月に吠える
ーeclipseによるＪａｖａアプリケーションの作成ー

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4月１７日

０．１版　２００５年2月１４日：一部をeclipse3.01向けに修正

スライドの構成

はじめに

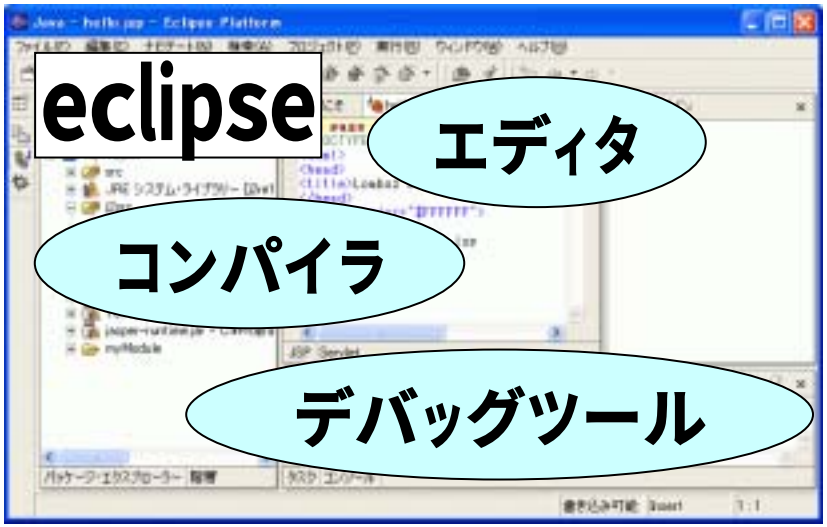
- 本スライドは、eclipseを用いてＪａｖａアプリケーションを作成する手順について記します。
- スライド（1）の「eclipseの立ち上げ方」以外は、Ｌｉｎｕｘ環境・Windows環境とも同じになります。
- 本スライドでは、Javaプログラムを作成する上で必要な最低限の機能に絞って説明しています。
- 岐阜経済大学では、第２演習室のLinuxPC、および全WindowsＰＣに、eclipseはインストールされています。本スライドでは、これらでの環境での操作を記します。
- なお、2005年度よりWindowsPCへは、eclipse3.01をインストールしています。

(1) eclipse

■eclipseは、Ｊａｖａに対応したＩＤＥ(Interated Development Environment)ソフトウェアです。

■ＩＤＥとは、コンパイラやエディタ、デバッグツールなど、プログラムの開発に必要なツールを統合したものです。

■eclipseを使うことで、Javaプログラム開発の一連の作業を、グラフィカルな分かりやすいインタフェースを通じて簡便に行うことが出来ます。

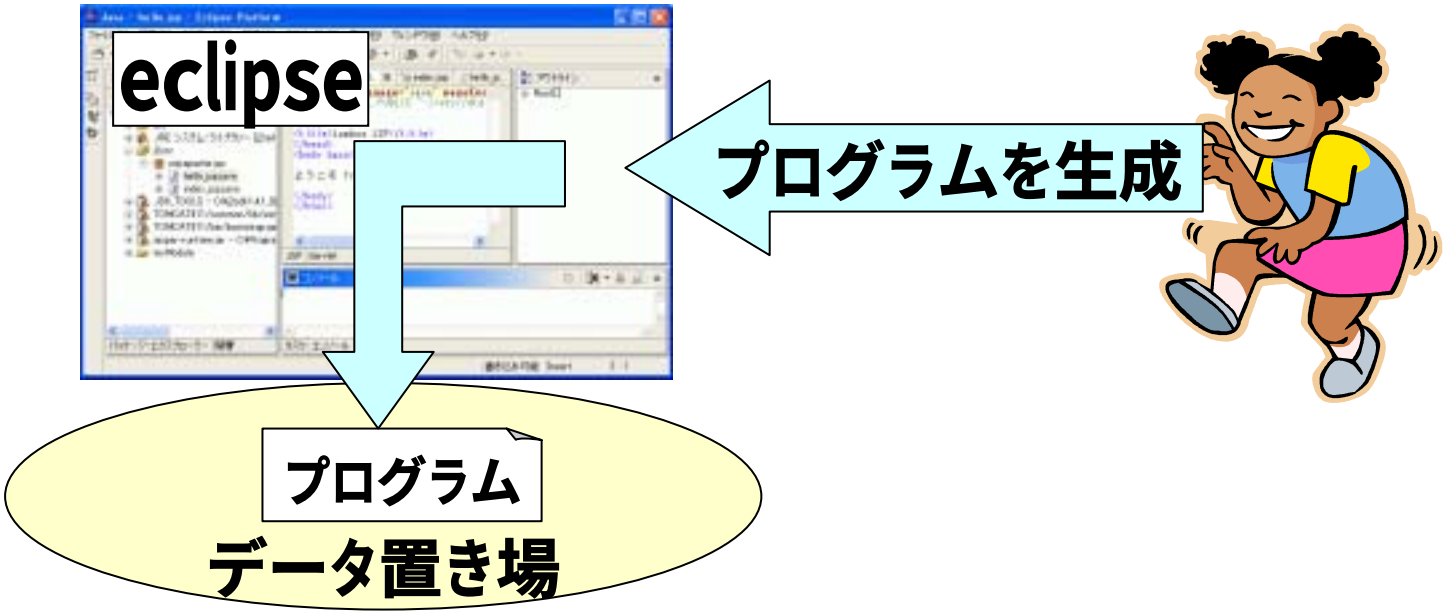


全部まとめて、わかりやすく、簡便な操作により、Ｊａｖａプログラムを作る！

(1．1) データの置き場所

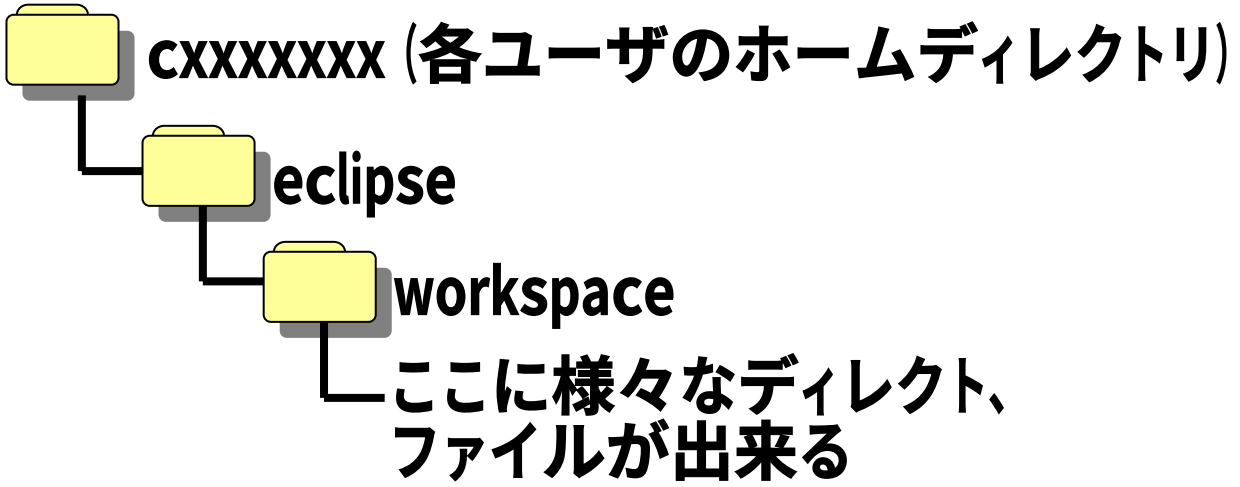
■eclipseでＪａｖａプログラムを開発する際には、eclipse内でＪａｖａファイルを管理します (すなわち、作成したり変更したりします)。

■このため、eclipseがプログラムを保持するためのデータの置き場所を決める必要があります (実際にはプログラム以外のデータも置かれます)。



(1．2) Linux PCでのデータ置き場

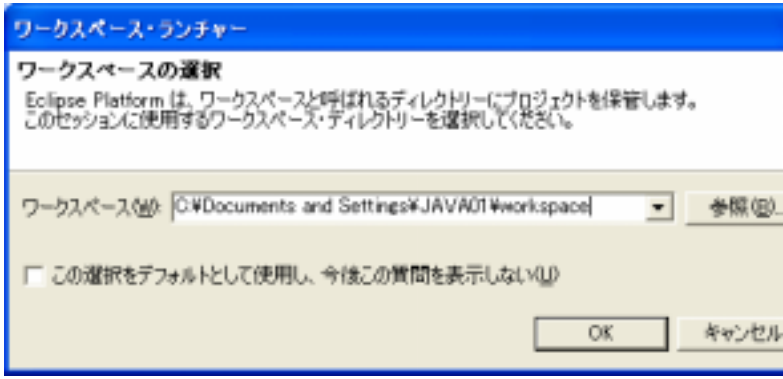
■Linux PCでのデータ置き場は、デフォルトで各ユーザのホームディレクトリ配下“eclipse/workspace”となるように設定されています。すなわち、eclipseを立ち上げると次のようなデータが出来ます。



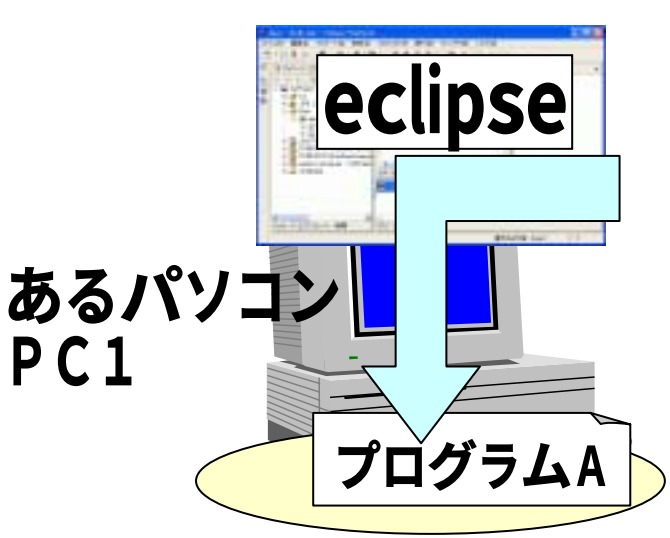
■各ユーザのホームディレクトリは、どのLinux PCからも同じように見えます。よって、Linux PCでは、ターミナルから“eclipse”と投入することで、毎回同じデータ置き場で動くeclipseを立ち上げることが出来ます。

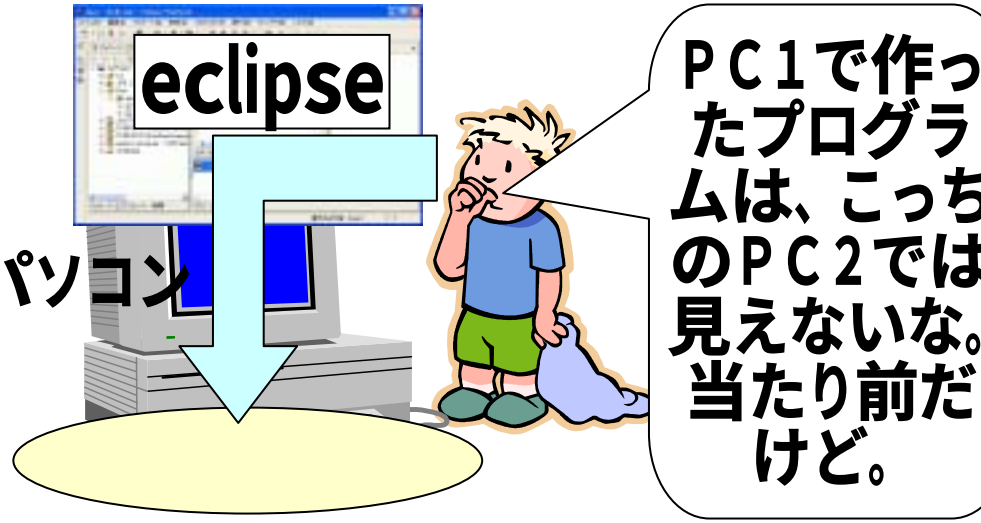
（1．3）Windowsでのデータ置き場

■Windowsでのデータ置き場は、eclipse立ち上げ時にダイアログにて尋ねてきます。



■ここでローカルのディレクトリを指定すると、異なるＰＣでは異なるデータ置き場からeclipseを立ち上げることになり、うまくありません。





(1．4) データ置き場を指定する (Windows)

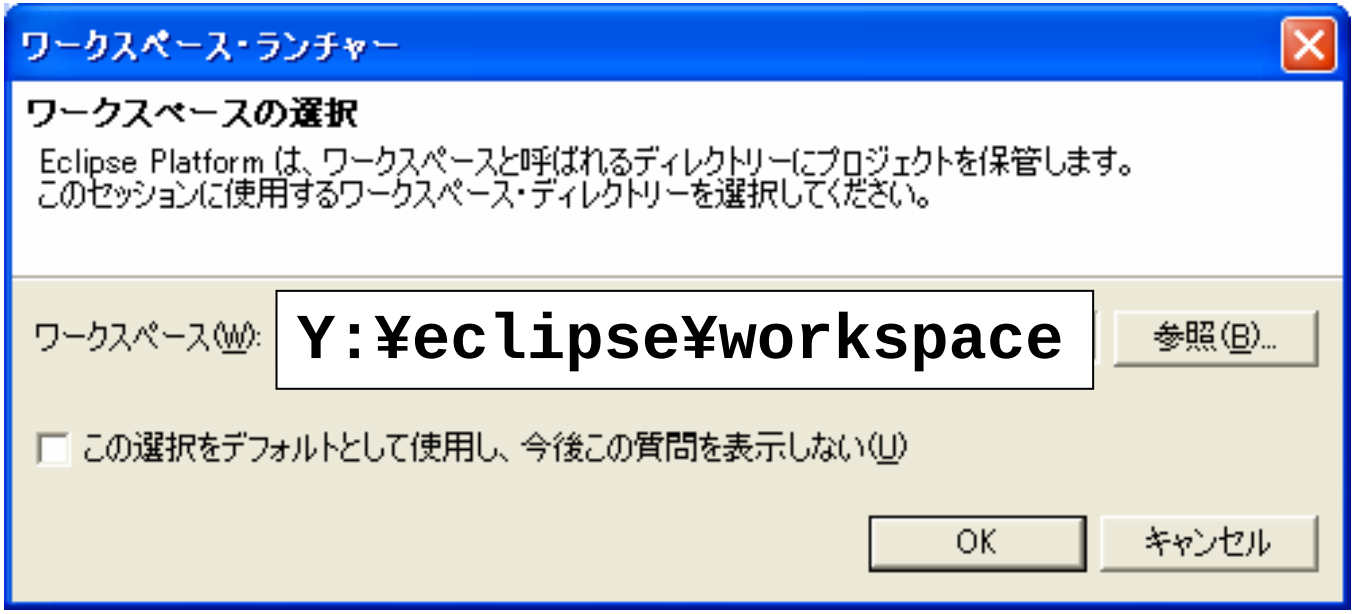
■スライド(1.3)のような状況では不都合なので、eclipseの起動時に、データ置き場を指定することにします。

■指定するデータ置き場は、どのＰＣからも見ることが出来るところでなければなりません。よって、次の2つの方法があります。

- 可般蓄積媒体を用いる。
  - フロッピーディスク (苦しい使い方になります) 、ＵＳＢメモリ、などを用います。
- 個人のネットワークドライブを用いる。
  - ネットワークドライブの利用の仕方ついては、次の資料を参考にしてください。
    中央フリーウェイ －個人のネットワークドライブ－
    (http://www.gifu-keizai.ac.jp/~ido/doc/literacy_text/net_drive_lit_s.pdf)

（1．5）eclipseの立ち上げ(Windows)

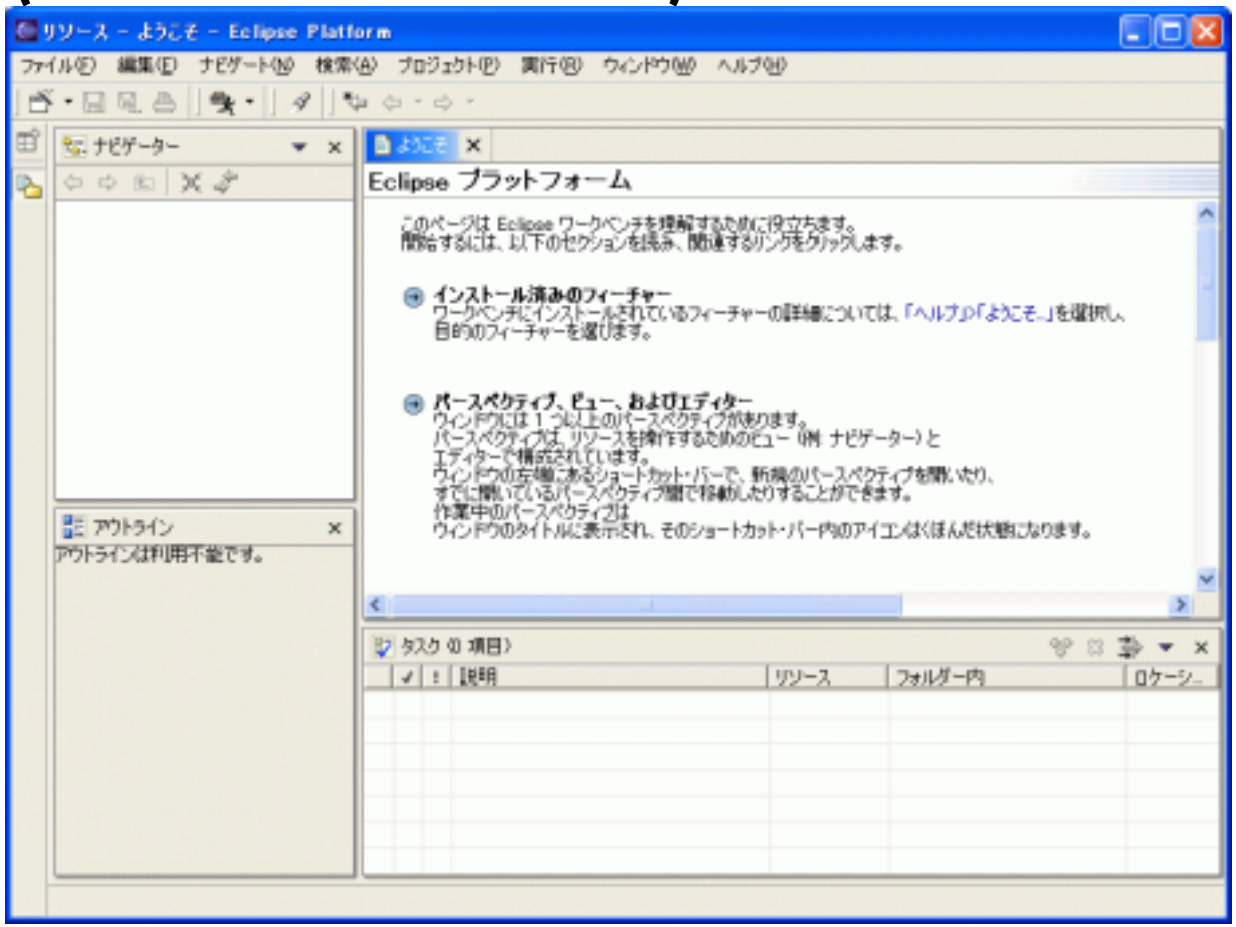
■ネットワークドライブが、ドライブレター“Y：”に割り当てられたとして、次のようにワークスペースを選択することとします。



■“\eclipse\workspace”という置き場所の名前は、他の名前でもＯＫです。しかしながら、上記の名前で統一しておくこととしたく、よろしくお願いします。

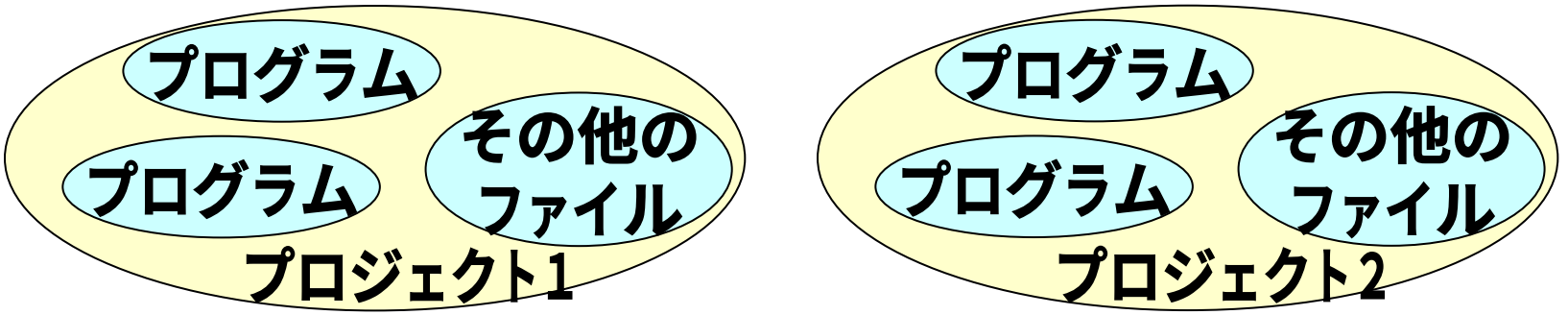
(1．6) 初回立ち上げ時のeclipse、ワークベンチ

■初回に立ち上げた際のeclipseは、次のようになります。eclipseを使う際に表示されるウインドウのことを、ワークベンチ(Work Bench：作業台)と言います。



(2) プロジェクト

■eclipseでは、プロジェクトという単位の中にプログラム(Javaではクラス)を作成します。プロジェクトにはプログラム以外の様々なファイルも含まれます。

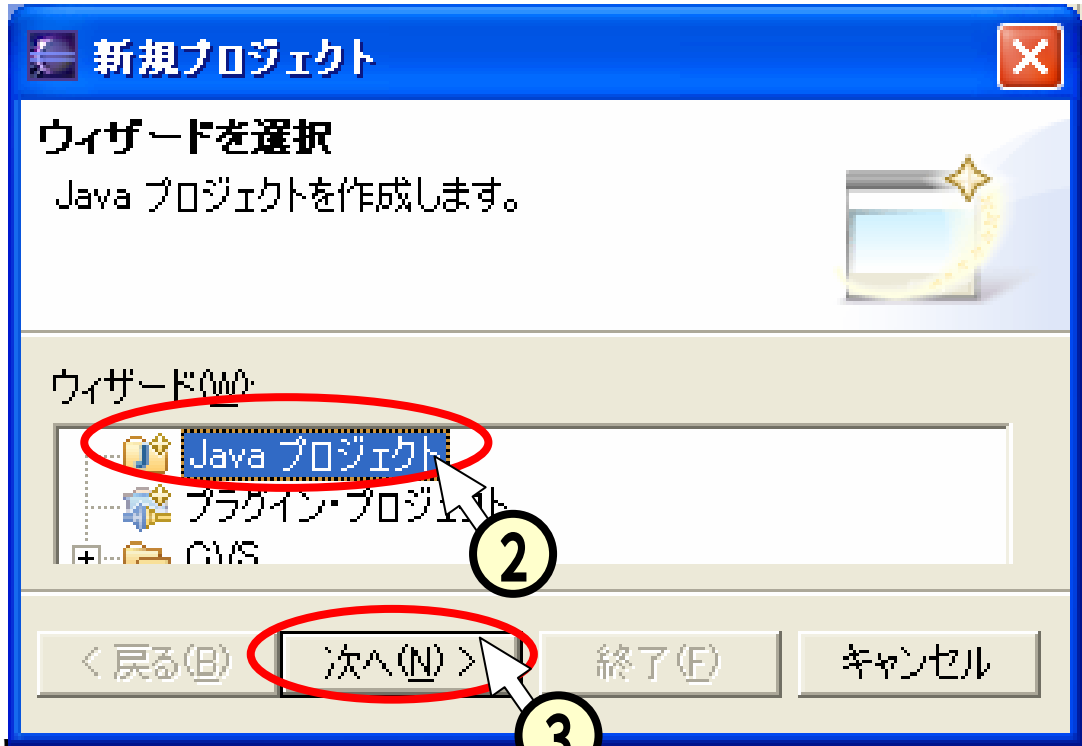


■ひとつのプロジェクトには、関連するプログラムを持つ複数のプログラムが含めることが普通ですが、関連がないプログラムを含めても問題があるわけではありません。これから“SampleProject”というプロジェクトを作成しますが、ここには雑多なプログラムを含めることにします。

(2．1．1) プロジェクトの作成　－1－

■[ファイル]-[新規]-[プロジェクト]をクリック (①) します。

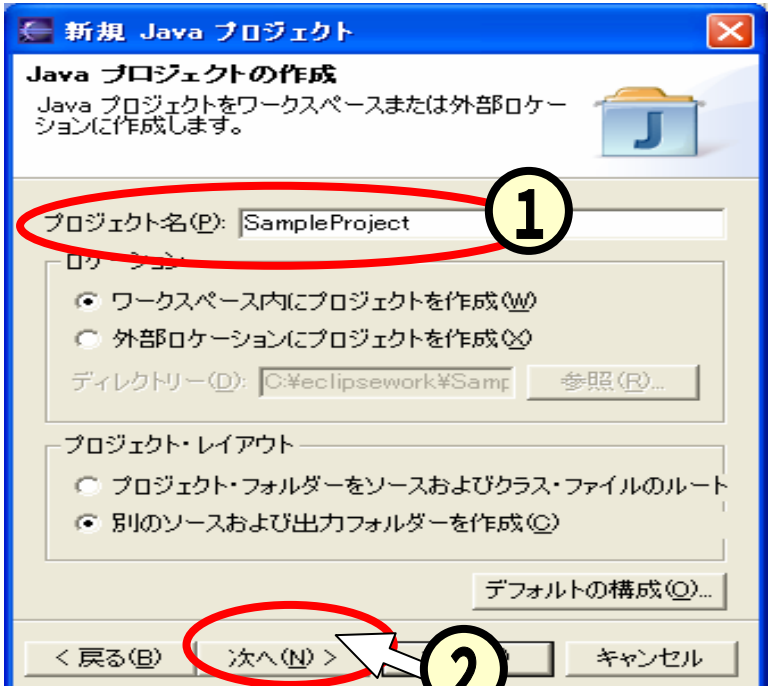
■「新規プロジェクト：選択」ウインドウ中、左ペインで[Java]をクリック (②) し、[次へ]をクリック (③) します。

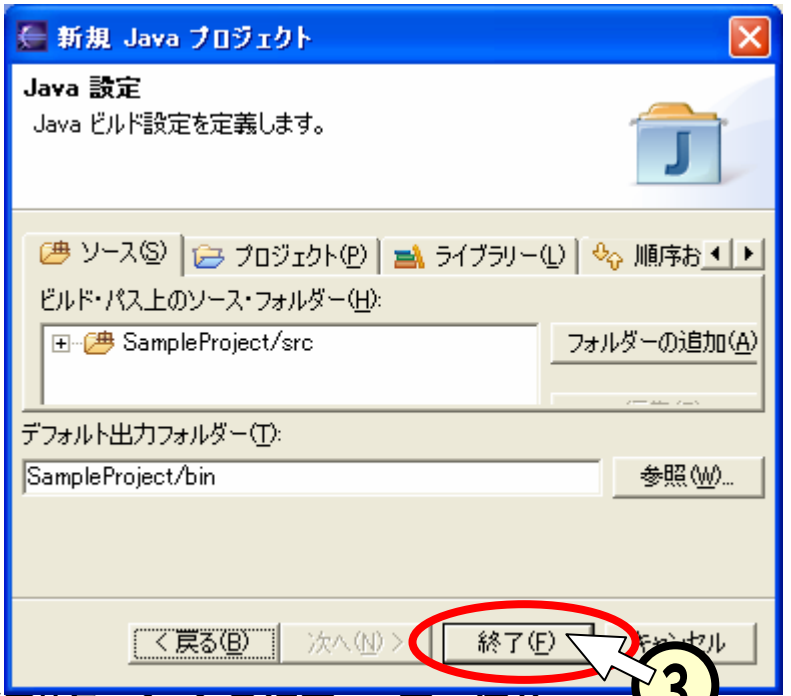


(2．1．2) プロジェクトの作成　－2－

■「新規Javaプロジェクト」ウインドウ中、プロジェクト名を入力 (①) する (右図では、“SampleProject”) 。[次へ]をクリック (②) する。

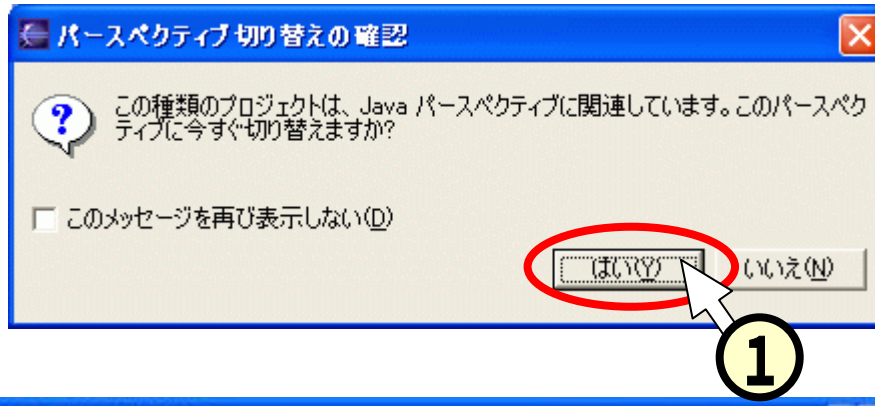
■ここでは標準の設定 (ビルド等) を用いるので、「新規Javaプロジェクト：Java設定」ウインドウでは、何も設定せず、[終了]をクリック (③) する。



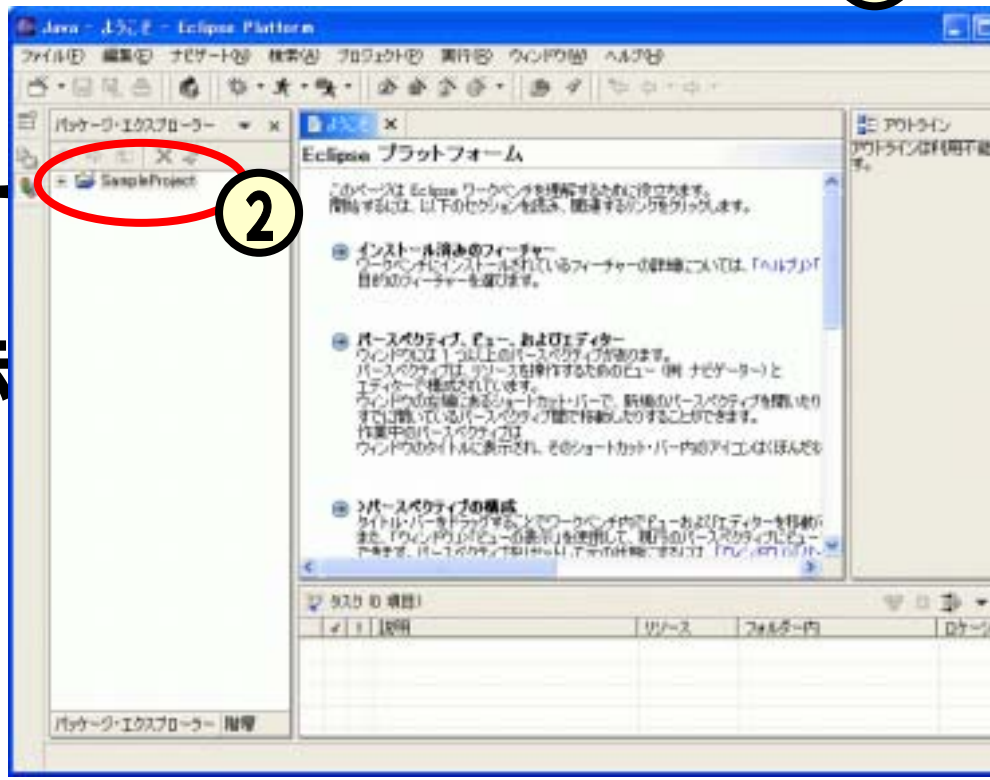


(2．1．3)プロジェクトの作成　－3－

■「パースペクティブ切り替え」ウインドウでは、Ｊａｖａパースペクティブに切り替えるという意味で、[はい]をクリック(①)する。



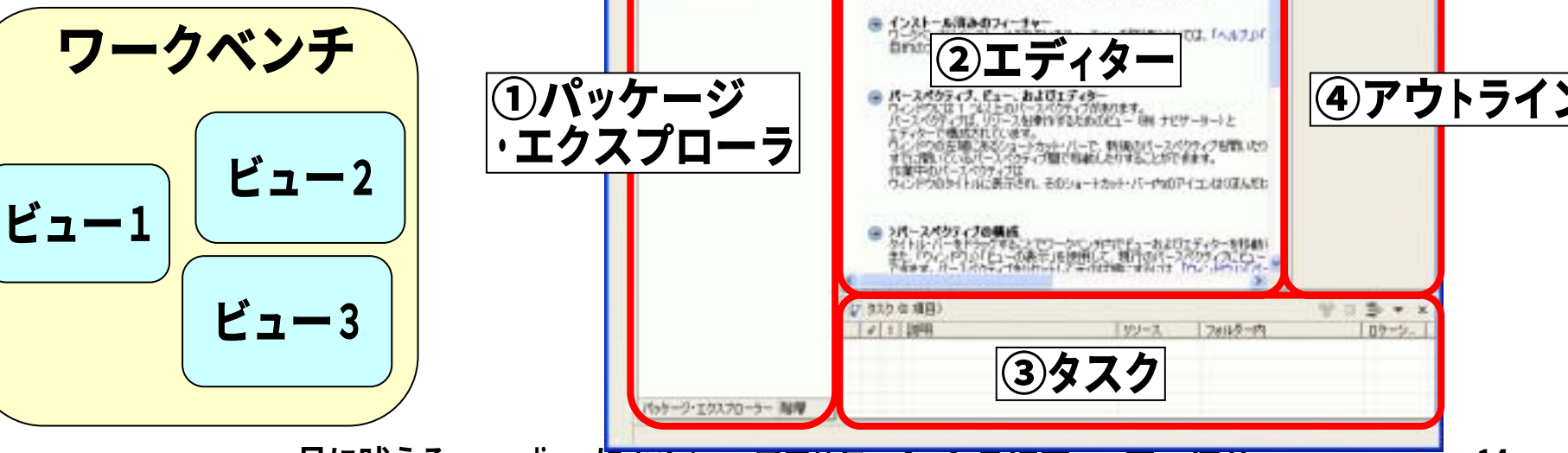
■パッケージエクスプローラ中に、“SampleProject”の表示(②)が見える、ワークベンチが現れます。



(2．2) ビュー

■eclipseを使う際に表示されるウインドウのことを、ワークベンチと言いましたが、ワークベンチの中の分割されたそれぞれの区画を、ビュー(view)と言います。

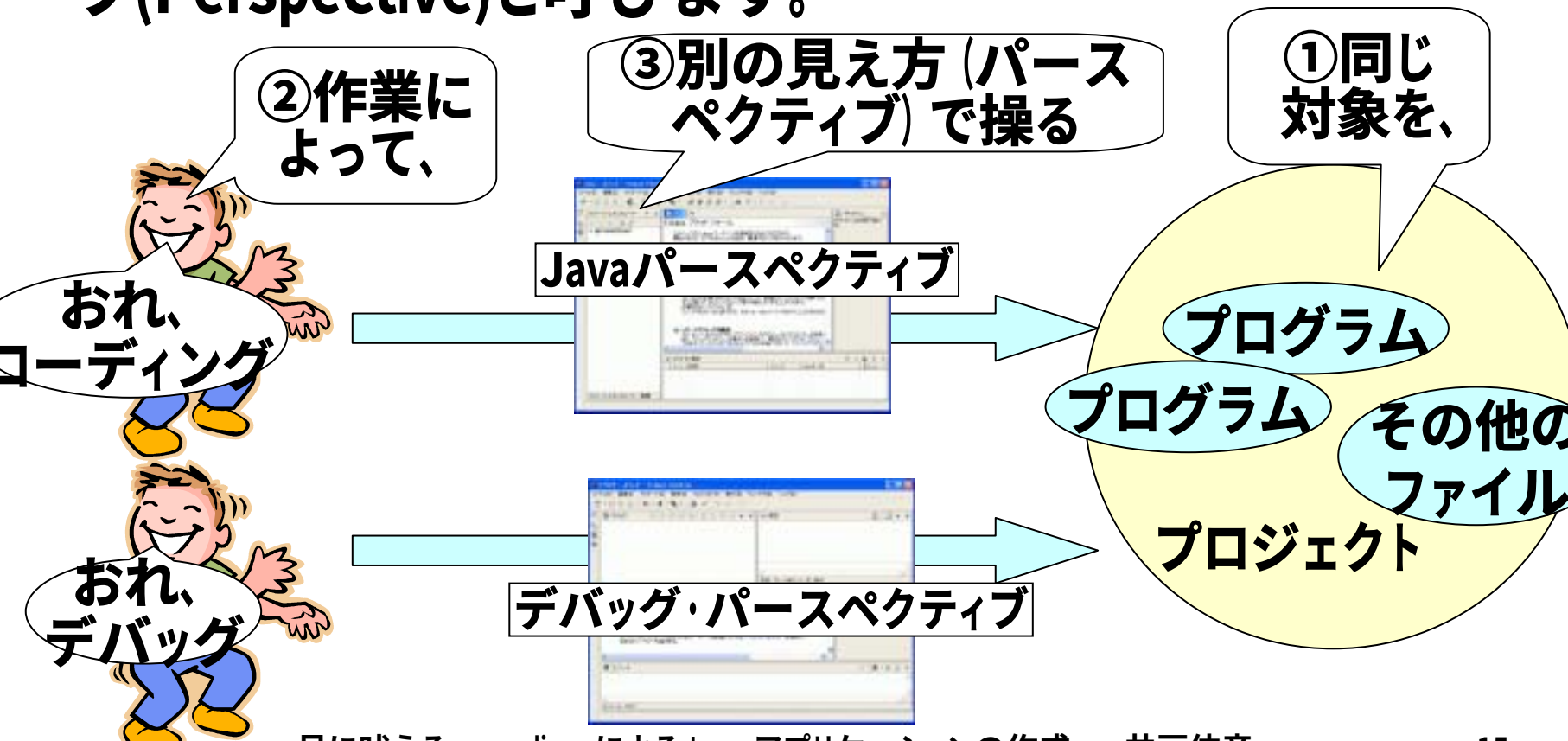
■例えば、下図のワークベンチでは、「①パッケージ・エクスプローラ」、「②エディター」、「③タスク」、「④アウトライン」の4つのビューがあります。



(2．3) パースペクティブ

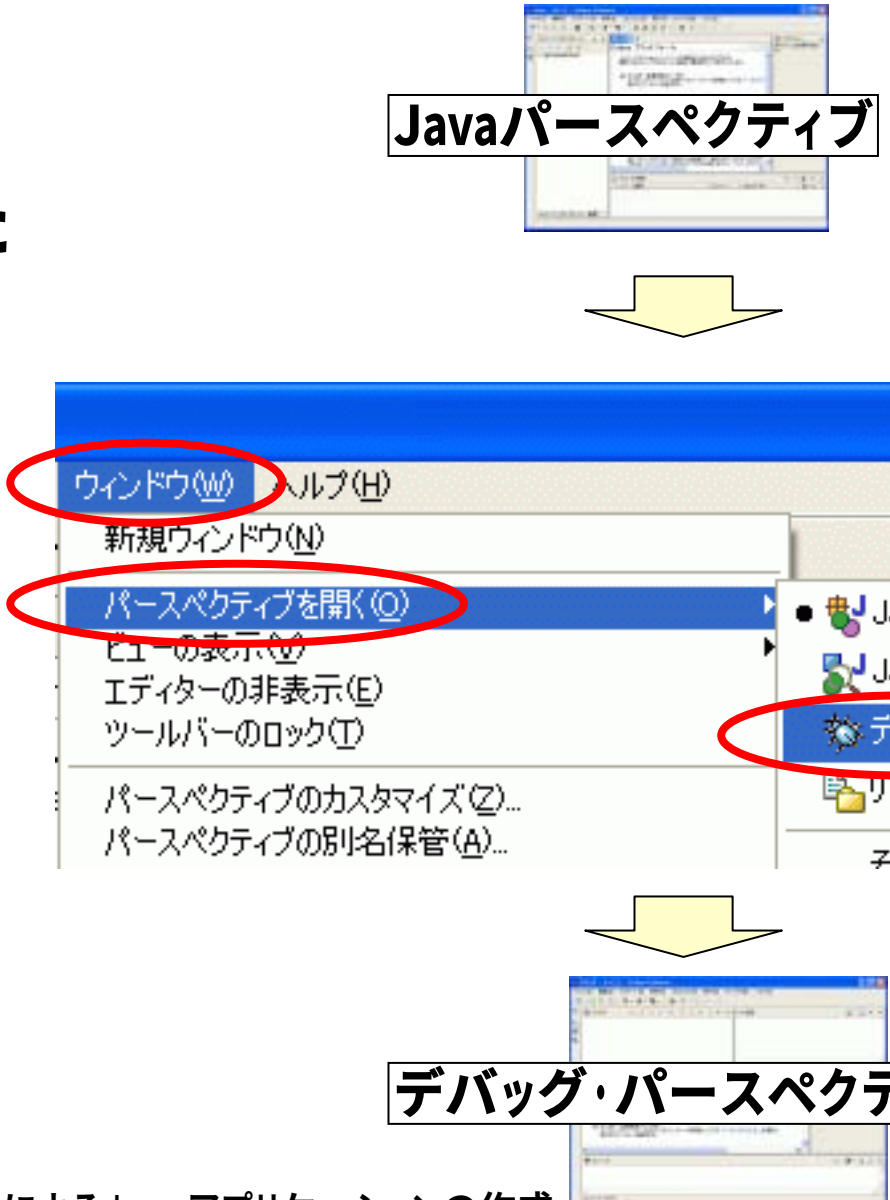
■ワークベンチの中のビューの構成は、一定ではなく、行う作業に都合の良いように、切り替わります。

■作業の都合に応じたビューの構成を、パースペクティブ(Perspective)と呼びます。



(2．4) パースペクティブの切り替え

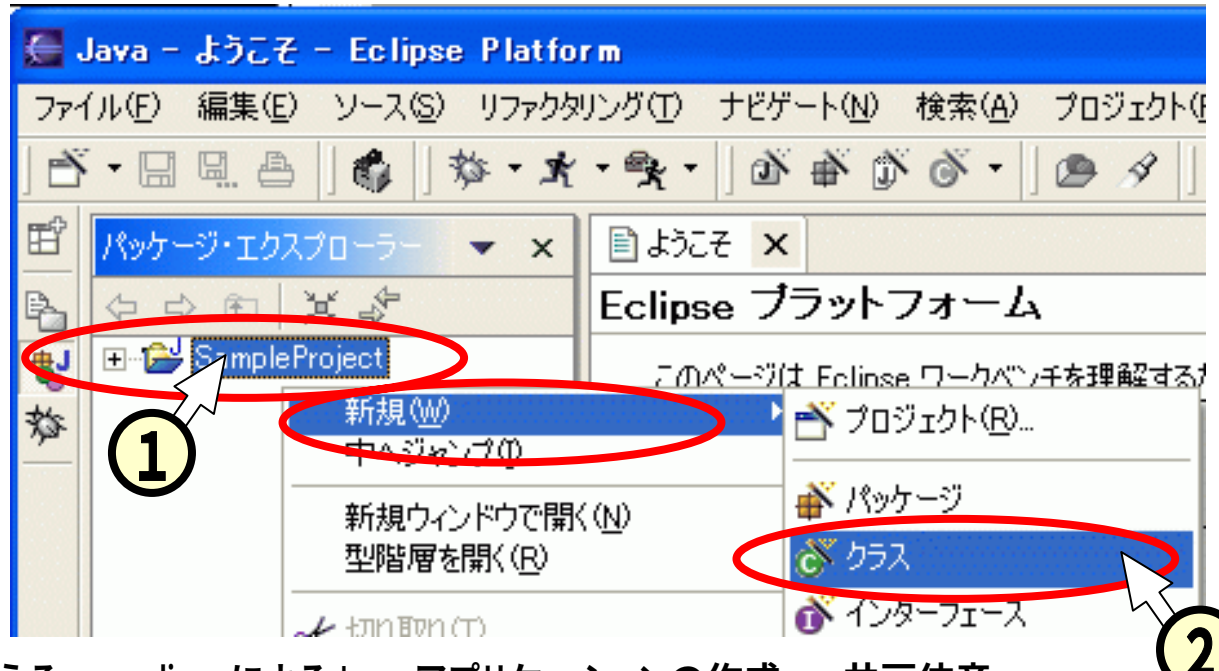
■[ウィンドウ]-[パースペクティブを開く]の先で、切り替えたい先のパースペクティブ (右図では、[デバッグ]) をクリック (①) ます。



(3．1．1)クラスの新規作成　－1－

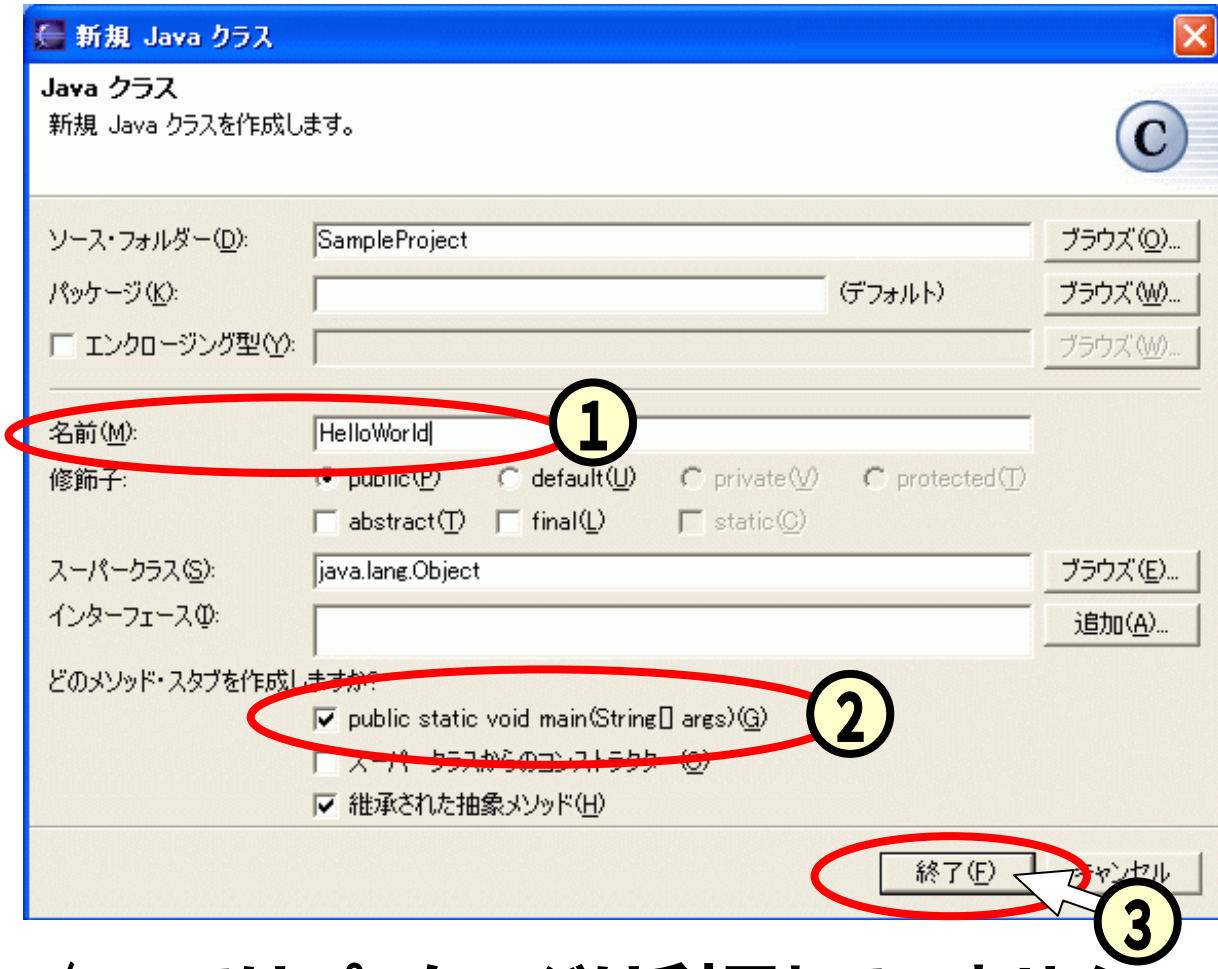
■スライド(2)で作成した“SampleProject”の中に、最初のプログラム“HelloWorld”クラスを作っていきます。

■Javaパースペクティブのパッケージエクスプローラ内に表示された、“SampleProject”を右クリック(①)し、現れたメニューから[新規]-[クラス]をクリックします(②)。



(3．1．2)クラスの新規作成　－2－

■「新規Javaクラス」ウィンドウにて、「名前」の欄に“HelloWorld”と入力(①)し、「どのメソッド・スタブを作成しますか」の[public static void main]にチェック(②)し、[終了]をクリック(③)します。

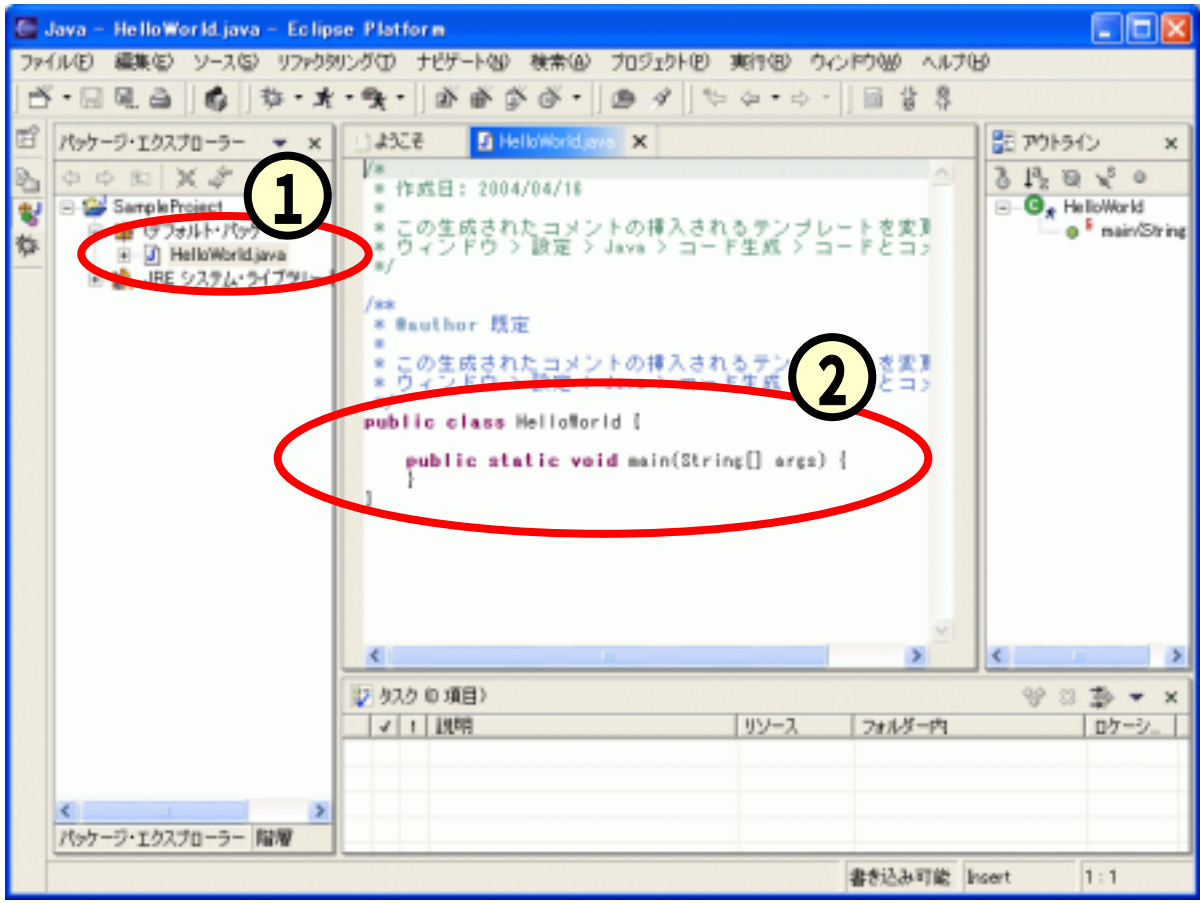


(ここではパッケージは利用していません。作成されるクラスはデフォルトパッケージに入ることになります。)

(3．1．3)クラスの新規作成　－3－

■パッケージエクスプローラ内に、作成された“HelloWorld.java”のファイルが見えます(①)。

■エディターでは、HelloWorld.javaのファイルが開いています。この中には、すでにメソッド“main”の雛形(②)が書かれています。前スライド(3.1.2)の②でチェックしたのは、この雛形を作るという意味です。

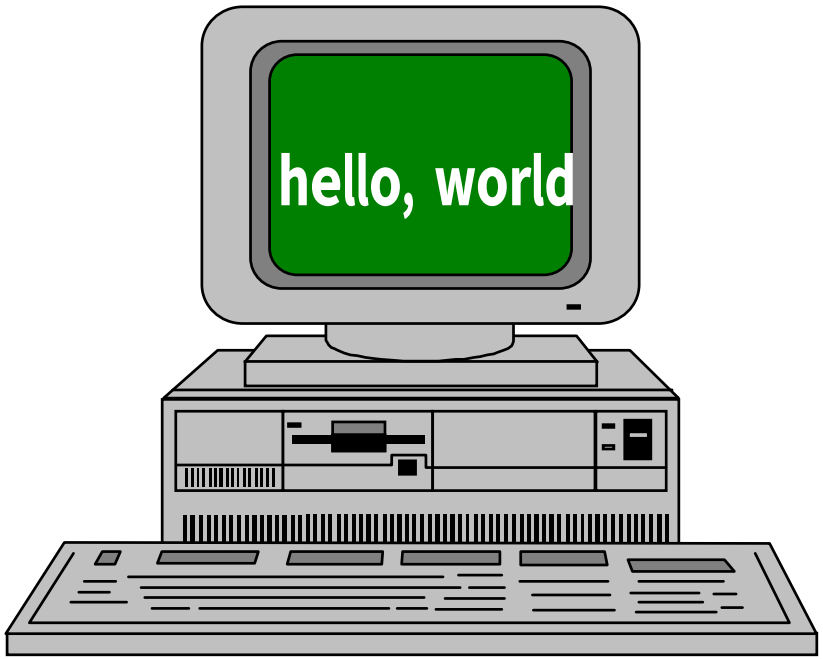


(3．2) プログラムの編集

■HelloWorldクラスのメソッドmainを、次のように編集します。

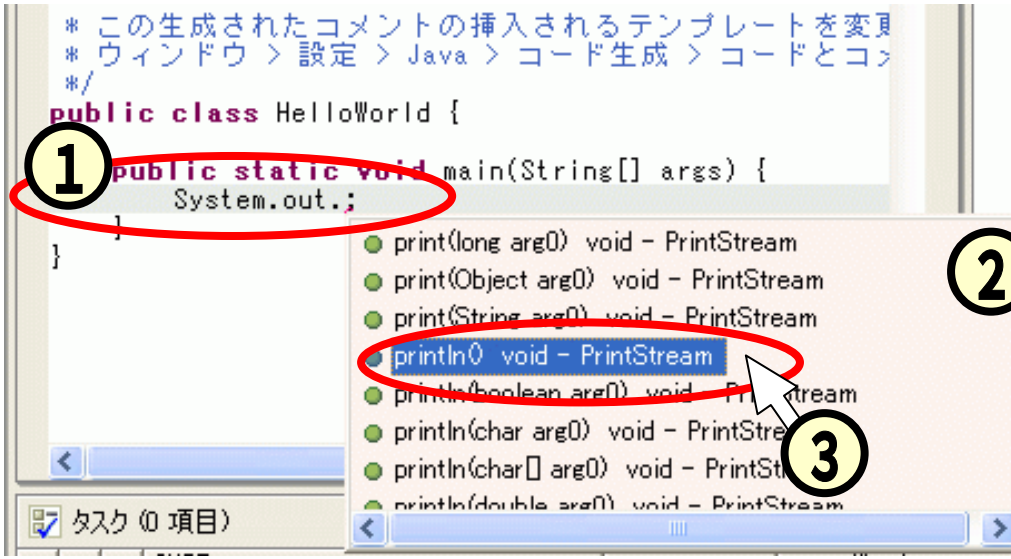
```
public static void main(String[] args) {
   System.out.println("hello, world. ");
}
```

■このプログラムは、コンソールに“hello, world.”と出力するプログラムです。

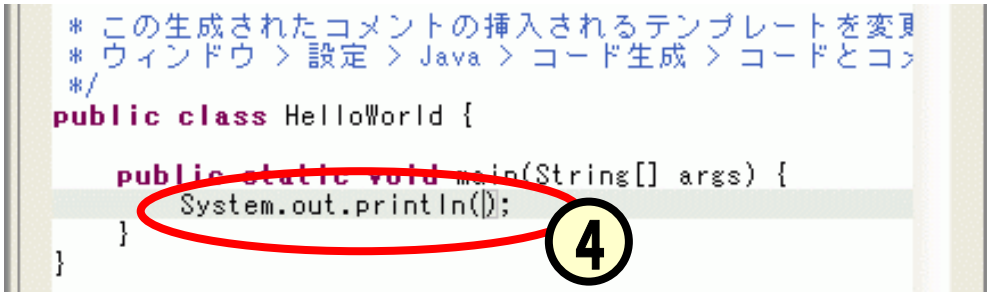


(3．3) コンテンツ・アシスト

■このプログラムの中では、“System.out.”と入力したところ (①) で、[Ctrl]+[Space]([Ctrl]キーを押しながら[Space]キーを押す)とすると、ポップアップメニューが表示されます (②)。

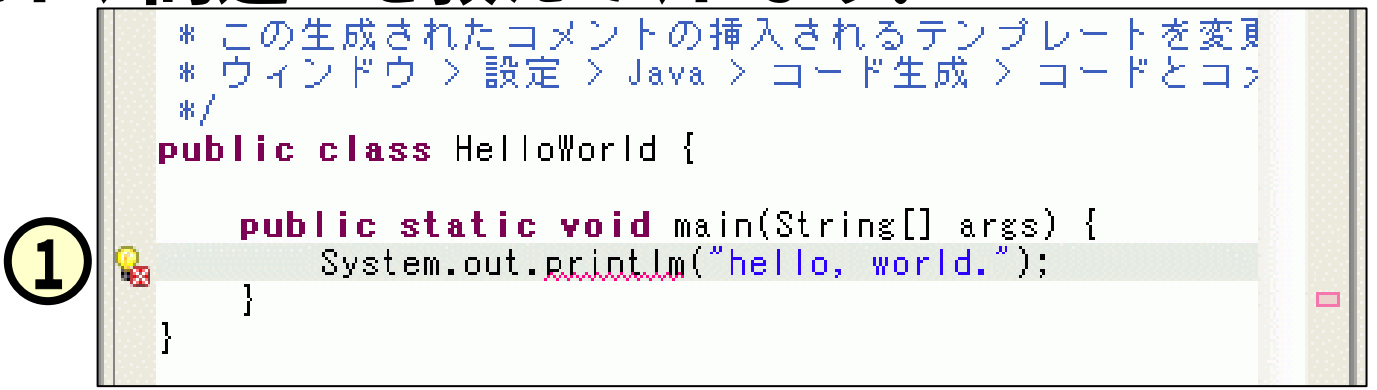


■メニューの中から選択したメソッド (右図では、“println()”) をクリック (③) すると、そのメソッドがソースの中に記述されます (④)。

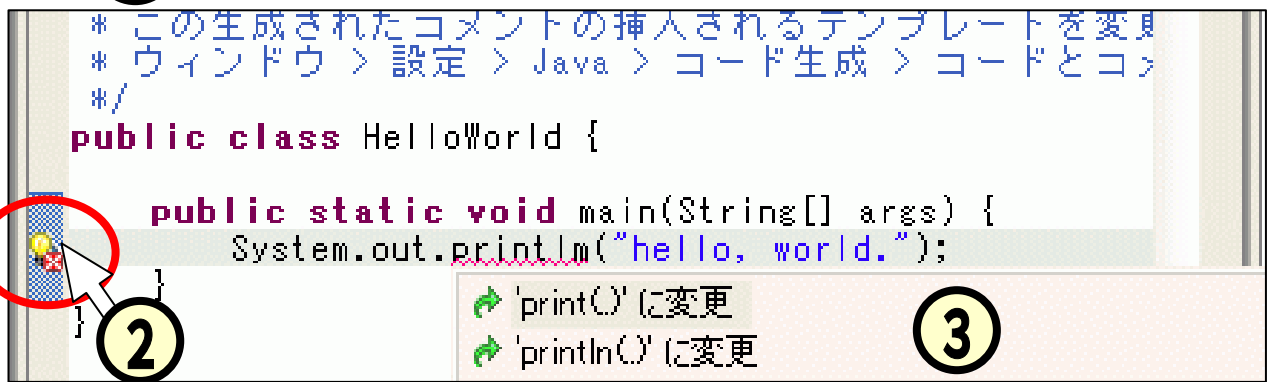


(3．4．1) クイック・フィックス　－1－

■例えば、“println”を、間違えて“printlm”と入力したとします。このとき、eclipseでは① のように警告アイコンが表示され、間違いを教えてくれます。

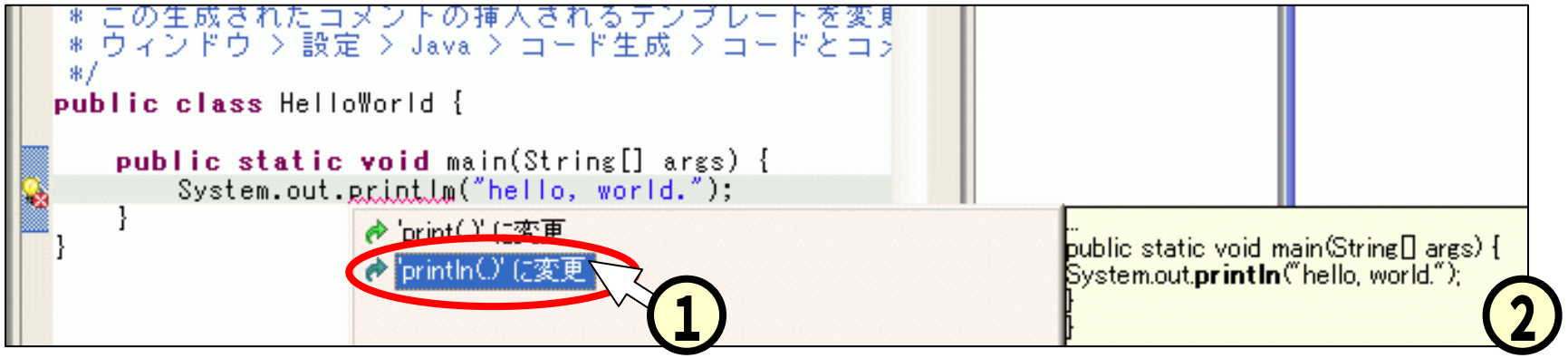


■警告アイコンをクリック (②) すると、修正案の候補一覧が表示 (③) されます。

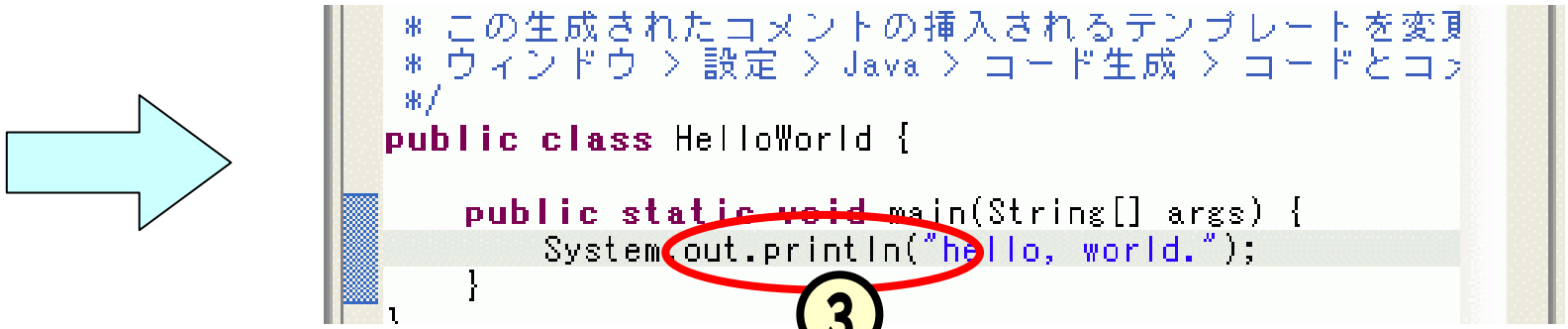


(3．4．2) クイック・フィックス　－2－

■修正案の“println”をクリック (①) すると、どのように修正されるのかがポップアップウィンドウに表示 (②) されます。



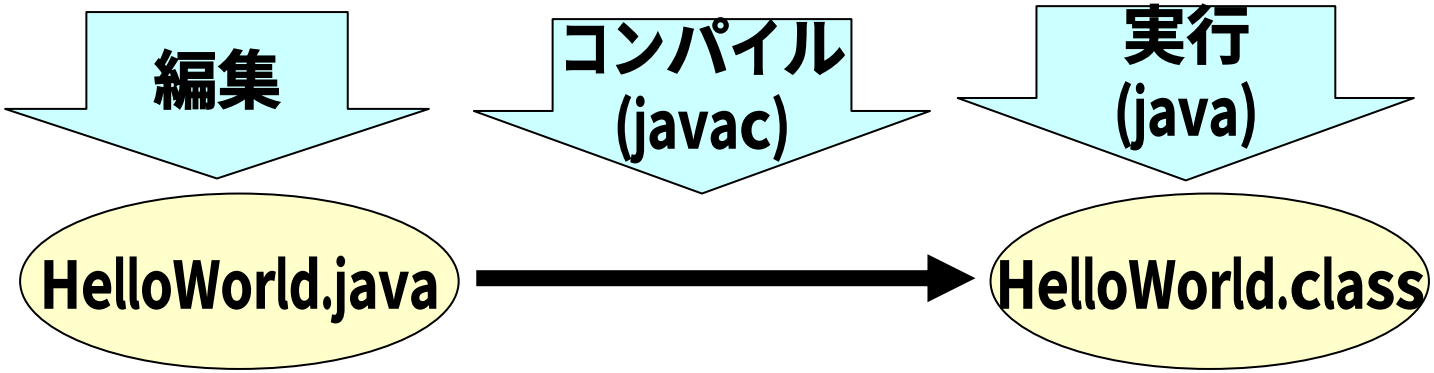
■修正案の“println”をダブルクリック (①) すると、実際に修正が実行されます (③)。



(4) ビルドと実行

■コマンドライン(Unix)上では、次のようにJavaプログラムを作成・実行していました。

```
$ emacs HelloWorld.java # エディタでプログラムを編集
$ javac HelloWorld.java # コンパイル
$ ls                    # ファイル一覧を表示
HelloWorld.java  HelloWorld.class
                        # classファイルが出来た
$ java HelloWorld       # プログラムを実行
```



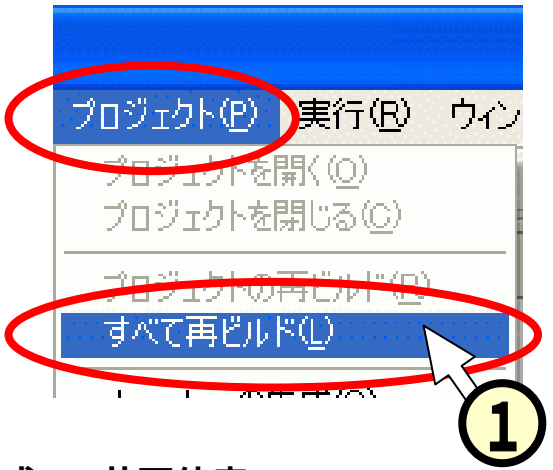
■実行出来るファイルを作成する(コンパイルを含む)ことを、ビルドと言います。

(4．1) eclipseでのビルド

■eclipseでは、①プログラムの実行を指示した際、②プログラムを保存した際に、自動的にビルドが行われます。よって、利用者がビルドの指示を行う必要はありません。

■ビルドだけを単独で行いたい時は、[プロジェクト]-[プロジェクトの再ビルド] (もしくは、[すべて再ビルド]) をクリック (①) します。

- プロジェクトの再ビルド
  - ◆変更のあったものだけビルド
- すべてビルド
  - ◆変更の有無に関わらずすべてビルド

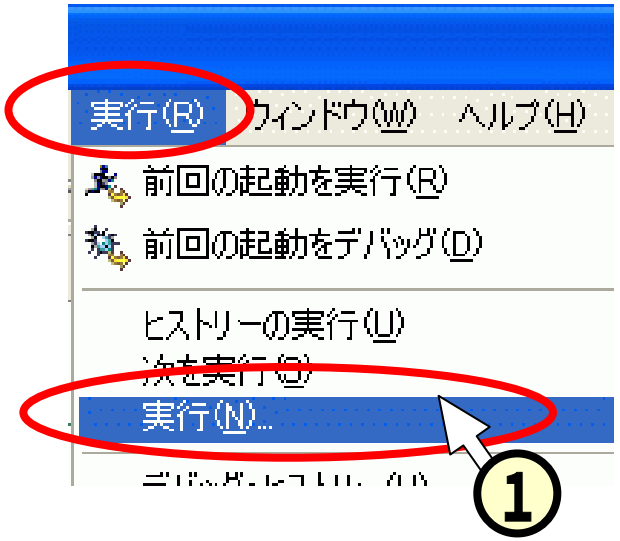


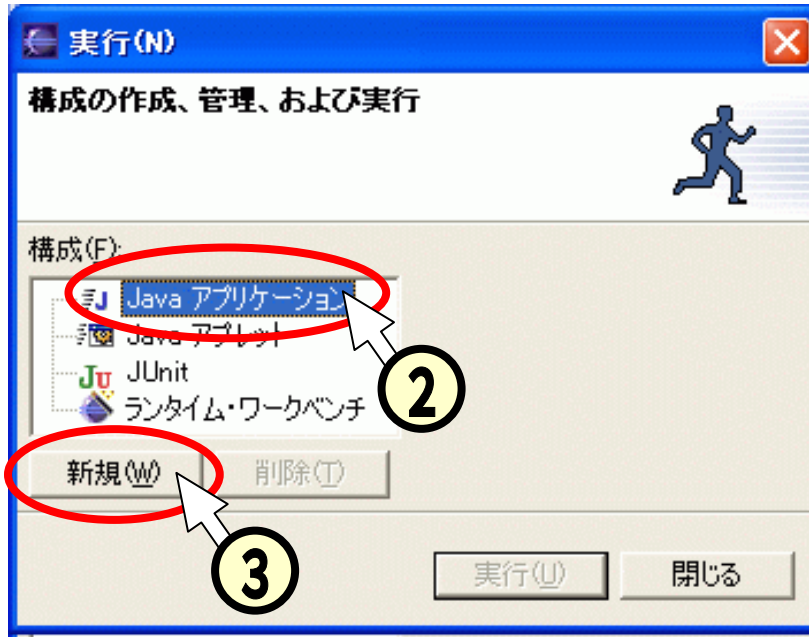
(4．2．1) 実行　－1－

■作成した、HelloWorldクラスを実行します。

■[実行(R)]-[実行(N)]をクリック(①)します。

■「実行」ウィンドウにて、「構成」の中から[Javaアプリケーション]をクリック(②)し、[新規]をクリック(③)します。



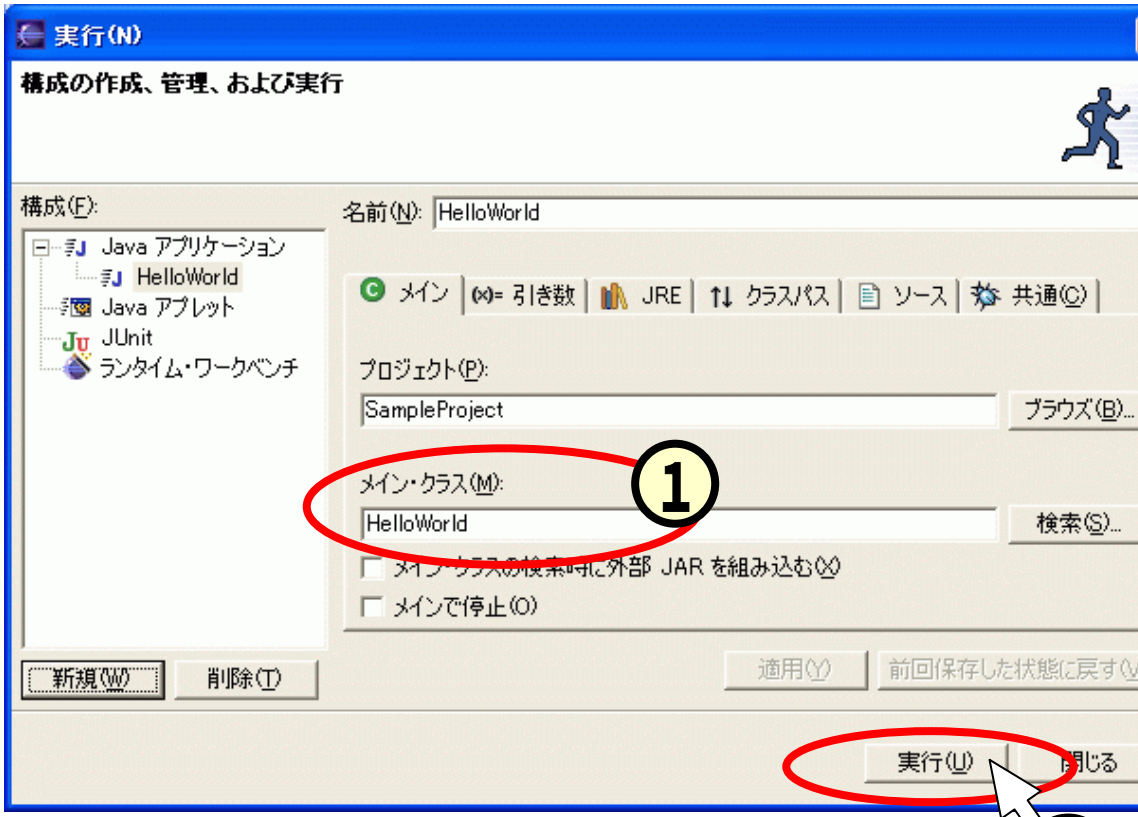


(4．2．2)実行　－2－

■実行可能なプログラムとして、“HelloWorld”が表示(①)されます。

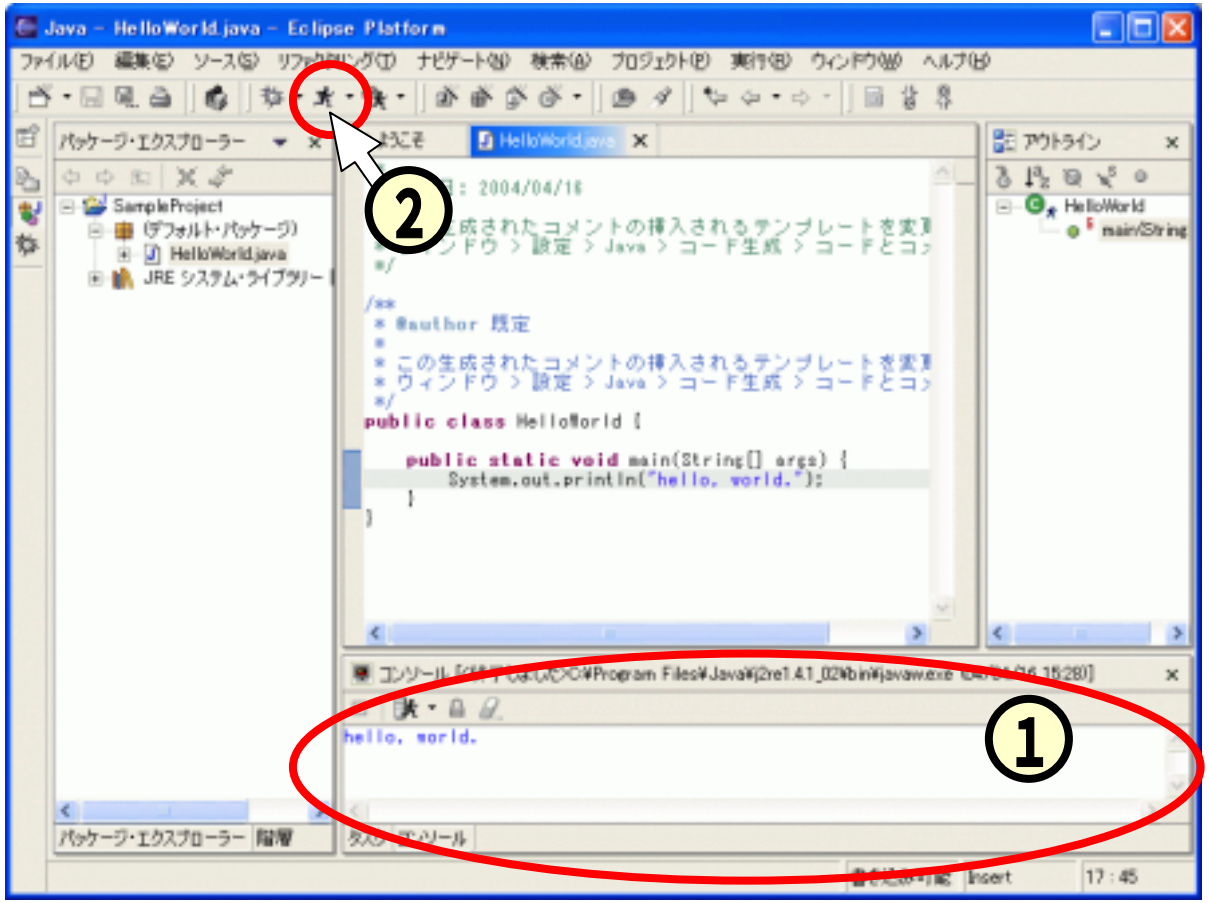
■そのまま[実行]をクリック(②)してください。

■「リリースの保管」ウィンドウにて、[すべて選択]をクリック(③)してから、[OK]をクリック(④)します。



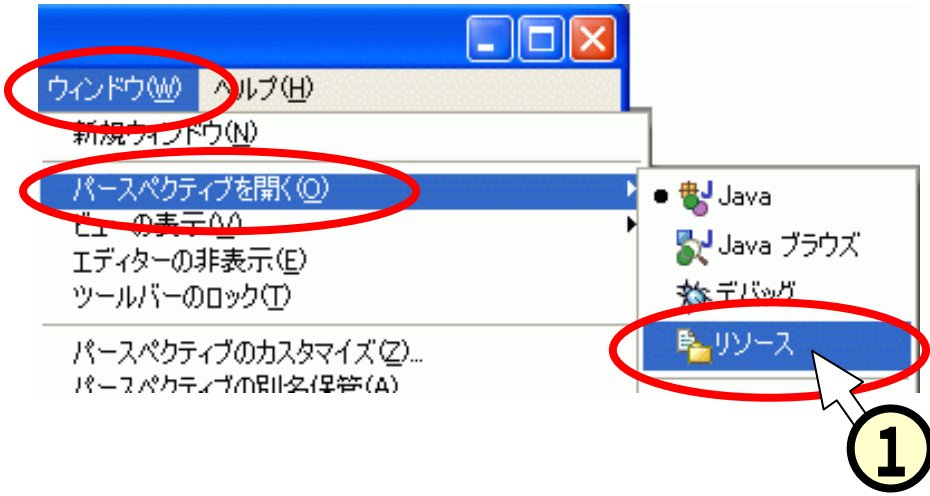
(4．2．3) 実行　－3－

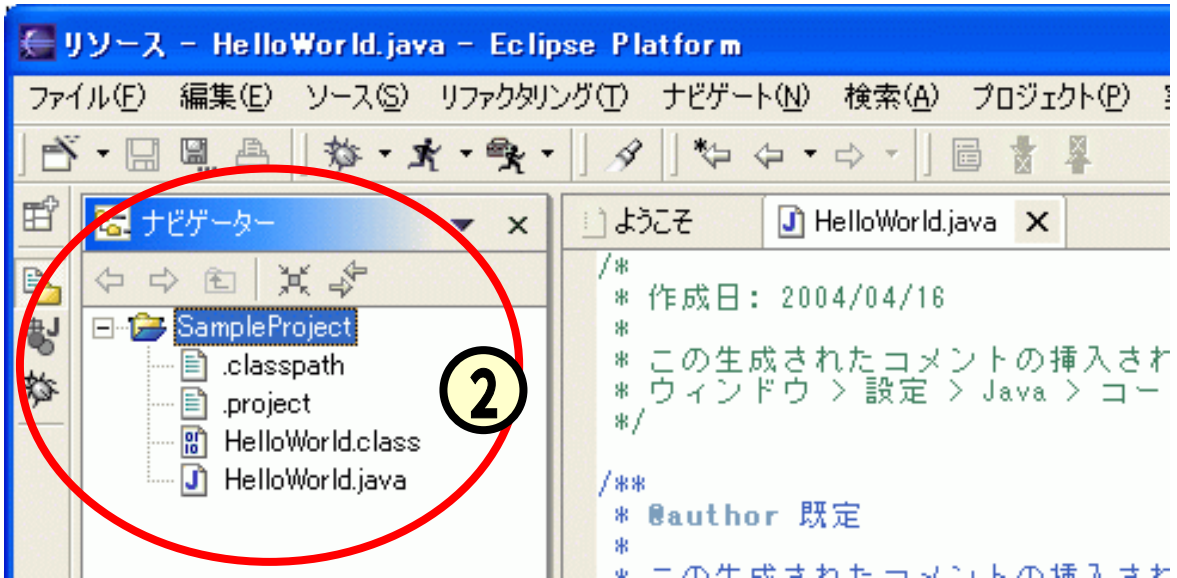
■実行結果は、「Java」パースペクティブの「コンソール」ビューに表示 (①) されます。2回目以降は、ボタンをクリック (②) することで実行することが出来ます。



(4．3) 作成したクラスファイルの確認

■「リソース」パースペクティブに切り替える([ウィンドウ]-[パースペクティブを開く]-[リソース] ①)と、実行したクラスファイルを、「ナビゲータ」ビュー(②)にて確認できます。

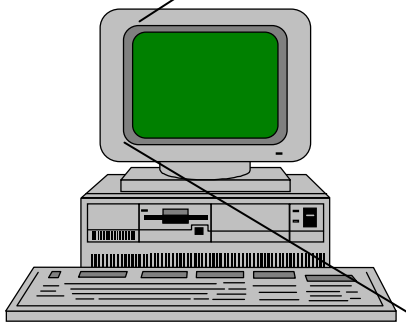




(5) デバッグ

■次のようなプログラムを考えます。

- 整数値“n”を引数として入力する。
- 1からnまでの数を順にコンソールに出力する。
- その際、プリントした数が2の倍数であれば、“2の倍数です”と出力し、3の倍数であれば、“3の倍数です”と出力する。



```
java TwoThree 6
<1>
<2>
2の倍数です。
<3>
3の倍数です。
<4>
2の倍数です。
<5>
<6>
2の倍数です。
3の倍数です。
```

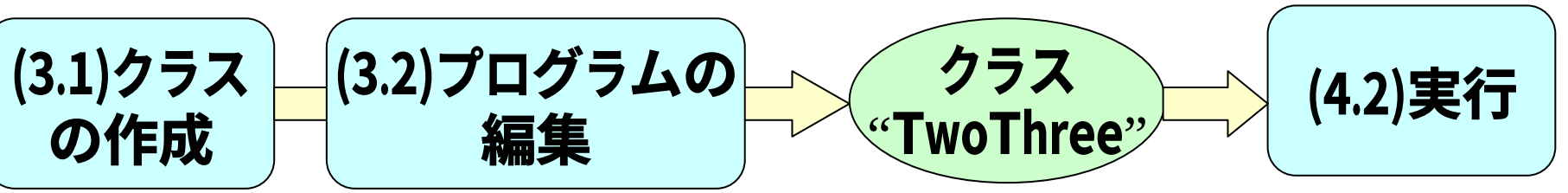
(5．1) 最初のソースコード

■次のようなソースコードを作成します。

```
public class TwoThree {
  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    for(int i=1;i<=n;i++){
      System.out.println("<"+i+">");
      if((i % 2)==0){
        System.out.println("2の倍数です。");
      }else if((i % 3)==0){
        System.out.println("3の倍数です。");
      }
    }
  }
}
```

（5．2）編集、ビルド、実行

■スライド(3)(4)に記した手順で、クラス“TwoThree”を作り、(5.1)のように編集し、ビルド、実行します。

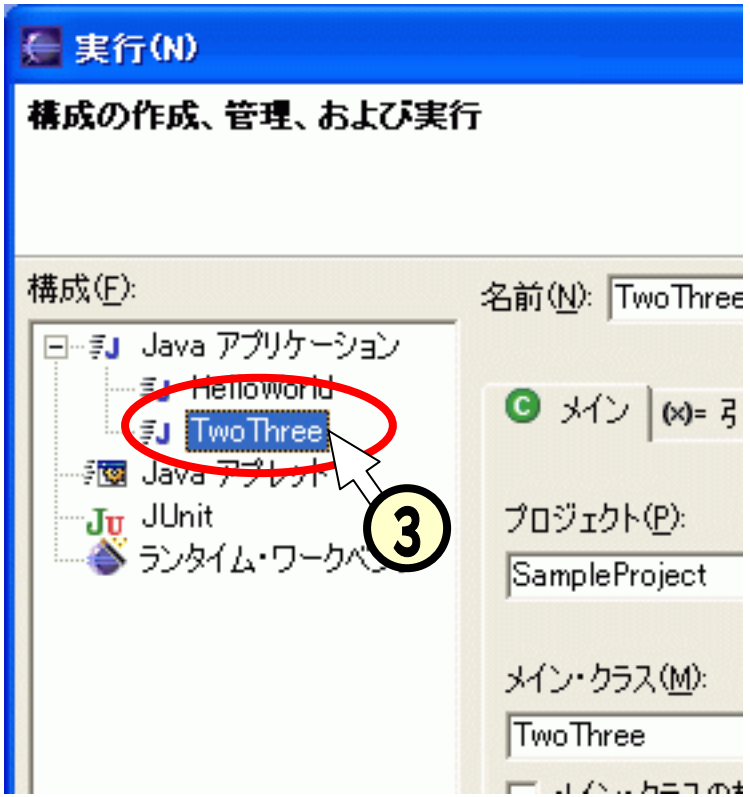


■実行の時点で、HellowWorldの時とは手順が少し異なります。以下、メニューより[実行(R)]-[実行(N)]をクリックして現れる「実行」ウインドウ中での手順の違いを説明します。

(5．2．1) 実行クラスの選択

■「実行」ウインドウの「構成」のペイン中、[Javaアプリケーション]をクリック(①)して、[新規]をクリック(②)すると、今回作成した“TwoThree”が表示されます。

■これの“TwoThree”をクリック(③)しておきます。

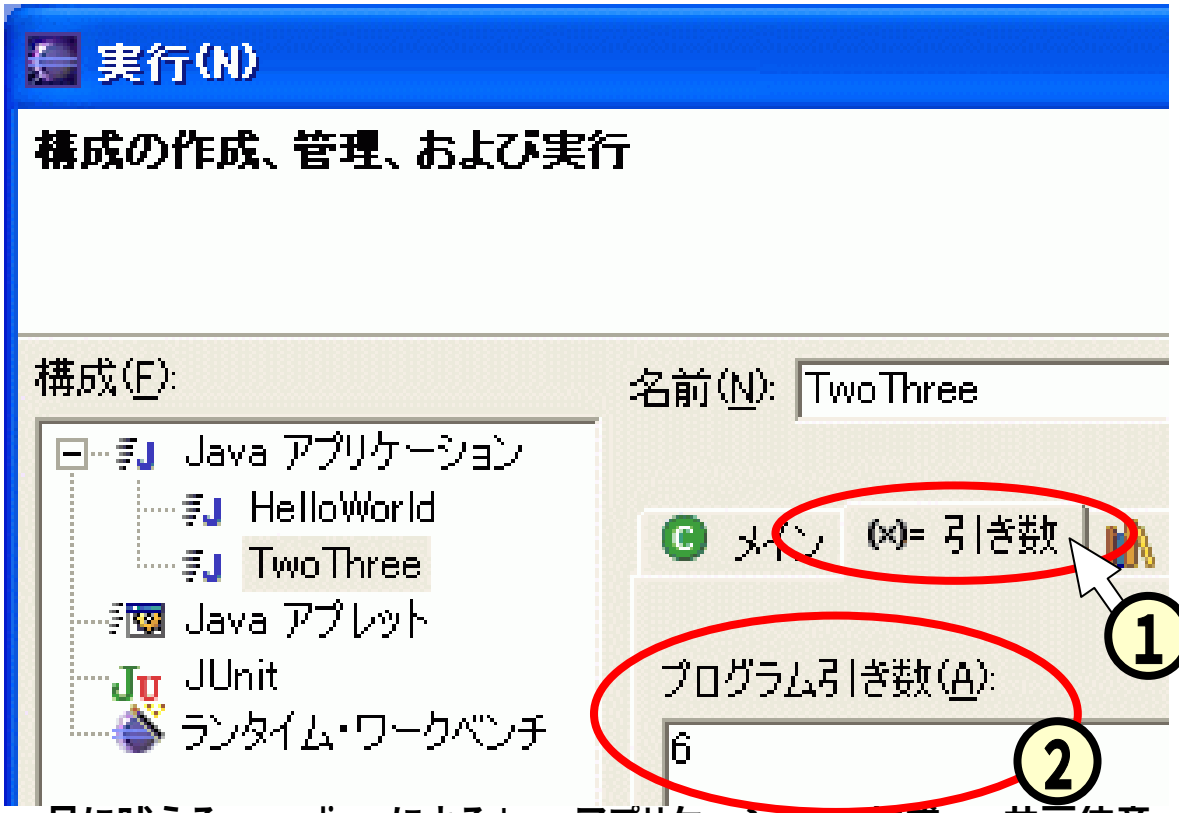


(5．2．2) 引数

■“TwoThree”のメイン･メソッドの引数を与えます。

■「実行」ウィンドウの、[引数]タグをクリック (①) します。

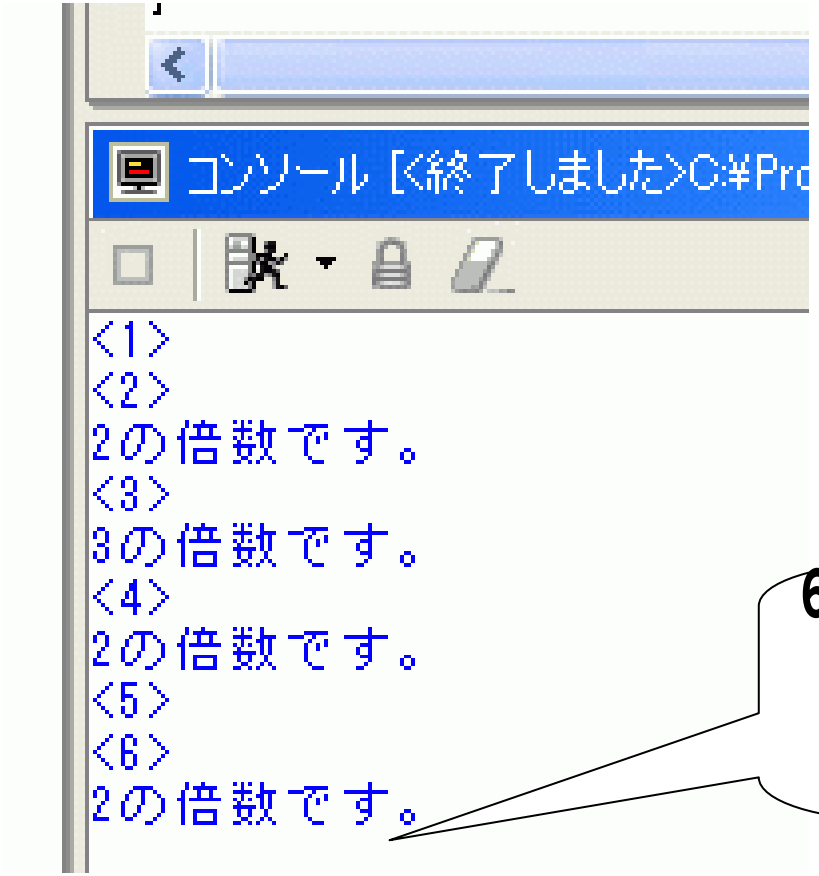
■[プログラム引数]の欄に、引数 (今回の場合は“6”) を入力 (②) します。



(5．2．3) 実行結果

■[実行]をクリックすると、ワークベンチ下側の「コンソール」ビューに、実行結果が表示されます。

■どうも、想定したものとは違っているようです。



6は3の倍数でもあるので、
“3の倍数です。”とも
出力したかったが、
されていない。

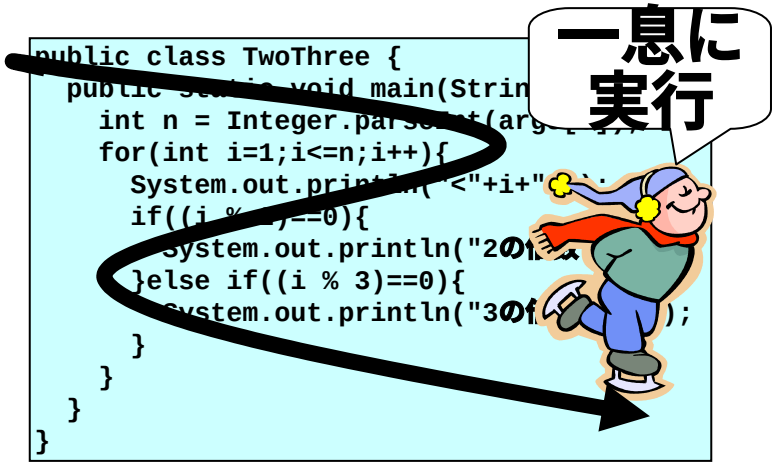
(5．3) ブレークポイントを用いたデバッグ

■スライド(5.1)で作成したプログラムには、何かしら“問題” (バグと言います) があるようです。

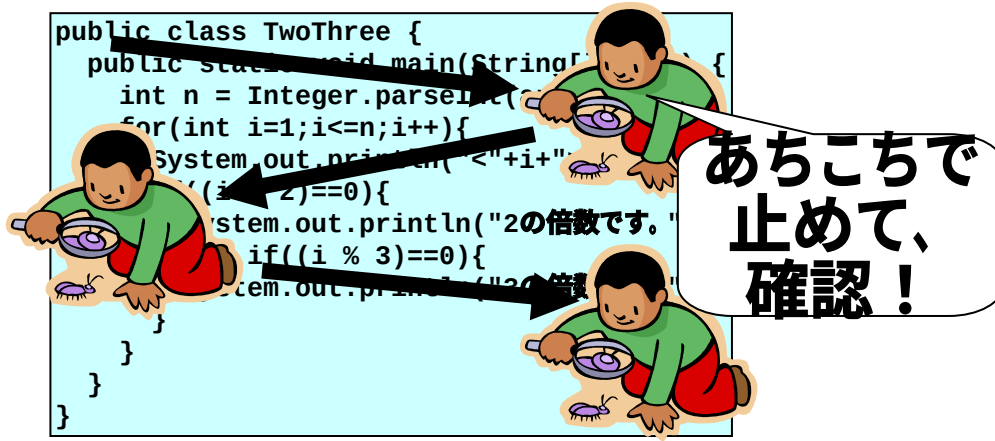
■この問題を除去する (デバッグといいます) ために、ブレークポイントを使ってみましょう。

■ブレークポイントを使ったデバッグでは、プログラムを一息に実行させるのではなく、あちらこちらで止めながら実行し、変数の値などを確認します。

＜通常のプログラム実行＞



＜ブレイクポイントを使ったデバッグ＞

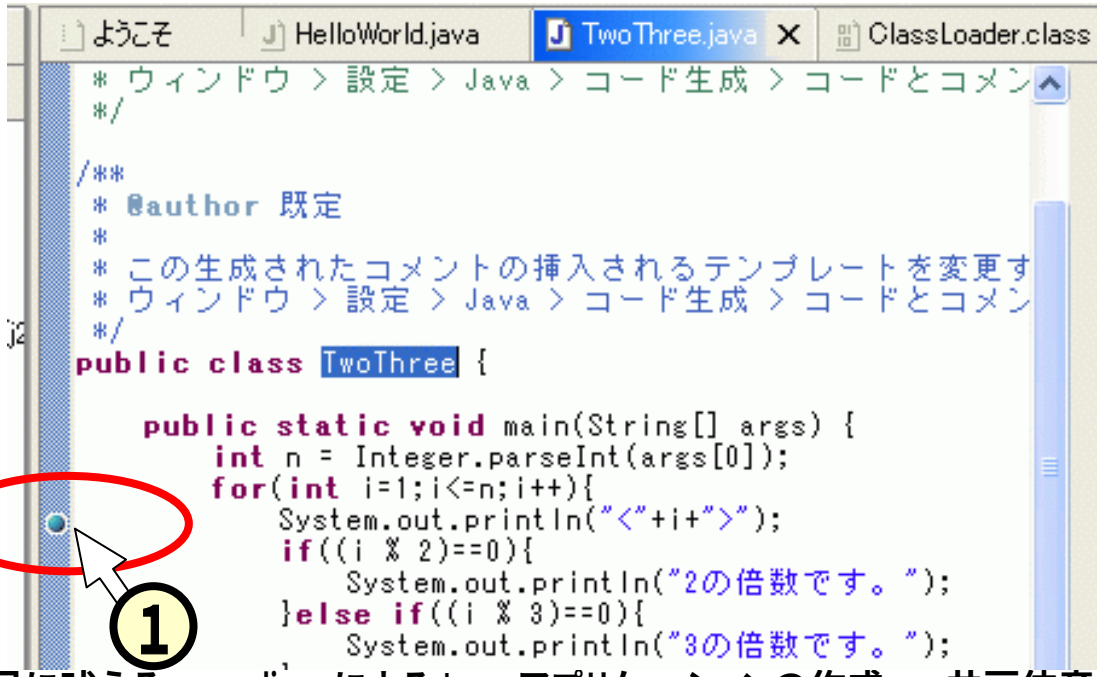


(5．3．1) ブレークポイントの設定

■デバッグでプログラムを止めるところを“ブレークポイント”といいます。

■暮らす“TwoThree”のデバッグでは、数字を出力するところにブレークポイントを設定することにします。

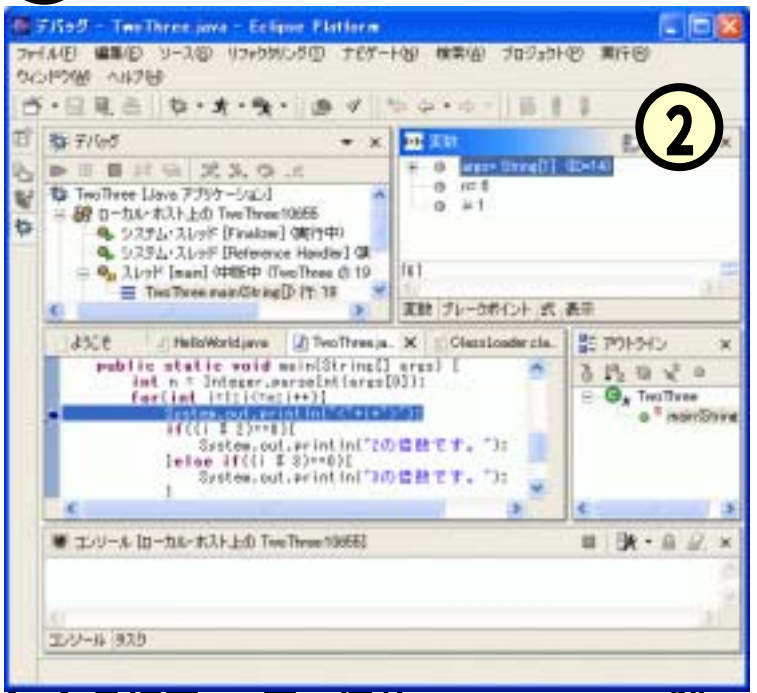
■プログラムの該当する行 (＝ライン) の左側の部分をダブルクリック (①) します。青い丸印が表示されます。



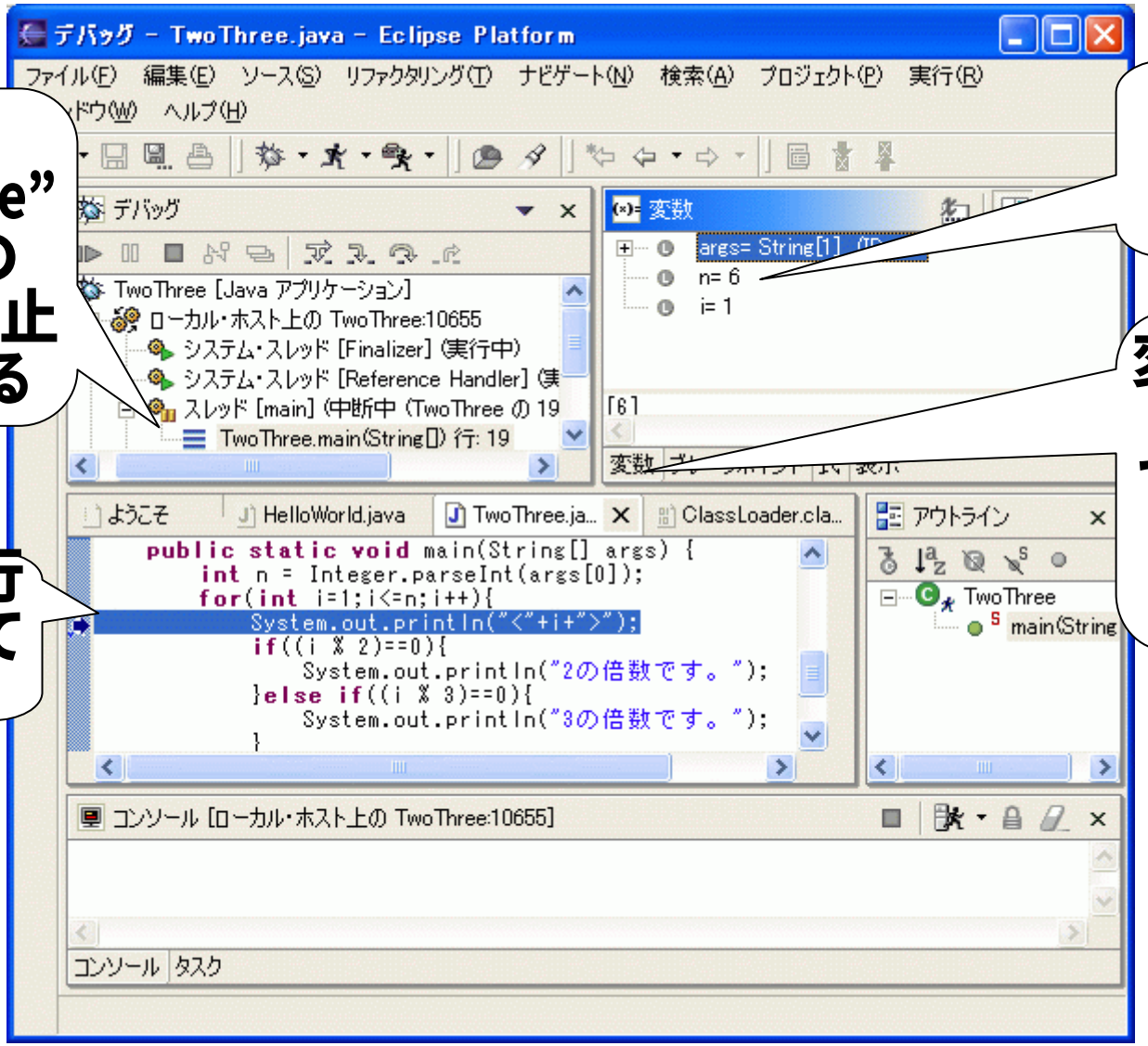
(5．3．2) デバッグ実行

■ブレークポイントで止まりながら実行する場合には、実行ボタンでなく、デバッグボタンをクリック (①) します。

■プログラムの「デバッグ」が開始され、パースペクティブに切り替わります (②)。



(5．3．3) デバッグ・パースペクティブ



クラス"TwoThree"のmainの19行目で止まっている

今、この行で止まっている

変数の値 nは6、iは1。

変数タグをクリックして、変数を表示させている

(5．3．4) 再開

■プログラムが止まった状態から、再度実行を開始させます。

■[実行]-[再開]をクリック（ 

、もしくは[F8]キー押下）します。

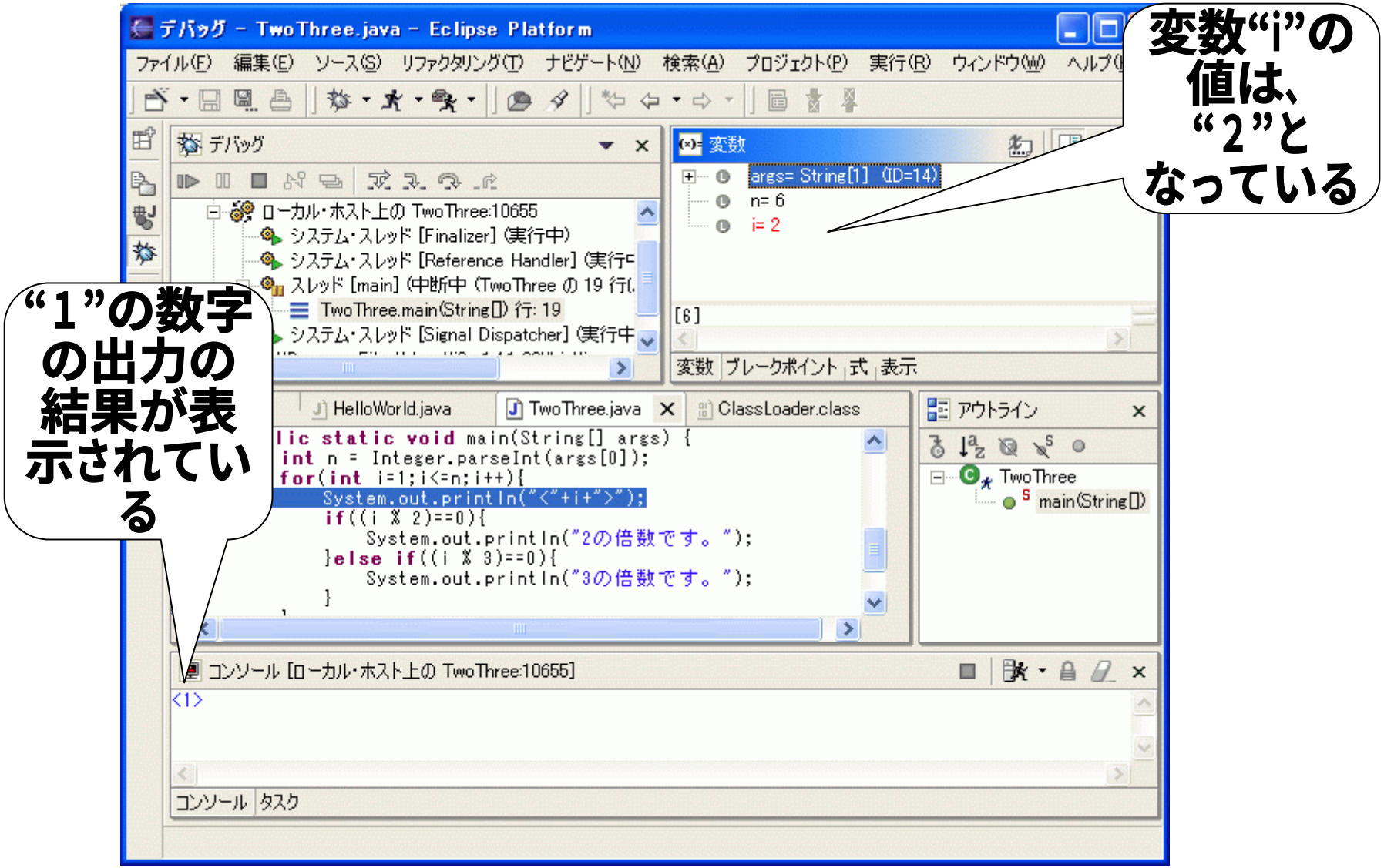
■すると、プログラムは再び同じブレークポイントに止まります。

```
public class TwoThree {
  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    for(int i=1;i<=n;i++){
      System.out.println("<"+i+">");
      if((i % 2)==0){
        System.out.println(i+"は2の倍数です。");
      }else if((i % 3
        System.out.pr
      }
    }
  }
}
```

①最初、ここのブレークポイントで止まり、

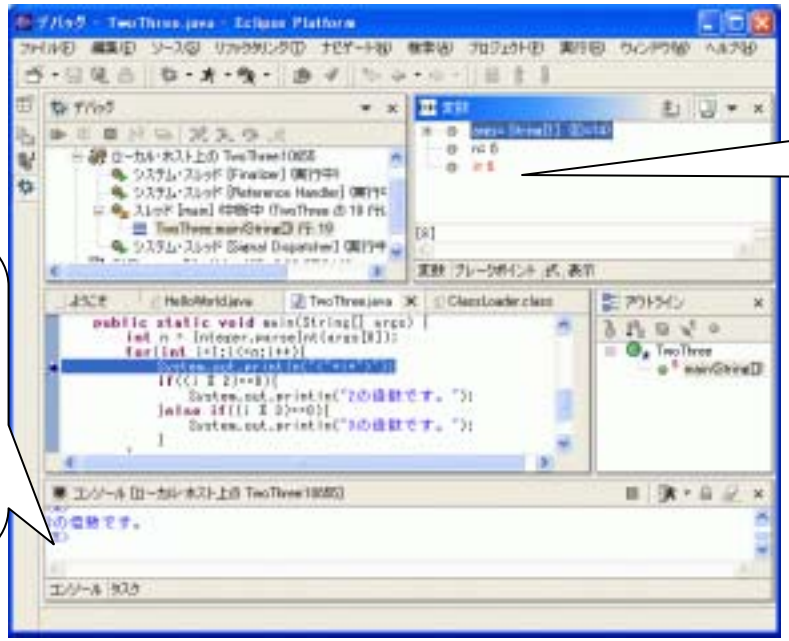
②再開させると、また同じブレークポイントに止まる。

(5．3．5) 2回目のブレークでは。。。



(5．3．6) 1行づつ実行する

■“再開”を繰り返して、6回までブレークさせます。

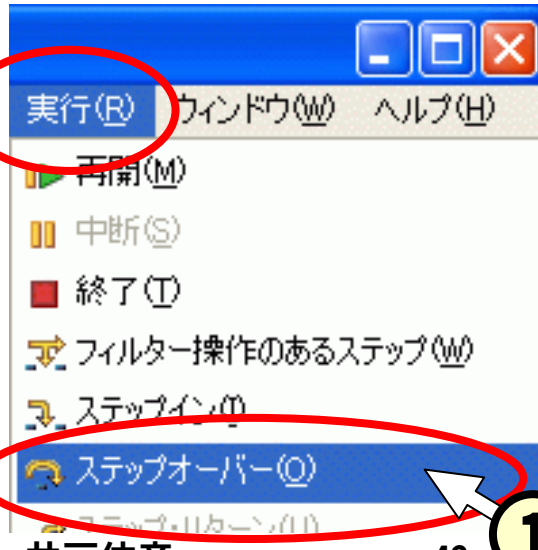


“5”の数字の出力の結果が表示されている

変数“i”の値は、“6”となっている

■この後は、1行ずつ実行することにします。

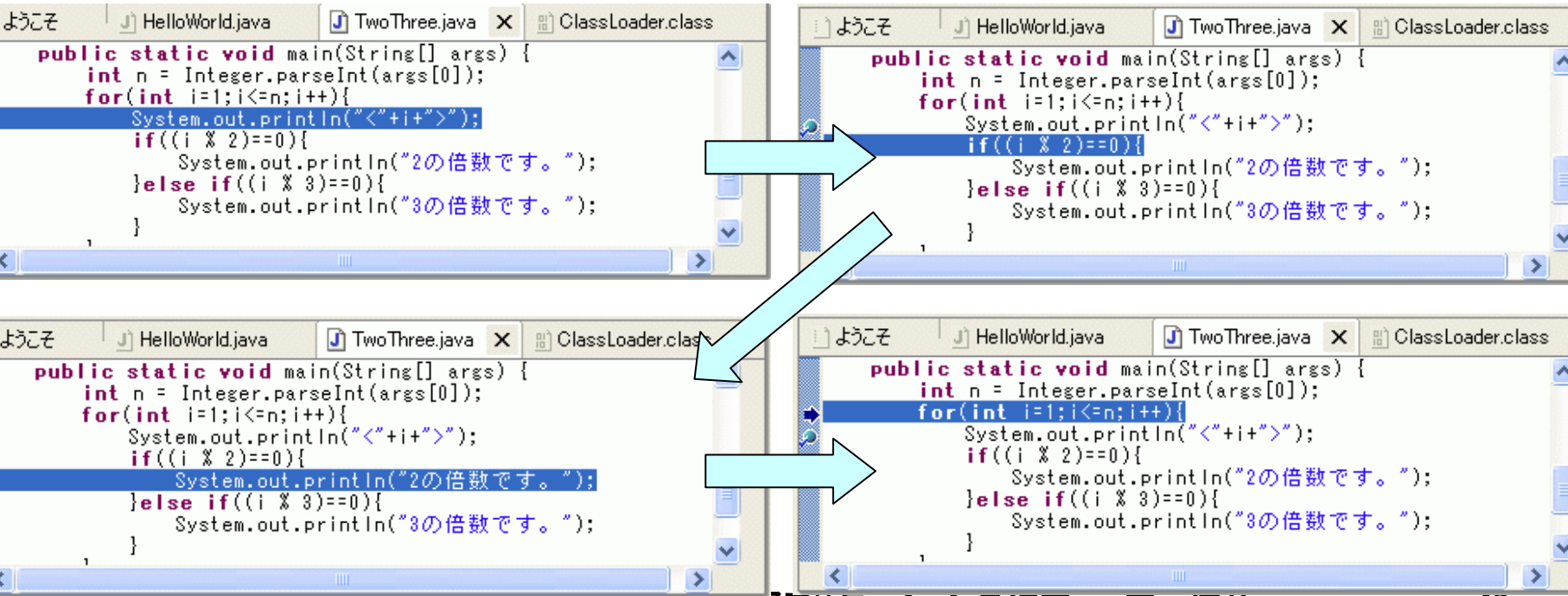
■1行ずつ実行するには、[実行]-[ステップ・オーバー]をクリック(①、もしくは、[F6]キー押下)します。



(5．3．7) ステップ･オーバーを繰り返す

■ステップ･オーバー ([F6]押下) を続けていくと、実行しているラインが順に動いていきます。

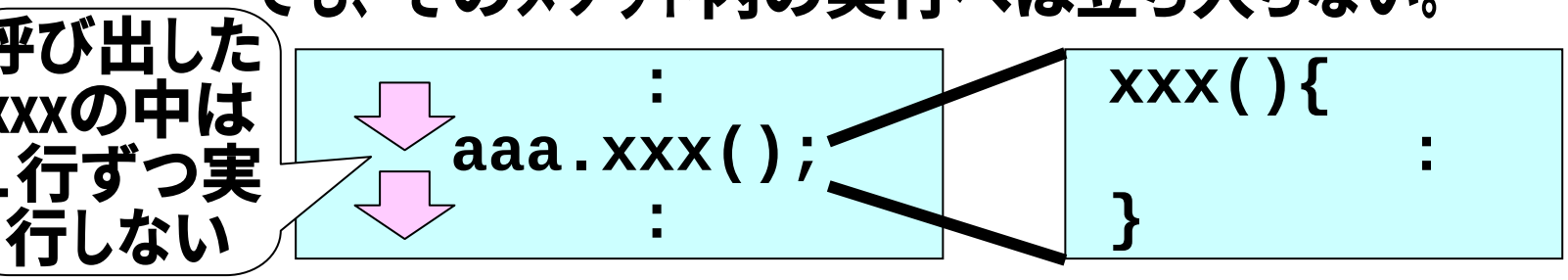
■3の倍数であることの判定が、“else if”となっているため、2の倍数で無い時しか3の倍数であることが出力されないことになっているようです。



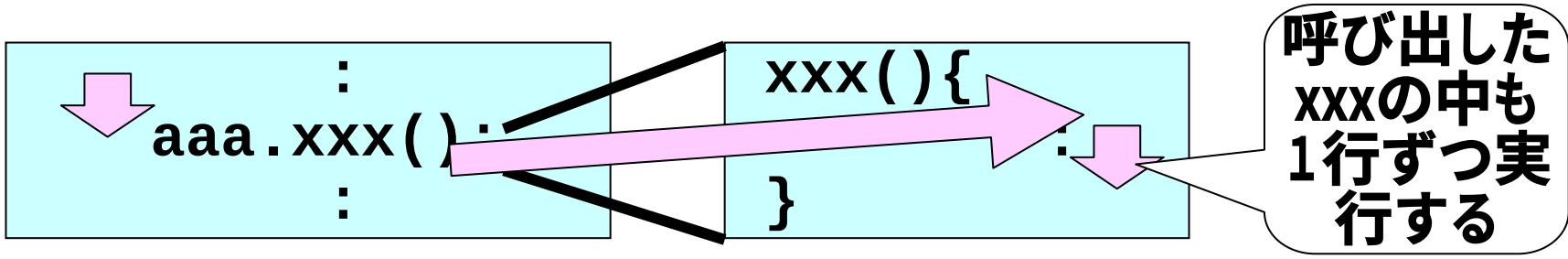
(3．5．8) ステップ･イン

■1行ずつ実行する方法には、次の2つがあります。

- ステップ･オーバー
  - ◆途中で他のメソッド (関数、サブルーチン) を呼び出していても、そのメソッド内の実行へは立ち入らない。



- ステップ･イン
  - ◆途中で他のメソッド (関数、サブルーチン) を呼び出している場合、そのメソッド内に移って、1行ずつ実行する。



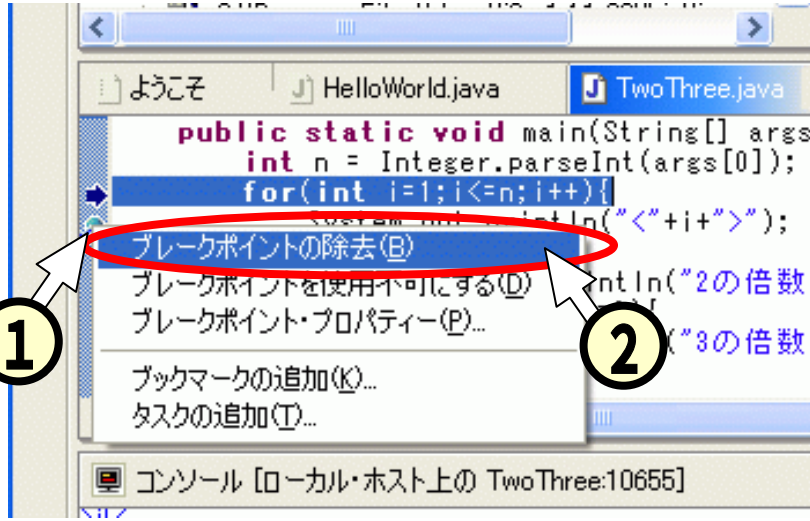
(5．3．8)ブレークポイントを除去する

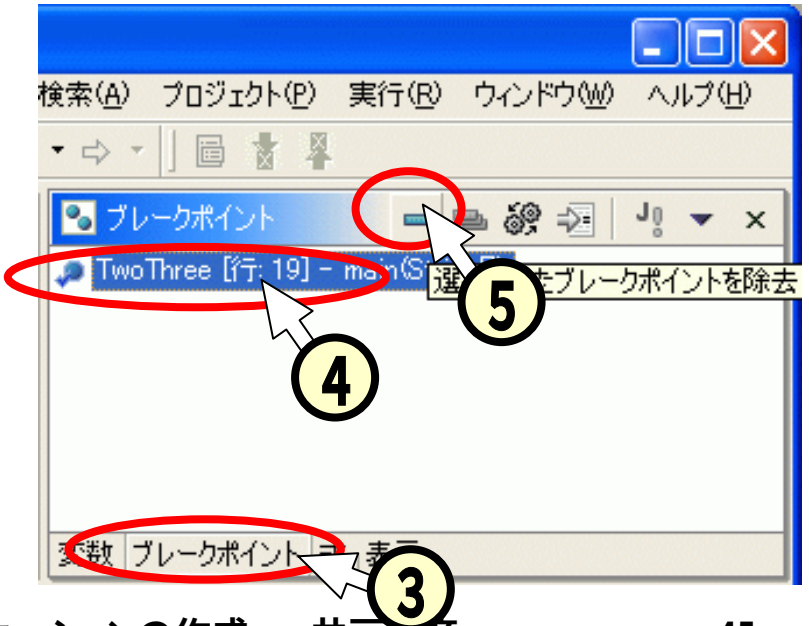
■一度ブレークポイントを設定すると、プログラムがそこを実行するたびに、何度でも止まります。

■もう止まらないようにするには、ブレークポイントを除去します。

■「エディタ」ビュー内のブレークポイントが設定された青い丸印を右クリック(　)し、ポップアップメニューか①ブレークポイントの除去]をクリック(　)します。

■もしくは、ワー②ベンチ右上の[ブレークポイント]のタグをクリック(③)し、ブレークポイントを選択(④)して、消去ボタンをクリック(⑤)してもＯＫです。





(5．4．1)ソースコードの修正

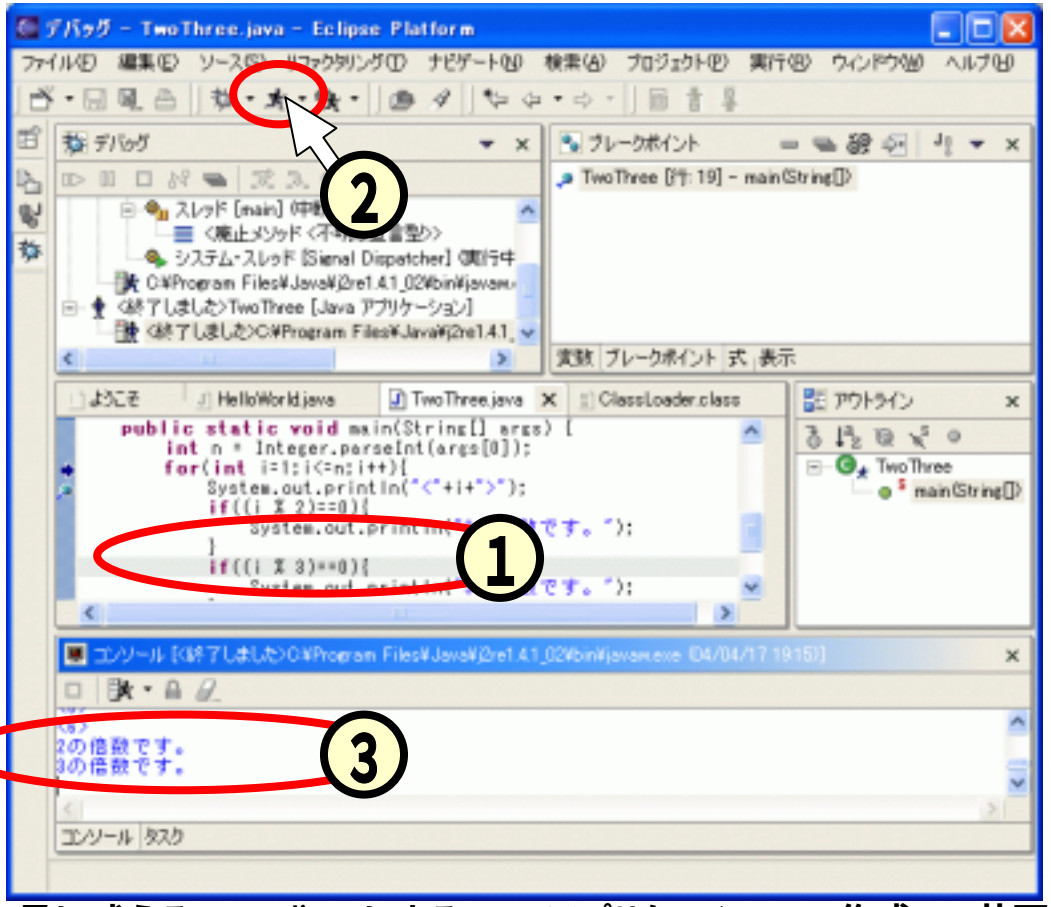
■ソースコードを次のように修正します。

```
public class TwoThree {
  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    for(int i=1;i<=n;i++){
      System.out.println("<"+i+">");
      if((i % 2)==0){
        System.out.println("2の倍数です。");
      }
      if((i % 3)==0){
        System.out.println("3の倍数です。");
      }
    }
  }
}
```

“else”を削除した

(5．4．2) 実行結果の確認

■「エディター」ビューでソースを修正し(①)、[実行ボタン]( 、②)をクリックすると、「コンソール」ビュー上に正しい実行結果が表示(③)されました。



(5．5) デバッグ一般について

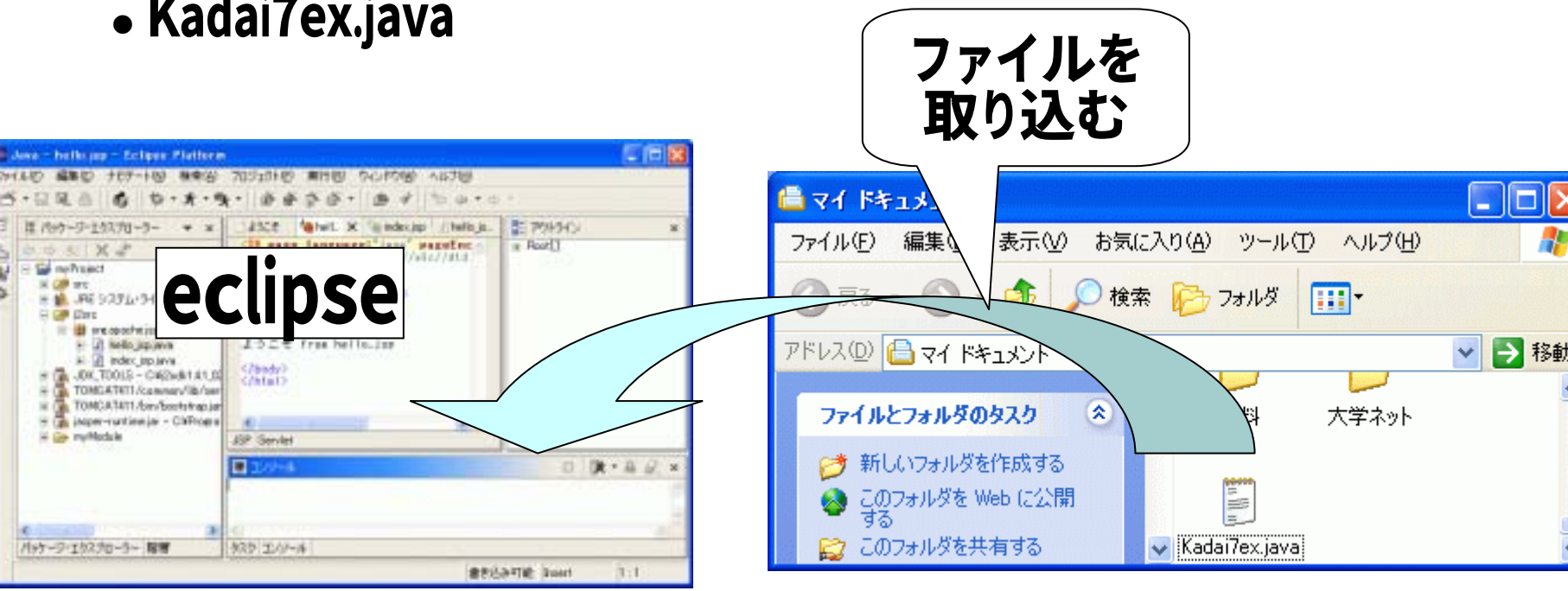
- (5) 項で示したバグは単純なものであり、また、デバッグで用いた技法も初歩的なものです。
- 実際のソフトウェア開発では、どのようなことが起こっているのか容易に理解出来ない事象も経験することになります。また、そのような事象の原因を突き止めていく方法に、分かりやすいセオリーがある訳でもありません。
- デバッグでどのような機能が使えるかは、スムースにソフトウェア開発を進められるか否かにおいて重要な意味を持ちます。
- ソフトウェア開発を自分の専門としたい方は、本スライドで紹介していないeclipseの機能についても、調べてみてください。

(6) ファイルのインポート

■eclipse上で扱えるよう、外部のファイルを取り込むことを、インポートといいます。

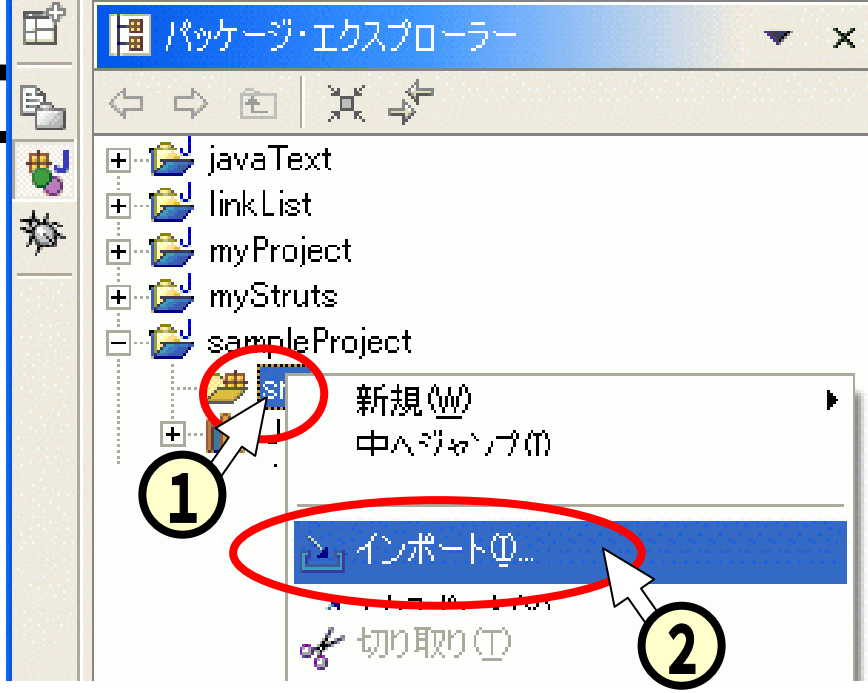
■マイドキュメント上にある次のＪａｖａのソースファイルを、eclipse内のプロジェクト“sampleProject”にインポートする (取り込む) 操作について説明します。

- Kadai7ex.java

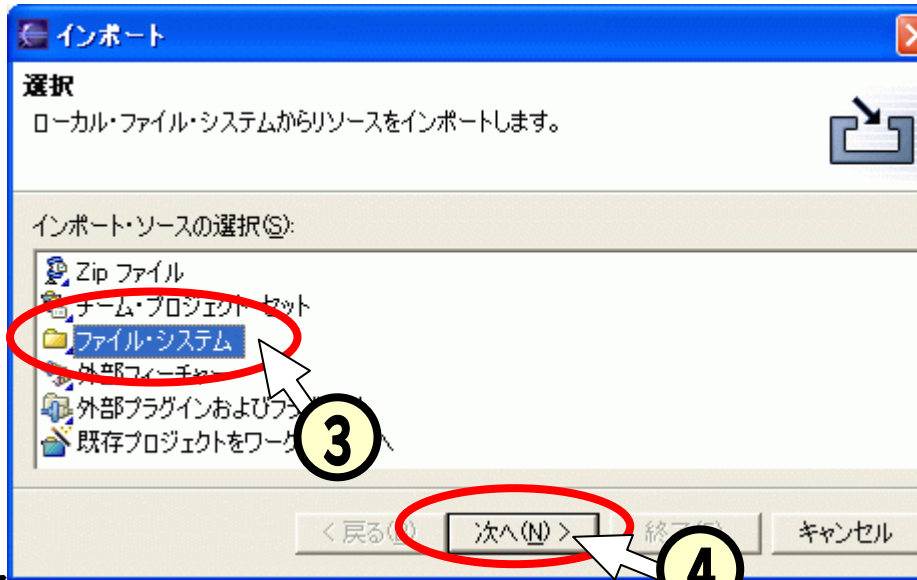


(6．1) インポートー1ー

■パッケージ･エクスプローラー中、SampleProjectのソースフォルダを右クリック (①) し、現れたメニューにてインポートをクリック (②) します。



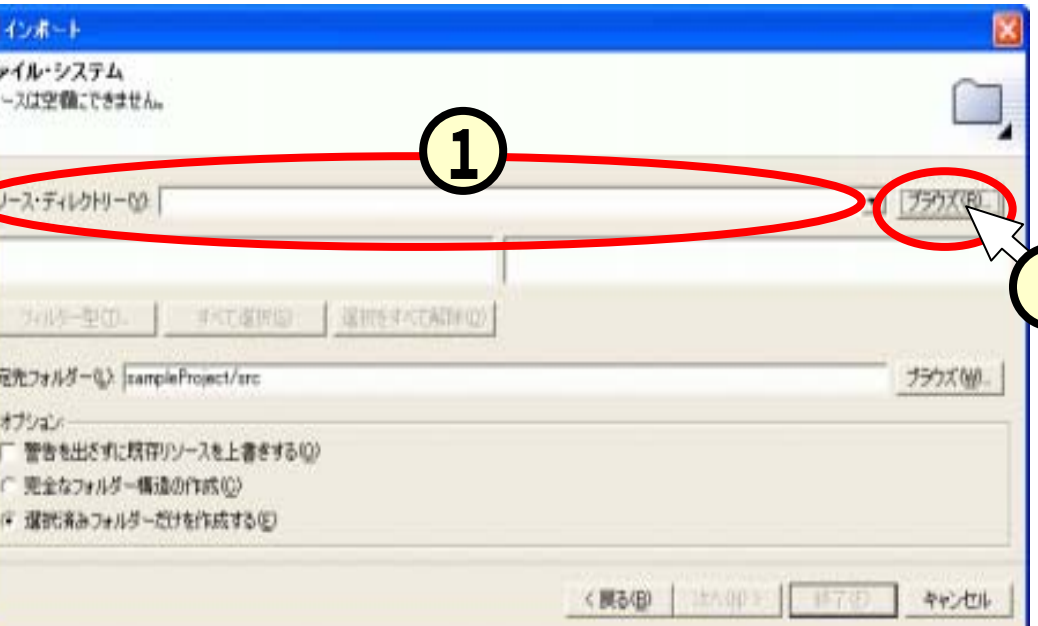
■「インポート：選択」のウインドウにて、[ファイルシステム]をクリック (③) し、[次へ]をクリック (④) します。



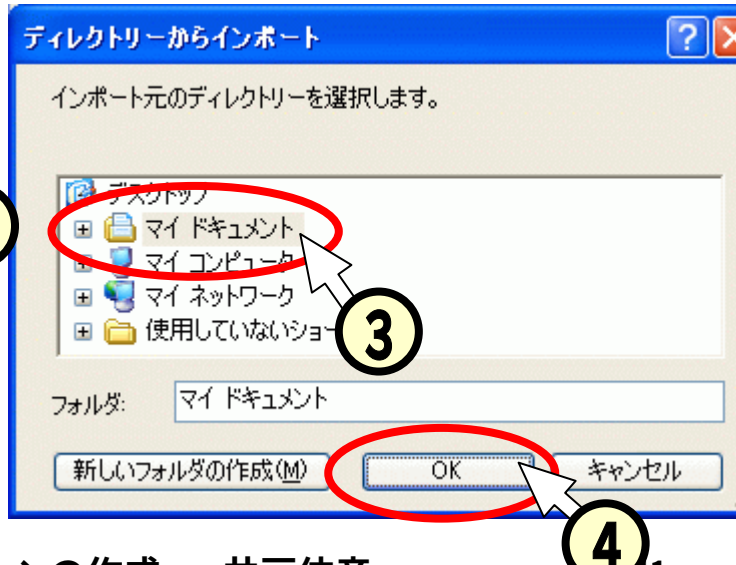
(6．2) インポート　－2－

■「インポート：ファイルシステム」のウインドウにて、次の手順で[ソースディレクトリ] (①) を入力します。

- 右の[ブラウズ]をクリック (②) します。
- 「ディレクトリーからのインポート」ダイアログにて、インポートするファイルのあるディレクトリ (この場合はマイドキュメント、③) をクリックします。
- [OK] をクリック (④) します。



②



(6．3) インポート　－３－

■選択したソースディレクトリが表示 (①) されるので、その配下の目的のファイルにチェック (②) して、[終了]をクリック (③) します。

■パッケージ･エクスプローラー中で、インポートしたファイルが確認出来ます (④)。