1806b 版

Eclipse の使い方

2019 年 1 月 21 日～1 月 25 日

（補助資料 1）

Microsoft CERTIFIED Solutions Developer
Web Applications
Windows Store Apps Using C#

Microsoft CERTIFIED Professional Developer
Windows Azure Developer
Windows Phone Developer
Windows® Developer 4
Enterprise Application Developer 3.5

2019 年 1 月 18 日

ニュートラル株式会社
林 広宣
http://www.neut.co.jp/
hayashi.hironori@neut.co.jp

（このページが、表紙の裏になるように印刷をお願いします。）

ご意見・ご質問は下記まで

ニュートラル株式会社　林広宣
hayashi.hironori@neut.co.jp

目　　次

ECLIPSE の使い方 **1**

目　　次 **3**

第 1 章. 簡単なアプリケーションの作成 **5**

1.1. ECLIPSE の起動 6
1.2. プロジェクトの作成 11
1.3. クラスの追加（ソースコードの記述） 18
1.4. プログラムの実行 21

第 2 章. プロジェクトのインポート **23**

2.1. プロジェクトのインポート 24

(空白ページ)

第1章. 簡単なアプリケーションの作成

1.1. Eclipse の起動

1.1.1.起動方法

Eclipse を実行するためには、JDK か JRE がインストールされており、そこにパスが通っている必要があります。

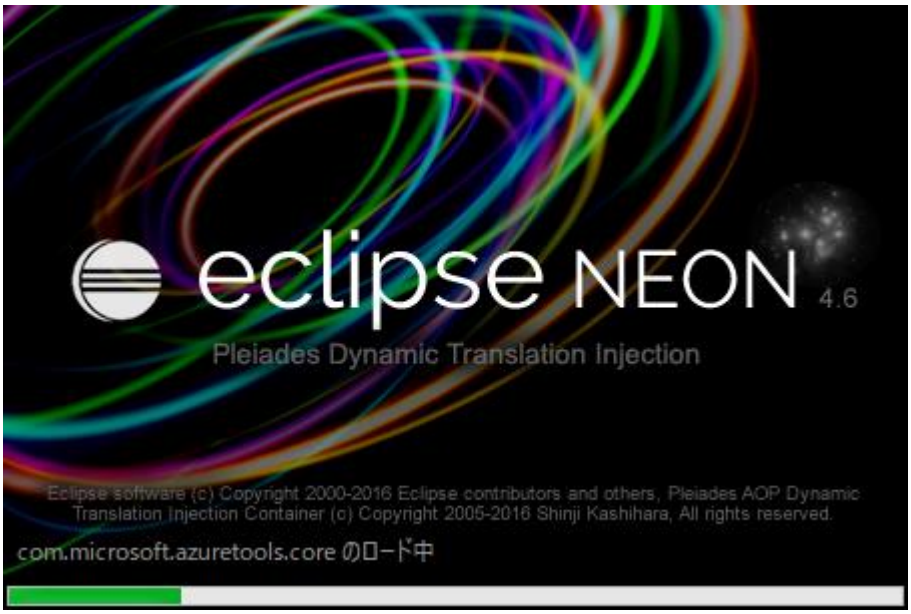
もし、JDK あるいは JRE を、Windows 用のインストーラでインストールした場合は、JDK あるいは JRE にパスが通るようにインストールされるため、eclipse.exe をダブルクリックなどするだけで、Eclipse が起動します。

一方、単なるフォルダコピーで JDK をインストールしたような場合は、eclipse.exe を実行する前に、java.exe が存在する場所にパスを通しておく必要があります。
この方法の詳細については、C:¥java¥bin フォルダにある「Eclipse の通常起動.bat」や「Eclipse の研修用起動.bat」ファイルなどの記述を参考にしてください。

いずれの方法にしても、Eclipse を起動すると、何秒かの間、次のようなスプラッシュ画面が表示されます。

スプラッシュ画面は、Eclipse のバージョンによって少しずつ異なります。(下図は、バージョン 4.6.3 のものです)

（この画面をスキップするには、 -nosplash オプションを指定して、 eclipse.exe -nosplash とします。）



1.1.2.ワークスペースの指定

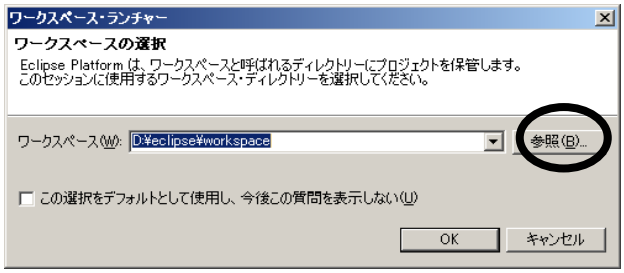
日本語化された Eclipse の場合は、引き続いて、次のような画面が表示されます。
（C:¥Java¥bin にある「Eclipse の研修用起動.bat」で起動した場合は、表示されず、スキップされます）

「ワークスペース」とは直訳すると「作業空間」となりますが、ここでは、我々が作成するソースファイルなどを保存しておくフォルダだと考えればよいでしょう。

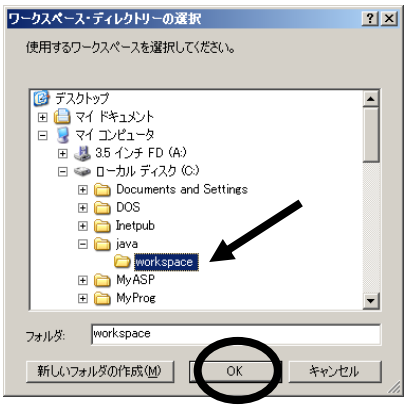
セミナーでは、ワークスペースは、次のフォルダにしてください。

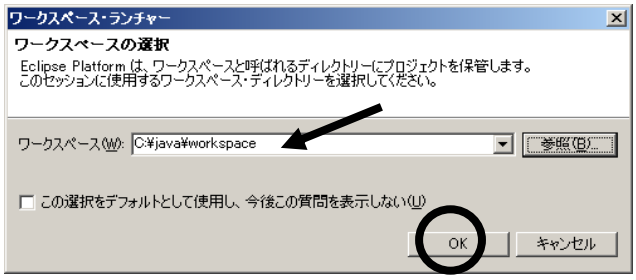
```
C:¥java¥workspace
```

（C:¥java¥bin フォルダにある「Eclipse の研修用起動.bat」ファイルを使用すると、自動的にこのフォルダをワークスペースに設定して Eclipse を起動します）



[参照]ボタンをクリックして、C:¥java¥workspace を選択します。





1.1.3.起動画面

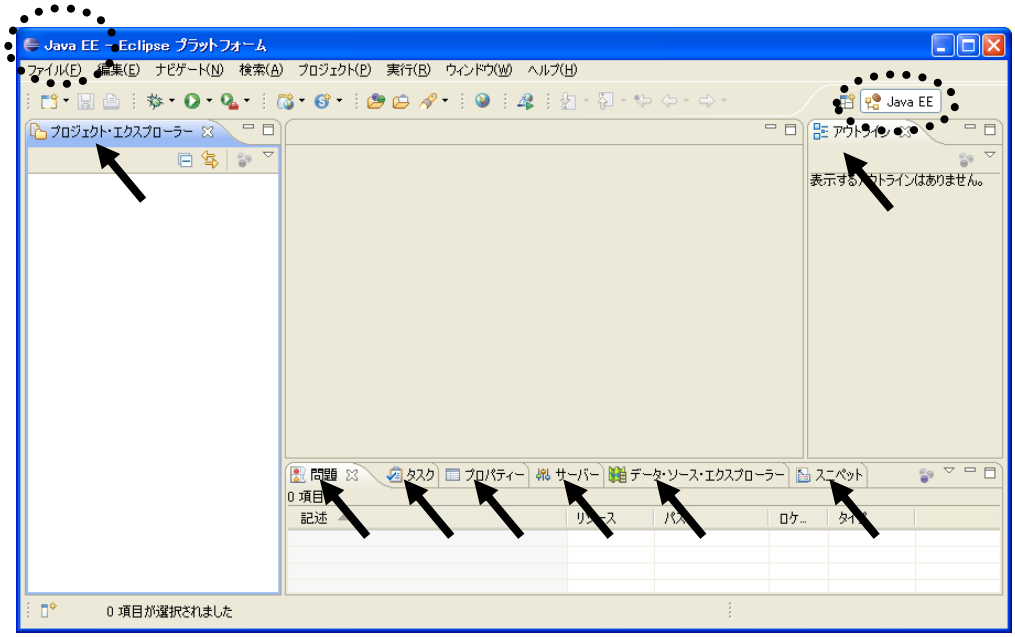
ここまでのステップを踏むと、下記のような起動画面が表示されます。
（下図は、バージョン 3.2.x のものです）

既に、ワークスペースにプロジェクトなどが作成されている場合は、この画面は表示されず、次の画面に進みます。

この画面は、下図右上の[閉じる]ボタン（×印の部分）をクリックして閉じてください。

ようやく、設計や開発が行える画面が表示されました。
この、Eclipse の統合開発環境の画面のことを「ワークベンチ」と言います。
下図の画面全体がワークベンチです。（矢印などが示す意味は、次のページで説明します）

（「ワークスペース」とはファイルを保存する場所のことで、「ワークベンチ」とは作業を行う画面全体のことです）



1.1.4.ワークベンチとパースペクティブ

Eclipse のワークベンチは「エディタ」や「ビュー」といった、いくつかのウィンドウ（ペインともいいます）から成り立っています。

前ページの図で、矢印で指し示したウィンドウ（ペイン）を「ビュー」と呼びます。
「エディタ」は、まだ現れていませんが、Java などのソースコードを入力する部分のことです。

これらのビューやエディタは、何十種類も用意されており、必要なものだけを画面に表示して使うようになっています。

ところで、Eclipse は、Java だけの開発環境ではありません。COBOL の開発や UML の設計などにも使用されます。(その他にも、C++、C#、PHP、XML などのプラグインがあります)
また、Java の開発にも何種類かあって、「通常のアプリケーション」「Java コンポーネント」「Web アプリケーション」などがあります。

このように、何種類もの形態の開発を行う場合に、いつも同じビューしか表示されないのでは使いづらくなってしまいます。
Java のアプリケーション開発の場合と UML 図の作成の場合では、異なった画面上のツール（つまり異なったビュー）を使いたいものです。

そこで、Eclipse では、ワークベンチを、「Java 開発向けの画面」、「UML 図作成用の画面」という風に、種類分けできるようになっています。

つまり、「Java 向けのビューは、これとこれ」、一方、「UML 向けのビューはこれとこれ」というように、あらかじめ適切なビューを組み合わせたものが用意されています。

この、ビューを組み合わせてセットにしたものを「パースペクティブ」と呼びます。
パースペクティブとは、直訳すると、「観点」「見通し」というような意味ですが、Eclipse では、ビューの組合せのことを言います。

前ページのワークベンチは、「JavaEE」と呼ばれるパースペクティブが使用されています。
（点線の楕円で囲まれた部分が、パースペクティブの名前です）
この JavaEE 用のパースペクティブは、Web アプリケーションなど JavaEE 対応のアプリケーションを作成する場合に使います。

同じ Java の開発でも、通常の Java アプリケーションの開発の場合は、[Java]パースペクティブを使用します。([JavaEE]パースペクティブでも作成可能です)

注 1
Eclipse のバージョンによっては、パースペクティブの名称が[J2EE]という名前になっているものもあります。

1.1.5.用語のまとめ

最後に、ここまでに登場した用語を列挙します。イメージが浮かぶ程度に覚えておくといいでしょう。

1. ワークスペース
2. ワークベンチ
3. ビュー
4. エディタ
5. パースペクティブ

上記の用語を読んで、イメージが沸くかどうか、自分自身に確認してみてください。

そして、もし、イメージが浮かんでこないようだったら、前のページの説明をもう 1 度読み返してください。

それでも解決しないようだったら、講師などに質問してみてください。

1.2. プロジェクトの作成

1.2.1.説明

ここでは、コンソールに文字列を表示する簡単な Java アプリケーションを作成しながら、Eclipse の操作を学習します。

1.2.2.プロジェクトの作成

Eclipse で作業を行う場合は、最初に、「プロジェクト」を作成します。

そして、プロジェクトを作成したのちに、そのプロジェクトに、Java のソースコードなど、さまざまなファイルを登録します。
プロジェクトの作成をスキップして、ファイルを追加したり、開いたりすることはできませんので、注意してください。

プロジェクトとは、簡単に言うと、複数のファイルを一元管理するためのしくみです。

例えば、X というアプリケーションを作成するためには、A と B と C という３つのファイルが必要だとします。
もし、プロジェクトという概念がないと、「X を作成するためには、A と B と C が必要」という情報を人間が（頭の中か、紙面などに書いて）記憶しておかなければなりません。
それは、面倒なので、プロジェクトとして管理するわけです。

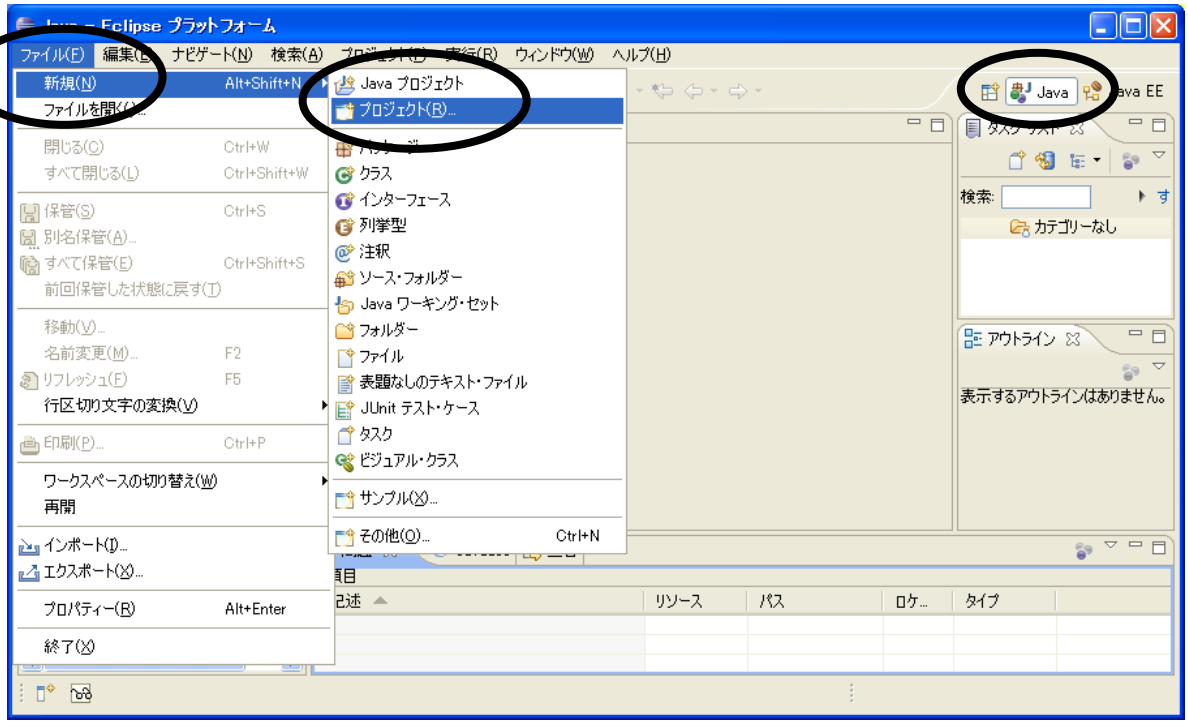
プロジェクトを作成するには、メニューから[ファイル]-[新規]-[プロジェクト]とたどって行きます。

ただし、現在表示されているパースペクティブによって、メニューの項目が若干異なるので注意してください。

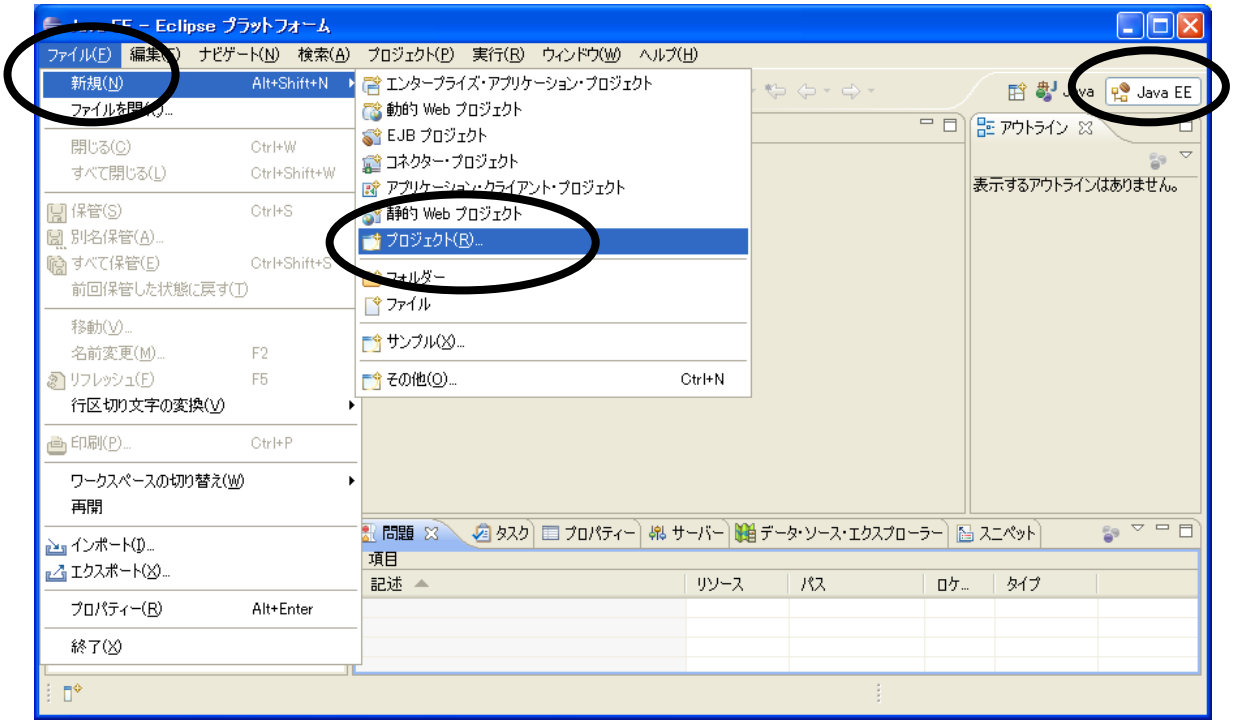
次のページに、[Java]パースペクティブと[JavaEE] パースペクティブの、[ファイル]-[新規]メニューを掲載します。

（セミナー環境ならば、[JavaEE] パースペクティブになっているかもしれません）

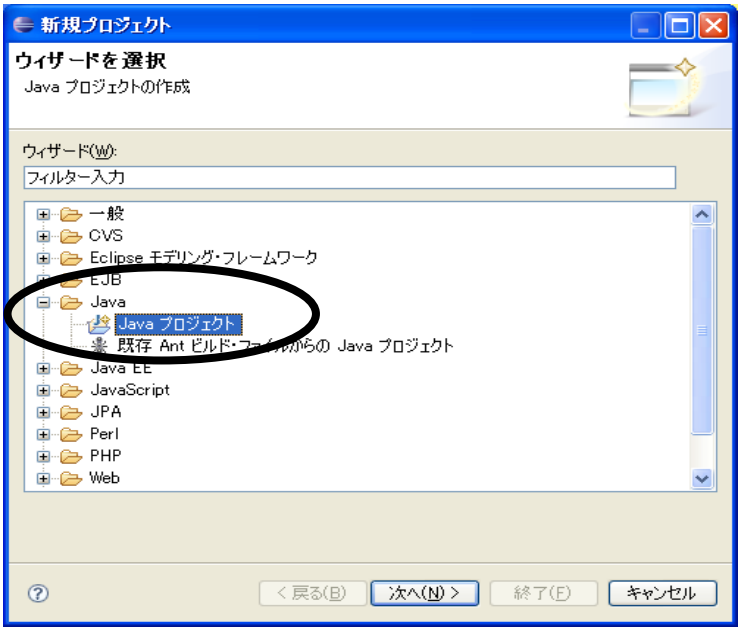
1. [Java]パースペクティブの[ファイル]-[新規]メニュー



2. [JavaEE] パースペクティブの[ファイル]-[新規]メニュー



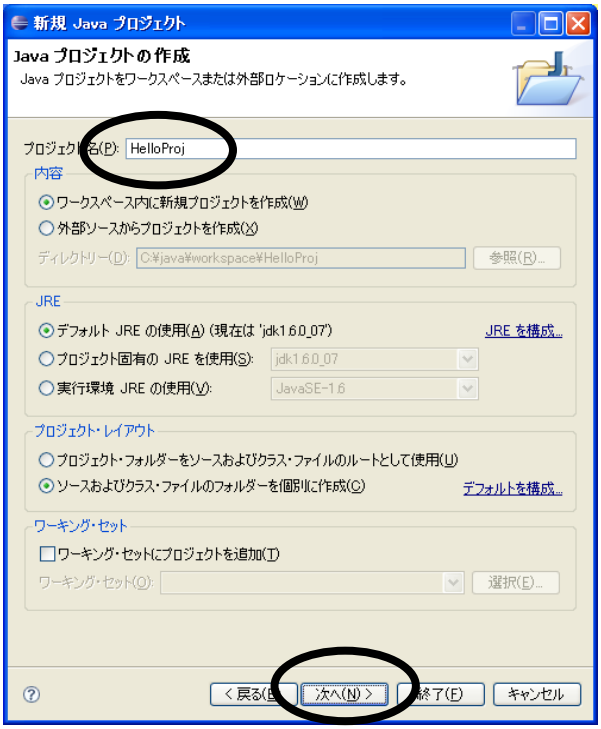
プロジェクトにはさまざまな種類がありますが、今回は、[Java]-[Java プロジェクト]を選びます。
選んだら、[次へ]ボタンをクリックします。



[プロジェクト名]には、毎回適切な名前を設定すべきですが、今回は簡単なサンプルなので、次のような名前にしました。

```
HelloProj
```

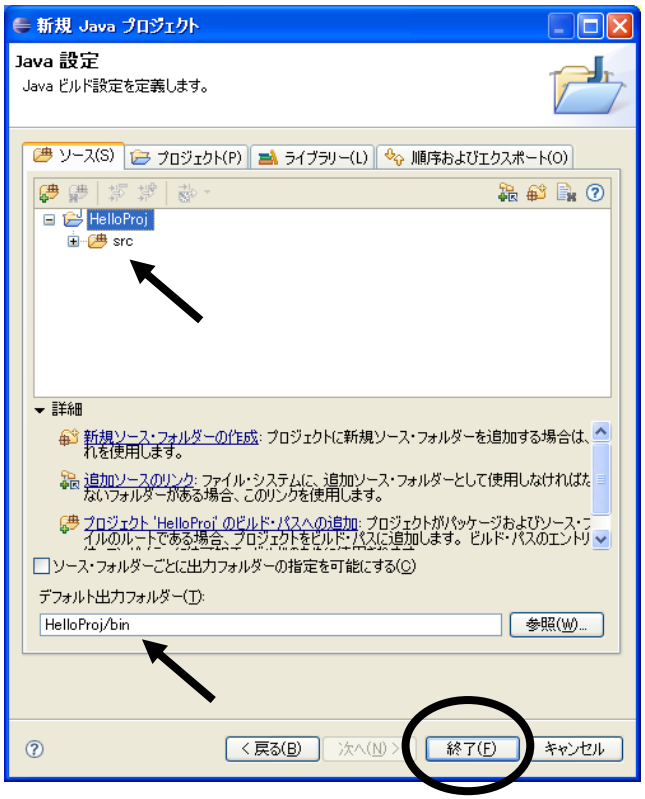
引き続いて、[次へ]ボタンをクリックします。
（慣れてきて、以降の画面で何をするか理解できてきたら、ここで、[終了]ボタンをクリックしてもかまいません）



次に、詳細な設定を行う画面が表示されますが、このまま[終了]ボタンをクリックしても構いません。（その場合は、前ページの画面で、[終了]ボタンをクリックしたことと同じになります）

もし、ここで何も変更せず[終了]ボタンをクリックすると、Java のソースファイル（*.java）とクラスファイル（*.class）が次のようなフォルダ内に作成されます。（注１、注２）

ソースファイル（*.java）	HelloProj/src
クラスファイル（*.class）	HelloProj/bin



注１：

出力フォルダ名は、Eclipse の場合、習慣的に bin という名前が使われますが、Java では classes という名前もよく使われます。

出力フォルダ名を変えたい場合は、上図の[デフォルト出力フォルダー]内を、直接編集し、"HelloProj/bin" ではなく、"HelloProj/classes" などと書き直しても構いません。
あるいは、上図で、[参照]ボタンをクリックして、classes フォルダをプロジェクト直下に作成してもかまいません。

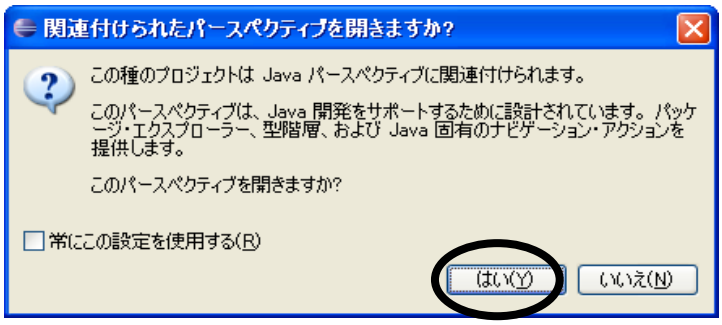
セミナーでは、自動生成された bin フォルダをそのまま使用します。

注２：

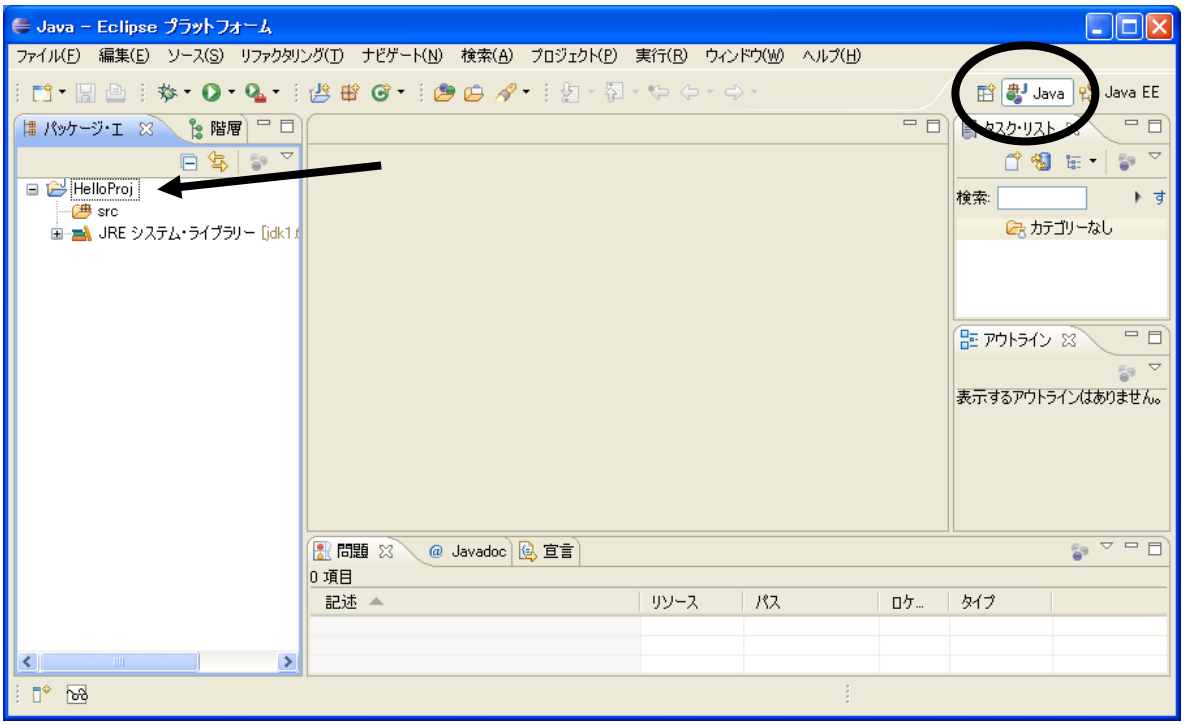
Eclipse3.2.x 以前のバージョンでは、ここで、自動的に src や bin というフォルダが作成されません。その場合は、上図の[新規ソース・フォルダの作成]というリンクボタンをクリックして、作成することができます。
なお、この節の最後では、ここで、src フォルダを作成しなかった場合に、あとから src フォルダを作成する方法を説明しています。

もし、現在、 [Java]以外のパースペクティブだと（例えば、[JavaEE] パースペクティブだと）、次のメッセージが表示されるので、[はい]をクリックしてください。

（[JavaEE] パースペクティブでも、Java アプリケーションの開発は可能ですが、ここでは混乱を避けるために、[Java] パースペクティブに統一します）

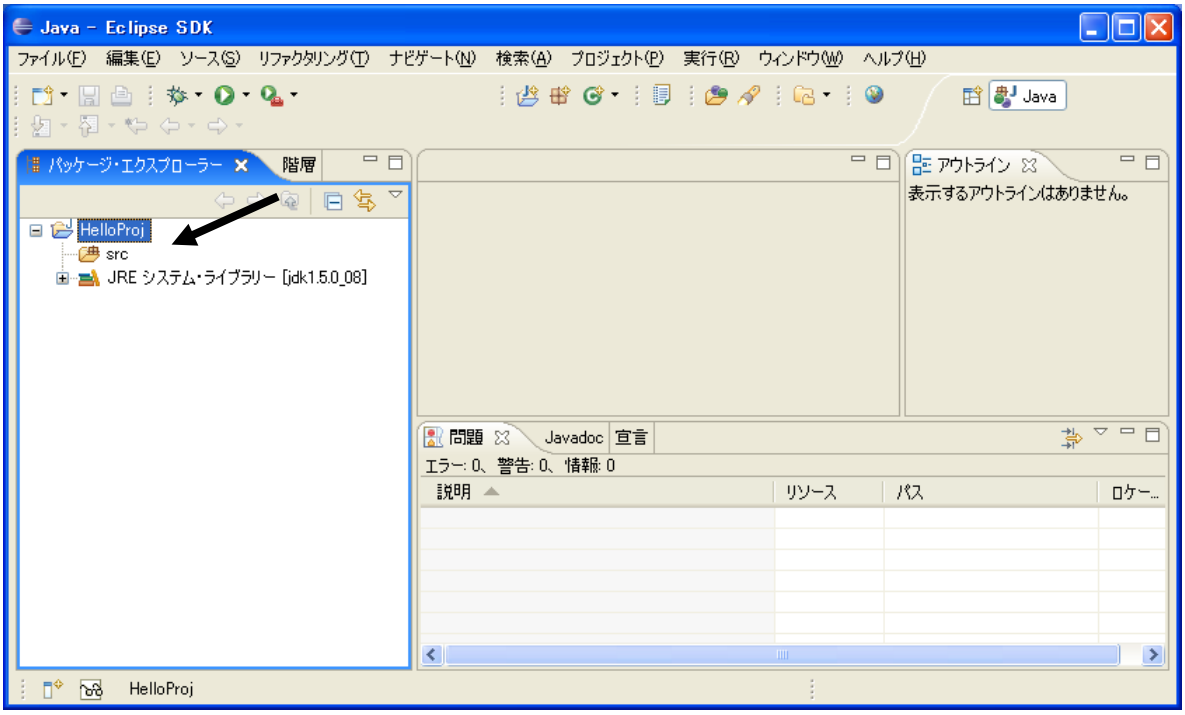


以上の操作で、HelloProj というプロジェクトが作成されました。



1.2.3.プロジェクトの確認

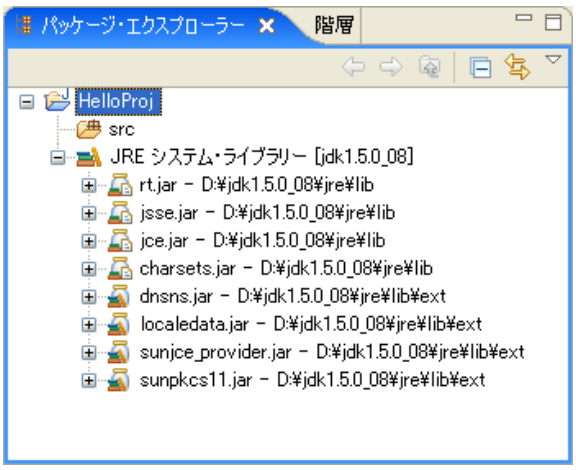
プロジェクトを作成すると、Java パースペクティブの[パッケージ・エクスプローラー]ビューに、プロジェクトの情報が表示されます。(下図)



通常、出力フォルダ(bin フォルダ)は、[パッケージ・エクスプローラー]ビューには表示されません。

また、[JRE システム・ライブラリー]という項目が作成されていますが、これは、Java のソースコードをコンパイルしたり実行したりするために必要な、JDK のライブラリファイル(*.jar)の一覧です。

(JDK のバージョンにより、表示されるファイルは異なります。下図は、JRE1.5 のものです)

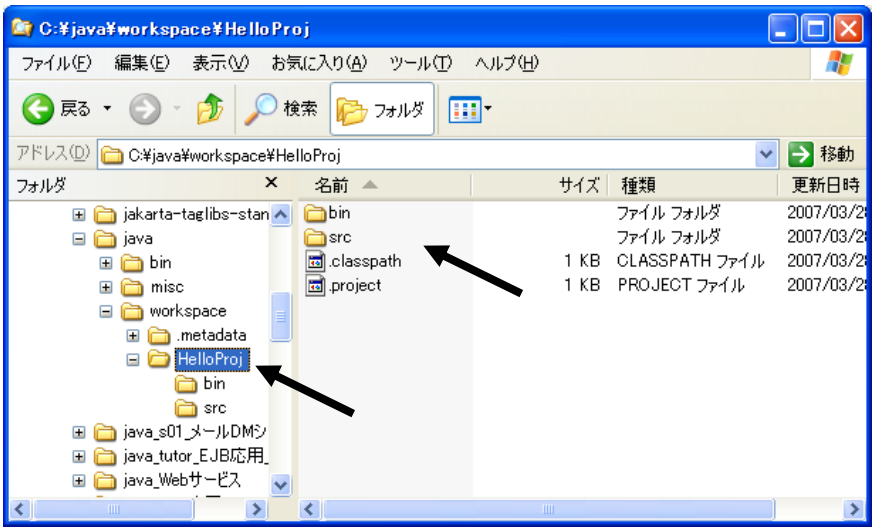


1.2.4.フォルダの確認

プロジェクトを作成したら、Windows のマイコンピュータやエクスプローラなどで、ワークスペースフォルダ（C:¥java¥workspace）の中を見て見ましょう。

プロジェクトフォルダが作成されているはずです。
さらに、そのフォルダ内に、src と bin も作成されています。

なお、プロジェクトフォルダ内の、.classpath や.project というファイルは、プロジェクトが必要なファイルです。削除しないように気をつけてください。



[備考]

Java パースペクティブと Java プロジェクトを混同している方はいませんか？
パースペクティブとは、画面上のウィンドウ（ペイン）のレイアウトのことで、プロジェクトとは、（画面とは関係なく）ファイルを管理する単位のことです。

つまり、Java プロジェクトを「PHP パースペクティブ」で開くこともできなくはないですし、PHP 用のプロジェクトを「Java パースペクティブ」で開くこともできます。

しかし、Java プロジェクトを「PHP パースペクティブ」で開いてしまうと、プログラム開発が困難になってしまうので、通常は、適したパースペクティブで開くようにします。

1.3. クラスの追加（ソースコードの記述）

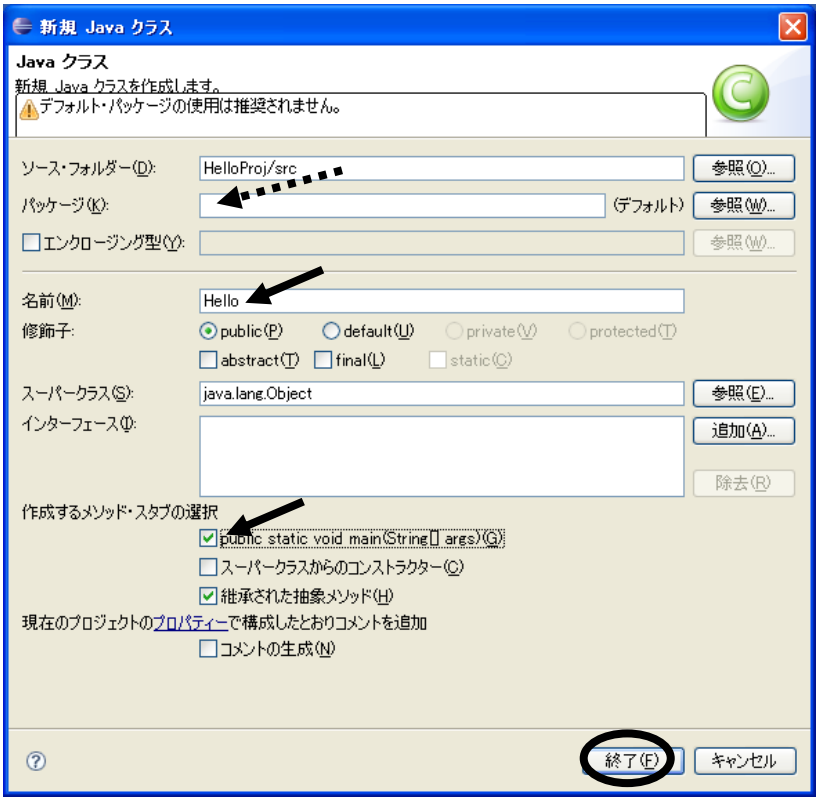
1.3.1.クラスの追加

プロジェクトを作成したら、Java のソースコードを記述できるようになります。

[パッケージ・エクスプローラー]で、プロジェクト（HelloProj）の src フォルダを[右クリック]し、[新規]-[クラス]を選択します。
あるいは、メニューから[ファイル]-[新規]-[クラス]とたどって行くこともできます。

すると、下図のようなウィンドウが開くので、次のように入力します。

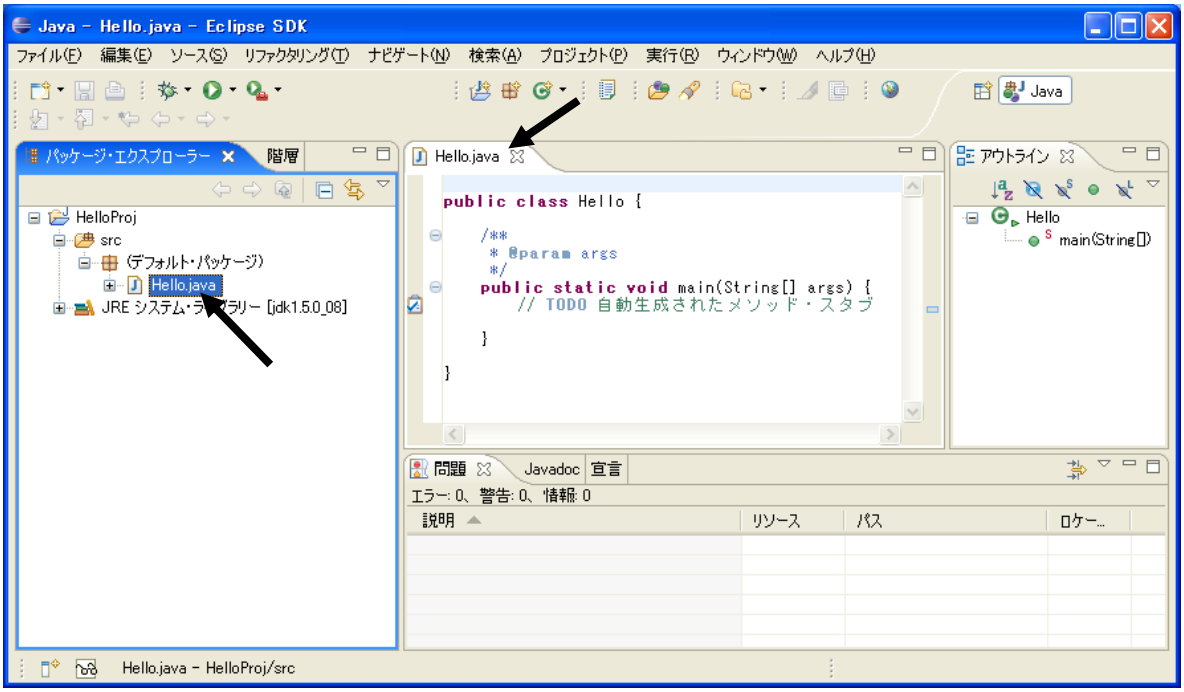
パッケージ(K)：　　（空白のまま）
名前(M)：　　***Hello***
作成するメソッド・スタブの選択：
　　public static void main(String[] args) にチェックを付ける



パッケージ名を空白のままにすると「デフォルトパッケージの使用は推奨されません」という警告が表示されますが、この時点ではまだ、「パッケージ」という概念を学習していない場合があるので、空白のままで結構です。

"Hello.java"というファイルが、プロジェクトに追加されました。(下図の[パッケージ・エクスプローラー]参照)

また、"Hello.java"のソースコードが、エディタで編集できるようになりました。(下図の中央部分が「エディタ」です)



下記は、Eclipse3.2.x によって自動生成されたコードです。(コメントは斜体にしてあります)

```
public class Hello {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ

    }

}
```

なお、このプリントでは、煩雑さを避けるため、自動生成されたコメントの部分は省略して表示することがあります。

1.3.2.コードの編集

例えば、下記のように、コードを追加してみます。
（下記のようでなく、お好きな文字列を表示するように記述してかまいません）

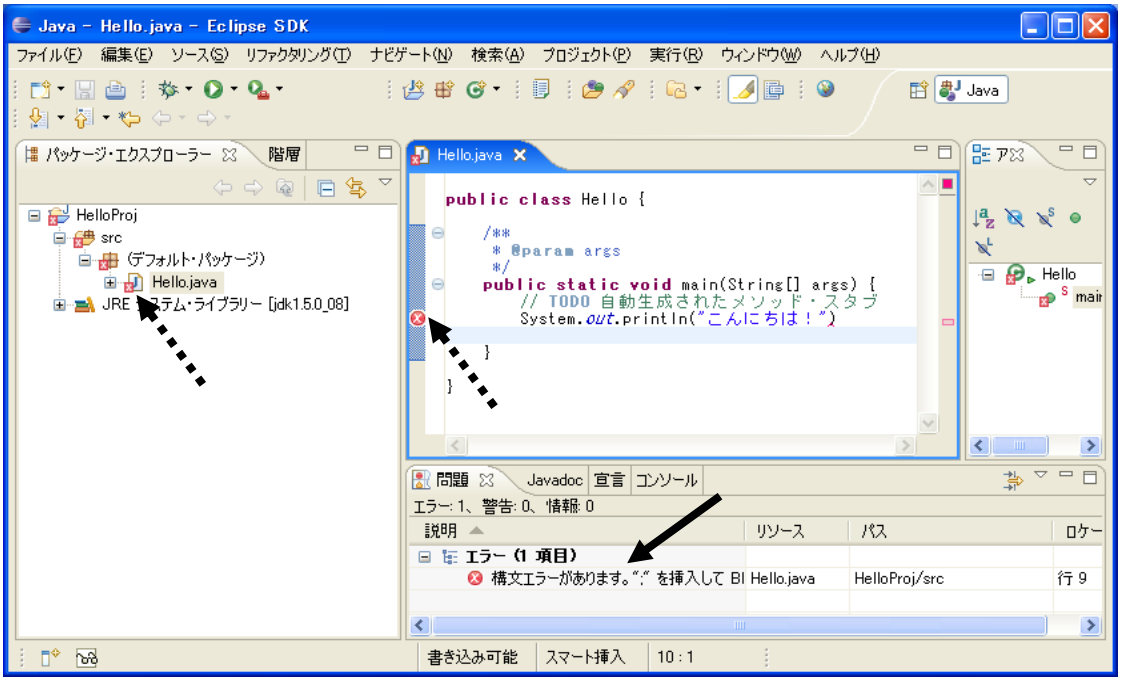
```
//この部分に、コメントが自動生成されていますが、このプリントでは、省略します。

public class Hello {

    public static void main(String[] args) {
        System.out.println("こんにちは！");
    }
}
```

1.3.3.コンパイルとエラーの確認

Eclipse では、コンパイルというメニュー項目はありません。ファイルを保存するとコンパイルも同時に行われます。

その時、もしコンパイルエラーがあると、下図のように、[問題]ビューにエラーメッセージが表示され、[パッケージ・エクスプローラー]やエディターにも、赤い×印のアイコンが表示されます。
（下図は、セミコロンを書き忘れた場合のエラーです）

エラーがなくなるまで、コードを修正してください。
なお、[問題]ビューで、エラーメッセージをダブルクリックすると、ソースコードのエラー行までジャンプしてくれます。

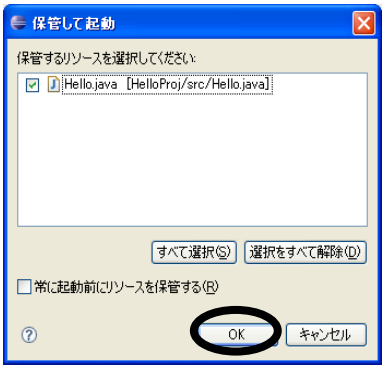


1.4. プログラムの実行

1.4.1.アプリケーションの実行

[パッケージ・エクスプローラー]で Hello.java を[右クリック]などして、[実行]-[Java アプリケーション]をクリックすると、選択したクラス（Hello.java）を実行することができます。

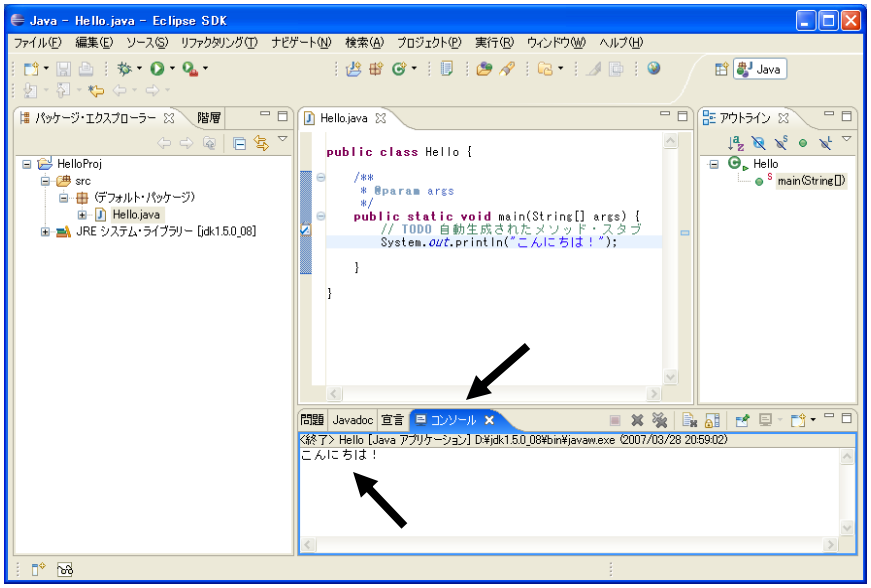
実行する前に、保存していないファイルがあると、次のように聞いてきますので、保存したいファイルにチェックを付け、[OK]ボタンをクリックします。



1.4.2.実行結果

System.out.println()は、「標準出力」と呼ばれる場所に出力しますが、Eclipse では、[コンソール]ビューが標準出力に対応付けられています。

[コンソール]ビューを見ると、「こんにちは！」などと、プログラムの結果が表示されているのがわかります。



（空白ページ）

第2章. プロジェクトのインポート

2.1. プロジェクトのインポート

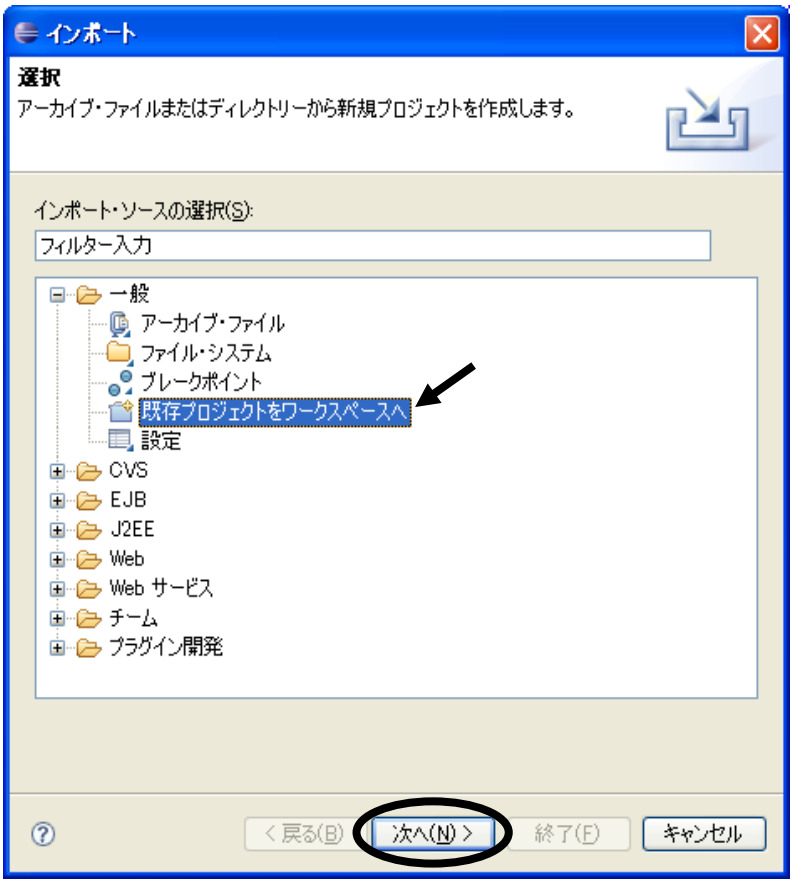
2.1.1.説明

研修で作成したプロジェクトや他の人が作成したプロジェクトを、現在作業中のワークスペースにコピー（インポート）する方法について述べます。

2.1.2.操作方法

メニューで[ファイル]-[インポート]を選択します。

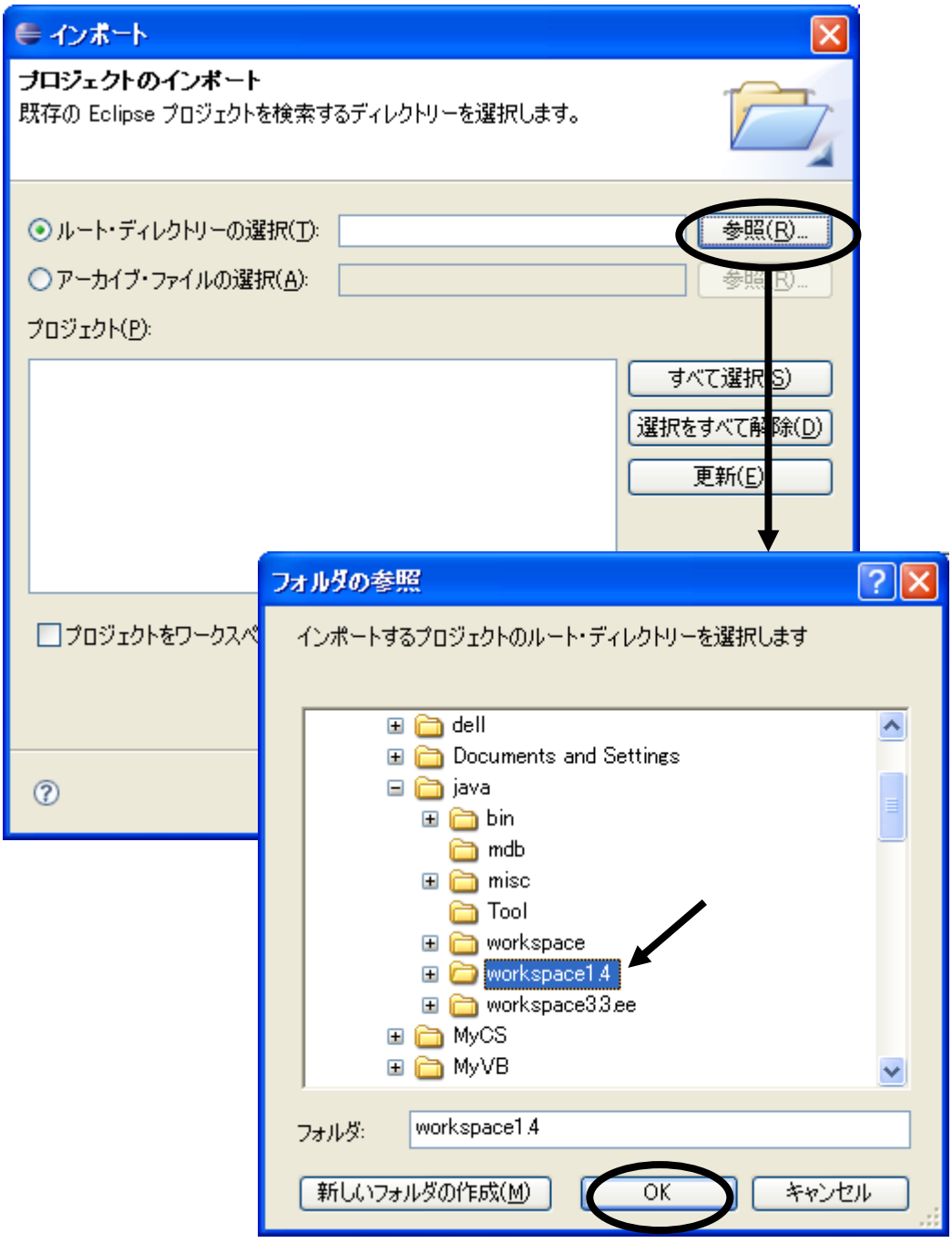
下記の画面で、[一般]-[既存プロジェクトをワークスペースへ]を選択し、[次へ]ボタンをクリックします。



[ルート・ディレクトリーの選択]の右側にある[参照]ボタンをクリックし、[フォルダの参照]ダイアログを開きます。

[フォルダの参照]ダイアログでは、インポートしたいプロジェクトが存在するディレクトリを選択し、[OK]ボタンをクリックします。

下図では、"C:¥Java¥workspace1.4"というフォルダを選択しています。



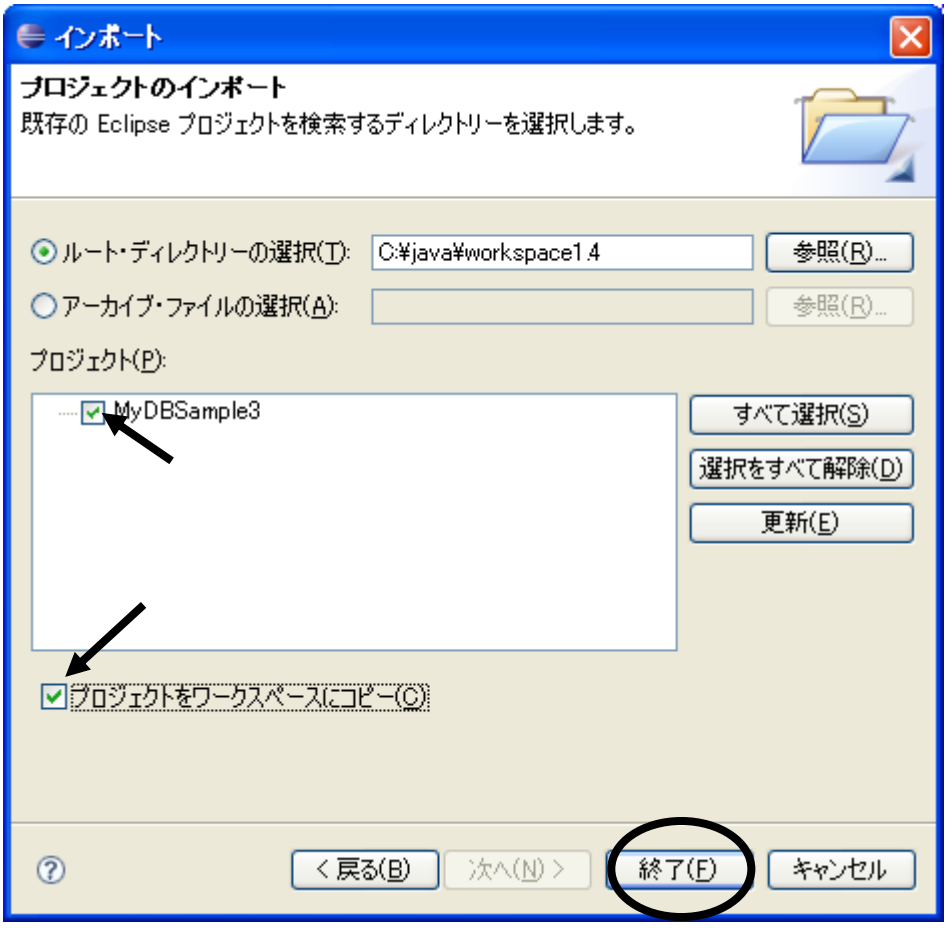
[プロジェクト]ボックスに、選択されたディレクトリ内のプロジェクト一覧が表示されるので、インポートしたいプロジェクトにチェックを付けます。

次に、[プロジェクトをワークスペースにコピー]のチェックを必要に応じて付けます。
チェックを付けたほうがいいでしょう。

このチェックボックスにチェックを付けると、選択されたプロジェクトのコピーが現在のワークスペース用のディレクトリ（C:¥Java¥workspace）にコピーされます。

チェックボックスのチェックを外すと、現在のワークスペース内にプロジェクトはコピーされず、もともとのフォルダのものが使用されます。

従って、workspace 以外のフォルダからインポートする場合はチェックを付け、workspace フォルダ（C:¥Java¥workspace）のものを使用する場合は、チェックを外すといいでしょう。



以上で、既存プロジェクトのインポート方法の説明は終わりです。