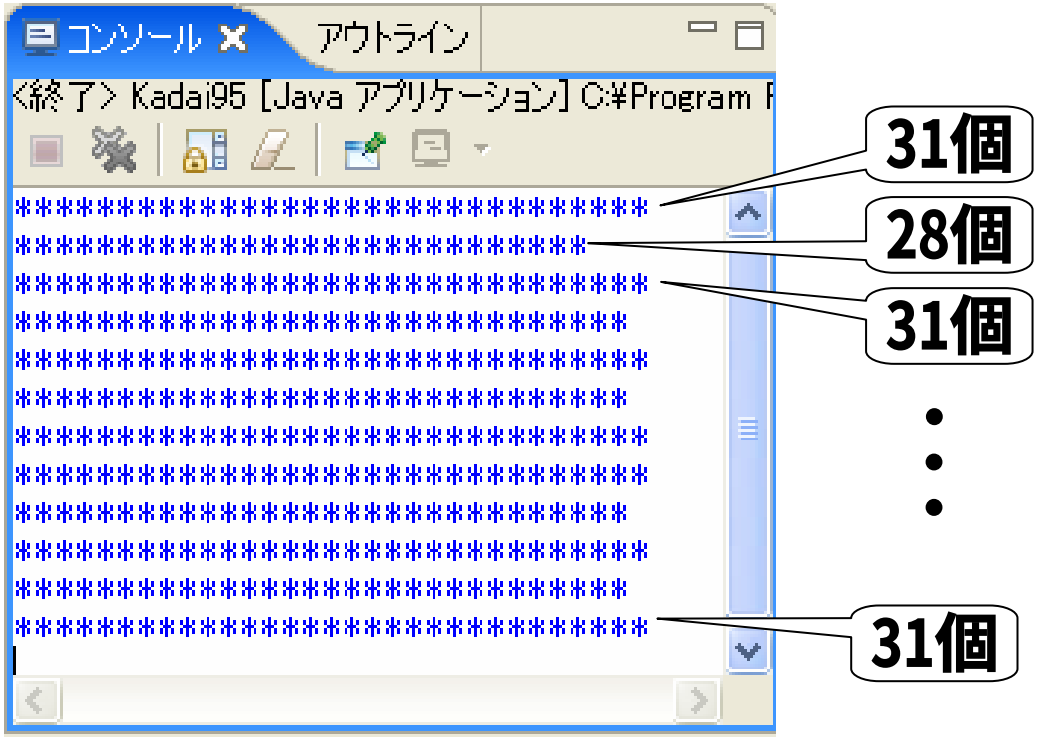
```
0123456789
1234567890
2345678901
3456789012
4567890123
5678901234
6789012345
7890123456
8901234567
9012345678

0123456789
1123456789
2223456789
3333456789
4444456789
5555556789
6666666789
7777777789
8888888889
9999999999
```

(9．11．4) 課題

■課題9－5

- 下のような配列“months”を宣言し、その下のような出力を得るプログラムを作成してください。

```
int[] months={31,28,31,30,31,30,31,31,30,31,30,31};
```

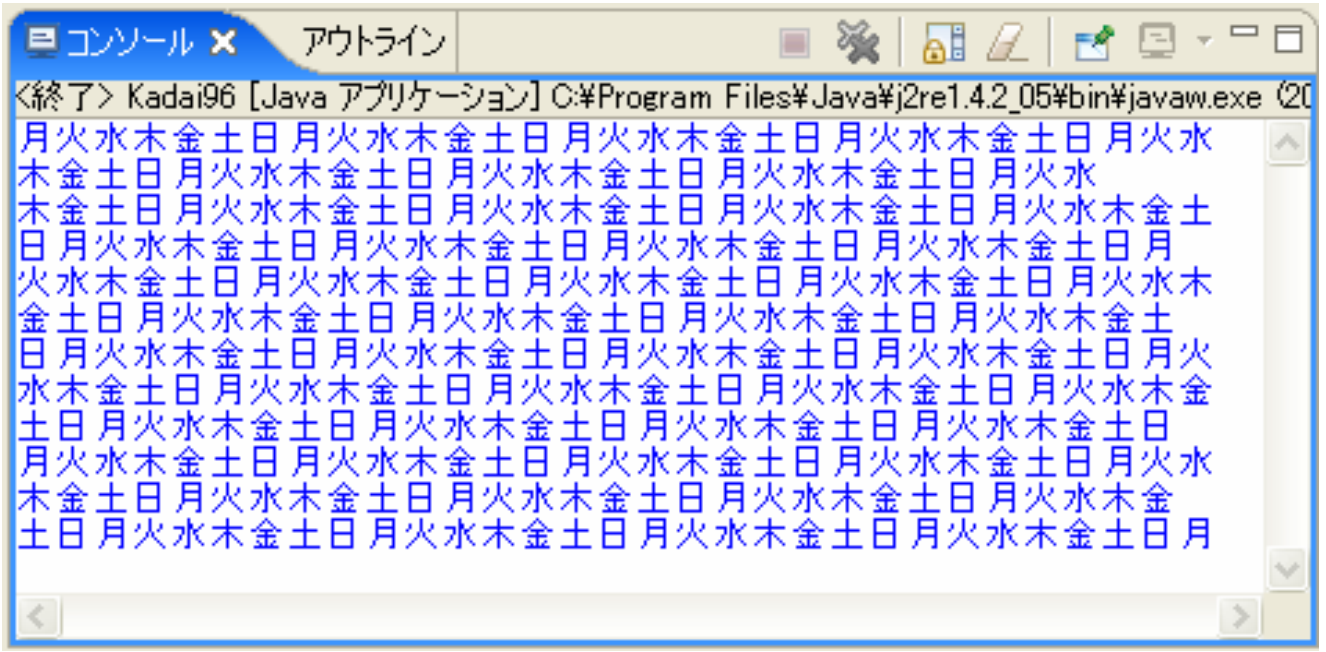


(9．11．5) 課題

■課題9－6

- 課題9－5のプログラムを改造して、1月1日が月曜日の場合の、各月の曜日の羅列を出力するプログラムを作成してください。ただし、次のような配列を宣言するものとします。

```
String[] days={"月","火","水","木","金","土","日",};
```



(10.1) プログラムの見栄え

■ここで説明することは、作成したプログラムの実行に直接関わることではありません。単にソースコードの“見た目”に関わるだけです。しかしながら、どうでも良いということではなく、重要なことです。

■次の2つのプログラムの断片を見て、どう感じますか？　どちらのプログラムの実行結果も同じですが、＜B＞のようなプログラムを書く人と共同でプログラムを開発する気は起こらないと思います。

＜A＞

```
int x=0
int y=3;
int z=5;
x = z-y;
System.out.println(“x=”+x);
```

＜B＞

```
int x=0
     int y=3;
  int z=5;
         x = z-y;
 System.out.println(“x=”+x);
```

(10.2)段下げ

■クラス、メソッド、if文、while文、for文では、その中身の実行文について通常4文字段下げを行うことが、見栄えの上でのルールとなっています。これを守らないプログラムを書く人は、ある意味前ページの＜B＞より酷いと言えます。“{”と“}”の位置にも注意してください。

クラス

メソッド

for文

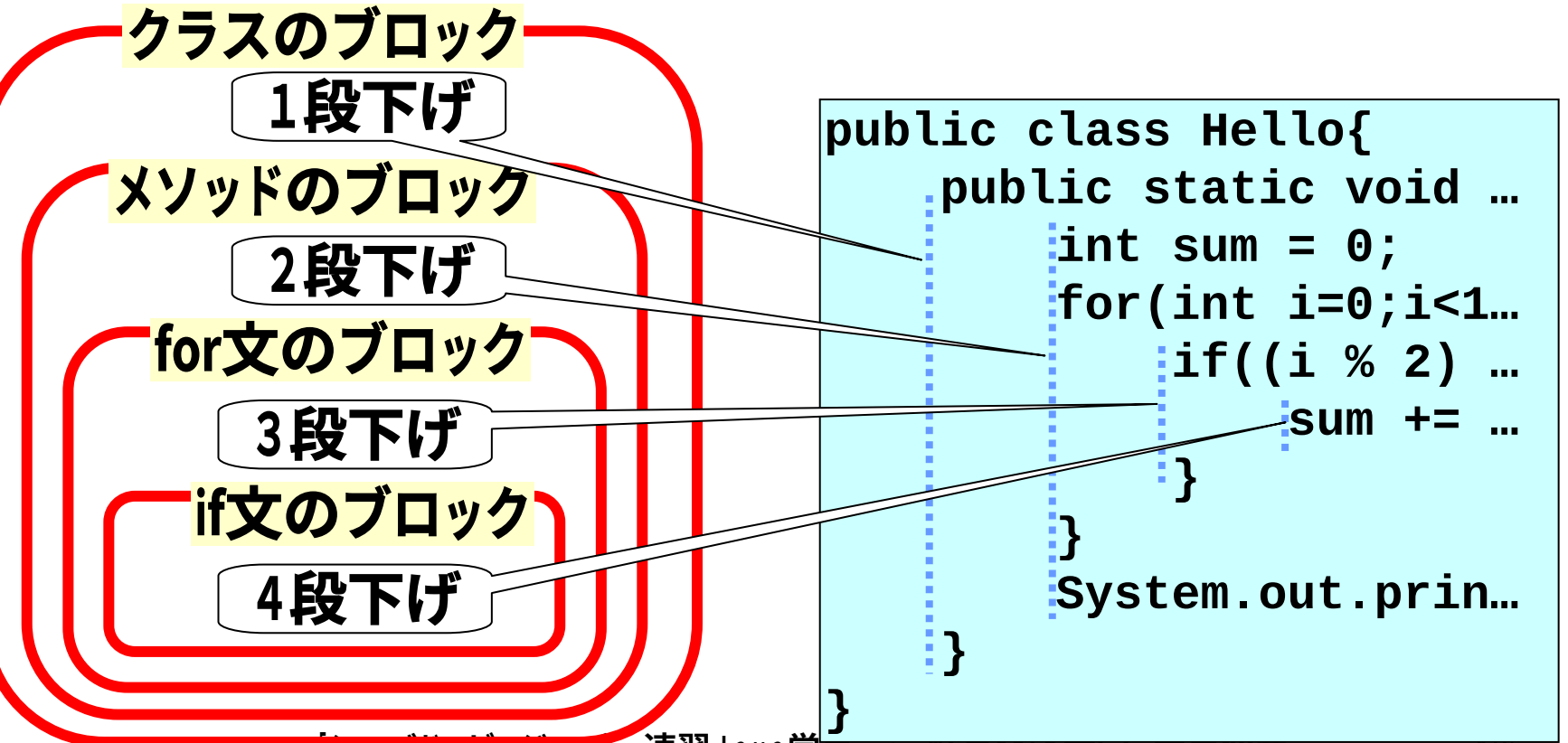
if文

```
public class Hello{
    public static void main(String args[]){
        int sum = 0;
        for(int i=0;i<10;i++){
            if((i % 2) ==0){
                sum += i;
            }
        }
        System.out.println(“sum=“+sum);
    }
}
```

(10．3)ブロック

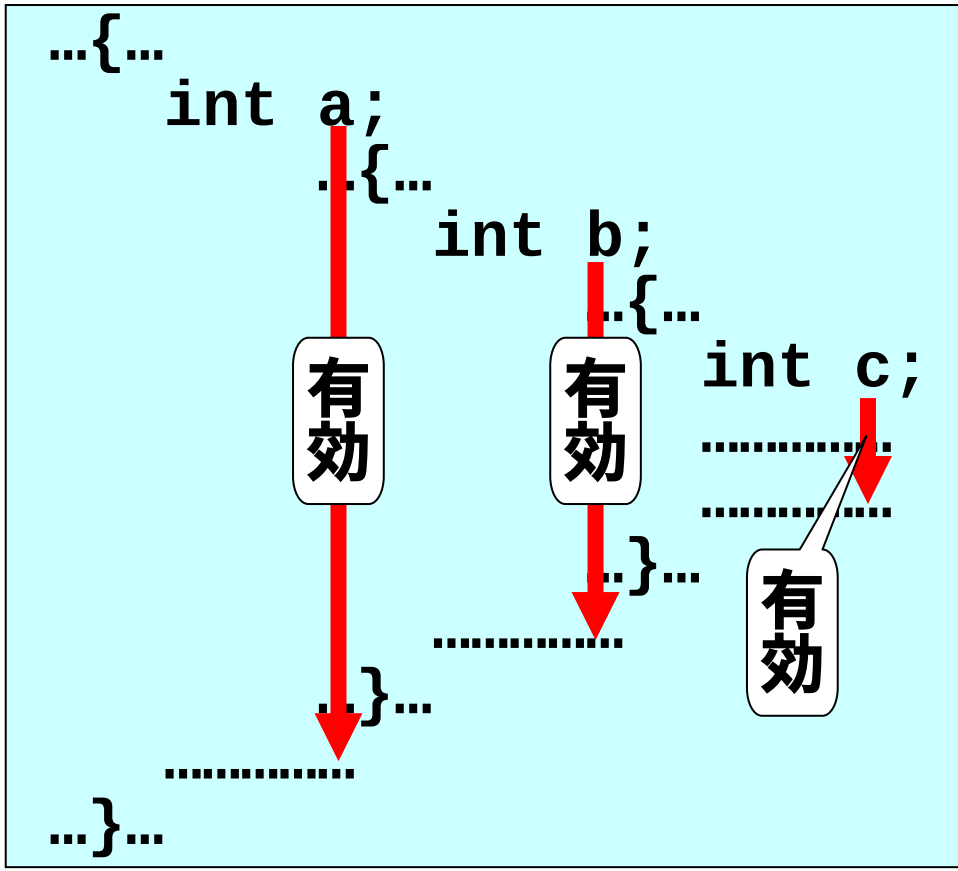
■“{”で始まり、これに対応する“}”で終わる範囲のことを、ブロックを言います。

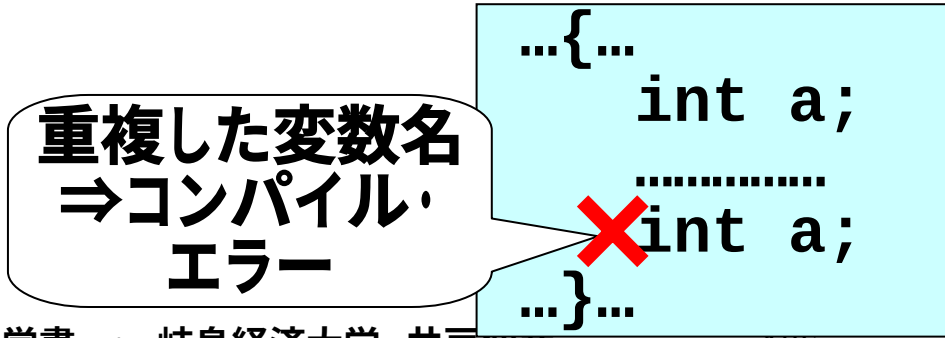
■どれだけ段下げを行うかは、いくつのブロックに囲まれているかにより決まる訳です。



(10．4) 名前空間 (変数の有効範囲)

- 名前空間とは、宣言した変数が使えるか範囲のことを指すと理解しておいてください。
- 原則として、変数はそれが宣言された行から、その行が属するブロックが終了するところまで有効となります。
- ある変数が有効な範囲で、同じ名前の変数を宣言すれば、それはコンパイル･エラーとなります。





(10.5)誤りの例

■次のプログラムでは、変数“str”が有効範囲外で利用されており、「strは解決できません」というコンパイルエラーとなります。

```
public class Rei102 {
  public static void main(String[] args) {
    int[] dec={1,2,3,4,5,};
    for(int i=0;i<dec.length;i++){
      if(dec[i]%2==0){
        String str=dec[i]+"は偶数です。";
      }
      System.out.println(str);
    }
  }
}
```

変数strの有効範囲はここで終わり。

strは宣言されていないこととなる⇒コンパイル・エラー

(10.6) 初期化 (名前空間とは直接関係なし)

■前スライドのプログラムを次のように直すと、やはり「初期化されていない可能性があります」というコンパイルエラーとなります。

```
1. public class Rei102 {
2.   public static void main(String[] args) {
3.     int[] dec={1,2,3,4,5,};
4.     for(int i=0;i<dec.length;i++){
5.       String str;
6.       if(dec[i]%2==0){
7.         str=dec[i]+"は偶数です。";
8.       }
9.       System.out.println(str);
10.    }
11.  }
12.}
```

strは値が設定されていない可能性がある
⇒コンパイル･エラー

■次のように、なにかの値を入れておくことで対処します。

```
5.       String str=null;
```

(10.7) “null” (名前空間とは直接関係なし)

■“null”(nullリテラル)は、「キーボードからの入力がある限り繰り返しを行う」プログラムで既に使ってきました。

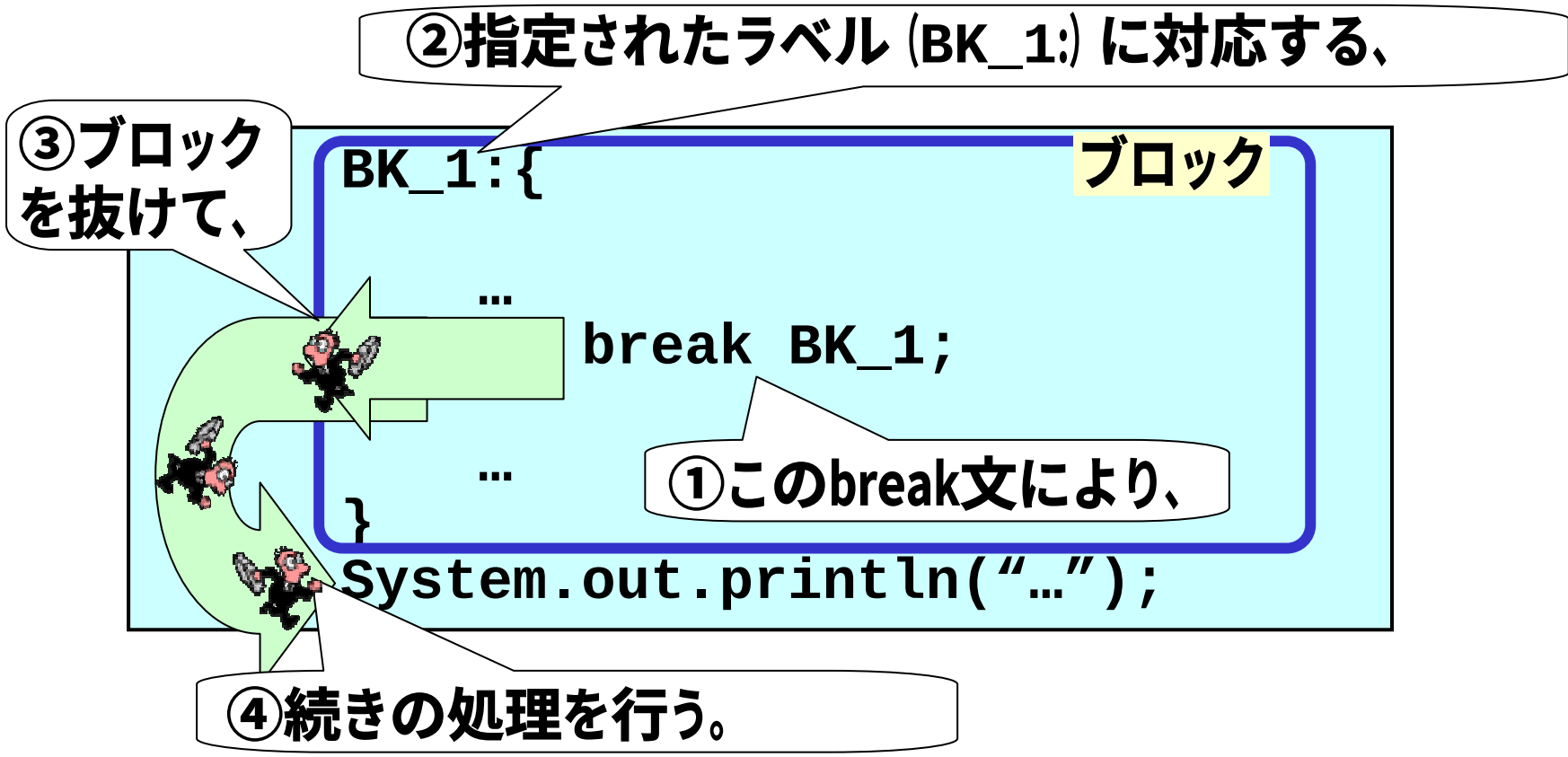
```
while((ss = kbd.readLine()) != null){
```

■“null”は、「変数が有効なインスタンスを保持していないこと」を表します。

- インスタンスについては、本スライドの後で学ぶスライドにて説明します。
- ここでは、「nullは“何もないこと”を表す」と理解しておいてください。
- なお、int型やchar型などの基本データ型に“null”は設定できません。初期値を与える必要がある場合は、適当な数などを代入しておいてください。

(10.8) ブロックに対するbreak文

■ (9) 章にて説明したbreak文は、for文やwhile文などの繰り返しに対して使うだけではなく、①if文のブロックや、②for文やwhile文無しのブロックだけにも使うことが出来ます。

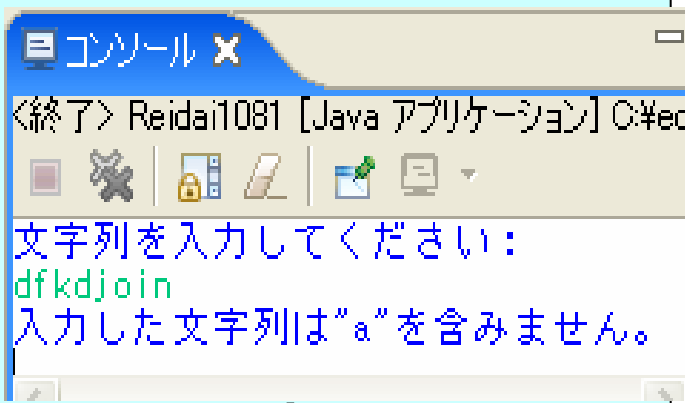


(10．8．1)“見つからない場合の処理”

■ブロックをbreak文で抜ける必要があるのかというと、必要な場合は時々あります。

■入力した文字列の中に文字“a”が含まれていない場合に“aを含みません”と出力する処理は次のように書くことができます。

```
(前略)
   System.out.println("文字列を入力してください:");
   String ss = kbd.readLine();
   BK_1:{
     for(int i=0;i<ss.length();i++){
       if(ss.charAt(i)=='a'){
         break BK_1;
       }
     }
     System.out.println("入力した文字列は¥"a¥"を含みません。");
   }
(後略)
```



(10．8．2) 処理の概要

■for文により文字‘a’を探し回って、

- 見つかれば、
  - ◆break文によりブロックを抜けることで、ブロック内のメッセージ出力は行われません。
- 見つかれば、
  - ◆for文の次にある、ブロック内のメッセージ出力を行います。

```
System.out.println(“…");
String ss = kbd.readLine();
BK_1:{
    for(int i=0;i<ss.length();i++){
        if(ss.charAt(i)=='a'){
            break BK_1;
        }
    }
    System.out.println(“入力した…");
}
```

ブロック

①探し回って、

②見つかればブロック“BK_1”を抜ける

③見つからなければ、ブロック内のメッセージ出力を行う。

(10.8.3) 変数の導入

■もちろん、「'a'を含むか否かを表す」変数を導入すれば、スライド(10.8.1)の処理は下のようにも書けます。

- これでも問題はありませんが、(10.8.1)の流儀で書きたくなる場面はよくあります。
- 「導入する必要のない変数は導入しない」という考えにより、筆者自身は(10.8.1)の処理を好みます。

```
(前略)
•   String ss = kbd.readLine();
•   boolean noA=true;
•   for(int i=0;i<ss.length();i++){
•     if(ss.charAt(i)=='a'){
•       noA=false;
•       break;
•     }
•   }
•   if(noA){
•     System.out.println("入力した文字列は¥"a¥"を含みません。");
•   }
(後略)
```

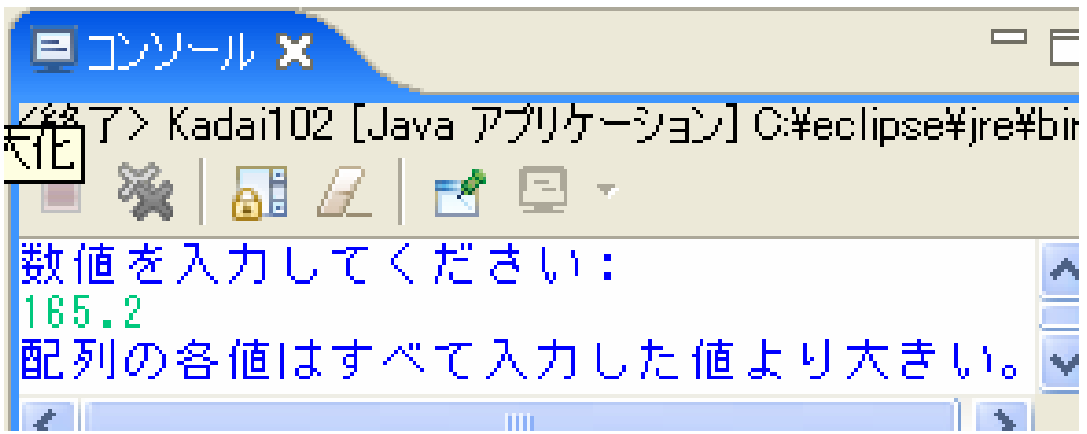
(10.9) 課題

■課題10－1

- ここまでに作成したすべてのプログラムについて、段下げが正しくなされているか否かをチェックしてください。

■課題10－2

- 下図のように初期化された配列を用意します。数値を入力して、配列内のすべての値が入力した値よりも大きい場合に、“配列の各値はすべて入力した値より大きい。”と出力するプログラムを作成してください（スライド(10.8.1)流のやり方で作成してください）。

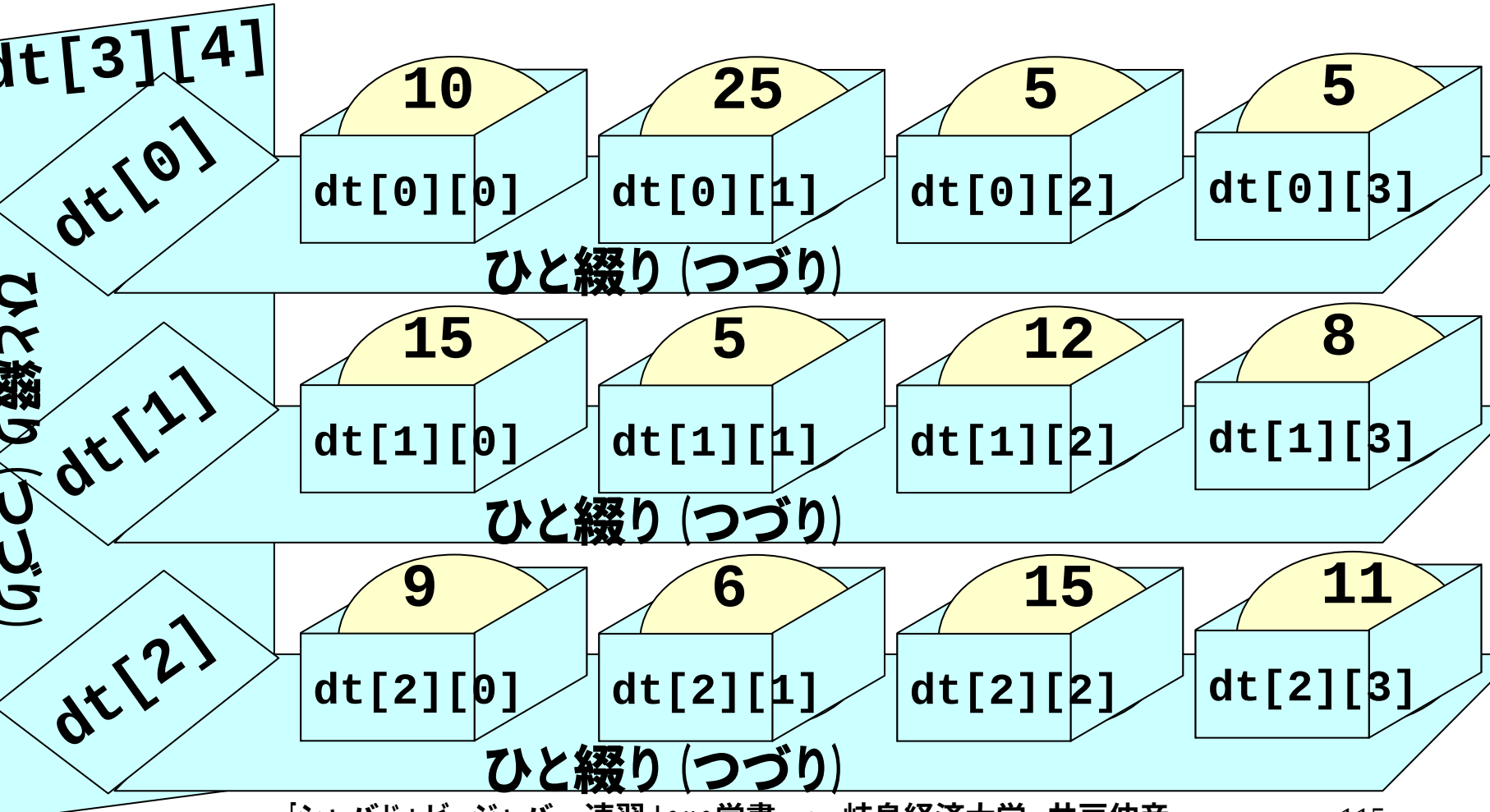



```
double dt[] = {172.3,168.5,177.8,181.5,174.8};
```

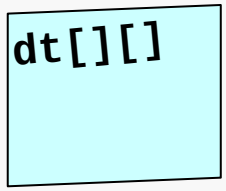
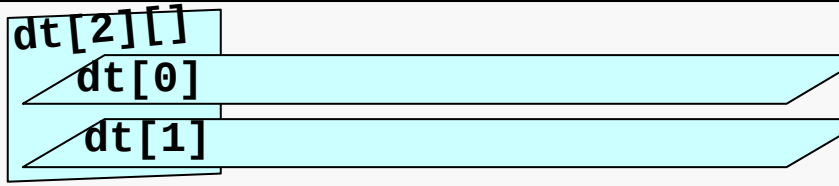
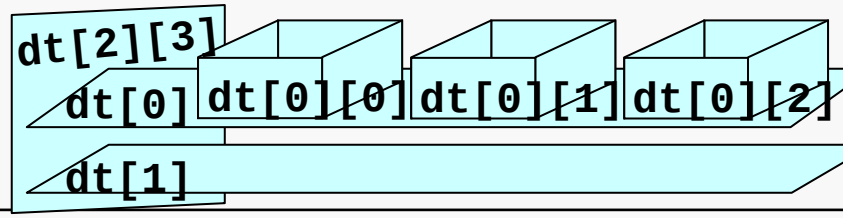
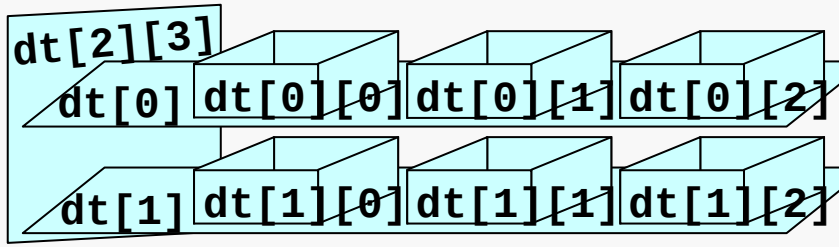
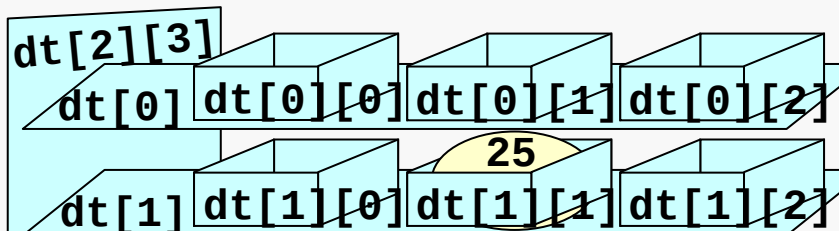
(11) 多次元配列

■配列は、多次元にすることも出来ます。

■3×4の2次元配列は、次のようなイメージとなります。



(11.1) 多次元配列の宣言

操作	記述例	操作跡の変数の内容
①整数型(int)の2次元配列変数dtを宣言する	`int dt[][];`	
②1次元目のひと綴りの実体を確保する	`dt = new int[2][];`	
③2次元目のひと綴りの実体を確保する(1つめ)	`dt[0] = new int[3];`	
④2次元目のひと綴りの実体を確保する(2つめ)	`dt[1] = new int[3];`	
番号(1,1)の要素(dt[1][1])に25を入れる	`dt[1][1]=25;`	

(11.2) 宣言の別のやり方

■前スライドのプログラムは、次のようにも書くことが出来ます。

```
int dt[][] = new int[2][];
dt[0] = new int[3];
dt[1] = new int[3];
dt[1][1] = 25;
```

```
int dt[][];
dt = new int[2][3];
dt[1][1] = 25;
```

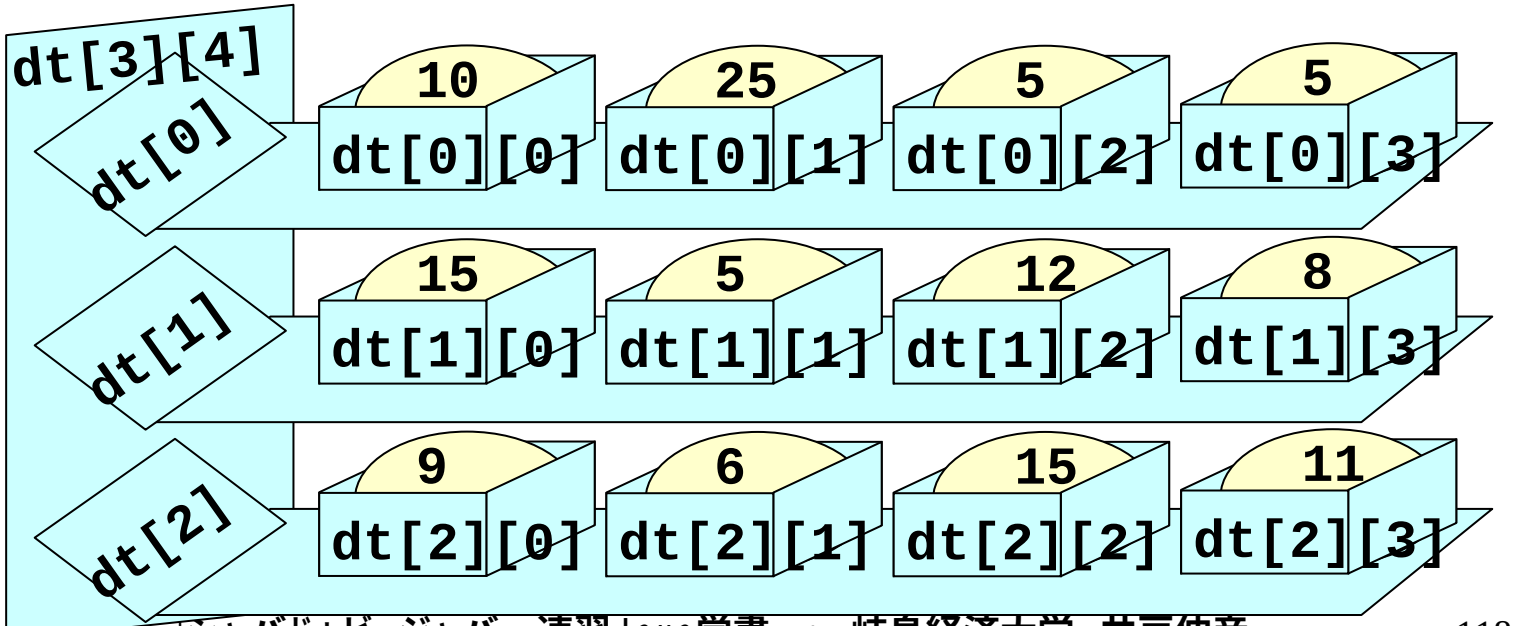
```
int dt[][] = new int[2][3];
dt[1][1] = 25;
```

(11.3) 初期化

■多次元配列も、次のように初期化が可能です。

```
int dt[][] = {
      {10,25, 5, 5},
      {15, 5,12, 8},
      { 9, 6,15,11},
};
```

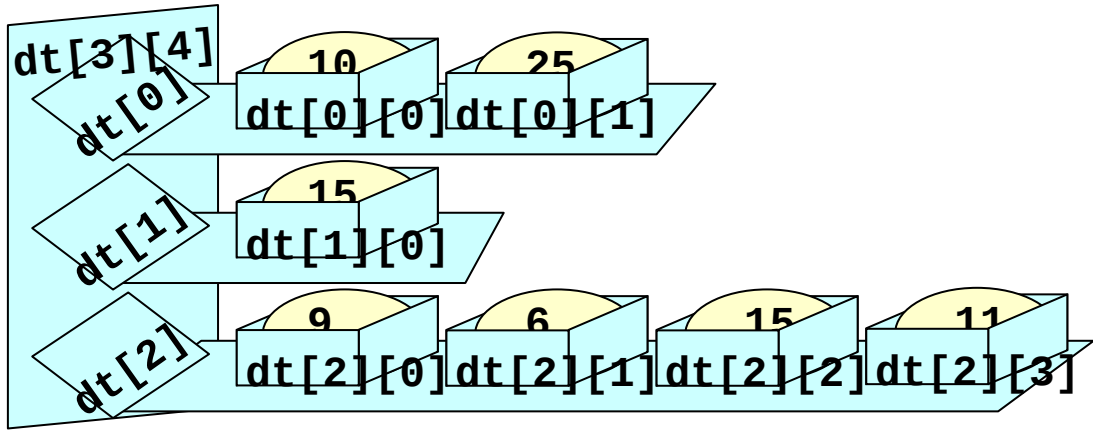
このカンマ(最終カンマ)はあってもなくてもOK。



(11．4) 非矩形の配列

■次のように、2次元目の長さが異なる配列とすることもOKです。

```
int dt[][] = {
    {10,25},
    {15},
    { 9, 6,15,11},
};
```



■初期化しない時は、次のように実体を確保します。

```
int dt[][] = new int[3][];
int dt[0] = new int[2];
int dt[1] = new int[1];
int dt[2] = new int[4];
```

■配列長は、右のように求めます。

```
dt.length  -> これは“3”
dt[0].length -> これは、“2”
dt[1].length -> これは、“1”
dt[2].length -> これは、“4”
```

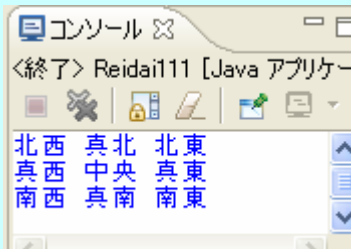
(11.5)例題11.1:二次元配列の打出し

■右の配列を順次打出すプログラムを作成してください。

```
String directions[][]={
   {"北西","真北","北東",},
   {"真西","中央","真東",},
   {"南西","真南","南東",},
};
```

■解答例:

```
public class Reidai111 {
  public static void main(String[] args) {
    String directions[][]={
      {"北西","真北","北東",},
      {"真西","中央","真東",},
      {"南西","真南","南東",},  };
    for(int i=0;i<directions.length;i++){
      for(int j=0;j<directions[i].length;j++){
        System.out.print(directions[i][j]+" ");
      }
      System.out.println("");
    }
  }
}
```



(11.6) 例題11.2:花文字

- “A”、“B”、“C” のいずれかの文字を入力して、入力した文字の花文字を出力するプログラムを作成してください。
- 花文字とは、右図のように文字を連ねて描いた文字のことです。“B”、“C”についても、適当に作ってください。
- 多次元配列を使ってください。

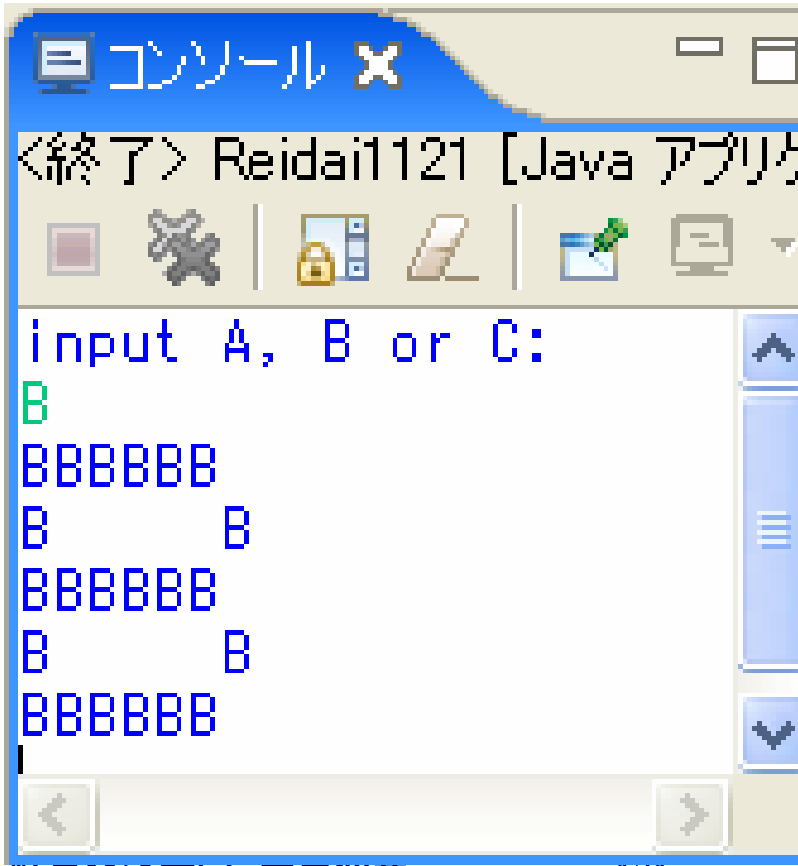
クラス名は、大文字から始めます。

```
$ java Kadai112
input A,B or C :
A
    A
   A A
  AAAAA
A     A
```

プログラムが出力する

入力する

プログラムが出力する



(11.6.1)解答例1(1／2)

```
import java.io.IOException;
public class Reidai1121 {
  public static void main(String[] args)
                                   throws IOException {
    String hana[][]={
      {
       "   A",
       "  A A",
       " AAAAA",
       "A     A",
      },
      {
       "BBBBB",
       "B   B",
       "BBBBB",
       "B   B",
       "BBBBB",
      },
   // C については省略します。
    };
```

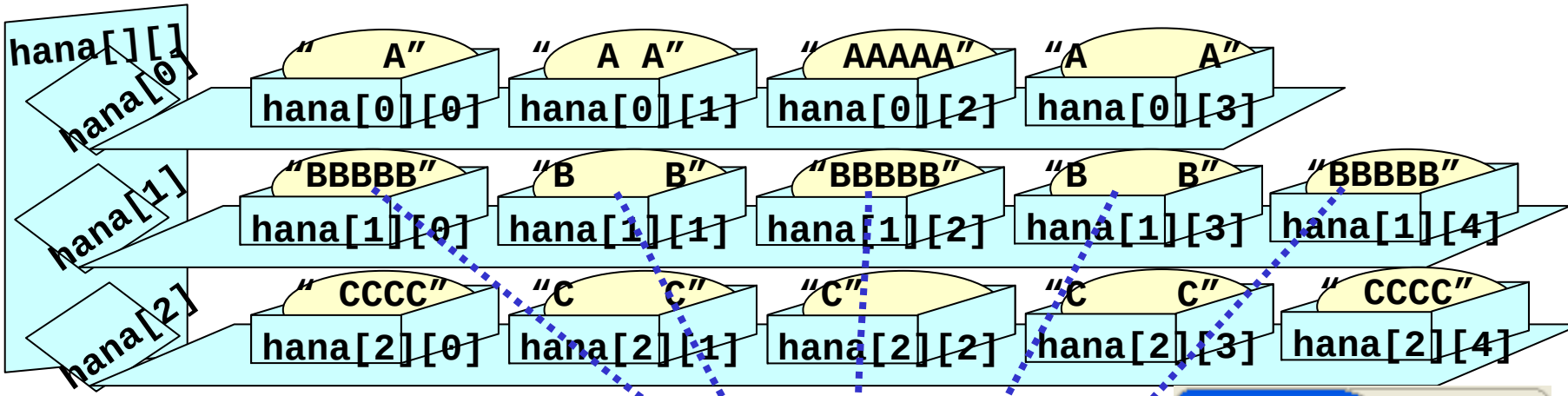
(11.6.2)解答例1(2／2)

```
20:     java.io.BufferedReader kbd
21:     = new java.io.BufferedReader(
22:       new java.io.InputStreamReader(System.in));
23:     System.out.println("input A, B or C:");
24:     String ss = kbd.readLine();
25:     if(ss.equals("A")){
26:       for(int i=0;i<hana[0].length;i++){
27:         System.out.println(hana[0][i]);
28:       }
29:     }else if(ss.equals("B")){
30:       for(int i=0;i<hana[1].length;i++){
31:         System.out.println(hana[1][i]);
32:       }
33:     }else if(ss.equals("C")){
34:       for(int i=0;i<hana[2].length;i++){
35:         System.out.println(hana[2][i]);
36:       }
37:     }else{
38:       System.out.println(“A,B,C以外は無効。");
39:     }
40:   }
41:}
```

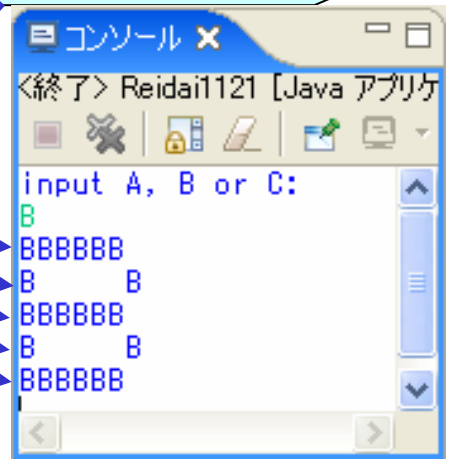
(11．6．3) 解答例1の概要

■Hana[][]は、下図のような配列です。

■入力された文字によって、下の図でのいずれかの1行を打出せば、花文字が出力される訳です。




```
    :
}else if(ss.equals("B")){
  for(int i=0;i<hana[1].length;i++){
    System.out.println(hana[1][i]);
  }
}……
```



(11.6.4)解答例1の問題点

■解答例1のでは、次のような同じような記述が繰り返されています。

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

```
for(int i=0;i<hana[1].length;i++){
  System.out.println(hana[1][i]);
}
```

```
for(int i=0;i<hana[2].length;i++){
  System.out.println(hana[2][i]);
}
```

■同じような記述があること自体無駄でもあり、また、この部分を変更する際には3箇所変更が必要になるなどの問題があります。

■上記の3つの部分の違いは ○ の部分だけなので、これを変数とすれば、1つにまとめることができます。

(11.6.5)解答例2(1／1)

```
1:import java.io.IOException;
2:public class Reidai1121 {
3:  public static void main(String[] args)
4:                                  throws IOException {
5:    String hana[][]={
      // 文字の定義は解答例1と同じです。
6:    };
7:    java.io.BufferedReader kbd
8:    = new java.io.BufferedReader(
9:                  new java.io.InputStreamReader(System.in));
10:   System.out.println("input A, B or C:");
11:   String ss = kbd.readLine();
12:   int ichar=0;
13:   if(ss.equals("A")){
14:     ichar=0;
15:   }else if(ss.equals("B")){
16:     ichar=1;
17:   }else if(ss.equals("C")){
18:     ichar=2;
19:   }else{
20:     System.out.println(“A,B,C以外は無効。");
21:     return;
22:   }
23:   for(int i=0;i<hana[ichar].length;i++){
24:     System.out.println(hana[ichar][i]);
25:   }
26:  }
27:}
```

(11.6.6)解答例2の概要

■どの文字を打出すかを“ichar”という変数に設定しています。

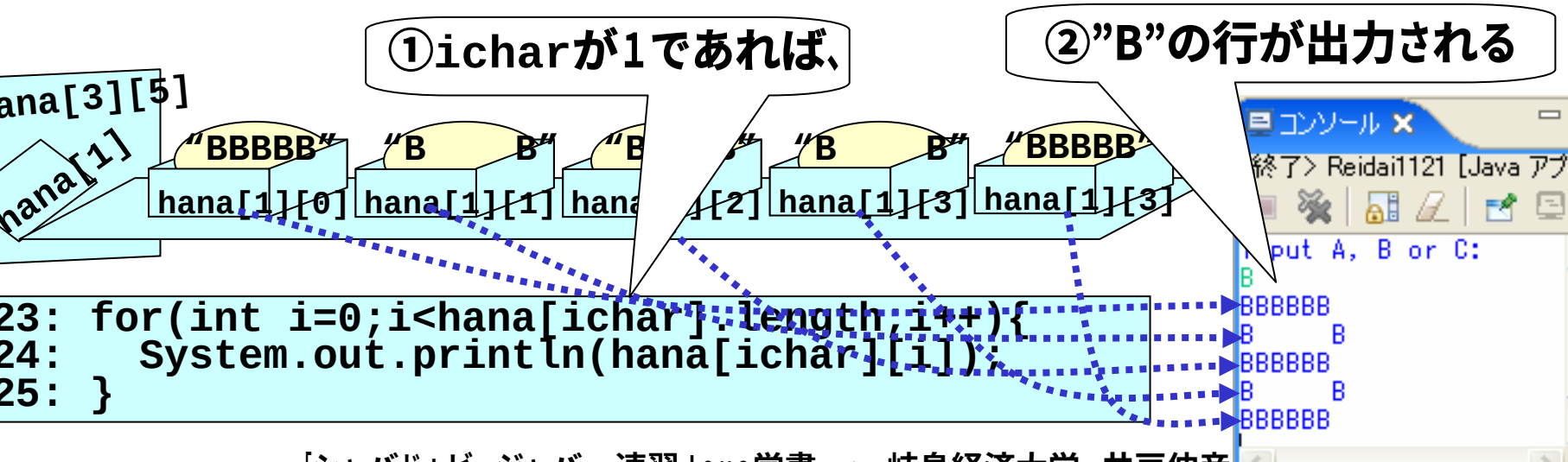
```
12:     int ichar=0;
13:     if(ss.equals("A")){
14:       ichar=0;
15:     }else if(ss.equals("B")){
16:       ichar=1;
17:     }else if(ss.equals("C")){
18:       ichar=2;
19:     }else{
20:       System.out.println(“A,B,C以外は無効。");
21:       return;
22:     }
```

入力が“A”ならば、0行目。

入力が“B”ならば、1行目。

入力が“C”ならば、2行目。

■“ichar”の値により、出力される行が決まります。



(11．6．7) 解答例2の問題点

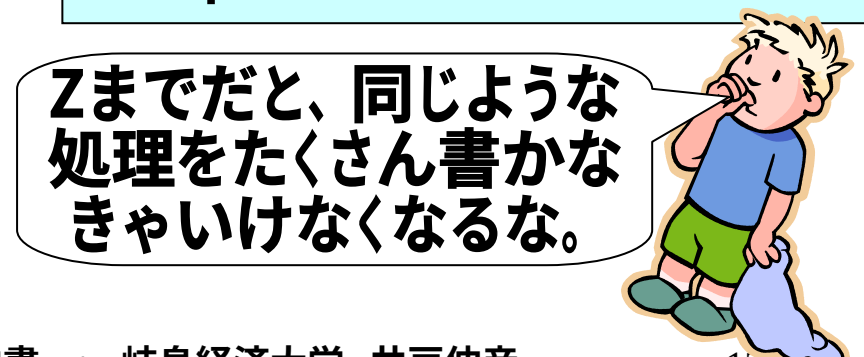
■例題では、AからCまでの3文字としましたが、AからZとした場合には、icharの値を決める(＝どの行を出力するかを決める)ための処理に長い記述が必要になってしまいます。

(AからZになった場合に、行例haraが大きくなることはしかたがありません。)

■これの問題に対応するのが、解答例3になります。

＜出力する行を求める処理＞

```
if(ss.equals("A")){
  ichar=0;
}else if(ss.equals("B")){
  ichar=1;
}else if(ss.equals("C")){
  ichar=2;
}else if(ss.equals(“D")){
  ichar=3;
}else if(ss.equals(“E")){
  ichar=4;
}
      :(中略)
}else if(ss.equals(“Z")){
  ichar=26;
}else{
   :
```



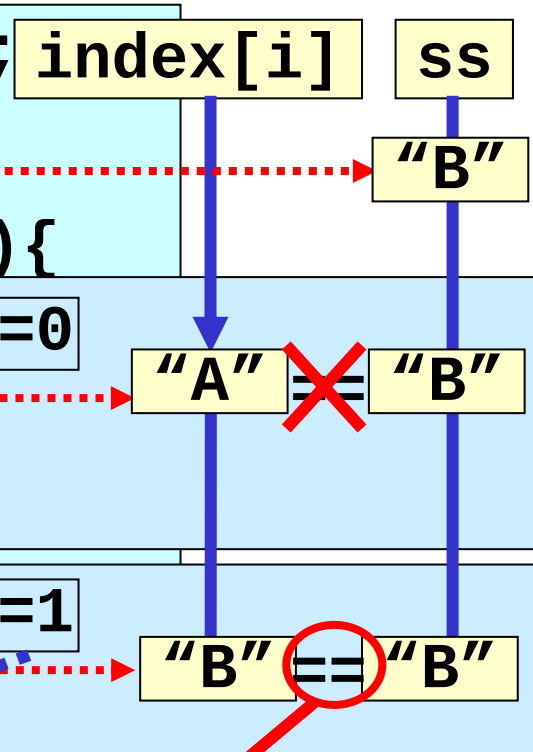
(11.6.8)解答例3(1／1)

```
 1:import java.io.IOException;
 2:public class Reidai1121 {
 3:  public static void main(String[] args)
 4:                                  throws IOException {
 5:    String hana[][]={
       // 文字の定義は解答例1と同じです。
 6:    };
 7:    String[] index={"A","B","C",};
 8:    java.io.BufferedReader kbd
 9:    = new java.io.BufferedReader(
10:      new java.io.InputStreamReader(System.in));
11:    System.out.println("input A, B or C:");
12:    String ss = kbd.readLine();
13:    int ichar=-1;
14:    for(int i=0;i<hana.length;i++){
15:      if(ss.equals(index[i])){
16:        ichar=i;
17:        break;
18:      }
19:    }
20:    if(ichar==-1){
21:      System.out.println(“A,B,C以外は無効。");
22:      return;
23:    }
24:    for(int i=0;i<hana[ichar].length;i++){
25:      System.out.println(hana[ichar][i]);
26:    }
27:  }
28:}
```

(11.6.9) 解答例3の概要

■解答例3では、次のようにicharの値（＝hanaの何行目を出力するか）を決めています（入力が“B”の場合）。

```
 7: String[] index={"A","B","C",};
12: String ss = kbd.readLine();

14: for(int i=0;i<hana.length;i++){
    15: if(ss.equals(index[i])){
    16:    ichar=i;
    17:    break;
    18: }

    15: if(ss.equals(index[i])){

    16:    ichar=i;
    17:    break;
    18: }
19: }
24:   //ichar=1となっている
```



（11．7．1）課題

■課題11－1

- 右のように初期化された配列のそれぞれの値の合計を求めるプログラムを作成してください。

```
int dt[][] = {
        {10,25, 5, 5},
        {15, 5,12, 8},
        { 9, 6,15,11},
};
```

■課題11－2

- 例題11．2の解答例1，2，3について、すべて作ってみてください。その際、次の点を確かめてください。
  - ◆解答例2が解答例1よりも優れている点は？
  - ◆解答例3が解答例2よりも優れている点は？

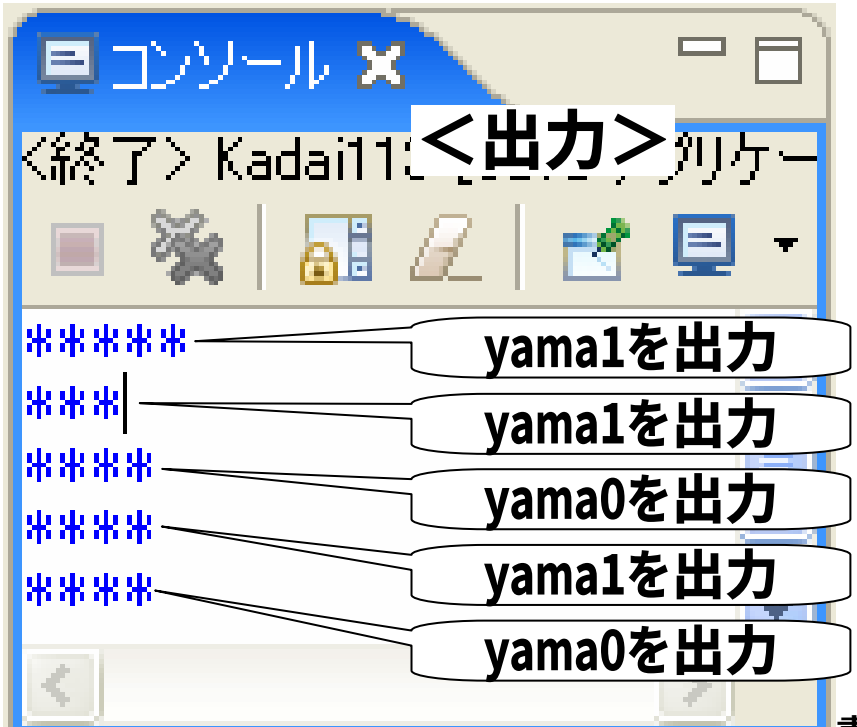
(11．7．2) 課題

■課題11－3

- 右のような配列“yama0”、“yama1”を宣言し、それぞれ対応する2次要素のうち長いほうを出力するプログラムを作成してください。

```
char yama0[][]={
  {'*','*'},
  {'*','*'},
  {'*','*','*','*'},
  {'*','*','*'},
  {'*','*','*','*'},
 };
char yama1[][]={
  {'*','*','*','*','*'},
  {'*','*','*'},
  {'*','*'},
  {'*','*','*','*'},
  {'*','*'},
 };
```

長さは2
長さは2
長さは4
長さは3
長さは4
長さは5
長さは3
長さは2
長さは4
長さは2



(12) メソッド

■ここでは、VBのプロシージャや、Cの関数と同じ意味でのメソッドの使い方を考えます。

■メソッド

- メソッドは、プログラムの実行文の最小単位です。どのような実行文も、メソッドの中にあります。
- ここまでのプログラムは、すべて、“public static void main(String args[])”という特別なメソッドの中に作ってきました。

クラス

メソッド

実行文

```
public class Hello{
    public static void main(String args[]){
        System.out.println("Hello, world.");
    }
}
```

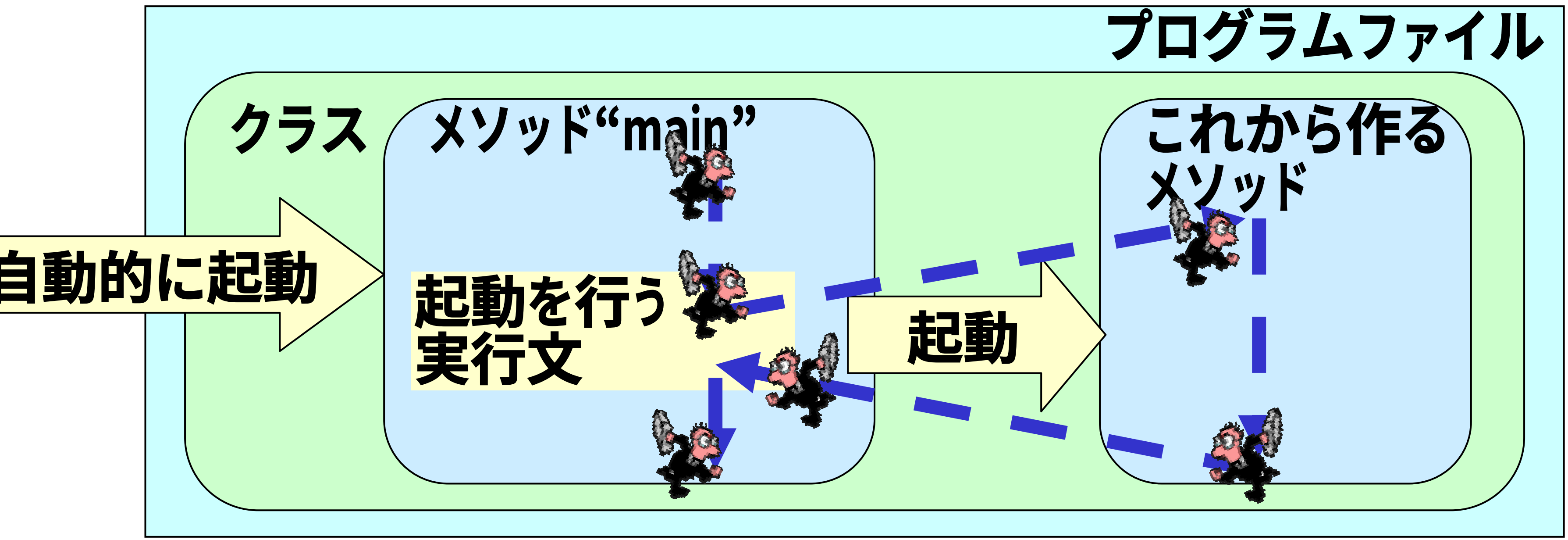
(12.1) メソッドを作る

■メソッド“main”は、次の意味で特別なメソッドです。

- プログラムを実行されたときに、自動的に起動される。

■これから作るメソッドは、実行文として起動することを記述することにより、実行されます。

■メソッドが起動され、その中の実行文が実行し終わると、起動元のメソッドに戻って処理が続けられます。



(12.2) メソッドの例

■次のようなメソッド“repeatStr”を作り、実行してみます。

＜プログラム＞

＜出力＞

クラス

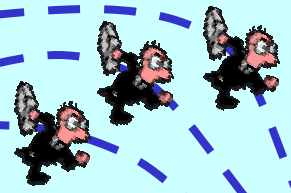
メソッド

```
public class PrintLines {
  public static void main(String[] args) {
    repeatStr("xy",3);
    repeatStr("yz",2);
    repeatStr("z ",3);
  }
```

メソッド

```
  public static void repeatStr(String s, int n){
    for(int i=0;i<n;i++){
      System.out.print(s);
    }
    System.out.println("");
  }
}
```

```
xyxyxy

yzyz

z z z
```



(12.3) メソッドの形式

```
// メソッドの呼び出し
repeatStr("xy",3);
```

メソッド名

実引数
(12.4)参照

```
// メソッド(呼び出される側)
public static void repeatStr(String s, int n){
        // 処理の記述
}
```

メソッド名

仮引数
(12.4)参照

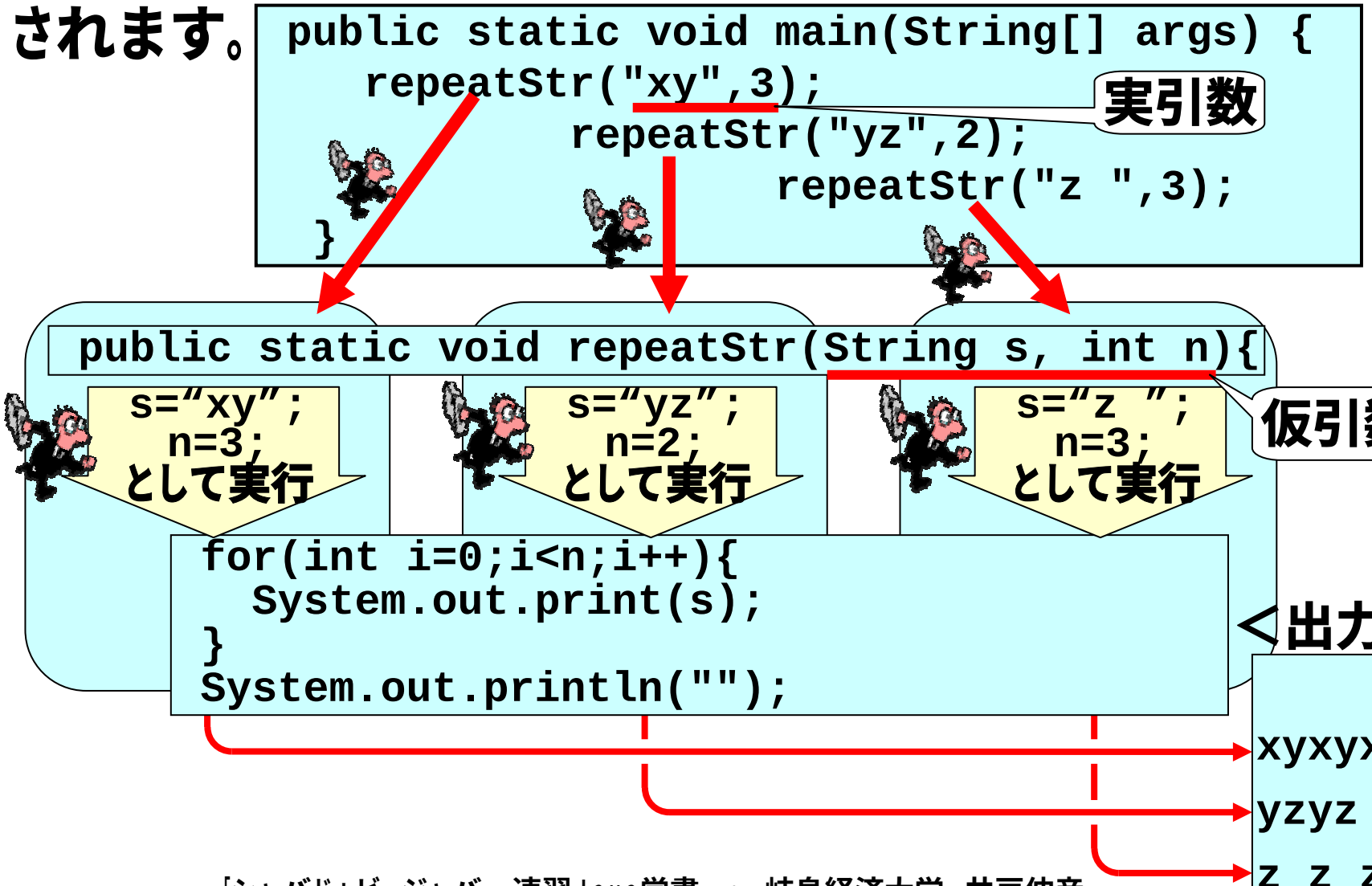
[修飾子]
“public”
→誰からでも呼び出せる
“static”
→クラスメソッドである

戻り値の型
“void”
→戻り値はない
(12.5.1)参照

これらについて、本スライドでは説明しません。

(12.4) 引数

■メソッドでの仮引数は、呼び元での実引数の値に設定されます。



(12．5．1) 戻り値

■次のプログラムにて、戻り値について考えます。

```
public class FactorEx {

  public static void main(String[] args) {
    int x = calFactor(4);

    System.out.println(x);
    x = calFactor(5);
    System.out.println(x);
  }

  public static int calFactor(int a){
    int fac = 1;
    for(int i=1;i<=a;i++){
      fac *= i;
    }
    return fac;
  }
}
```

戻り値

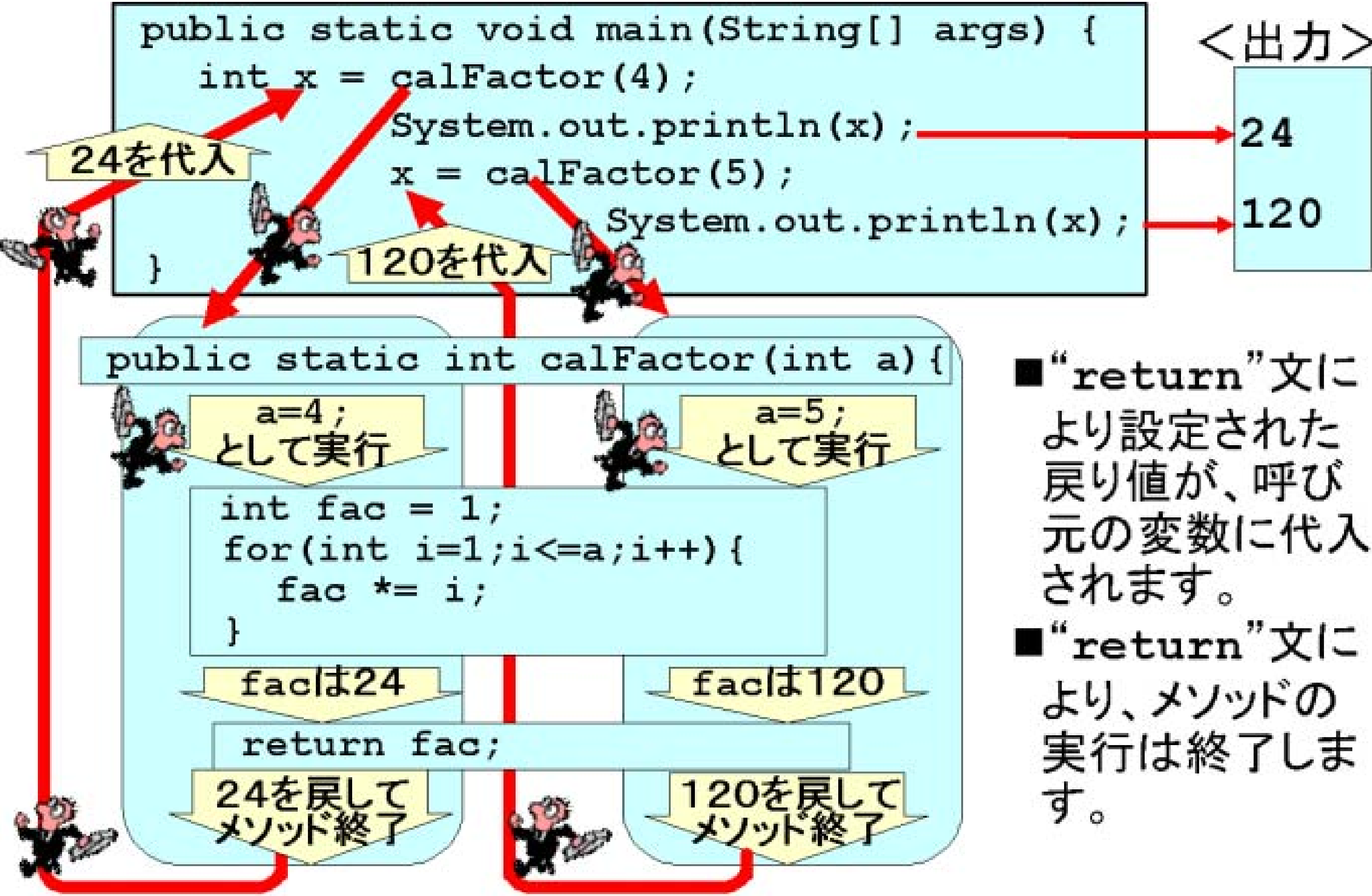
戻り値の型“int”

リターン文

＜出力＞

```
24
120
```

(12．5．2) 戻り値



- ■“return”文により設定された戻り値が、呼び元の変数に代入されます。
- ■“return”文により、メソッドの実行は終了します。

(12.5.3) 戻り値がない場合

■スライド(12.2)のプログラムで見たように、戻り値がないメソッドの、戻り値の型は、“void”となります。

■戻り値のないメソッドでの“return”文では、戻り値は返さず、単にメソッドを終了することになります。

戻り値がないことを示す

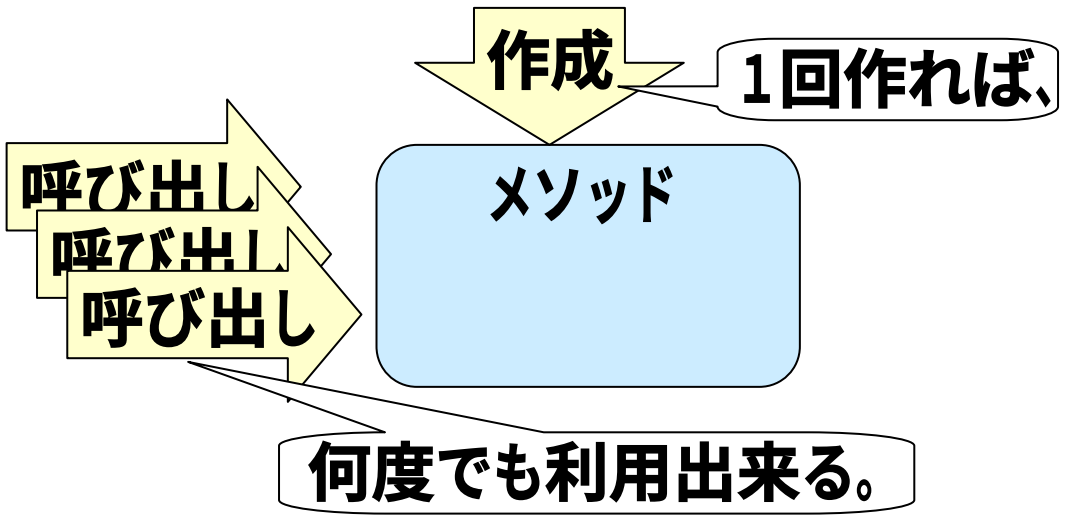
```
public static void repeatStr(String s, int n){
        // 処理の記述
        return;
}
```

戻り値は指定せず、
単にメソッドを終了する

(12.6) どうしてメソッドが必要か？

- スライド(12.2) のプログラムを、メソッドを使わずに作成すると、右下のようなプログラムになります。
- このプログラムでは、同じような実行文が繰り返し書かれており、これは無駄であるとともに、プログラムも見通しの悪いものになっています。
- メソッドを用いることで、効率的で見通しの良いプログラムを作成することが出来ます。

```
public class PrintLines {
  public static void main(String[] args) {
    for(int i=0;i<3;i++){
      System.out.print(“xy”);
    }
    System.out.println("");
    for(int i=0;i<2;i++){
      System.out.print(“yz”);
    }
    System.out.println("");
    for(int i=0;i<3;i++){
      System.out.print(“z ”);
    }
    System.out.println("");
  }
}
```

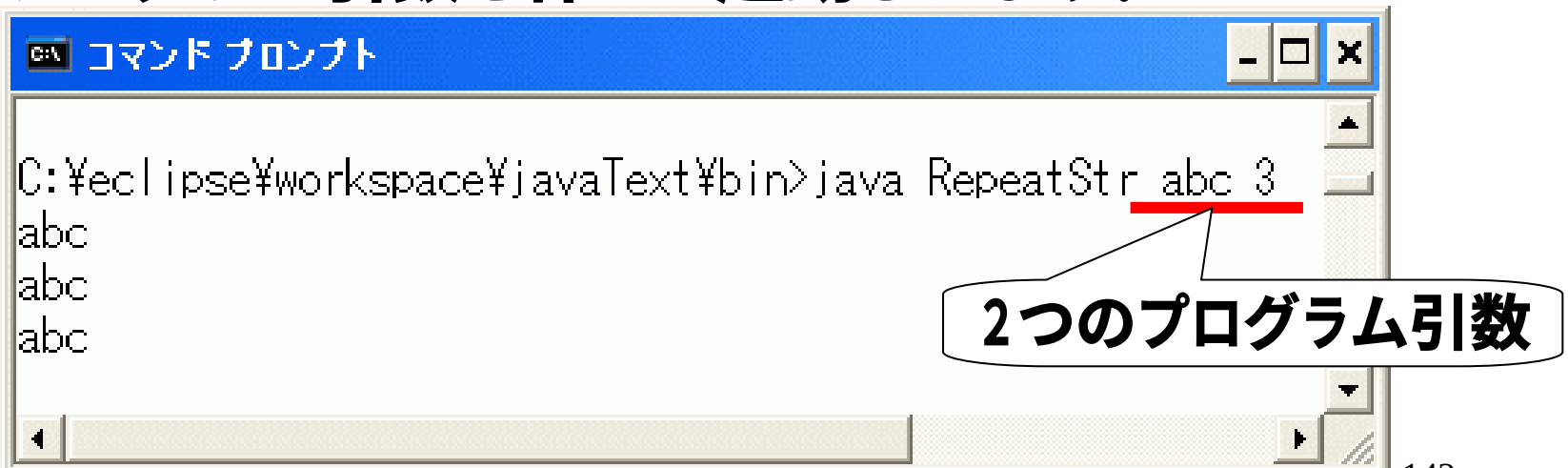


(12．7) プログラム引数

■右の簡単なプログラムを考えます。

```
public class RepeatStr {
  public static void main(String[] args){
    String str = args[0];
    int n = Integer.parseInt(args[1]);
    for(int i=0;i<n;i++){
      System.out.println(str);
    }
  }
}
```

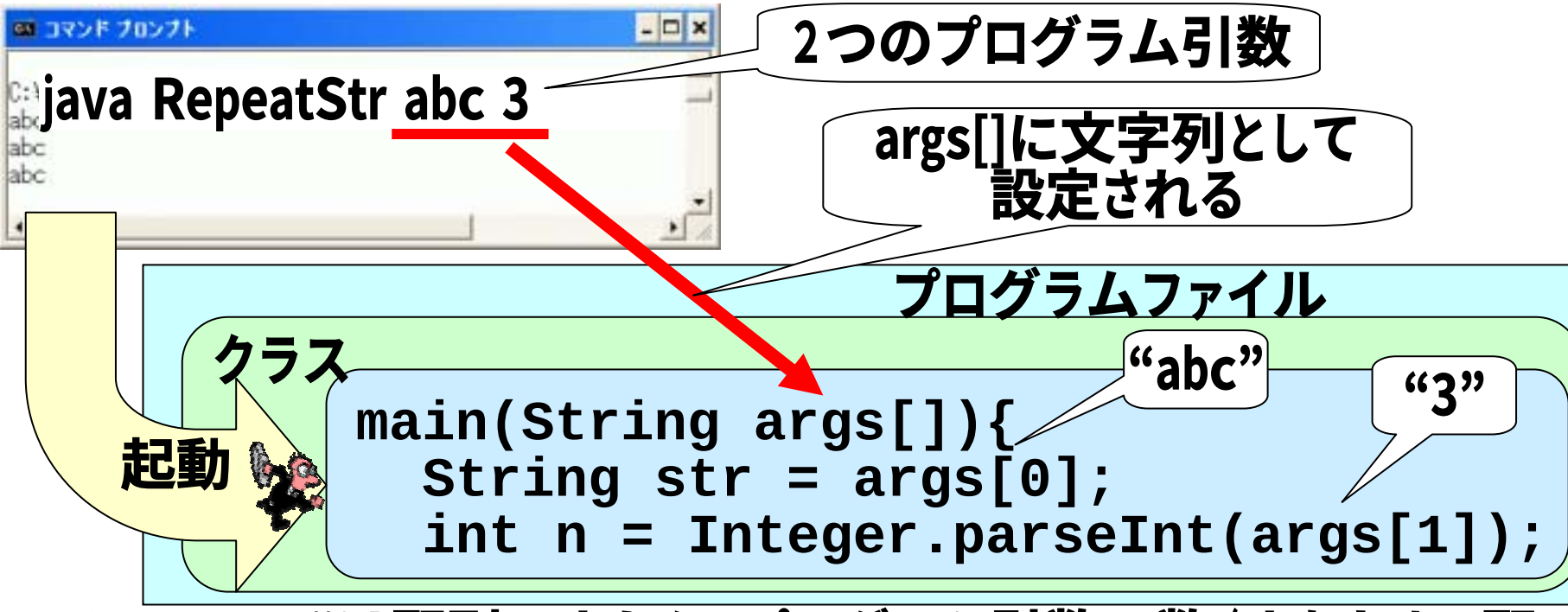
■このプログラムは、次のようにコマンドライン上で2つのプログラム引数を伴って起動されます。



(12．7．1) args[]

■起動時に指定されたプログラム引数は、mainメソッドの引数“args[]”に設定されて、mainプログラムに渡されます。
“args[]”は、文字列の配列です。

■プログラムでは、この値を見て処理を行っています。



■“args[]”は配列ですから、プログラム引数の数（すなわち、配列の長さ）は、“args.length”で調べることが出来ます。

(12．7．2) eclipseでのプログラム引数

■eclipseでのプログラム引数の与え方は、次のスライドに説明があります。

- 月に吠える　－eclipseによるJavaアプリケーション作成－
  (http://www.gifu-keizai.ac.jp/~ido/doc/java/eclipse_application.pdf)
  スライド (5.2.2) 引数

■すなわち、スライド(12.7)のプログラム引数は、次のように設定します。

- 「実行」ウインドウ中、
  [引き数]タグをクリック (①)。
- [プログラム引き数]欄に、
  “abc 3”と入力 (②)。

(12.8) 例題12.1：アスタリスクとシャープ

<出力>

```
#####
*####
**###
***##
****#
*****
```

■右のような出力を行うプログラムを作成してください。但し、次のようなメソッドを使うこととします。

```
pulic static void drawSyms(int w, String s)
```

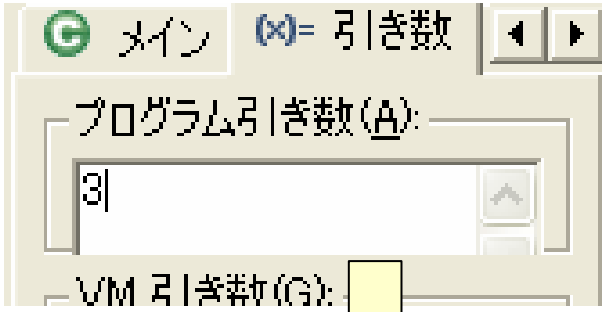
- ◆このメソッドは文字列sを、w回だけ出力する。
- ◆例えば、sが”#”、wが5のとき、“#####”を出力する。

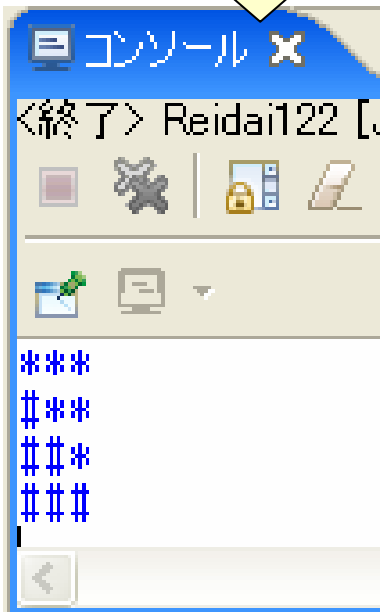
■解答例：

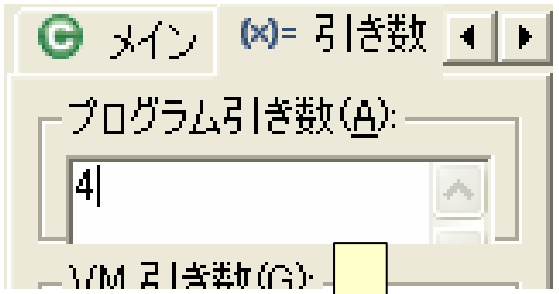
```
public class Reidai121 {
  public static void main(String[] args) {
    for(int i=0;i<6;i++){
      drawSyms(i,"#");
      drawSyms(5-i,"*");
      System.out.println("");
    }
  }
  private static void drawSyms(int w,String s){
    for(int i=0;i<w;i++){
      System.out.print(s);
    }
  }
}
```

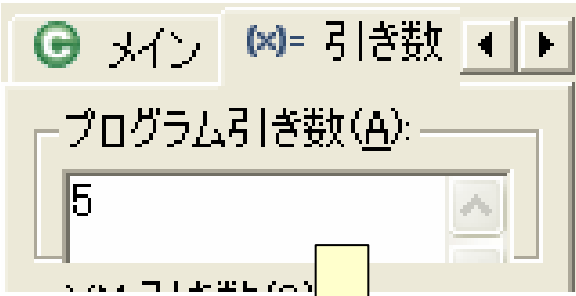
(12．9．1) 例題12．2：サイズを変える

■プログラム引数で前の例題にて、描いた文字による図形のサイズを変更できるようにしてください。









(12.9.2)例題12.2:サイズを変える

■解答例:

```
public class Reidai121 {
  public static void main(String[] args) {
    int d = Integer.parseInt(args[0]);
    for(int i=0;i<d+1;i++){
      drawSyms(i,"#");
      drawSyms(d-i,"*");
      System.out.println("");
    }
  }
  private static void drawSyms(int w,String s){
    for(int i=0;i<w;i++){
      System.out.print(s);
    }
  }
}
```

（１２．１０．１）課題

■課題１２－１

- スライド(12.2)、(12.5.1)のプログラムを作成し、実行してみてください。

■課題１２－２

- 右のような出力を行うプログラムを作成してください。
- 例題１２－１のdrawSymsメソッドを使うこととします。

```
****#****
***###***
**#####**
*#######*
**#####**
***###***
****#****
```

(12.10.2) 課題(続き)

■課題12－3

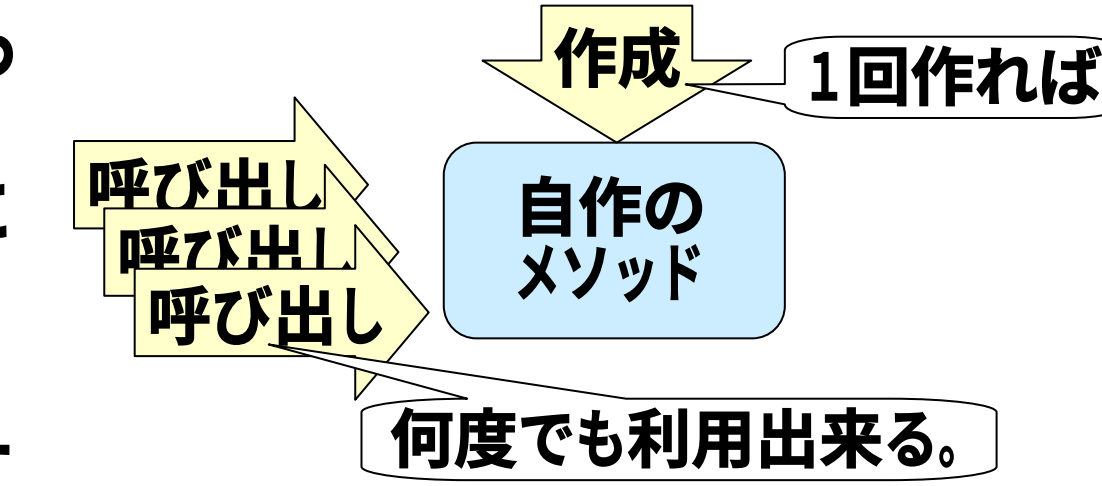
- 文字を2つ入力して、下図のような出力を得るプログラムを作成してください。

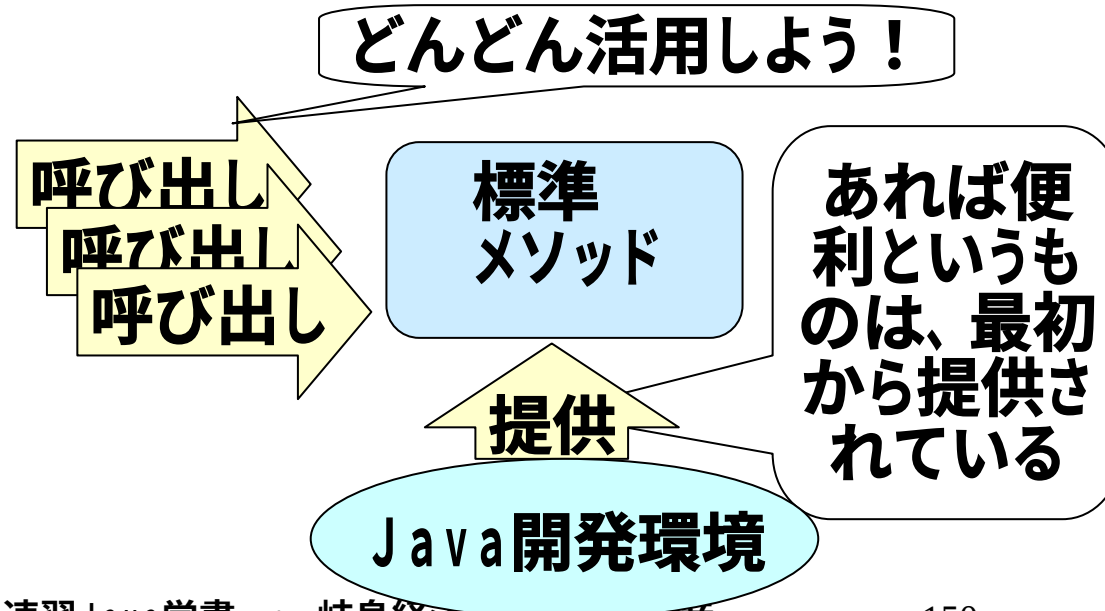
```
$ java Kadai123 & =
====&====
===&&&===
==&&&&&==
=&&&&&&&=
==&&&&&==
===&&&===
====&====
```

プログラム引数として、2つのシンボルを与える

(13) 標準のメソッド

- メソッドは、一度作ってしまえば何度でも利用出きる点で役に立ちます。
- 標準のメソッドは、Ｊａｖａの開発環境にすでに用意されている、“あれば便利な”メソッドのことです。
- 以下、代表的な標準のメソッドの使い方を説明していきます。

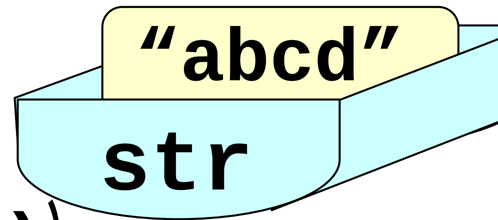




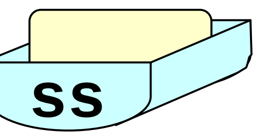
（１３．１）文字列処理メソッド

■文字列を扱うStringクラスについては、既に“equals(String)”を学んでいます。（スライド(8.3)）

文字列型変数

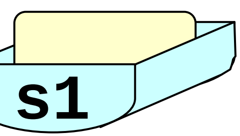


■文字列の等価判定（equals(String)）

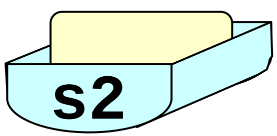


が“abc”と同じか？ ⇒

```
if(ss.equals(“abc”)){
```



が



と同じか？ ⇒

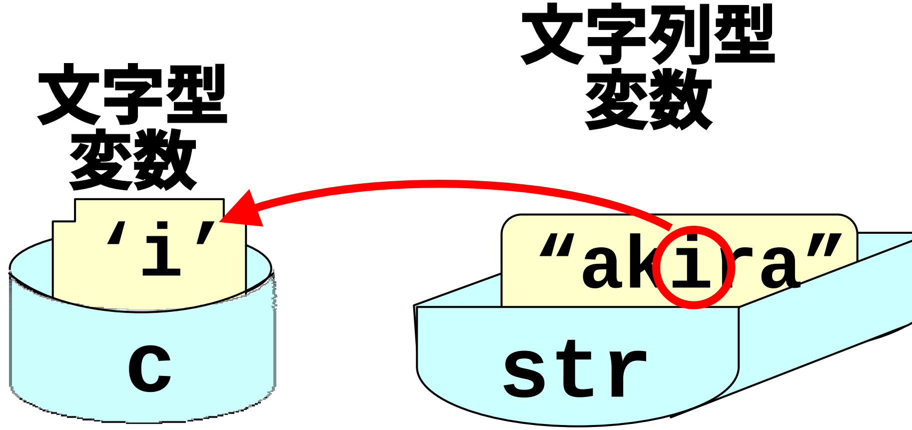
```
if(s1.equals(s2)){
```

（ `if(s2.equals(s1)){` でも同じ）

(13.1.1) 文字の取得、文字列長の取得

■文字の取得(charAt())

```
String str = “akira”;
char c = str.charAt(2);
// cには、’i’が入る
```

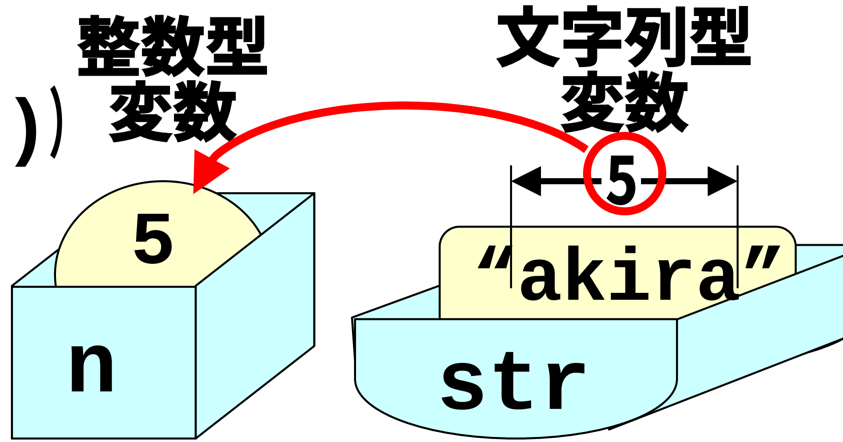


- 文字の数える数は、先頭が0になることに注意！
- 他のメソッドでも、文字は0から数えます。

0	1	2	3	4
a	k	i	r	a

■文字列長の取得(length())

```
String str = “akira”;
int n = str.length();
// nには、数値の5が入る。
```

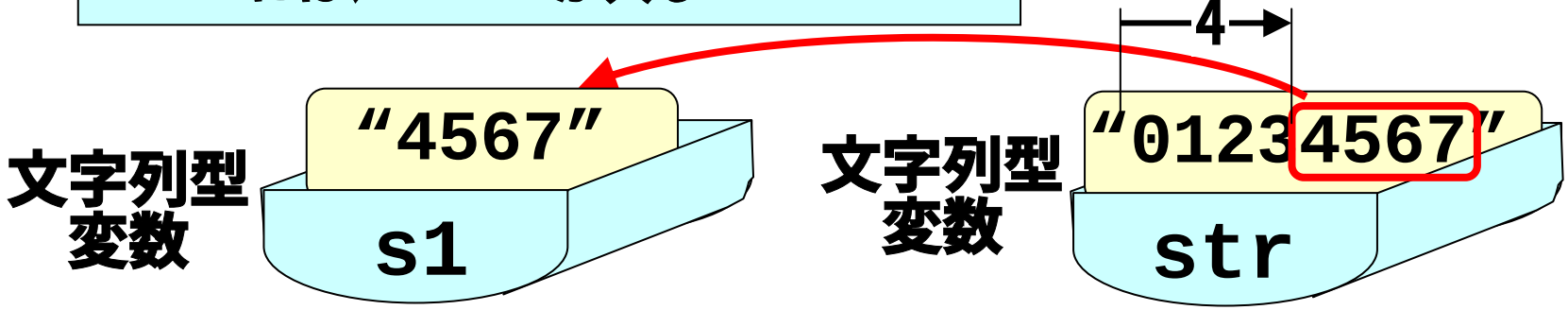


(13.1.2) 部分文字列の取得

■末尾までの取得(substring(int))

```
String str = “01234567”;
String s1 = str.substring(4);
// s1には、“4567”が入る
```

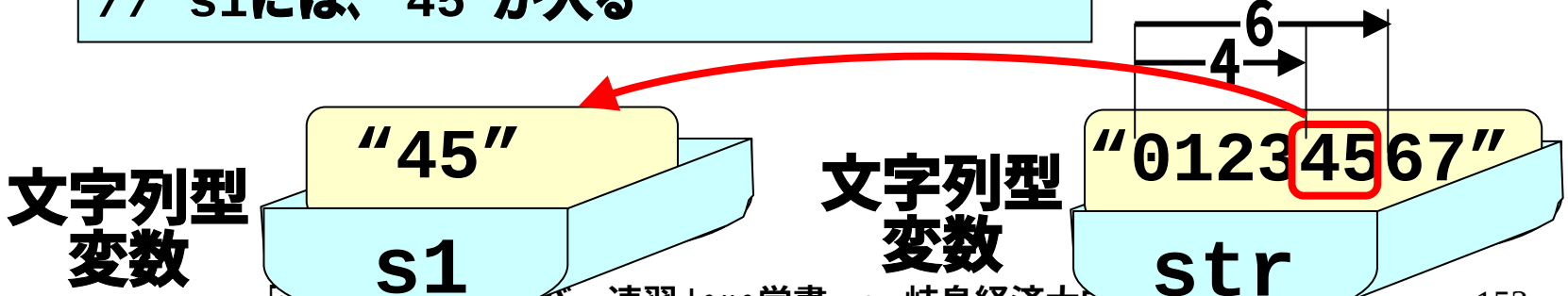
位置4から
最後まで



■指定位置までの取得(substring(int,int))

```
String str = “01234567”;
String s1 = str.substring(4,6);
// s1には、“45”が入る
```

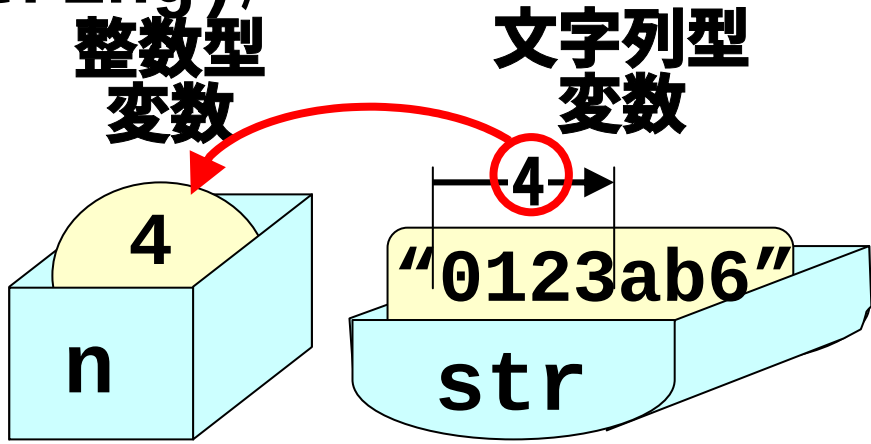
位置4から
位置5(6の前)まで



(13.1.3)文字列検索、文字置換

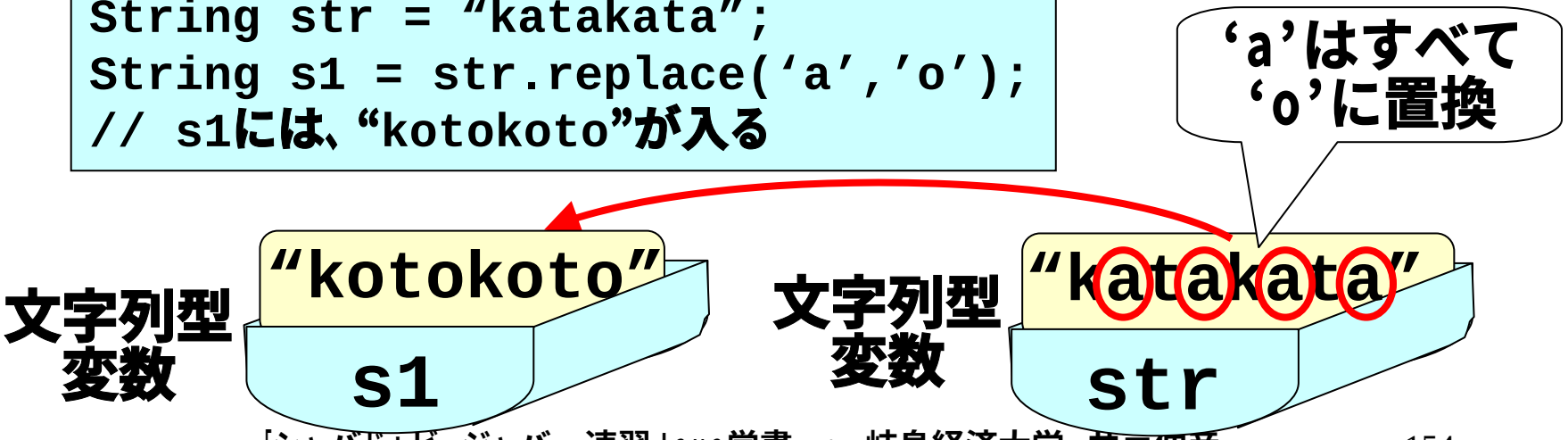
■文字列検索(indexOf(String))

```
String str = “0123ab6”;
int n = str.indexOf(“ab”);
// nには、数値の4が入る。
int n = str.indexOf(“cd”);
// nには、数値の-1が入る。
```



■文字置換(replace(char,char))

```
String str = “katakata”;
String s1 = str.replace(‘a’,’o’);
// s1には、“kotokoto”が入る
```



（13．1．4）その他のメソッド

■その他にも、Stringクラスには豊富なメソッドが揃っています。「ありそうなものは全てある」と思ってください。

■ですから、「ありそうなもの」を自分で作ることは馬鹿馬鹿しいので避けてください。

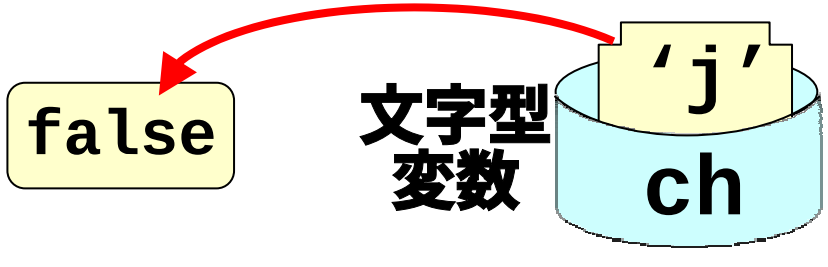
■どのようなメソッドがあるかについては、Web利用すれば、メソッドの使い方を含めて楽に調べることができます（これはStringクラス以外のクラスについても同じです）。

■例えば、googleにて“Java”、“String”で検索してみてください。

(13．2) 文字処理メソッド

■文字の種類の判定

```
if(Character.isDigit(ch)){   //chが数字ならば真
```

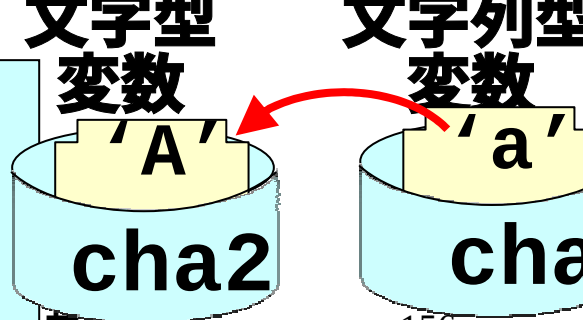


- 他にもいろいろな判定がある。

```
if(Character.isLetter(ch)){   ...    //chが英字ならば真
if(Character.isUpperCase(ch)){ ... //chが大文字ならば真
if(Character.isWhilespace(ch)){... //chが空白系文字ならば真
```

■文字の大小変換

```
char cha = 'a'; char chb = 'B';
char cha2 = Character.toUpperCase(cha);
char chb2 = Character.toLowerCase(chb);
// cha2には'A'が入る、chb2には'b'が入る。
```



（１３．３．１）数値処理メソッド

■数値処理メソッド以外のものを含めて、文字列／文字／数値の変換について見て行きます。

■文字列から数値への変換は、既にスライド(6.5)で見ました。

- 整数値への変換

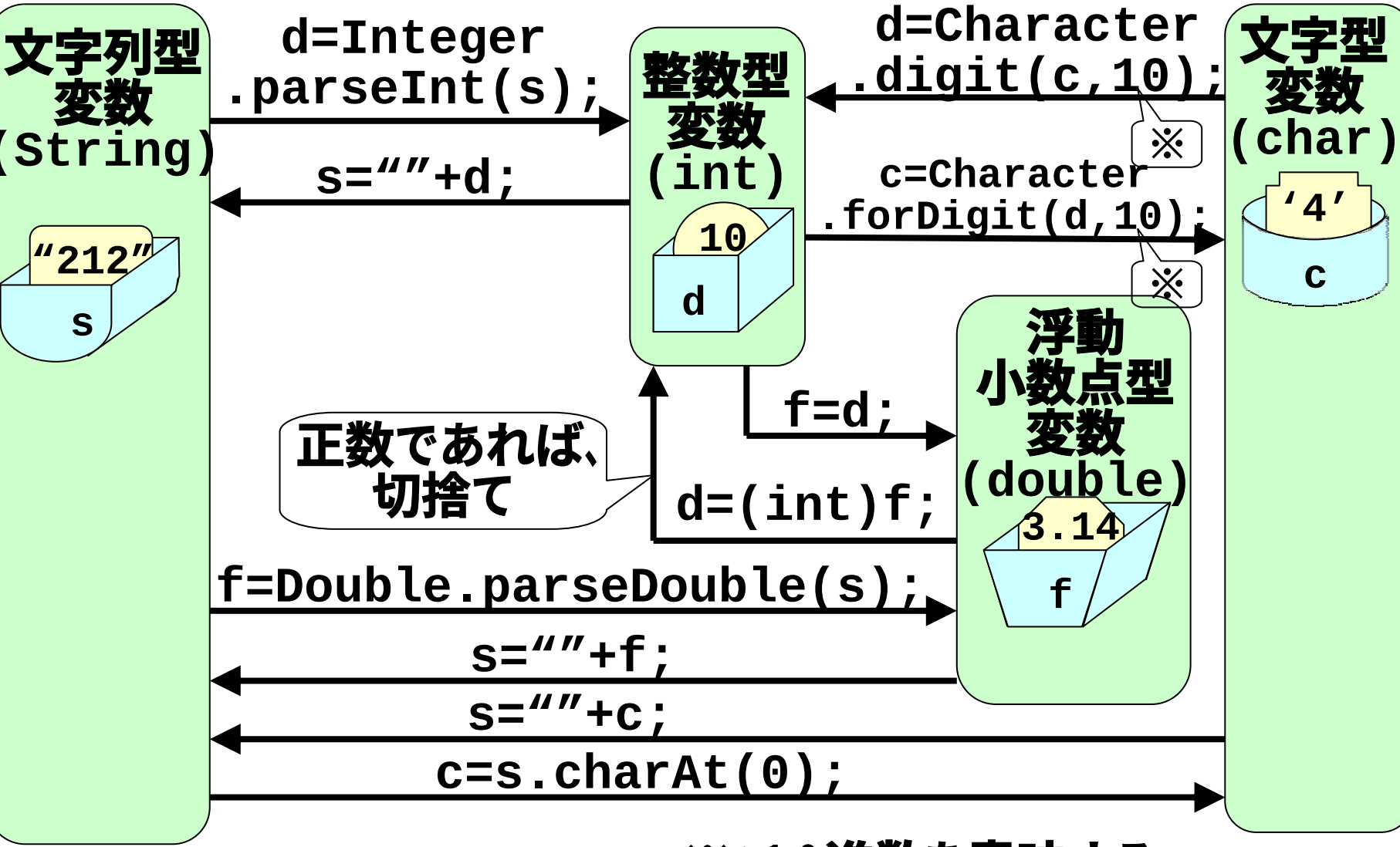
```
d = Integer.parseInt(ss);
```

- 浮動小数点数値への変換

```
f = Double.parseDouble(ss);
```

■次スライドに、主な変換を示します。

(13.3.2) 主な変換



※：10進数を意味する

(13.4.1) 課題

■課題13－1

- 英数字の文字列を入力して、小文字母音("a","i","u","e","o")がそれぞれいくつ含まれるかを出力するプログラムを作ってください。母音は配列に入れてください。

```
$ java Kadai131
input string :
I refused him absolutely.
a:1, i:0, u:2, e:3, o:1
```

■課題13－2

- 上記のプログラムで、母音の大文字("A","I","U","E","O")と小文字("a","i","u","e","o")とを区別しないで数えるようにしてください。

（１３．４．２）課題

■課題１３－３

- 課題１３－１を、次のような配列を使って作成してください。
（すでに使っている場合は、この課題は実施する必要はありません。）

```
char vowel[] = {‘a’,’i’,’u’,’e’,’o’};
int counter[] = {0,0,0,0,0};
```

■課題１３－４

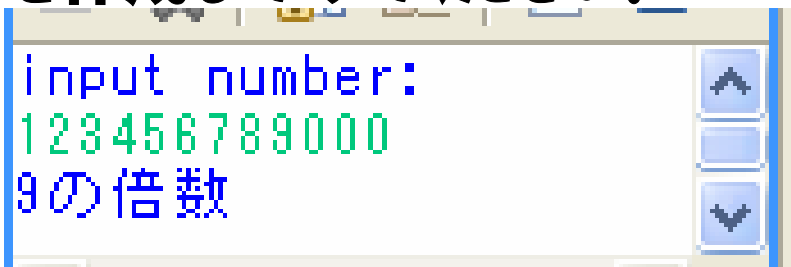
- 文字列を入力して、数字だけを出力するプログラムを作ってください。

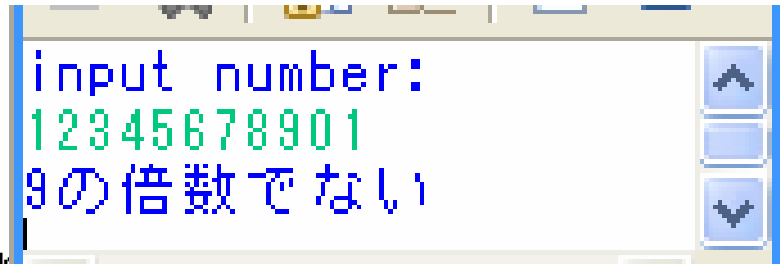
```
$ java Kadai134
input string :
23fjaksi55f 8989da
23558989
```

(13．4．3) 課題

■課題13－5

- 数字による長い文字列を入力して、それが9の倍数であるか否かを出力するプログラムを作成してください。但し、9の倍数であることの判定は、次のように行うこととします。すなわち、% (あまり) は使わないこととします。
  - ◆423711は、9の倍数である。
    - □4＋2＋3＋7＋1＋1＝18
    - □1＋8＝9　⇒　9となれば、9の倍数
  - ◆38572は、9の倍数でない。
    - □3＋8＋5＋7＋2＝25
    - □2＋5＝7　⇒　9未満となれば、9の倍数ではない。
- “d=Integer.parseInt(s);”を用いると、大きな桁数の入力についてエラーとなってしまいます。そうならないようなプログラムを作成してみてください。





(13．4．4) 課題

■課題：暗号解読

- 次のように、アルファベットの順番をずらして得られる変換による暗号を、シーザー暗号といいます。
- 文字列と数字nとを入力して、文字列のうちの小文字のアルファベットをn個ずらして得られるシーザー暗号を出力するプログラムを作成してください (小文字のアルファベット以外の入力はそのまま出力します)。

```
暗号を入力してください:
hal
何文字ずらすかを入力してください:
1
ibm
```

```
暗号を入力してください:
d rdnc d rzmz v wdmy.
何文字ずらすかを入力してください:
5
i wish i were a bird.
```

(13．4．5) 課題

■課題13－6

- 次のような考えで、暗号解読のプログラムを作ってください。
  - ◆入力した文字列の最初の文字から、最後の文字まで、一文字づつ処理する。

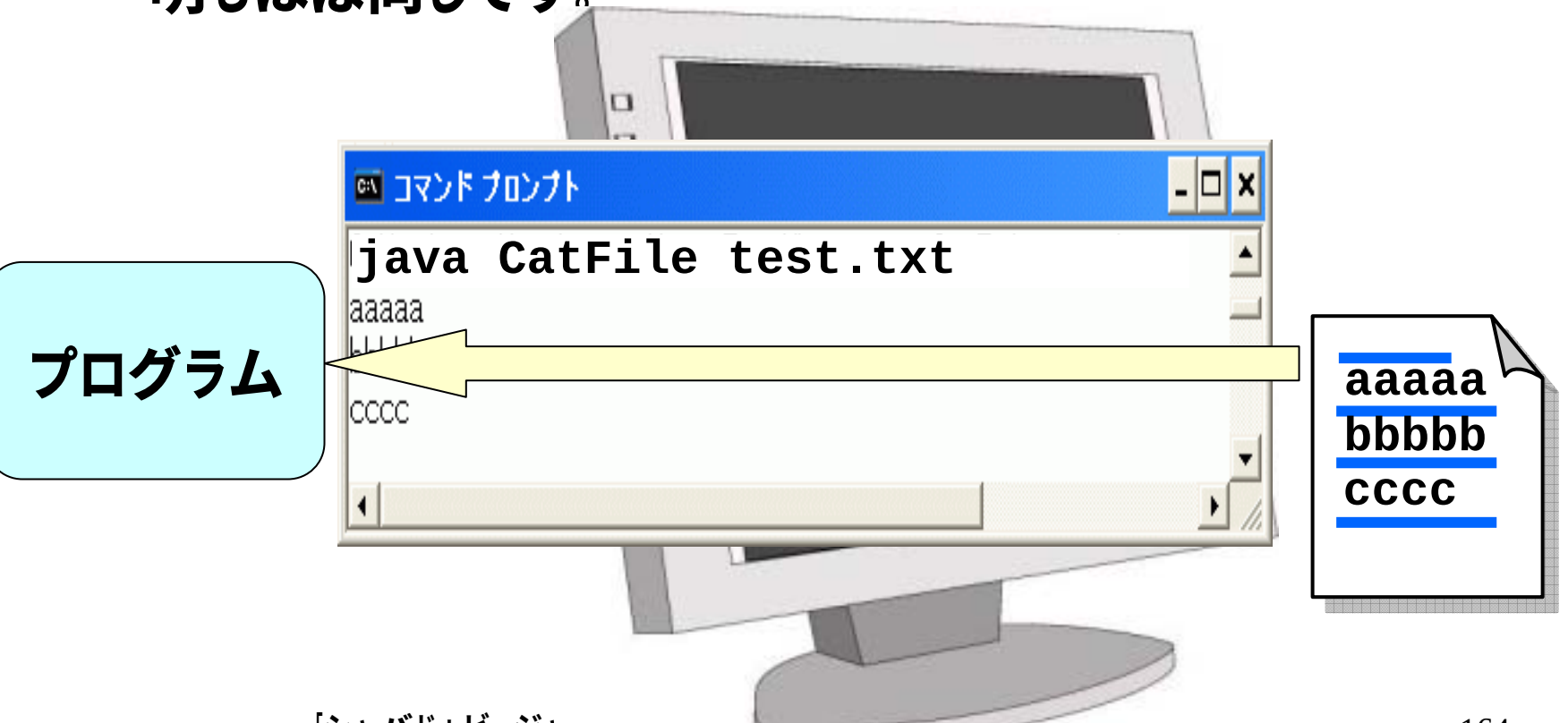
■課題13－7

- 次のような考えで、暗号解読のプログラムを作ると、うまく行かないことを確認してください。
  - ◆入力文字列中の文字‘a’を、n個ずらした文字に置き換える。
  - ◆入力文字列中の文字‘b’を、n個ずらした文字に置き換える。
  - ◆以下、同様に‘z’までの文字について実行する。
  - ◆置き換えた文字列を出力する。

(14) ファイルからの入力

■何をしたいのか？

- テキスト･ファイルから、1行ずつ読み込む処理を作成します。
- 読み込んだ各行を、コマンドラインに出力します。
- スライド(6)「データの入力」と似たようなプログラムになり、説明もほぼ同じです。



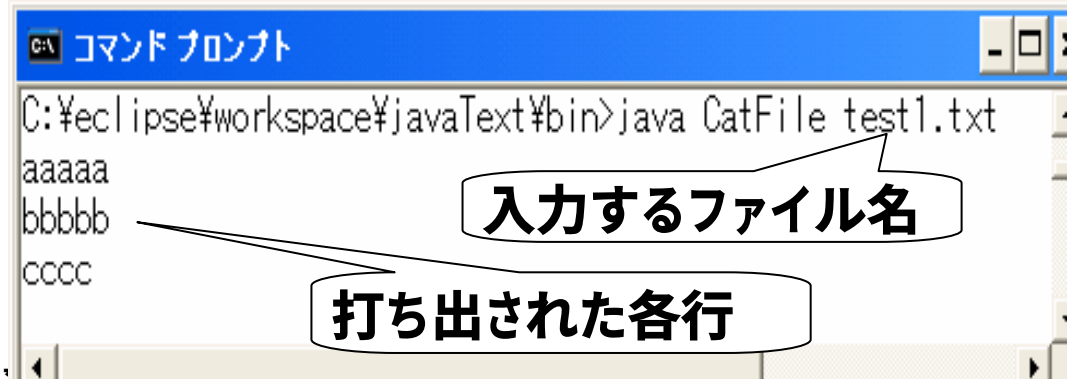
(14．1) ファイル入力を行うプログラム

■次のようなプログラムを考えます。

```
import java.io.*;
public class CatFile {
  public static void main(String[] args) throws IOException {
    String fileName = args[0];
    String ss;
    BufferedReader fin = new BufferedReader(
                                        new FileReader(fileName));
    while((ss = fin.readLine())!=null){
      System.out.println(ss);
    }
    fin.close();
  }
}
```

プログラム引数を用いています
(12.7.1)参照

■このプログラムは、
ファイルより入力した
文字列1行を
1行ずつ打ち出します。

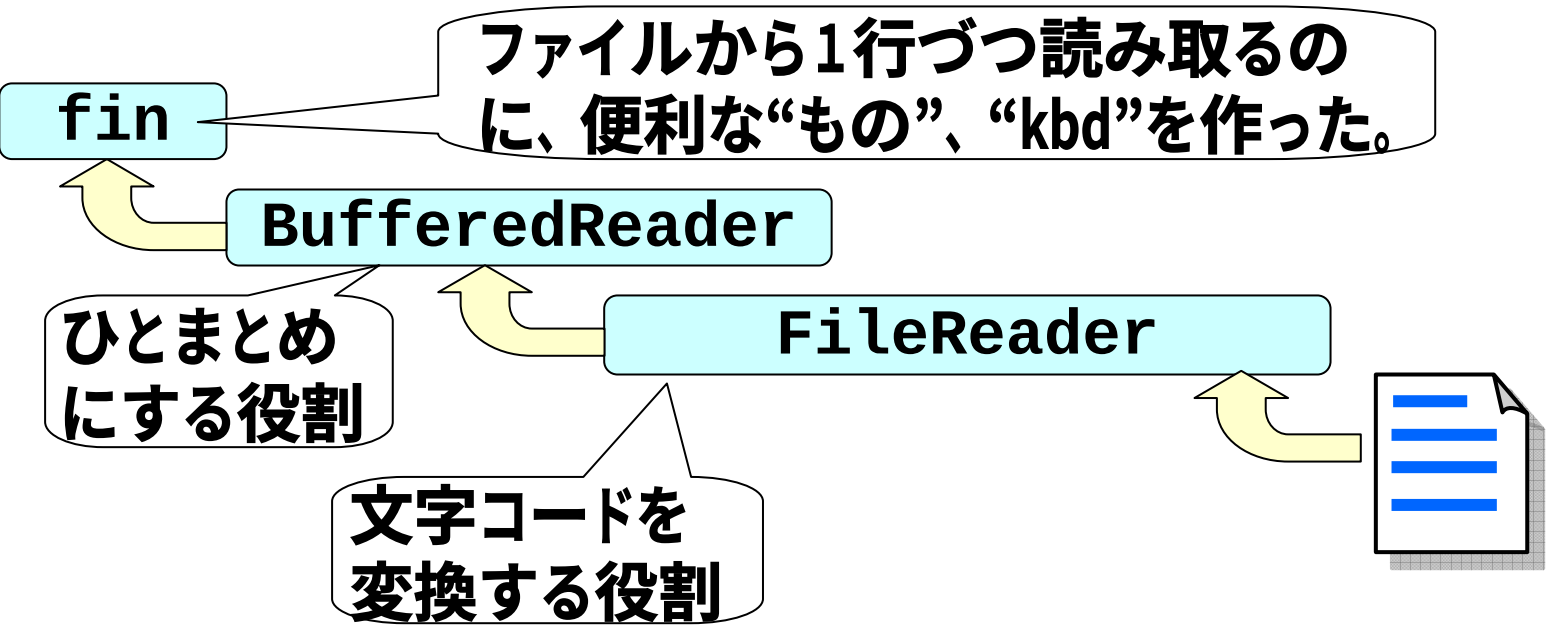


(14．2．1) ファイル入力を行う2行　－1－

■ファイルから入力を行うプログラムは2行だけです。

■最初の行では、ファイルから1行分を入力するために便利なものである、“fin”を作っています（少し荒っぽい言い方ですが）。

```
BufferedReader fin = new BufferedReader(
                         new FileReader(fileName));
```

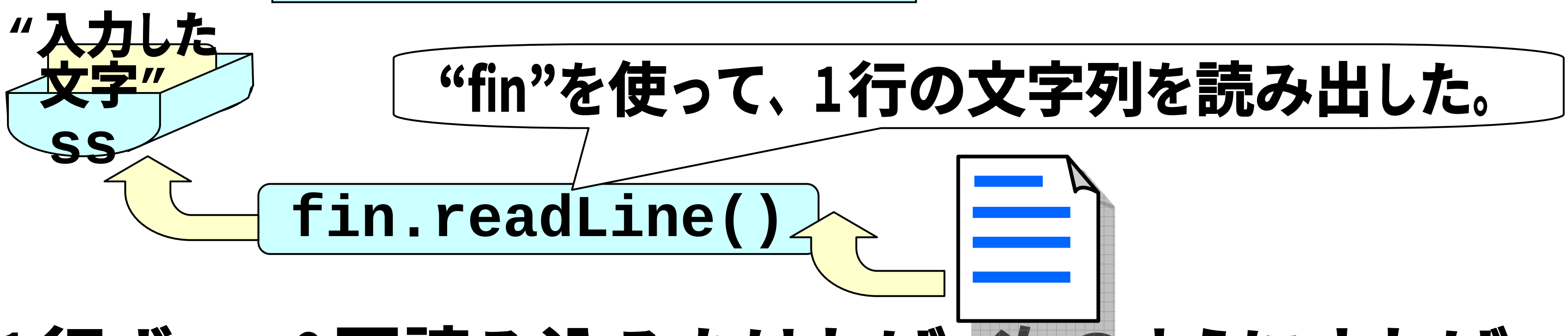


■“bufferedReader”は、コマンドラインからの入力で使用したもの（スライド(6)）と同じです。

(14.2.2) ファイル入力を行う2行 －2－

■1行分を入力するために便利なもの“fin”を使えば、1行を読み込む処理は、次のように簡単です。

```
ss = fin.readLine()
```



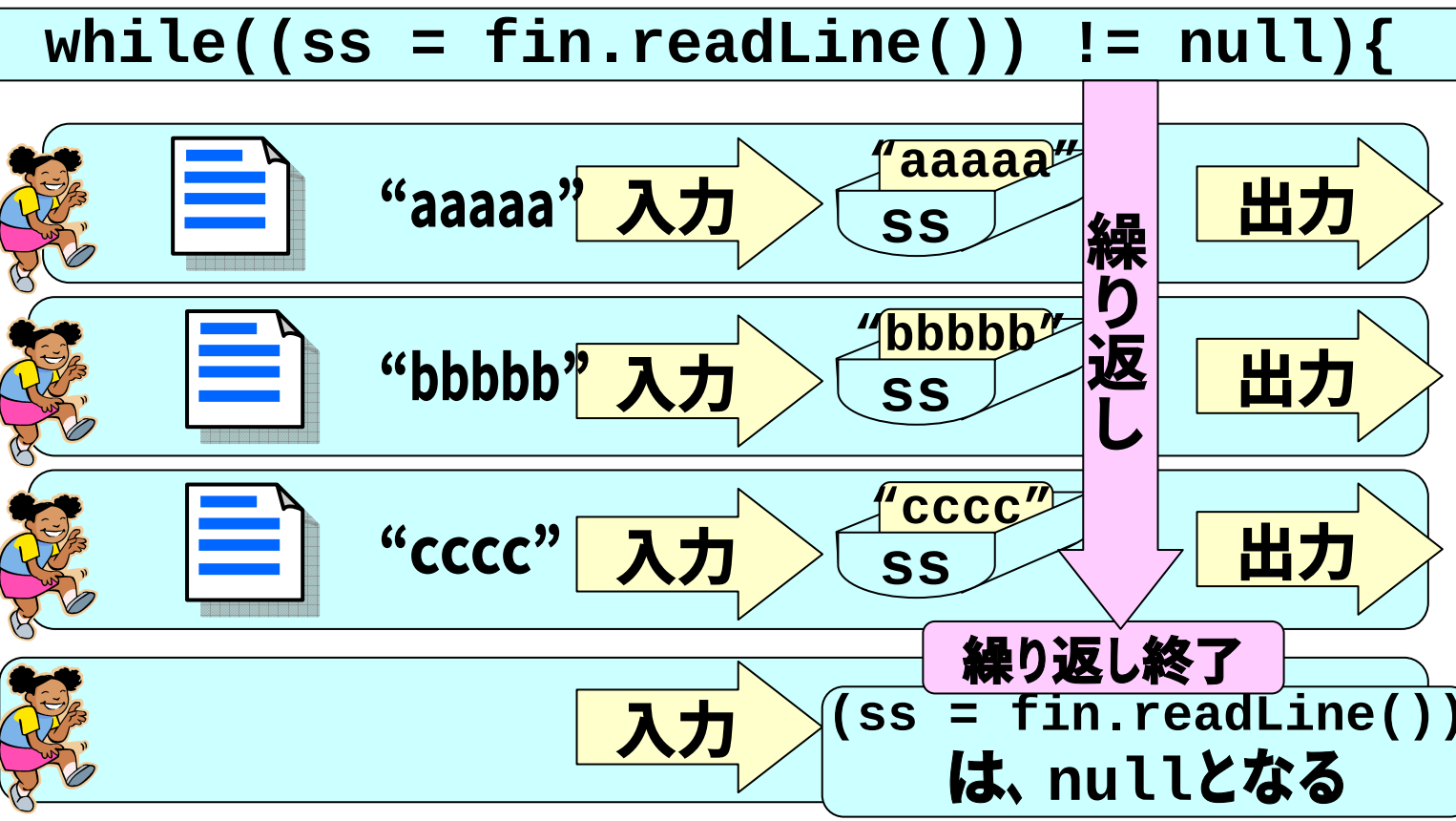
■1行ずつ、2回読み込みたければ、次のようにすればＯＫです。

```
String ss1,ss2;
BufferedReader fin =
  new BufferedReader(new FileReader(fileName));
ss1 = fin.readLine();
ss1 = fin.readLine();
```

(14.3) 行がある限り。。。

■スライド(14.1)のプログラムでは、ファイルに行がある限り、繰り返しを行っています。

```
while((ss = fin.readLine()) != null){
```



＜出力＞

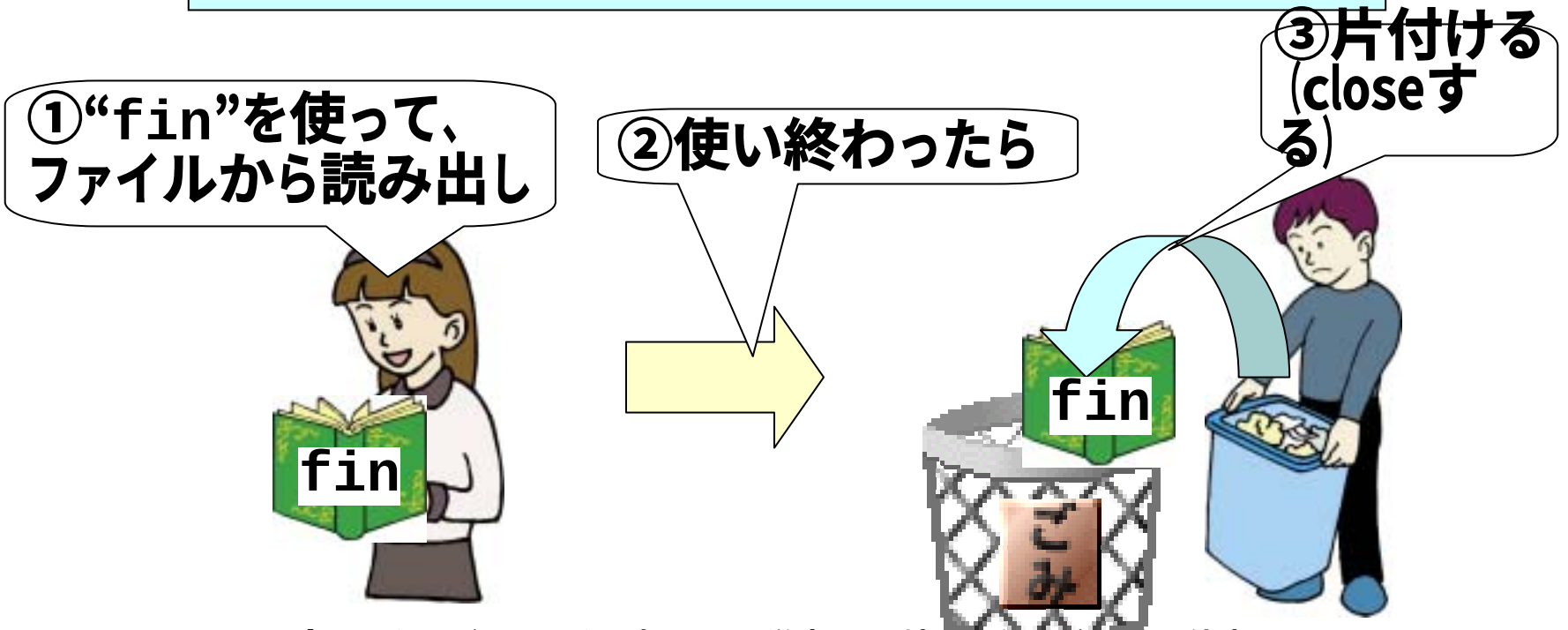
```
aaaaa
bbbbb
cccc
```

■スライド(9.5.2)で見たwhile文の使い方ですね。

(14．4) ファイルのクローズ

■ファイルから1行分を入力するために便利なもの
“fin”は、使い終わったら片付けます。

```
        : // ファイルからの読み込み処理

fin.close();
```



(14.5)日本語のファイルの読み取り

■日本語を含むファイルについては、行を読み取った後、次のように変換するようにしてください。

```
String ss;
BufferedReader fin =
  new BufferedReader(new FileReader(fileName));
ss = fin.readLine();
ss = new String(ss.getBytes("iso-8859-1")
                ,"JISAutoDetect");
```

(14.6.1) 課題

■課題14－1

- スライド (14.1の) プログラムを作成してください。
- 入力するファイルの作り方については、スライド(14.7)を参照してください。

■課題14－2

- 次のようなプログラムを作成してください。
  - ◆入力パラメータでファイル名を指定された、右のようなファイルを入力します。
  - ◆コンソールに下のようなグラフを出力します。
  - ◆項目の数は3つ (A、B、C) の固定としてOKです。
  - ◆形式が間違っているファイルに対する考慮も無用です。

＜test.txt＞

```
A
12
B
28
C
20
```

```
$ java Kadai142 test.txt
 A |************
 B |****************************
 C |********************
   +---------+---------+---------+
   0         10        20        30
```

（１４．６．２）課題

■課題１４－３

- ２つのファイルを読み込んで、一致する行のみを出力するプログラムを作成してください。

（２つのファイルで行数が異なる場合は、余分となる行は一致しないと考えてください。）

<test1.txt>

```
Abc
1234
dEf
5678
ghi
910
```

<test2.txt>

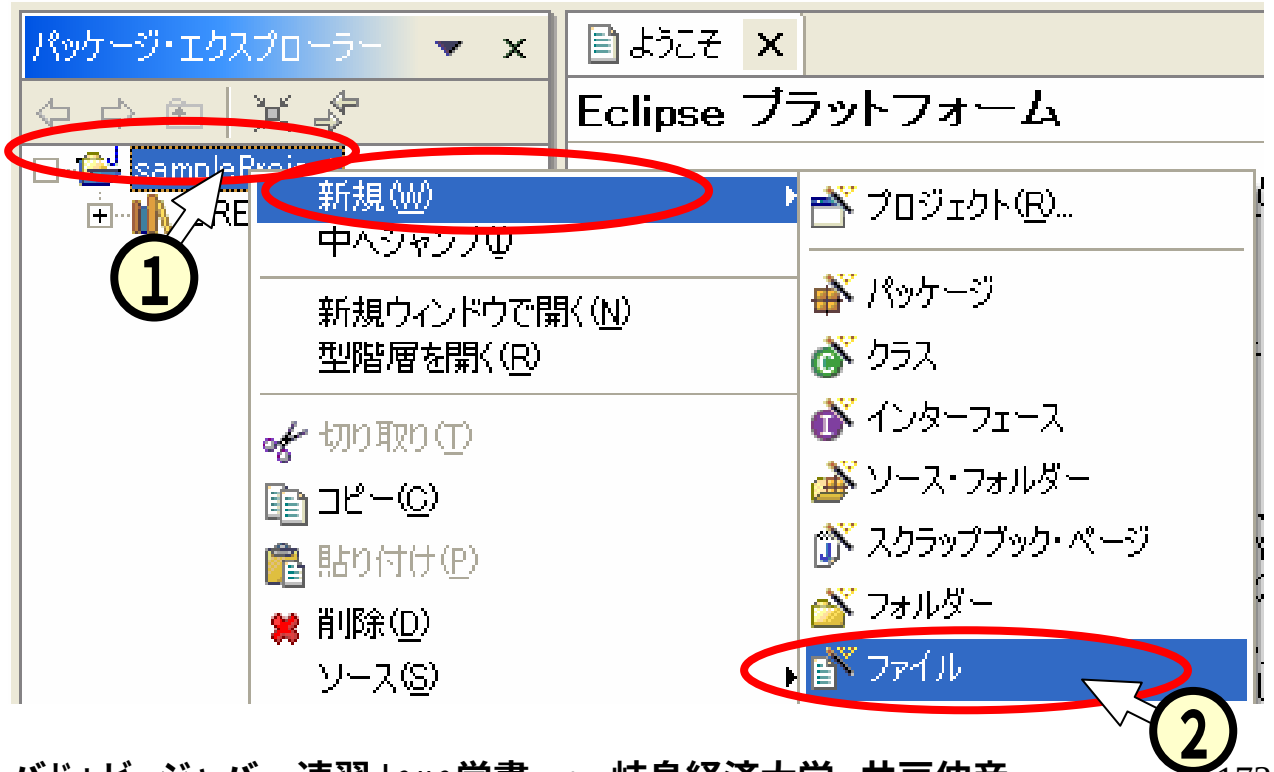
```
Abc
12345
dEf
5678
ghijk
910
```

```
$ java Kadai143 test1.txt test2.txt
Abc
dEf
5678
910
```

(14．7．1) eclipseでのファイル作成ー1ー

■プロジェクトの直下にファイルを作成します。

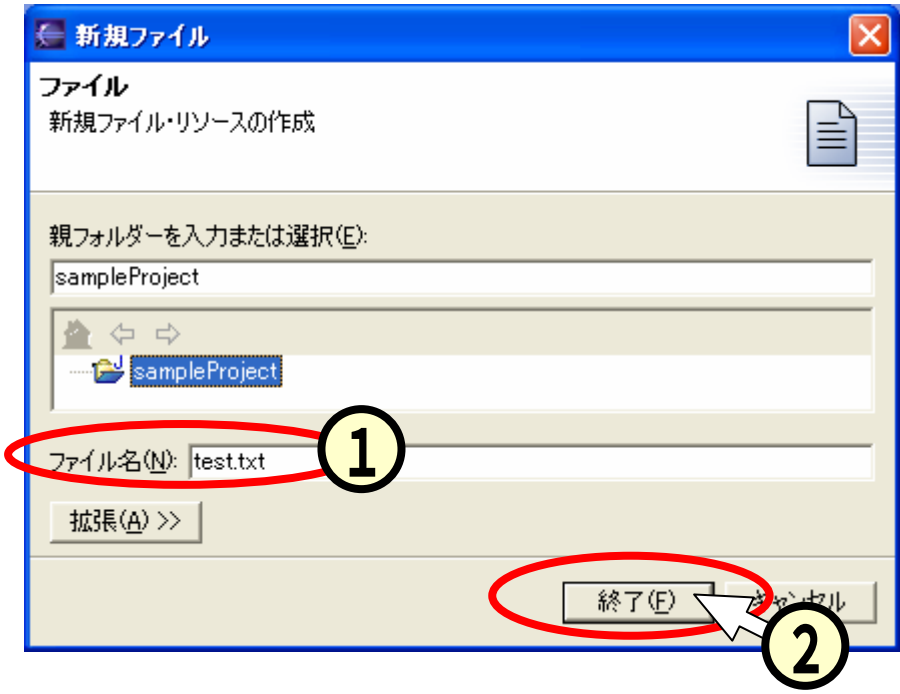
■パッケージ･エクスプローラー中、プロジェクト(この場合、sampleProject)を右クリック(①)し、現れたポップアップメニューから[新規]-[ファイル]をクリック(②)します。

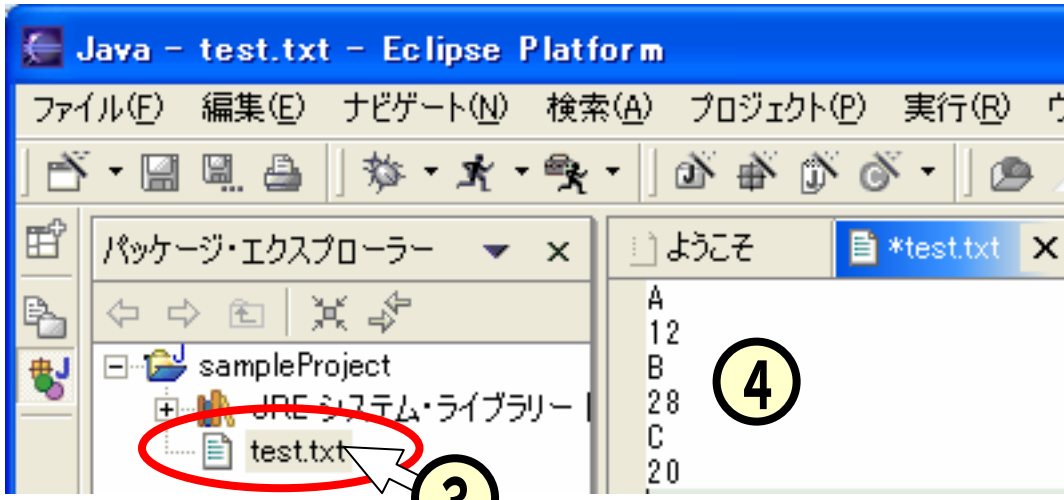


(14．7．2) eclipseでのファイル作成ー2ー

■「新規ファイル」のウィンドウにて、ファイル名を入力(①)し、終了をクリック(②)します。

■プログラムファイルと同様に、パッケージ・エクスプローラー上で選択(③)し、エディタ上で編集(④)出来ます。

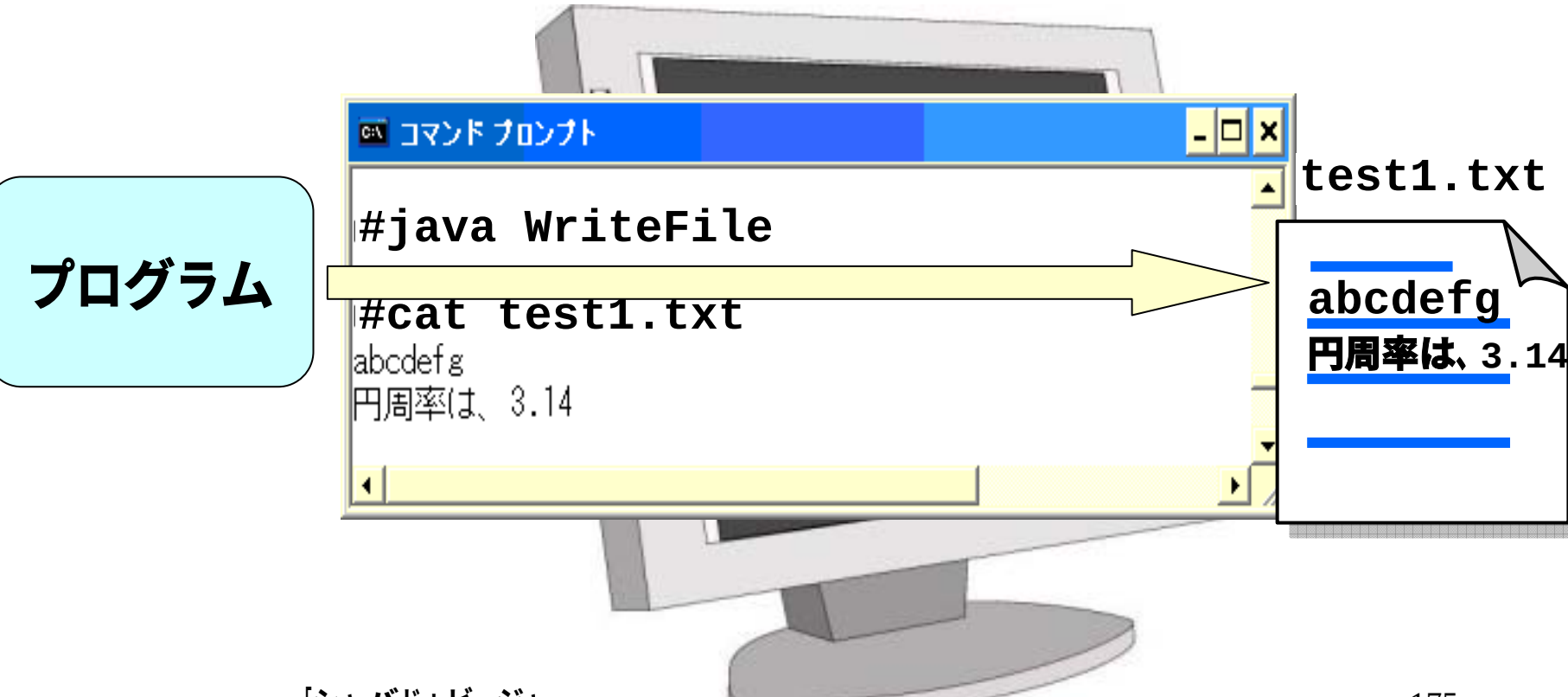




(15) ファイルへの出力

■何をしたいのか？

- テキスト･ファイル“test1.txt”へ、1行ずつ書き込む処理を作成します。



(15.1) ファイルへ出力を行うプログラム

■次のようなプログラムを考えます。

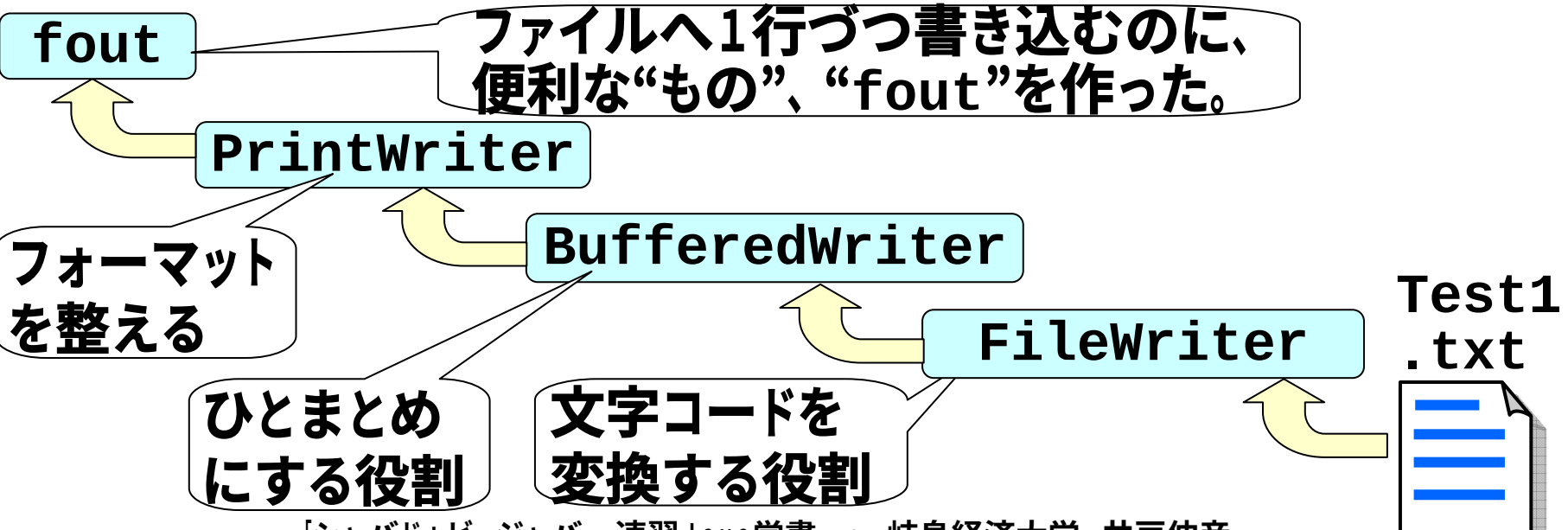
```
import java.io.*;

public class WriteFile {

  public static void main(String[] args)
                         throws IOException {
    PrintWriter fout = new PrintWriter(
                 new BufferedWriter(
                 new FileWriter("test15.txt")));
    fout.println("abcdefg");
    fout.println("円周率は、"+3.14);
    fout.close();
  }
}
```

(15．2．1) ファイル出力を行う2行　－1－

■出力を行うプログラムは2行だけです。

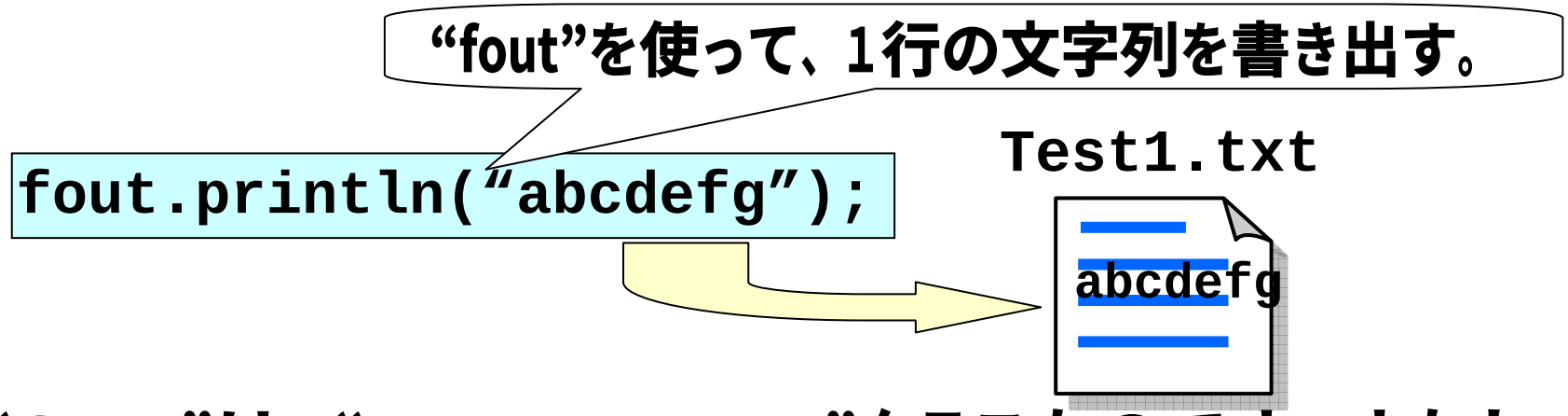
■最初の行では、1行分を出力するために便利なものである、“fout”を作っています (少し荒っぽい言い方ですが)。

```
PrintWriter fout = new PrintWriter(
                   new BufferedWriter(
                   new FileWriter("test1.txt")));
```



(15．2．2) ファイル入力を行う2行　－2－

■1行分を出力するために便利なもの“fout”を使えば、1行を書き込む処理は、次のように簡単です。



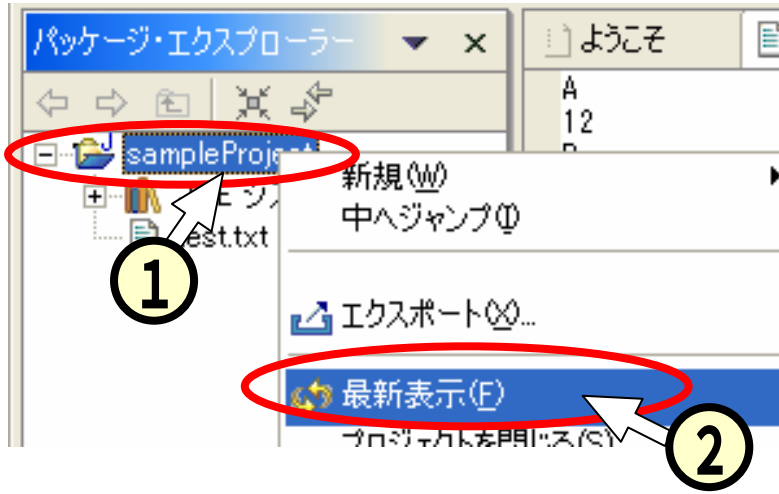
■“fout”は、“PrintWriter”クラスなのです。すなわち、スライド(2)「データの出力」で用いた、“System.out”とまったく同じ操作が可能です
(“System.out.println”、　”System.out.print”)。

■クラスの話は、別スライドにて説明します。

(15.3.1) 課題

■課題15－1

- スライド(15)のプログラムを作成してください。
- 作成されたファイルをパッケージ･エクスプローラー上に表示させるには、プロジェクトを右クリック(①)して出てきたメニューから、[最新表示]をクリック(②)します。



■課題15－2

- コンソールからファイル名と内容を入力して、ファイルを作成するプログラムを作成してください。
- “.”のみの行が入力された場合、内容の入力が終了したと判断することにします。

```
$ java Kadai152
input file name:
test152.txt
input lines:
大和撫子七変化
素顔のほうが嘘つきね
.
```

<test.152.txt>

```
大和撫子七変化
素顔のほうが嘘つきね
```

(15.3.2) 課題

■課題15－3

- ファイルを読み込んで、改行位置を変換したファイルを出力するプログラムを作成してください。

```
$ java Kadai153 test1.txt test2.txt
```

入力するファイル名（test1.txt）

出力するファイル名（test2.txt）

＜test1.txt＞
このファイルは
予め作っておく

```
Abc
1234
dEf
5678
ghi
910
```

変換 ⇒

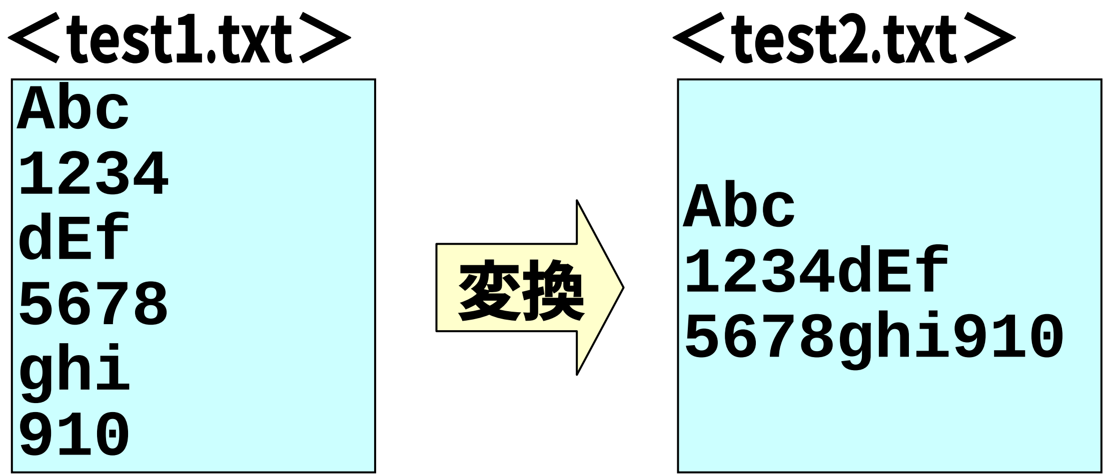
＜test2.txt＞
このファイルを
プログラムにより
作成する

```
Abc1234
dEf5678
ghi910
```

（１５．３．３）課題

■課題１５－４

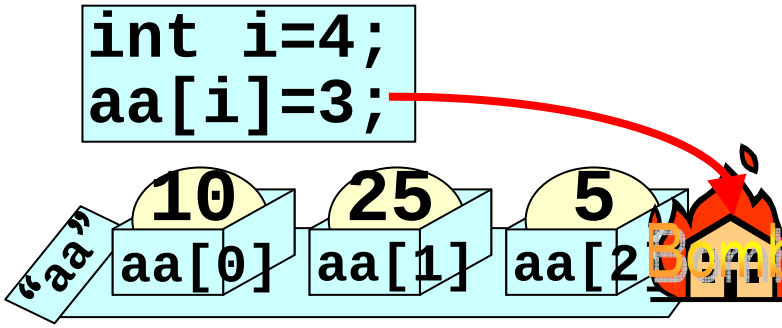
- 変更後の改行位置が、1行ごと、2行ごと、3行ごと、....というようになるよう課題１５－３のプログラムを変更してください。



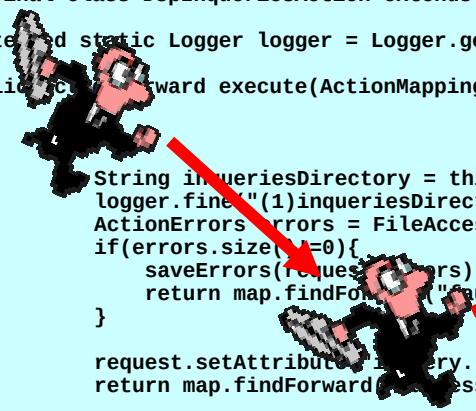
(16) 例外処理

■例外処理とは？

- プログラムを実行しているときに、次のような思い通りでないことが発生することがあります。
  - ◆ファイルが見つからない。
  - ◆長さ3の配列の、7番目の要素にアクセスした。
- このようなこと(例外)が発生した場合に、どのような処理を行うかを指定することを、例外処理といいます。




```
public final class DspInquiriesAction extends Action{

    protected static Logger logger = Logger.getLogger(LoggingPlugIn.class.getName());

    public ActionForward execute(ActionMapping map,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response)
                                 throws IOException, ServletException{
        String inquiriesDirectory = this.servlet.getInitParameter("inquiry-directory");
        logger.fine("(1)inquiriesDirectory="+inquiriesDirectory);
        ActionErrors errors = FileAccess.setInquiryList(inquiriesDirectory,inquiryList);
        if(errors.size()!=0){
            saveErrors(request,errors);
            return map.findForward("fault");
        }

        request.setAttribute("inquiry.list",inquiryList.toArray());
        return map.findForward("success");
    }
```



走ってみたら、
思い通りでないことが
発生する場合がある。
このときどうするか？
⇒例外処理

(16．1) 例外処理の方法

■例外が発生する恐れのある実行文の扱い方に、2つの方法があります。

- 例外処理を書く

  try文で例外が発生する恐れのある実行文を指定し、catch文にて例外発生時の処理を指定する。

- 例外を投げる(スローする)

  throws文により、例外が発生した時の処理を呼び出し元に任せる

```
try{
   //
   //
}catch(Exception e){
   //
   //
}
```

例外が発生する恐れのある実行文

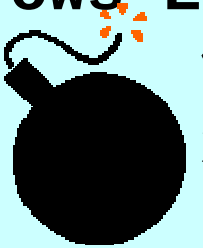
例外が発生した場合の処理

```
//呼び出し元
xx();
```

呼び出し

例外を投げる (throws)

```
public static void xx()
    throws Exception{
   //
   //
}
```



例外が発生する恐れのある実行文

（16．2）例外処理が必要な実行文

■スライド(6.1)に示したデータを入力するプログラムでは、throw文による例外処理をしていました。

■スライド(6.1)のプログラムの“throws IOException”の部分を右のように削除してコンパイルすると、下のようなエラーが発生します。すなわち、データ入力では例外処理が絶対に必要で、省略出来ません。

```
import java.io.*;
public class Input {
  public static
  void main(String[] args) {
        :
    ss = kbd.readLine();
        :
  }
}
```

“throws IOException”を削除

＜eclipseでのコンパイルエラー表示＞

		!	説明	リソース	フォルダー内	ロケーション
			処理されない例外の型 IOException	Input.java	javaText/src	行 21

＜コマンドラインでのコンパイルエラー表示＞

```
Input.java:21: 例外 java.io.IOExceptionは報告されません
スローするにはキャッチまたは、スロー宣言をしなければなりません。
                ss = kbd.readLine();
                     ^
エラー1個
```

(16．3) try, catch

■スライド(14.1)の“ファイル入力を行うプログラム”に、try～catch～の例外処理を入れたプログラムを、次に示します。

```
mport java.io.*;
ublic class CatFile {
 public static void main(String[] args){
   String fileName = args[0];
   String ss;
   BufferedReader fin = null;
   try{
     fin = new BufferedReader(new FileReader(fileName));
     while((ss = fin.readLine())!=null){
       System.out.println(ss);
     }
     fin.close();
   }catch(Exception e){
     System.out.println("ファイル("+fileName
                  +“)読み取りで例外が発生しました。“);
     System.out.println("e="+e);
   }
 }
```

“throws Exception”は削除

try節：例外が発生する恐れのある処理

Catch節：例外が発生した場合に実行される処理

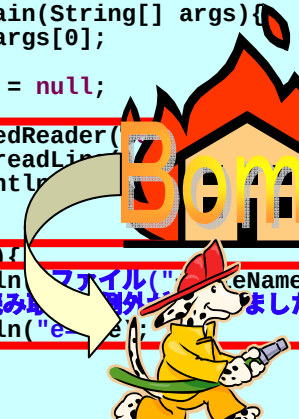
(16．3．1) 存在しないファイルを指定する

■引数に存在しないファイル名を指定すると、次のような実行結果となります。

```
# java CatFile test2.txt
ファイル(test2.txt)読み取りで例外が発生しました。
e=java.io.FileNotFoundException: test2.txt (指定されたファイルが見つかりません。)
```

■try節（“try{“と”}”で囲まれた部分）で例外が発生して、catch節（“catch{“と”}”で囲まれた部分）の部分が実行された訳です。

```
import java.io.*;
public class CatFile {
  public static void main(String[] args){
    String fileName = args[0];
    String ss;
    BufferedReader fin = null;
    try{
      fin = new BufferedReader(
      while((ss = fin.readLi
        System.out.println
      }
      fin.close();
    }catch(Exception e){
      System.out.println("
                +")読み
      System.out.println("e=
    }
  }
}
```



①Try節で例外が発生し、

②catch節の処理が実行される

（16．3．2）Exception

■catch節では、次のように“Exception（例外）”クラスのインスタンスである、“e”を用いています（クラスについては、別スライドで説明します）。

```
    }catch(Exception e){
                        :
      System.out.println("e="+e);
    }
```

■例外“e”には、例外に関するさまざまな情報が詰まっています。“e”を、そのまま出力すると、例外の内容が表示されます。

```
System.out.println("e="+e);
```

例外“e”

・どのような例外か？
・どのように呼ばれた場合に発生したか？

java.io.FileNotFoundException: test2.txt （指定されたファイルが見つかりません。）

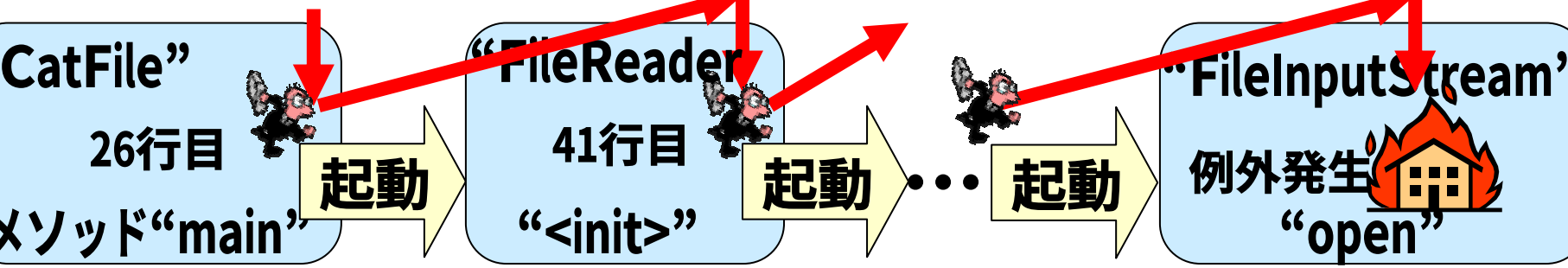
(16．3．3) スタック･トレース

■例外処理でよく出力されるものに、スタック･トレースがあります。

```
e.printStackTrace();
```


```
at java.io.FileInputStream.open(Native Method)
at java.io.FileInputStream.<init>(FileInputStream.java:103)
at java.io.FileInputStream.<init>(FileInputStream.java:66)
at java.io.FileReader.<init>(FileReader.java:41)
at CatFile.main(CatFile.java:26)
```

■これは、例外が発生するまでにどのようなメソッドが呼ばれているかを示すデータです。



(16．4．1) スロー宣言

■スロー宣言の使用例を、次に示します。

```
import java.io.*;
public class CatFile2 {
  public static void main(String[] args) {
    String fileName = args[0];
    try {
      fileAccess(fileName);
    } catch (Exception e) {
      System.out.println("ファイル("+fileName
                         +")読み取りで例外が発生しました。");
      System.out.println("e="+e);
    }
  }

  public static void fileAccess(String fileName)
                                              throws Exception {
    BufferedReader fin;
    fin = new BufferedReader(new FileReader(fileName));
    String ss;
    while((ss = fin.readLine())!=null){
      System.out.println(ss);
    }
    fin.close();
  }
}
```

例外を投げてくるメソッドの
呼び出しを、try～,catch～で囲む

例外が発生する恐れのある処理

(16．4．2) 呼び元に任せる

■スロー宣言をしているメソッド“fileAccess”を呼ぶことは、例外を発生する恐れがある処理になります。

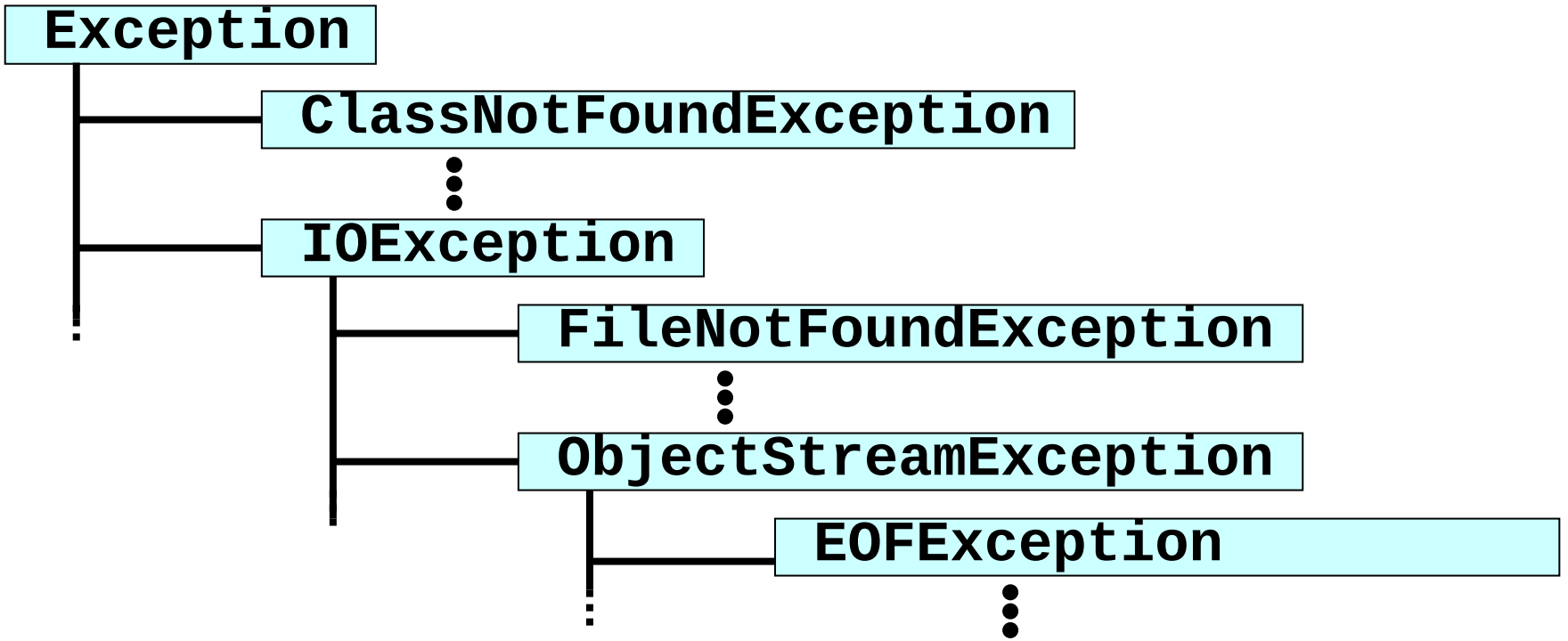
```
public static void main(String[] args) {
        :
 try {
 // 例外をスローするメソッド
 // の呼び出しは、例外を発生
 // する恐れのある処理となる
 // ので、try節とし、

   fileAccess(fileName);

 } catch (Exception e) {

 // 例外が発生した場合の処理
 // をcatch節に記す。

 }
}
```

起動

```
public static void fileAccess
        (String fileName) throws Exception {
 // 例外が発生する恐れが
 // ある処理

}
```

(16.5.1) 例外の種類

- ここまで、例外はすべてクラス“Exception”として扱ってきましたが、実際にはさまざまな種類(クラス)があります。
- これらは親子の関係に整理されており、いちばん親にあたる“Exception”を指定すれば、どのような例外も含むことになります。



(16.5.2)例外の種類による処理

■発生した例外の種類により、例外処理を分けることも出来ます。

```
try {
        :
} catch (FileNotFoundException e) {
  // ファイルが見つからない場合
  System.out.println("ファイル("+fileName
                          +")は見つかりません。");
} catch (IOException e) {
  // 上記の“FileNotFoundException”以外の
  // 入出力装置例外が発生した場合
  System.out.println("ファイル("+fileName
                          +")読み取りで例外が発生しました。");
}
```

■最後のcatch節にて、“Exception”を受けておけば、これより前で処理されなかった例外は、そこで処理されます。

(１６．６．１) 課題

■課題１６－１

- スライド (16.3) のプログラムを作成してください。
- プログラムを実行し例外を発生させて、動作を確認していください。

■課題１６－２

- スライド (16.4.1) のプログラムを作成してください。
- プログラムを実行し例外を発生させて、動作を確認していください。

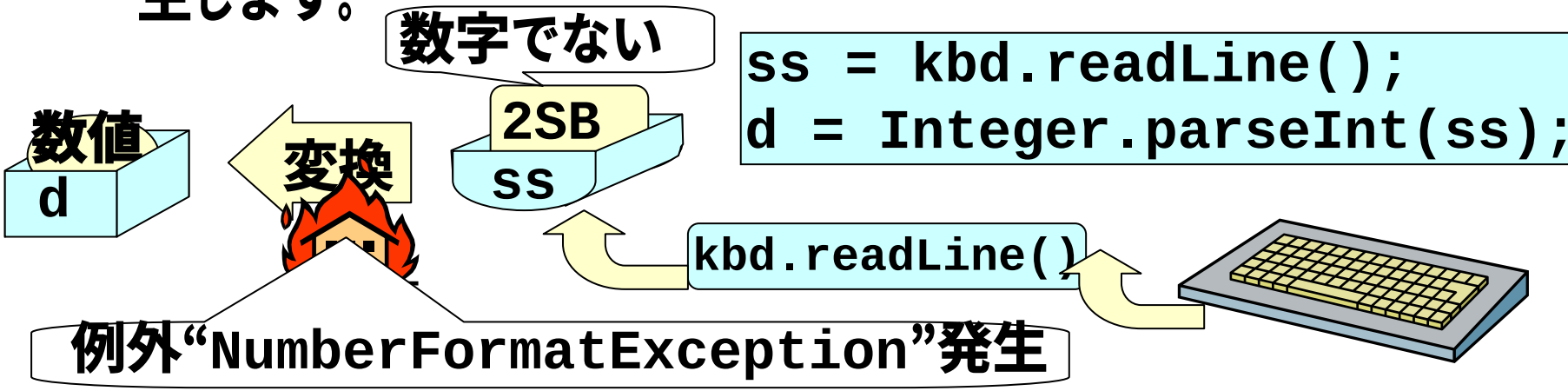
(16.6.2) 課題

■課題16－3

- 課題6－1のプログラムについて、次の変更を行ってください。
  - ◆スロー宣言をやめて、try節catch節で例外処理する。

■課題16－4

- 文字列を数値に変換する際、変換元の文字列に数字以外が含まれていると、例外“NumberFormatException”が発生します。



- キーボードから数字以外が入力された際には、“数字以外が入力されました”と表示するように、課題16－3のプログラムを変更してください。

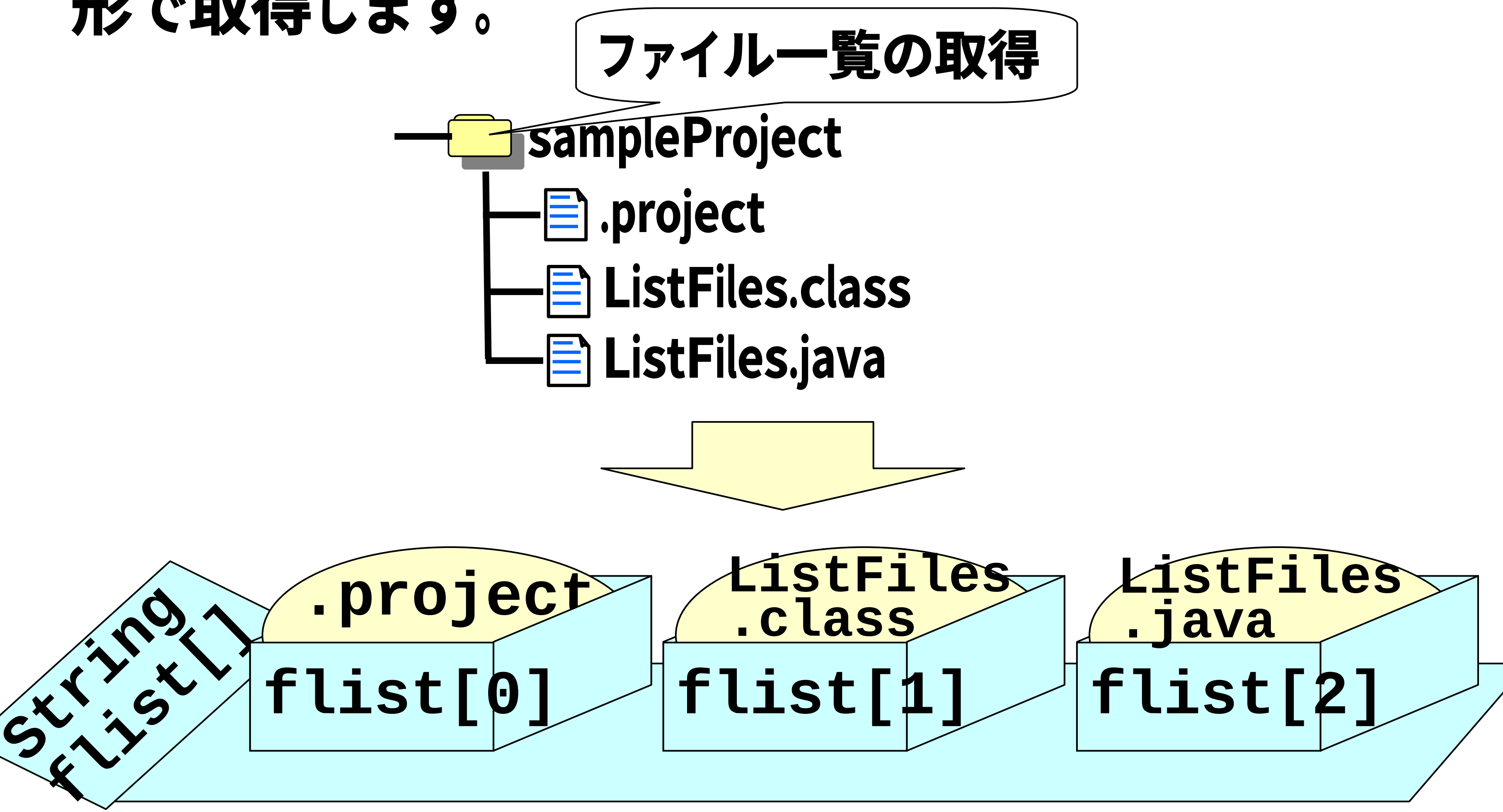
（17）ディレクトリ・ファイルの操作

■ここでは次の項目について説明します。

- （17．1）ファイル一覧の取得
- （17．2）ディレクトリの判定

（17．1）ファイル一覧の取得

■ディレクトリ（フォルダ）配下のファイル一覧を、配列の形で取得します。



(17．1．1) プログラム

■つぎのようなプログラムを考えます。

```
import java.io.File;
public class ListFiles {
  public static void main(String[] args) {
    try{
      File dir = new File(".");
      String flist[] = dir.list();
      if(flist==null){
        System.out.println("ファイル一覧取得は失敗。");
        return ;
      }
      for(int i=0;i<flist.length;i++){
        System.out.println(flist[i]);
      }
    }catch(Exception e){
      System.out.println("ファイル一覧取得で例外発生。e="+e);
    }
  }
}
```

実行結果 ⇒

```
$ java ListFiles
.project
ListFiles.class
ListFiles.java
```

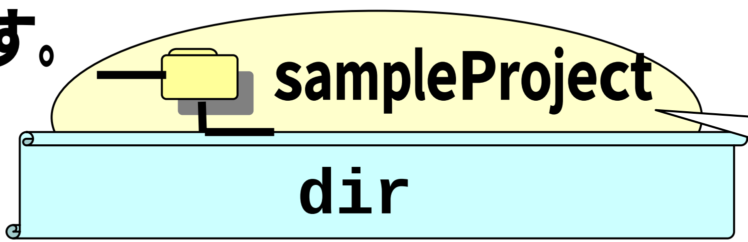
（17．1．2）動作

- ■ファイル一覧取得にかかわる処理は、2行です。
- ■ファイル（ディレクトリ）を取得します。

```
File dir = new File(".");
```

ファイル（ディレクトリ）　　パス名（カレントディレクトリ）

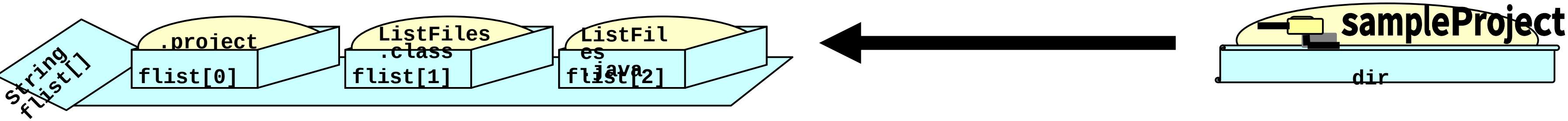
- ここでいう“ファイル”とは、普通のファイルに加えて、ディレクトリも含めています。上記の場合、パス名にディレクトリを指定しているので、取得されるファイルは実際はディレクトリです。



カレントディレクトリが“sampleProject”の場合

- ■ディレクトリから、配下のファイルの一覧を取得し、配列に格納します。

```
String flist[] = dir.list();
```



(17.1.3) 課題

■課題17－1

- スライド (17.1.1) のプログラムを実行してください。

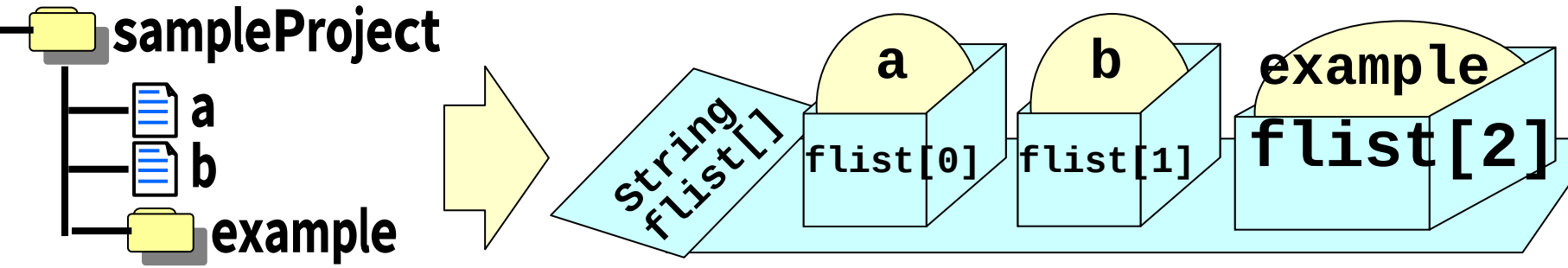
■課題17－2

- 次のようなプログラムを作成してください。
  - ◆ディレクトリと文字列とを入力パラメタとする。
  - ◆入力したディレクトリ配下の、次の条件を満たすファイル一覧を出力する。
    - ▫ファイル名が、入力した文字列で始まっている。

```
$ java Kadai172 C:¥eclipse abc
abcedf.txt
abcOk.java
```

(17.2)ディレクトリの判定

■取得したファイル一覧には、ディレクトリも含まれます。



■ファイル一覧のそれぞれがファイルであるか否かは次のように判定します。

```
for(int i=0;i<flist.length;i++){
        パス名から
        ファイルを生成する          ファイルで
                                    あるか否か？

  if((new File(“.¥¥”+flist[i])).isFile()){
    // ファイルである場合のみ、ファイル名を出力する
    System.out.println(flist[i]);
  }
}
```