プログラミング入門1

第2回

式、型、変数

授業開始前に

ログオンして待機して

ください

不要ファイルの掃除

- 前回デスクトップにファイルをダウンロードした場合、次のものを削除してください
  - jtaf.zip
  - week01.zip

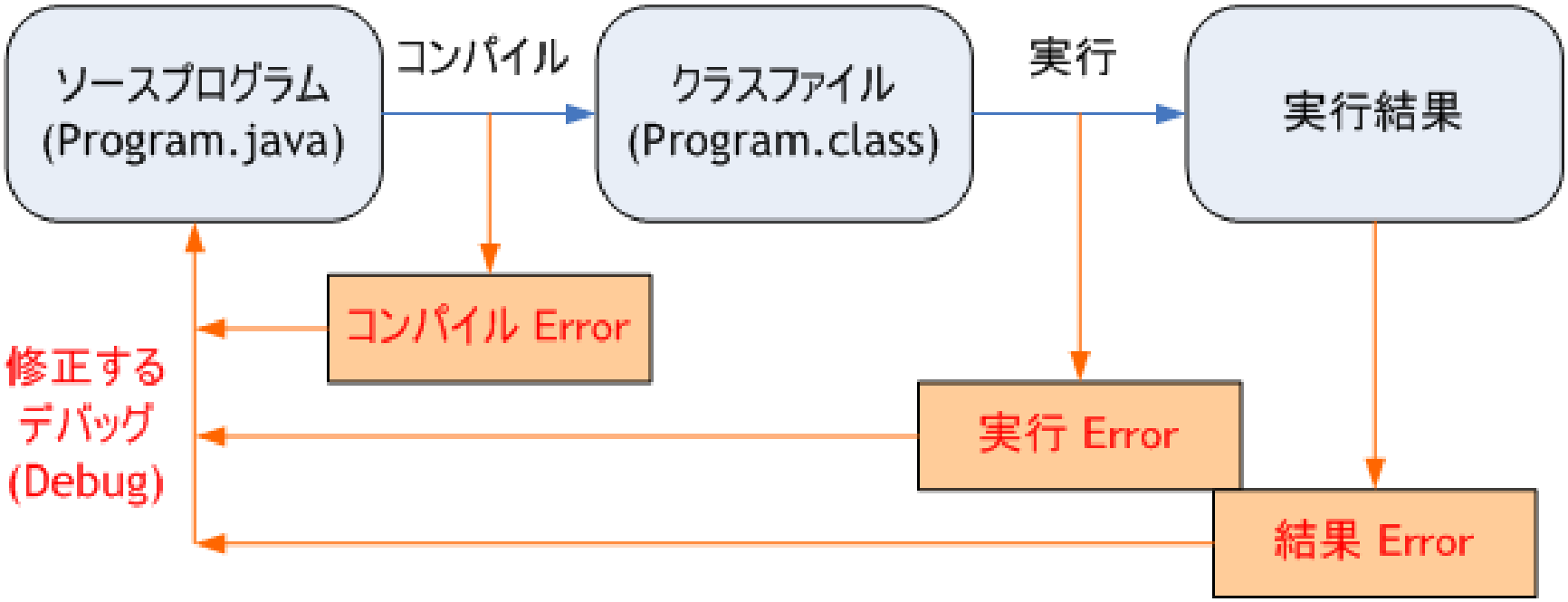
- デスクトップにファイルを置きすぎると、コンピュータをシャットダウンできなくなります

授業を始めます

前回の課題は

うまくできましたか？

復習：プログラムを書く・実行する



復習：最も簡単なプログラムの書き方

```
public class プログラムの名前 {
    public static void main(String[] args) {
        System.out.println("Hello!");

        ここに命令をいくつ並べてもよい

    }
}
```

復習:print 命令

```
public class Hello {
    public static void main(String[] argv){
        System.out.print("Hello, ");
        System.out.print(“world!");
    }
}
```

の実行結果は

```
Hello, world!
```

復習：println　命令

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

の実行結果は

```
Hello,
world!
```

復習:文字列の書き方

- 文字列は「"」から始まり「"」で終わる。

  "Hello" (Hello)

  "a" (a)

  "" (空の文字列)

  など
- 先ほど使用した文字列は "Hello, world!"

復習終わり

思い出しましたね。

今回のテーマ

- 算術演算
  - 加法、減法、乗法、除法などの計算をコンピュータにやらせる
- 型と変数
  - 計算の結果やデータをしまう場所が変数
  - 変数には型がある

算術演算

記号	意味	備考
+	加算	文字列を連結する演算にも使われる
-	減算	
*	乗算	
/	除算	整数での計算は端数が切り捨てられる
%	剰余	割った余り

算術式の例

```
(1 + 2) * (3 + 4) / 5
```

このような計算を実行して結果を
表示するには

```
System.out.println( (1+2)*(3+4)/5 );
```

算術演算のデータ
データには型がある

- 整数型(int)

  `0, 1, 3, 563, -32, -1024`

- 実数型

  `1.0, 3.14, -73.5, 18.(=18.0), .5`(=0.5)

  – 小数点つきの数
  – 正確には浮動小数点数（floating point number）と呼ばれる

算術演算の結果

- 整数と整数の演算結果は整数である。
  12 + 34 の結果は整数の46
- 実数と実数の演算結果は実数である。
  4.0 + 1.0 の結果は実数の 5.0
- 整数と実数、あるいは実数と整数の演算結果は実数である。
  7.0 / 3 の結果は実数の2.3333333333333335

整数の割り算

- 7/3 の結果は 2.333... という実数ではない。端数が捨てられて 2 という整数が 7/3　という算術式の値になる。
- 7割る3 は 2 余り 1 という計算をする。
- 算術式 7/3 の値は商の 2
- 余りは 7%3 という算術式で表し、値は 1

整数の値の範囲

- 最小値　-2^{31} ＝ -2147483648
- 最大値　$2^{31}-1$ ＝ 2147483647
- 整数の値は32ビットで表現されている。
  2ビットあれば00、01、10、11の4通り($4=2^2$)、
  nビットあれば2^n通り、
  32ビットあれば2^{32}通りの値の表現が可能。
  半分ずつを正負の整数で分け合う。
- 演算結果がその範囲を超えた場合は、正しい値が得られない。

実数の精度

- 実数の値は誤差がある。
- 有効数字１６桁くらい(10進数に換算)。
- コンピュータ内部では２進数

　0.1 のような１０進数としては「きりのよい」数も２進数では無限小数なので「きりのよい」数とは言えない。実際、精度の範囲内で最も近い数に「変更」される（丸められる）。その結果、0.1とそのコンピュータ内部での表現は数学的な意味で等しくはない。実数の値の比較は常に精度に注意を払う必要がある。

算術演算の優先度

- 小学校で習った通り
- +, - より *, / の優先度が高い
- 1+2*3-4/2 は以下のように計算される。
- 1+(2*3)-(4/2)

基本的な関数と定数

- 平方根sqrt、三角関数のsin、cos、tanなど、乱数random、べき乗pow
- PI = 3.14159265358979323846、自然対数の底 E = 2.718281828459045235 4
- Math.sqrt(2.0)　は平方根2
- Math.sin(Math.PI)　は sin π

計算して結果を表示するには

- 下の命令文は100.5 を表示するはず

```
System.out.println(100 + 0.5);
```

- 半径2.0の円の面積

```
System.out.println(2.0*2.0* Math.PI);
```

- 辺の長さが3.0の正方形の対角線の長さ

```
System.out.println(Math.sqrt(Math.pow(3.0,
  2.0) + Math.pow(3.0, 2.0)));
```

- 1 から 10 までの総和

```
System.out.println(1+2+3+4+5+6+7+8+9+10);
```

文字列の加法
連結ができる

```
System.out.print("2.0の平方根は");
System.out.print(Math.sqrt(2.0));
System.out.println("である");
```

加法演算は連結を意味するので同じ結果になる。

```
System.out.println("2.0の平方根は" + Math.sqrt(2.0) + "である");
```

同じ計算を繰り返す
同じデータを繰り返し使う

1 から 10 までの総和を計算し、それを5回表示する

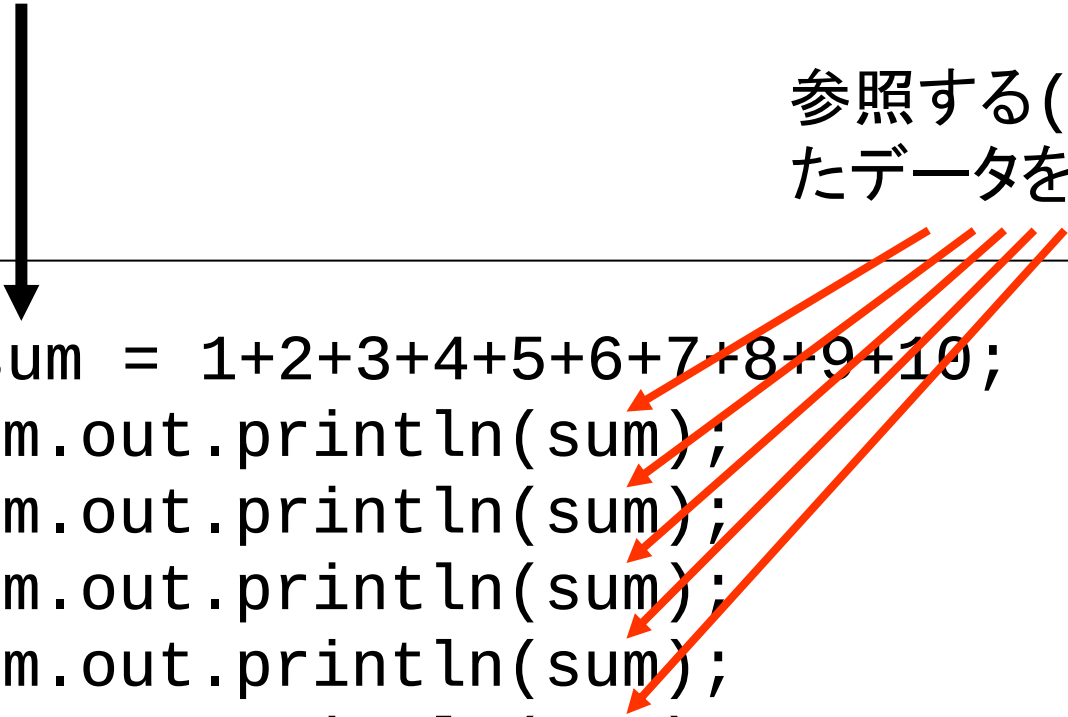
プログラムを書く場合、以下のように書くのは面倒である。

```
System.out.println(1+2+3+4+5+6+7+8+9+10);
System.out.println(1+2+3+4+5+6+7+8+9+10);
System.out.println(1+2+3+4+5+6+7+8+9+10);
System.out.println(1+2+3+4+5+6+7+8+9+10);
System.out.println(1+2+3+4+5+6+7+8+9+10);
```

計算は1度
結果はしまっておく

変数というものに計算の結果を入れておき、

参照する(変数にしまわれたデータを利用する)



```
int sum = 1+2+3+4+5+6+7+8+9+10;
System.out.println(sum);
System.out.println(sum);
System.out.println(sum);
System.out.println(sum);
System.out.println(sum);
```

変数と型

- 変数は何かを入れておく箱のようなもの
- 変数は名前が必要
- どんな種類(整数や実数など)のデータを入れる箱であるかを予め決めなければならない。
- これを型という
- 代入：　変数にデータを入れること
- 参照：　変数に入っているデータを利用すること

宣言：変数を用意する

int型の変数 x を宣言する

```
int x;
```

のようにすると、整数(integer)を入れるための変数 x が用意される。

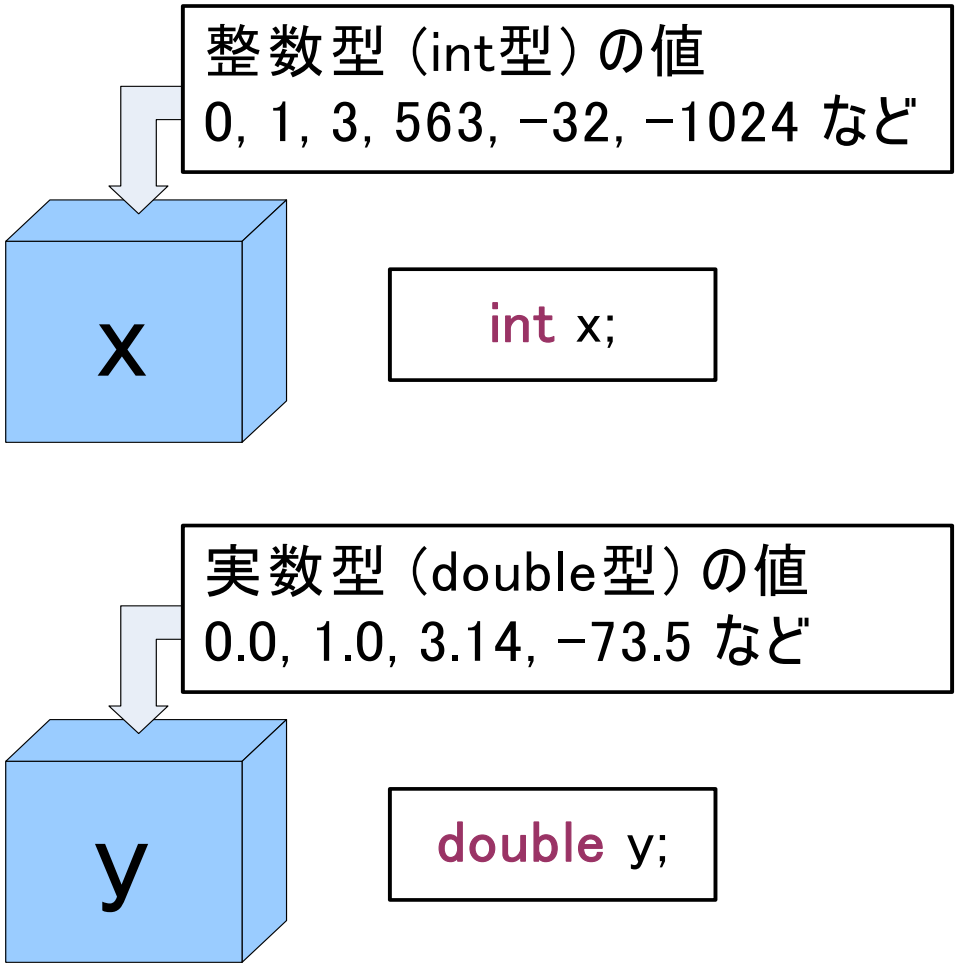
double型の変数 a を宣言する

```
double a;
```

のようにすると、実数を入れるための変数 a が用意される。

行末のセミコロン（;）を忘れないように。

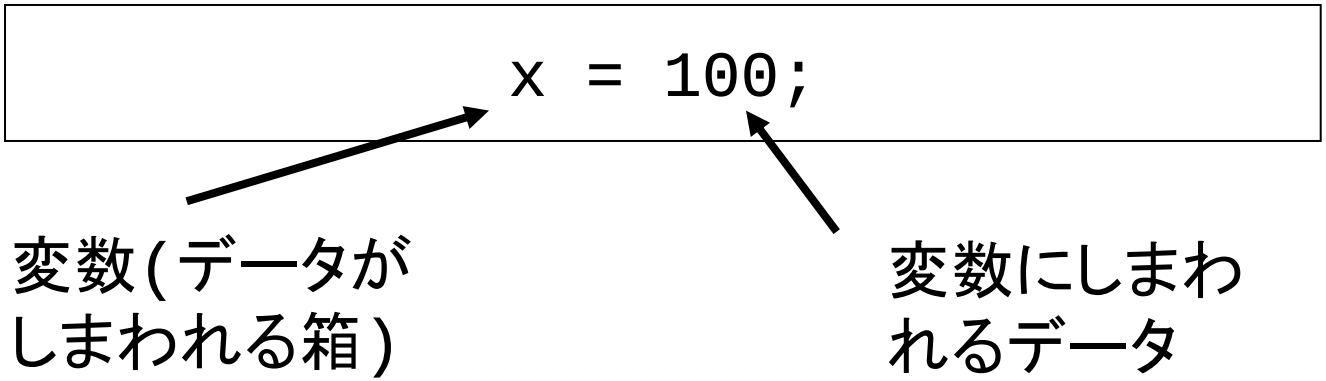
変数と型のイメージ


整数型（int型）の値
0, 1, 3, 563, −32, −1024 など
x
int x;
実数型（double型）の値
0.0, 1.0, 3.14, −73.5 など
y
double y;

算術演算に使われる型

型	サイズ	特徴
int	32bit	最もよく使われる整数型 (-2147483648 ~ +2147483647)
double	64bit	最もよく使われる実数型
byte	8bit	絶対値の小さな (-128 ~ +127) 整数型
short	16bit	絶対値の小さな (-32768 ~ +32767) 整数型
long	64bit	絶対値の大きな (-9223372036854775808 ~ +9223372036854775807) 整数型
float	32bit	精度の低い実数型
char	16bit	アルファベット、日本語を問わず1文字を表す
boolean	8bit	trueとfalseの2値のみを表現する型

代入：変数にデータをしまう



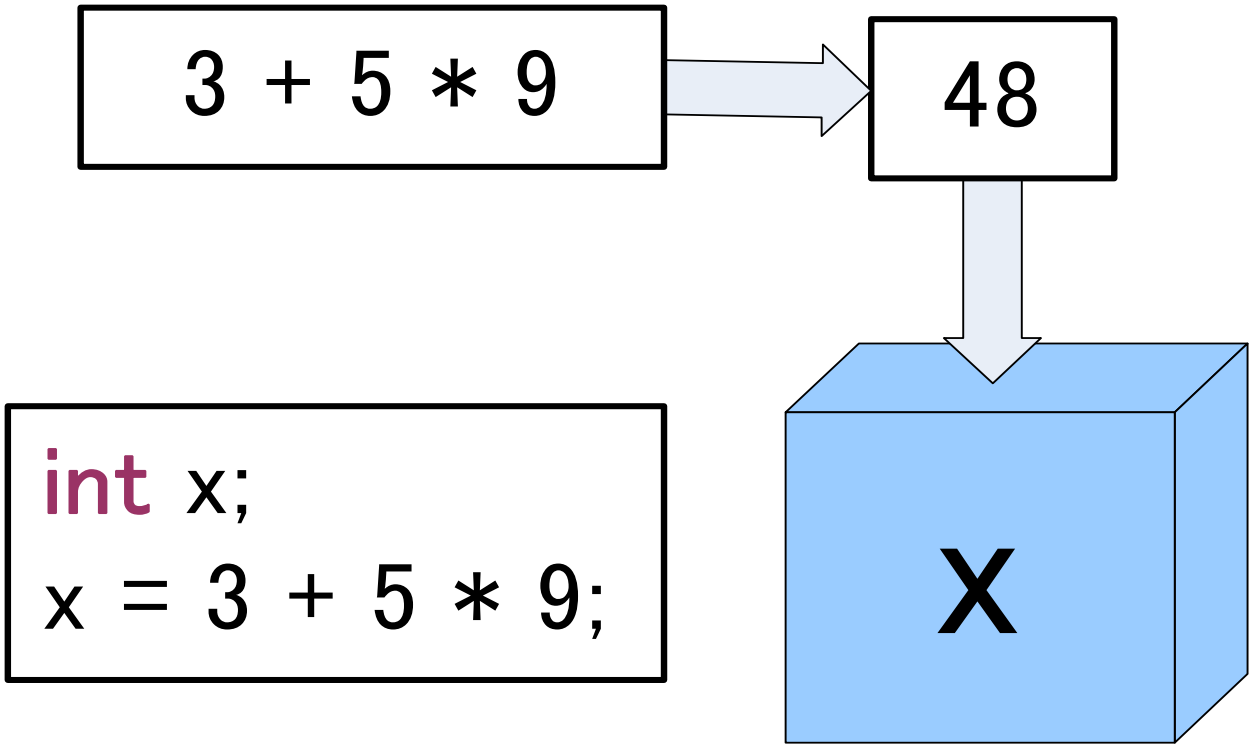
代入命令の書き方

- 左辺には変数名
- 右辺にはデータ(算術式もOK)

命令文はいつもセミコロン「;」で終わる。

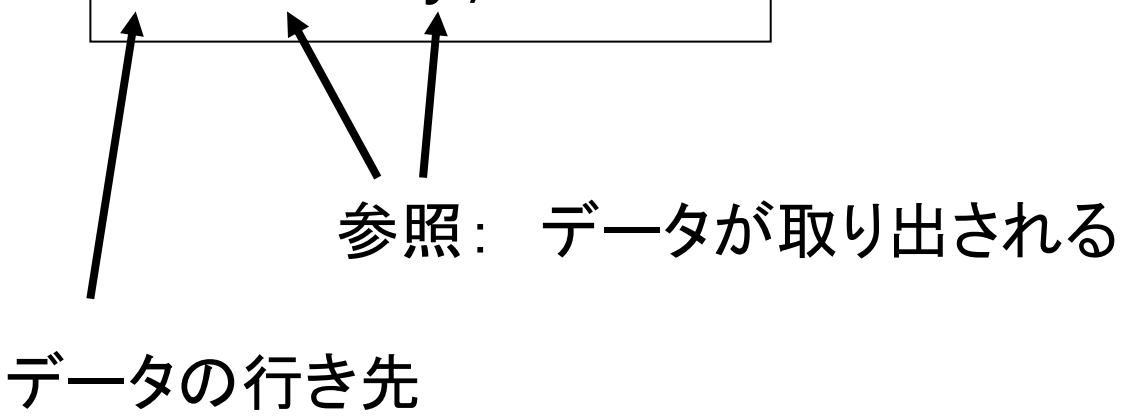
代入のイメージ

- 計算式ではなく、計算結果が格納される



参照：変数のデータを利用する

```
int x;
int y;
x = 100;
y = 200;
int z;
z = x + y;
```



x = x+1;

- 代入命令

  x と x+1 が等しいという意味ではない

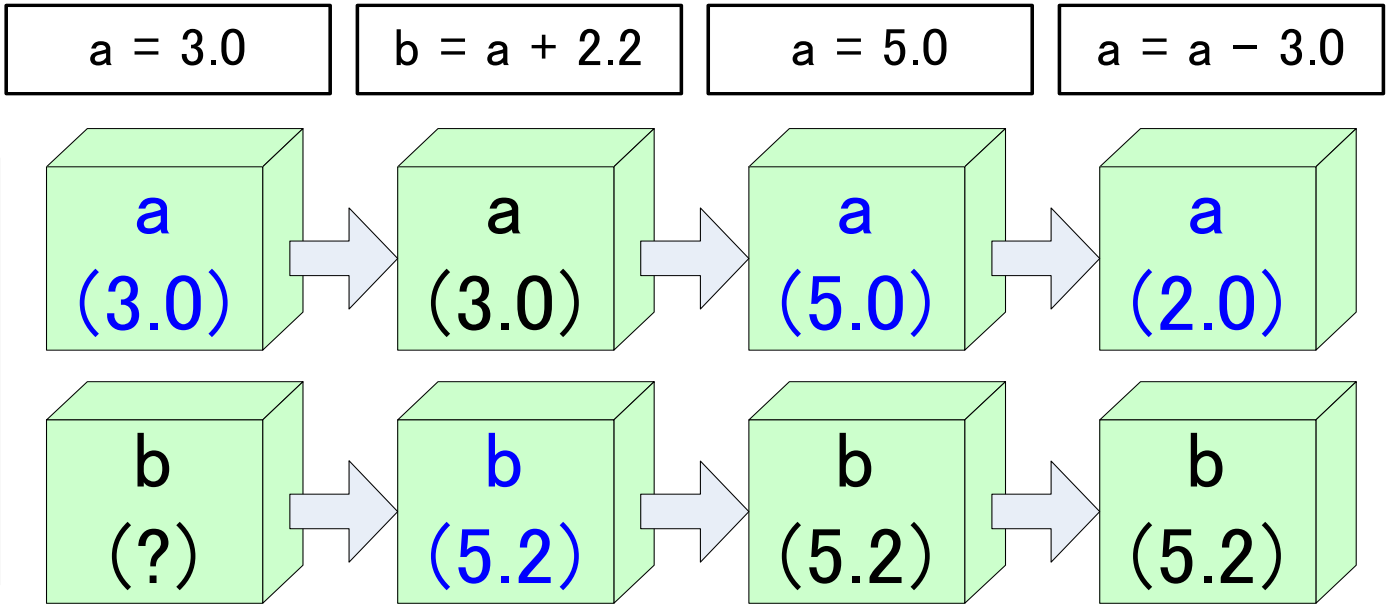
  左辺の x はデータをしまう場所を示す

  右辺の x は参照、つまりデータの取り出し
- 変数 x に入っているデータと 1 を足し算して、その結果を変数 x に代入するという意味。

代入と参照のイメージ

- 並べた順に、上から実行されていく

```
double a,b;
a = 3.0;
b = a + 2.2;
a = 5.0;
a = a - 3.0;
```



演算つき代入

```
a += 10;
```

変数 aに入っているデータを、それに 10 が加算されたデータに置き換える。

```
a = a + 10;
```

と同じ。

```
a -= 10;
```

変数 x に入っているデータを、それから 10 引かれたデータに置き換える。

```
a = a – 10;
```

と同じ。

変数の中身の増減

```
x++;
```

変数 x に入っているデータを、それに 1 が加算されたデータに置き換える。

```
x += 1;
```

と同じ。

```
x--;
```

変数 x に入っているデータを、それから 1 が減算されたデータに置き換える。

```
x -= 1;
```

と同じ。

変数の宣言と初期化

```
int zero = 0;
```

同じ

```
int zero;
zero = 0;
```

識別子：変数の名前

- アルファベットではじまること
- ２文字目以降は英数字なんでもよい
- 正確なルールは後に学ぶ
- abc、b747、Abc、i503d　などはよろしい
- 503d はだめ

一緒に試してみよう

- これから変数を使ったプログラムを作成します。
- 第１回目に出席していない人、あるいは第１回の設定作業が完了していない人は、すぐにTAを呼んでください。以下の作業を一緒にやることはできません。

演習の準備

- 今回の演習で使うテストドライバを第１回目と同じようにしてインストールします。
- 一緒に作業をしますので、指示に従ってやってください。

テストドライバの導入

1. プロジェクト「java20XX」にある「test」の左側の「+」をクリック
2. ツリーが展開されるので「install-libraries.xml」を右クリック
3. 「実行(R)」にマウスカーソルを合わせる
4. 「1 Ant ビルド」をクリック
5. 「コンソール」タブに"BUILD SUCCESSFUL"と表示されれば成功
6. eclipseの画面でプロジェクト「java20XX」を右クリック
7. メニューが表示されるので、「更新」あるいは「最新表示」をクリック
8. week02.zip をデスクトップなどにダウンロード
9. eclipseの画面でプロジェクト「java20XX」を右クリック
10. メニューから「インポート(I)」を選択
11. 「インポート」ウィンドウが表示されるので、「Zip ファイル」を選択
12. 「次へ(N)」をクリック
13. 宛先フォルダー(L): が「java20XX」になっていることを確認
14. From archive file: の右側にある「参照(R)...」あるいは 「ブラウズ(R)...」をクリック
15. ファイルダイアログが表示されるので、ダウンロードした "week02.zip" をダブルクリック
16. 前の画面に戻るので、From archive file: のエリアに正しいパスが入力されていることを確認
17. フォルダ「/」の左にチェックがついていることを確認 (ついていなければチェックボックスをクリック)
18. 「警告を出さずに既存リソースを上書き」にチェックがついていることを確認 (上書きしたくないファイルがある場合はチェックを外す)
19. 「終了 (F)」をクリック

テストドライバの導入に成功すると

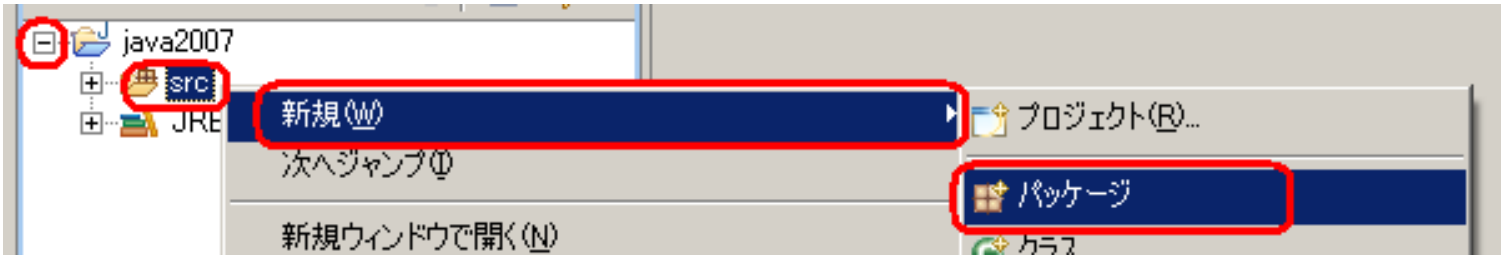
- プロジェクト「java20XX」の中の「test」というフォルダに「 j1.lesson02.xml 」という名前のファイルが作成される。
- このファイルには今週使用するテスト一式が記述されている。

パッケージ

- 前回は j1.lesson01 というパッケージを作成した。演習で書いたプログラムは、この「入れ物」に入っているはず。
- 今回は j1.lesson02 というパッケージを作成する。
- パッケージの作成は毎回の演習の最初に行うことを原則とする。

パッケージの作成

1. 「java20XX」プロジェクトの左側にある「+」をクリック
2. ツリーが展開されるので、「src」の上で右クリック
3. マウスカーソルを「新規」に合わせる
4. 「パッケージ」をクリック

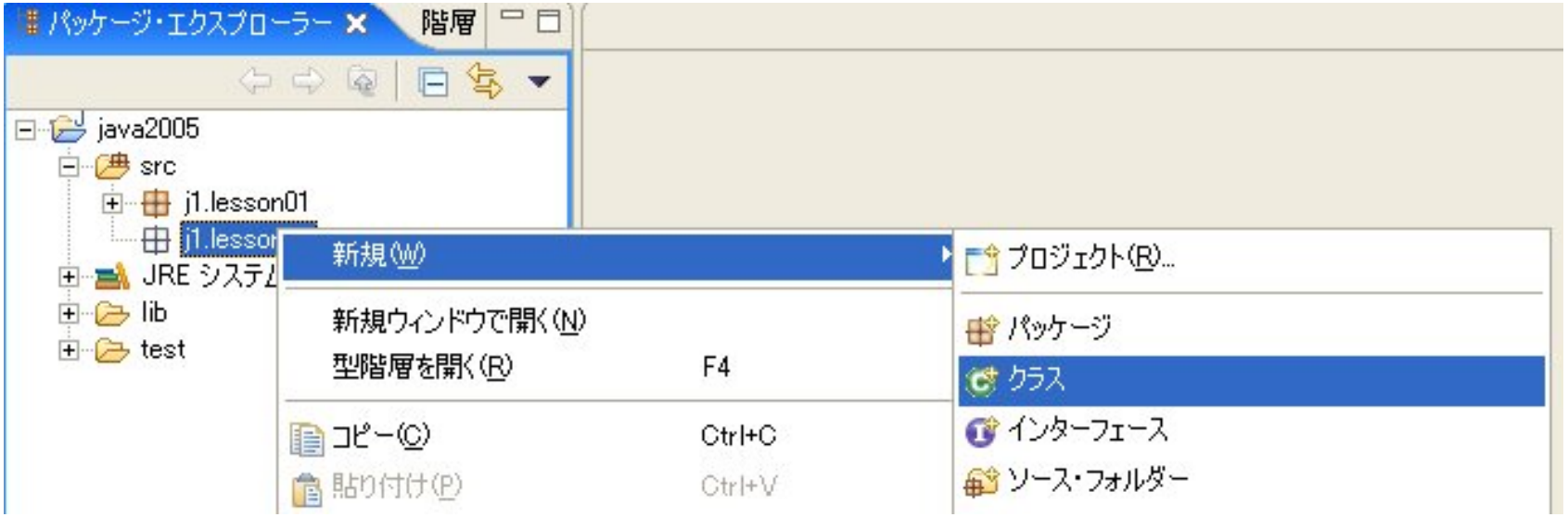


パッケージ名の入力

1. 「新規 Java パッケージ」というウィンドウが開くので、
2. 「名前」欄に**半角文字** で
3. 「j1.lesson02」と入力する 。
4. 正しく入力できたことを確認したら、右下の「終了」ボタンをクリックして確定する。

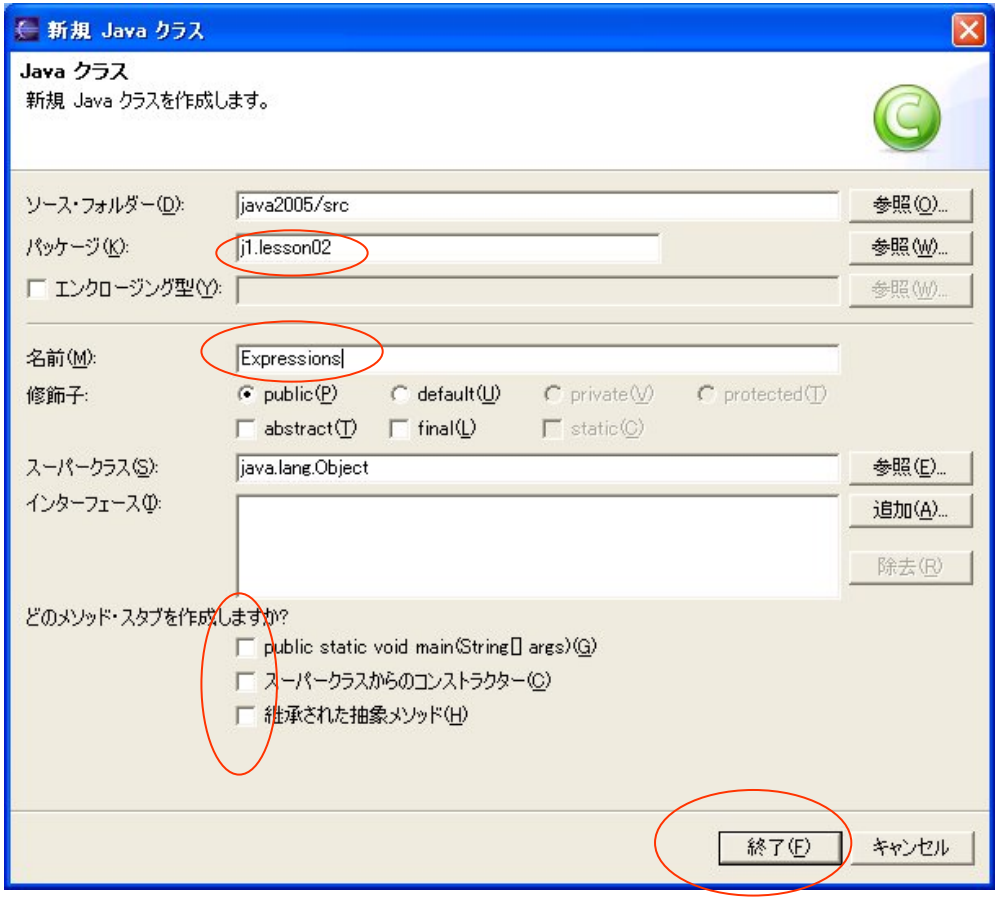
クラスの作成
クラス = プログラム　と思ってよい

1. 先ほど作成したパッケージ「j1.lesson02」の上で右クリック
2. マウスカーソルを「新規」に合わせる
3. 「クラス」をクリック

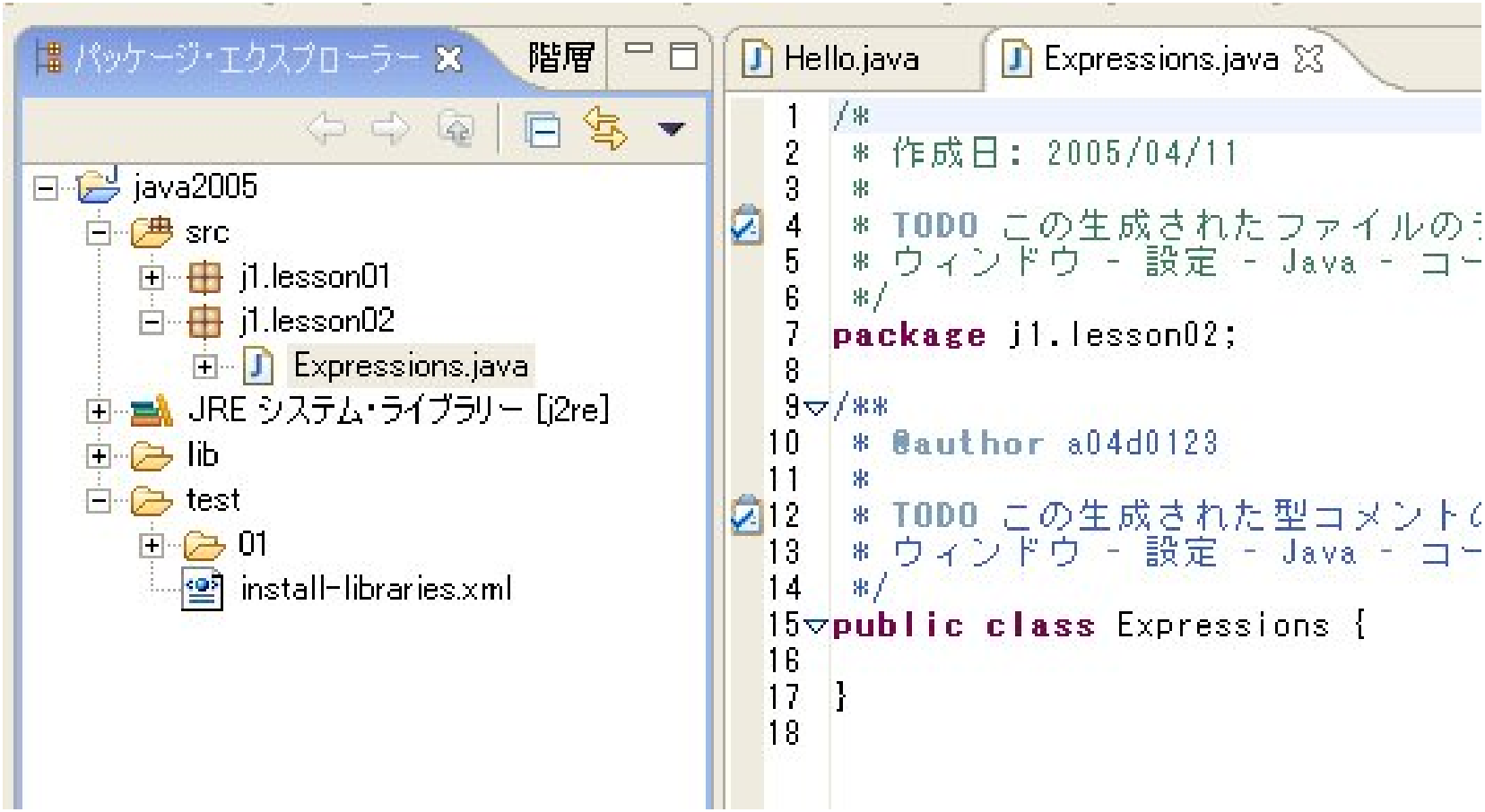


新規クラスの名前などを設定
クラス名 = プログラム名　と思ってよい

1. 新規 Java クラス」というウィンドウが開くので、
2. 「パッケージ(K)」が先ほど入力した「j1.lesson02」になっていることを確認すること。間違っていた場合は、ここに「j1.lesson02」と入力する。
3. 「名前(M)」に「Expressions」と入力する。
4. 「どのメソッド・スタブを作成するか?」という項目では、全てのチェックがはずれていることを確認。
5. 最後に、全ての項目を確認した後、右下の「終了」ボタンをクリックして確定する。



このような画面が現れるはず



実際のソースファイル

- この教室でパッケージ j1.lesson02 に Experssionsクラスを作成すると

  U:¥Eclipse3.1¥Windows¥java20XX¥src¥j1¥lesson02¥Expressions.java

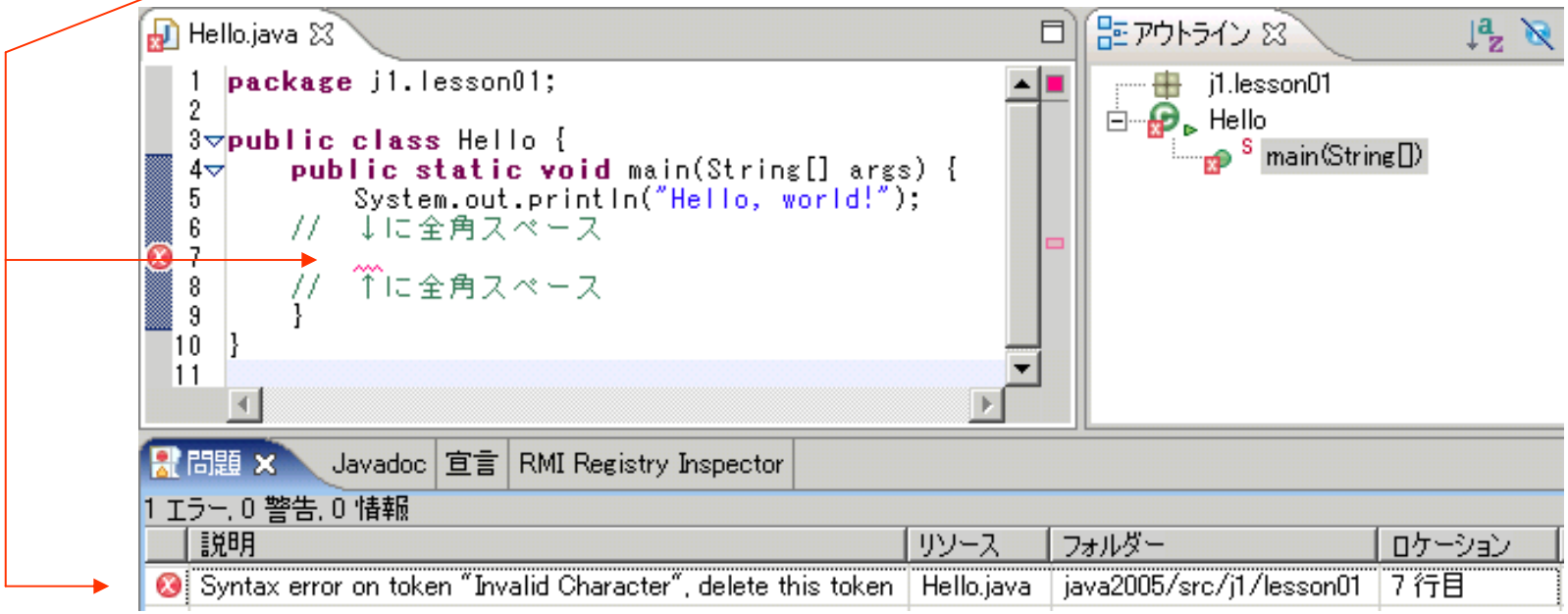
  というファイルができる

- Eclipseを使っていれば、あまり気にしないでもプログラムは書ける

注意　全角文字と半角文字の区別

- 全角文字(日本語文字)がプログラムの中で使えるのは、以下のところだけ
  - 引用符で囲まれた部分、つまり文字列のデータを表している部分
  - コメントの部分
- プログラムの地の部分、つまり引用符で囲まれた部分の外では半角文字(英字、数字)だけが許されている。
- 空白にも全角と半角があるので、上の注意があてはまる。
  - Eclipseではエラー表示が出る

```
System.out.println("こんにちは");
```

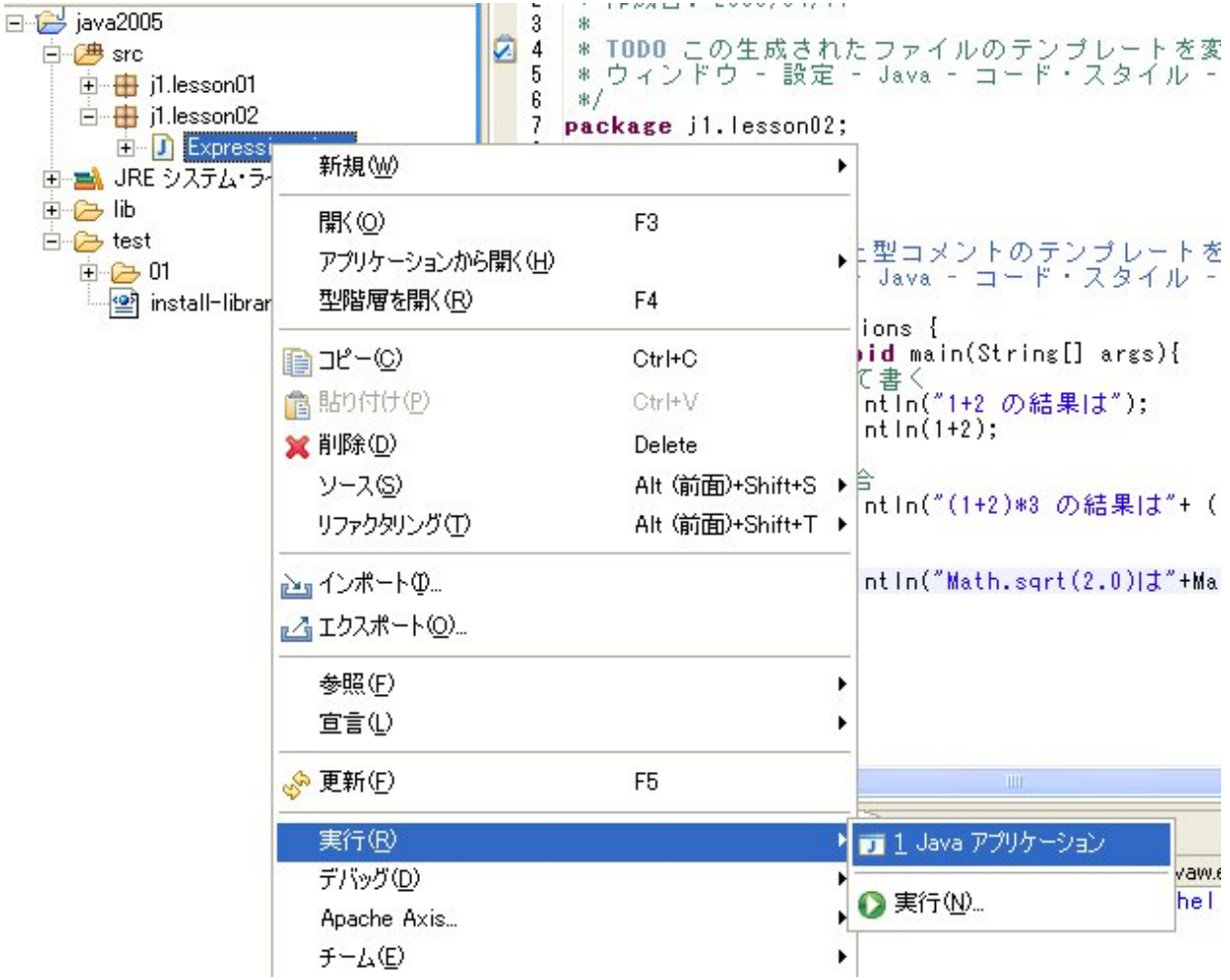


では以下のように書いてみよう
1行書く度に保存(自動コンパイル)　Ctrl+S

```
package j1.lesson02;

public class Expressions {
    public static void main(String[] args){
        // 2行に分けて書く
        System.out.println("1+2 の結果は");
        System.out.println(1+2);

        // 文字列の結合
        System.out.println("(1+2)*3 の結果は"+ ((1+2)*3) +"です");

        // Math.sqrt
        System.out.println("Math.sqrt(2.0)は"+Math.sqrt(2.0)+"です");
    }
}
```

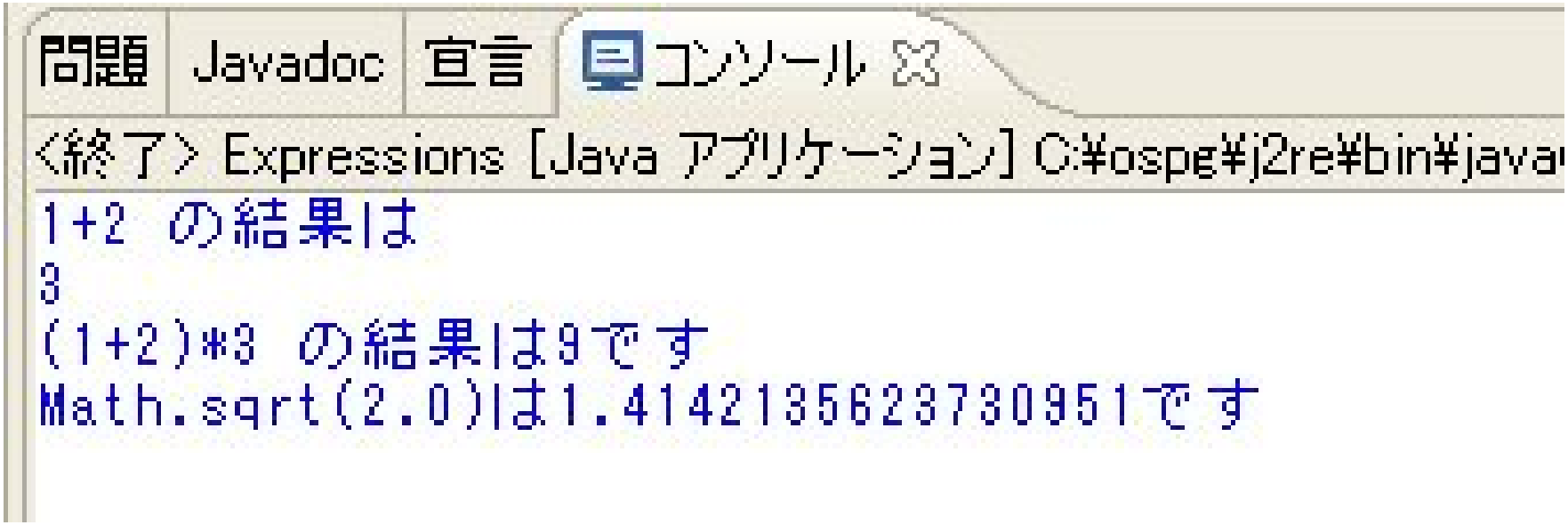
プログラムの実行

1. 「Expressions.java」の上で右クリック
2. メニューが表示されるので、「実行(R)」にマウスカーソルを合わせる
3. 「1 Java アプリケーション」をクリック



成功すると
右下の小さいエリアに

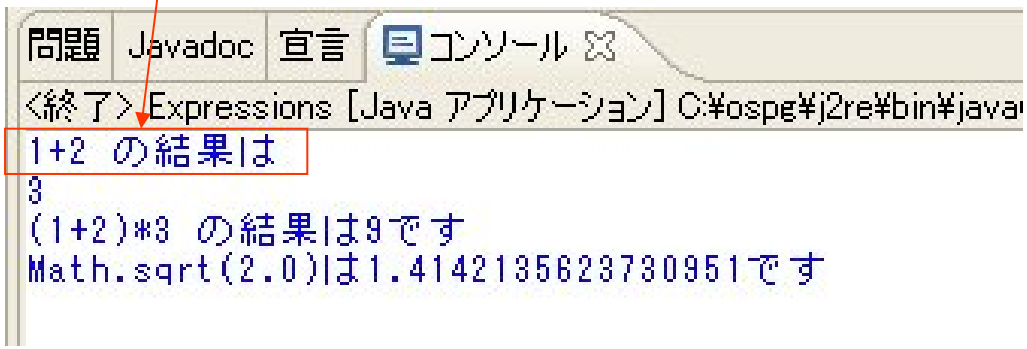
表示されない場合は右下の「コンソール」をクリックする



なぜこういう結果になるか

```
public class Expressions {
    public static void main(String[] args){
        // 2行に分けて書く
        System.out.println("1+2 の結果は");
        System.out.println(1+2);

        // 文字列の結合
        System.out.println("(1+2)*3 の結果は"+ ((1+2)*3) +"です");

        // Math.sqrt
        System.out.println("Math.sqrt(2.0)は"+Math.sqrt(2.0)+"です");
    }
}
```

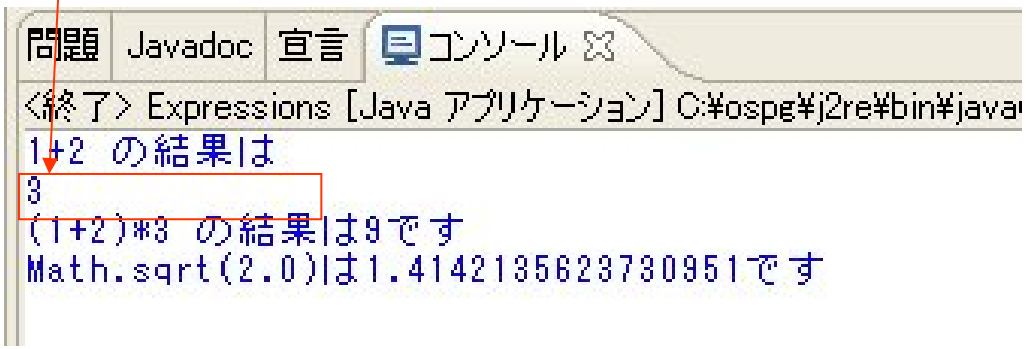
文字列そのまま



なぜこういう結果になるか

```
public class Expressions {
    public static void main(String[] args){
        // 2行に分けて書く
        System.out.println("1+2 の結果は");
        System.out.println(1+2);

        // 文字列の結合
        System.out.println("(1+2)*3 の結果は"+ ((1+2)*3) +"です");

        // Math.sqrt
        System.out.println("Math.sqrt(2.0)は"+Math.sqrt(2.0)+"です");
    }
}
```

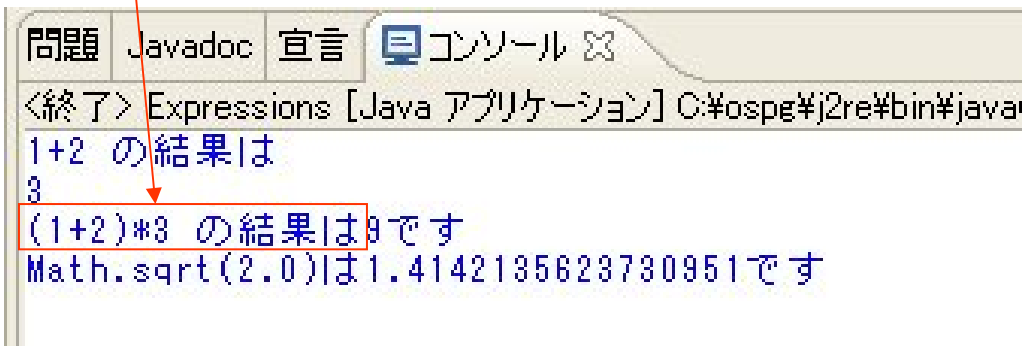
算術式の計算結果が文字列となる



なぜこういう結果になるか

```
public class Expressions {
    public static void main(String[] args){
        // 2行に分けて書く
        System.out.println("1+2 の結果は");
        System.out.println(1+2);

        // 文字列の結合
        System.out.println("(1+2)*3 の結果は"+ ((1+2)*3) +"です");

        // Math.sqrt
        System.out.println("Math.sqrt(2.0)は"+Math.sqrt(2.0)+"です");
    }
}
```

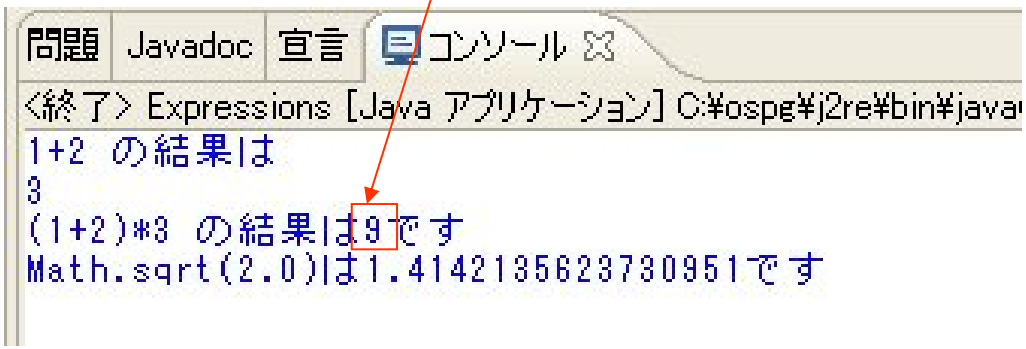
文字列そのまま



なぜこういう結果になるか

```
public class Expressions {
    public static void main(String[] args){
        // 2行に分けて書く
        System.out.println("1+2 の結果は");
        System.out.println(1+2);

        // 文字列の結合
        System.out.println("(1+2)*3 の結果は"+((1+2)*3)+"です");

        // Math.sqrt
        System.out.println("Math.sqrt(2.0)は"+Math.sqrt(2.0)+"です");
    }
}
```

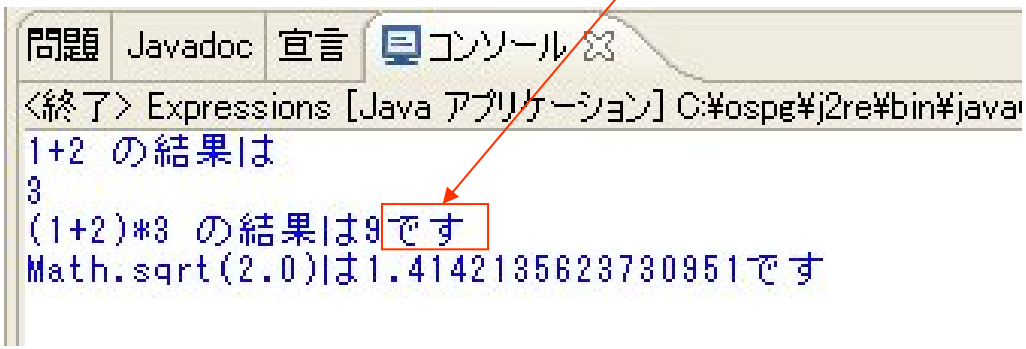
算術式の計算結果が文字列となる



なぜこういう結果になるか

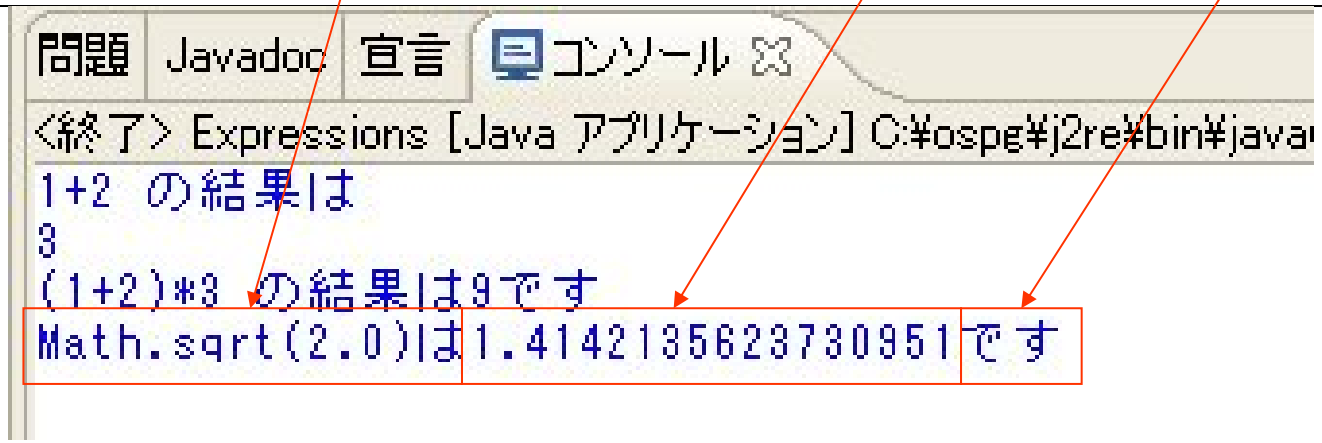
```
public class Expressions {
    public static void main(String[] args){
        // 2行に分けて書く
        System.out.println("1+2 の結果は");
        System.out.println(1+2);

        // 文字列の結合
        System.out.println("(1+2)*3 の結果は"+ ((1+2)*3) +"です");

        // Math.sqrt
        System.out.println("Math.sqrt(2.0)は"+Math.sqrt(2.0)+"です");
    }
}
```

文字列そのまま



なぜこういう結果になるか

```
public class Expressions {
    public static void main(String[] args){
        // 2行に分けて書く
        System.out.println("1+2 の結果は");
        System.out.println(1+2);

        // 文字列の結合
        System.out.println("(1+2)*3 の結果は"+ ((1+2)*3) +"です");

        // Math.sqrt
        System.out.println("Math.sqrt(2.0)は"+Math.sqrt(2.0)+"です");
    }
}
```



ではもうひとつ練習

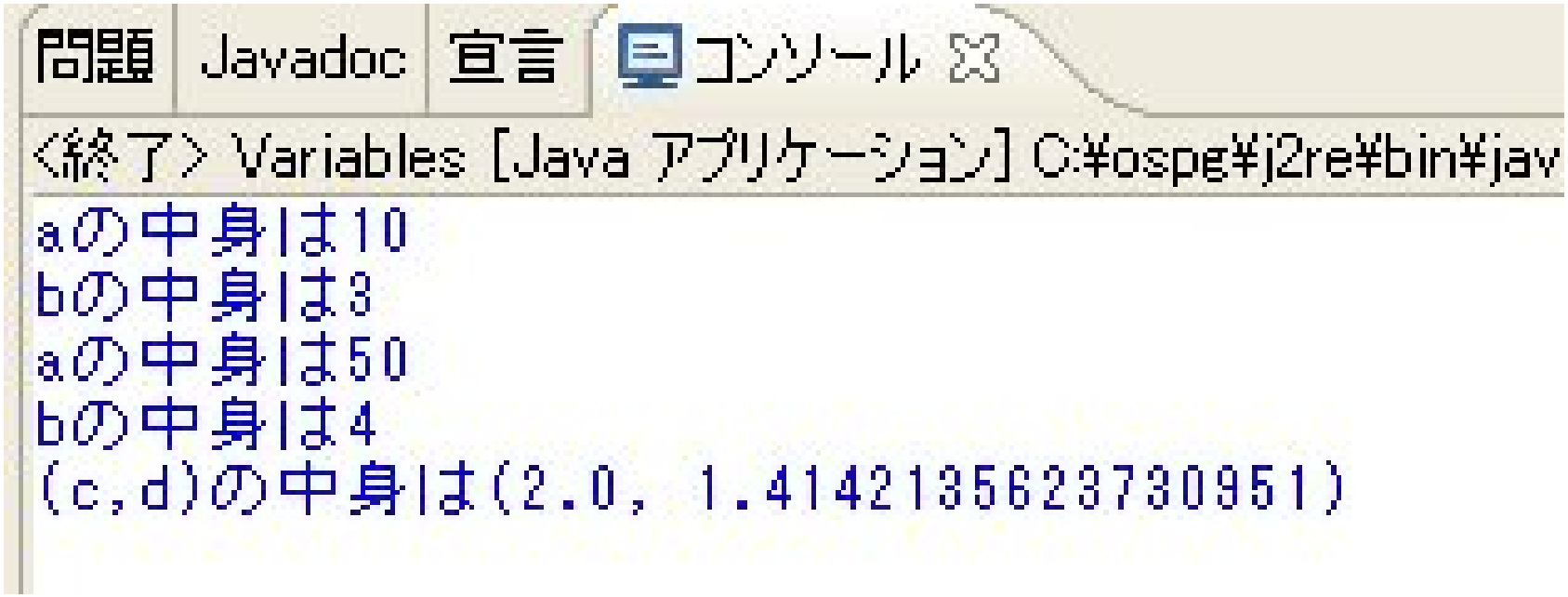
1. 先ほど作成したパッケージ「j1.lesson02」の上で右クリック
2. マウスカーソルを「新規」に合わせる
3. 「クラス」をクリック
4. クラス名は Variables とする
5. 次のスライドに ある命令文を main メソッドの中に書く
6. そして先と同様に実行してみよ

注意　// は、以下行末までコメントであることを示す。

main メソッドの中身

```
int a, b;     // int型の変数の宣言
a = 10;      // 初期化
System.out.println("aの中身は" + a);

b = a / 3;  // 代入
System.out.println("bの中身は" + b);

a *= 5;      // 代入その2
System.out.println("aの中身は" + a);

b++;         // 自動インクリメント
System.out.println("bの中身は" + b);

// double型変数
double c = 2.0;
double d = Math.sqrt(c);
System.out.println("(c,d)の中身は(" + c + ", " + d + ")");
```

実行結果

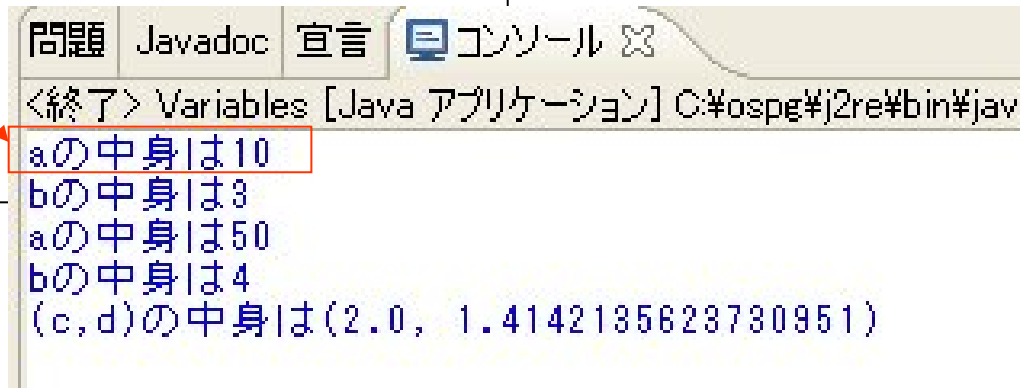

問題
Javadoc
宣言
コンソール
<終了> Variables [Java アプリケーション] C:¥ospe¥j2re¥bin¥jav
aの中身は10
bの中身は3
aの中身は50
bの中身は4
(c,d)の中身は(2.0, 1.4142135623730951)

うまく行きましたか

- なぜ、このような結果になるのか丁寧に解説します。

出力の1行目

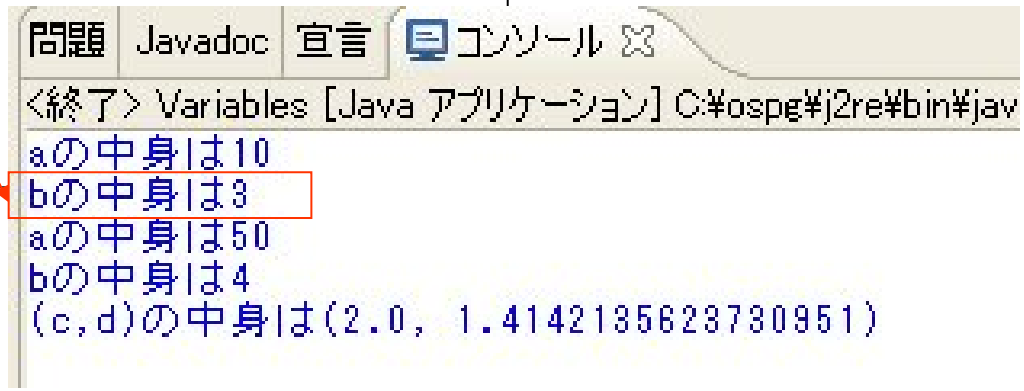
```
int a, b;    // int型の変数の宣言
a = 10;      // 初期化
System.out.println("aの中身は" + a);

b = a / 3;  // 代入
System.out.println("bの中身は" + b);

a *= 5;     // 代入その2
System.out.println("aの中身は" + a);

b++;        // 自動インクリメント
System.out.println("bの中身は" + b);

// double型変数
double c = 2.0;
double d = Math.sqrt(c);
System.out.println("(c,d)の中身は(" + c
```



出力の2行目

```
int a, b; // int型の変数の宣言
a = 10;      // 初期化
System.out.println("aの中身は" + a);

b = a / 3;  // 代入
System.out.println("bの中身は" + b);

a *= 5;      // 代入その2
System.out.println("aの中身は" + a);

b++;         // 自動インクリメント
System.out.println("bの中身は" + b);

// double型変数
double c = 2.0;
double d = Math.sqrt(c);
System.out.println("(c,d)の中身は(" + c
```

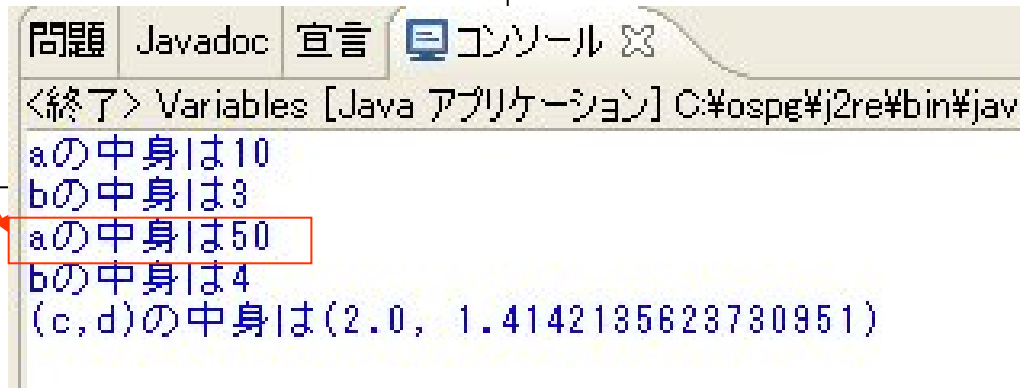
現在 a には10が入っている



出力の3行目

```
int a, b; // int型の変数の宣言
a = 10;      // 初期化
System.out.println("aの中身は" + a);

b = a / 3;  // 代入
System.out.println("bの中身は" + b);

a *= 5;      // 代入その2
System.out.println("aの中身は" + a);

b++;         // 自動インクリメント
System.out.println("bの中身は" + b);

// double型変数
double c = 2.0;
double d = Math.sqrt(c);
System.out.println("(c,d)の中身は(" + c
```

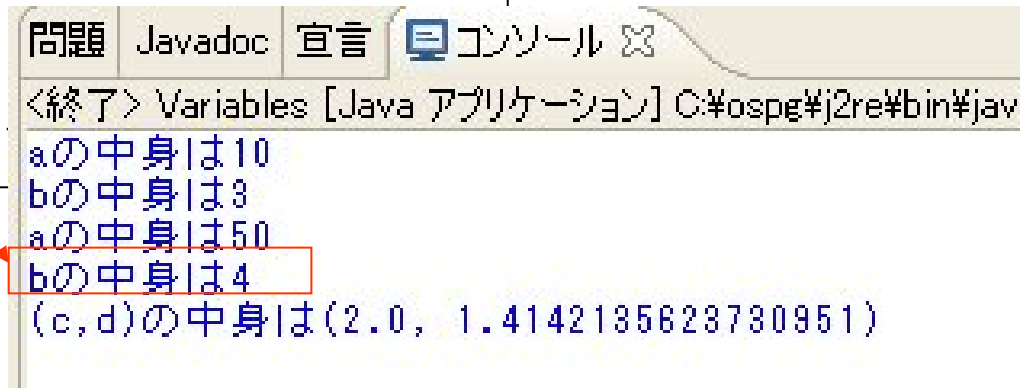
先ほどまで a には10が入っていた



出力の4行目

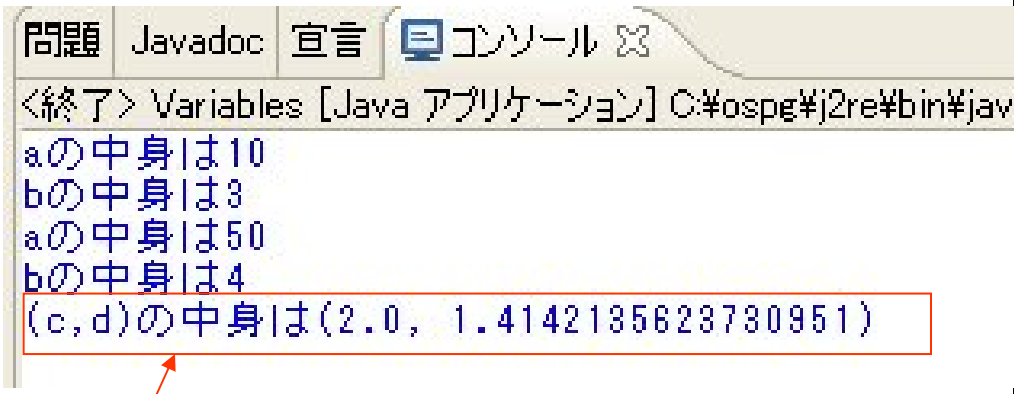
```
int a, b; // int型の変数の宣言
a = 10;      // 初期化
System.out.println("aの中身は" + a);


b = a / 3;  // 代入
System.out.println("bの中身は" + b);

a *= 5;      // 代入その2
System.out.println("aの中身は" + a);

b++;            // 自動インクリメント
System.out.println("bの中身は" + b);

// double型変数
double c = 2.0;
double d = Math.sqrt(c);
System.out.println("(c,d)の中身は(" + c
```

先ほどまで b には3が入っていた



出力の5行目は少しややこしい

```
int a, b; // int型の変数の宣言
a = 10;      // 初期化
System.out.println("aの中身は" + a);


b = a / 3;  // 代入
System.out.println("bの中身は" + b);

a *= 5;      // 代入その2
System.out.println("aの中身は" + a);

b++;         // 自動インクリメント
System.out.println("bの中身は" + b);
```




```
// double型変数
double c = 2.0;
double d = Math.sqrt(c);
System.out.println("(c,d)の中身は(" + c + ", " + d + ")");
```

(c,d)の中身は(	2.0	,	1.4142135623730951	)

課題

各自のペースで
「第02週目の課題」をやってみよう

1. http://java.cis.k.hosei.ac.jp/ をブラウザで開く
2. 第02回 - Expressions, Types and Variables - をクリック
3. 第02週目課題をクリック

第02回目終了

- 課題を時間内あるいは本日中にやり切れなかった人は
- 次回までに自習してください。
- 質問は随時歓迎