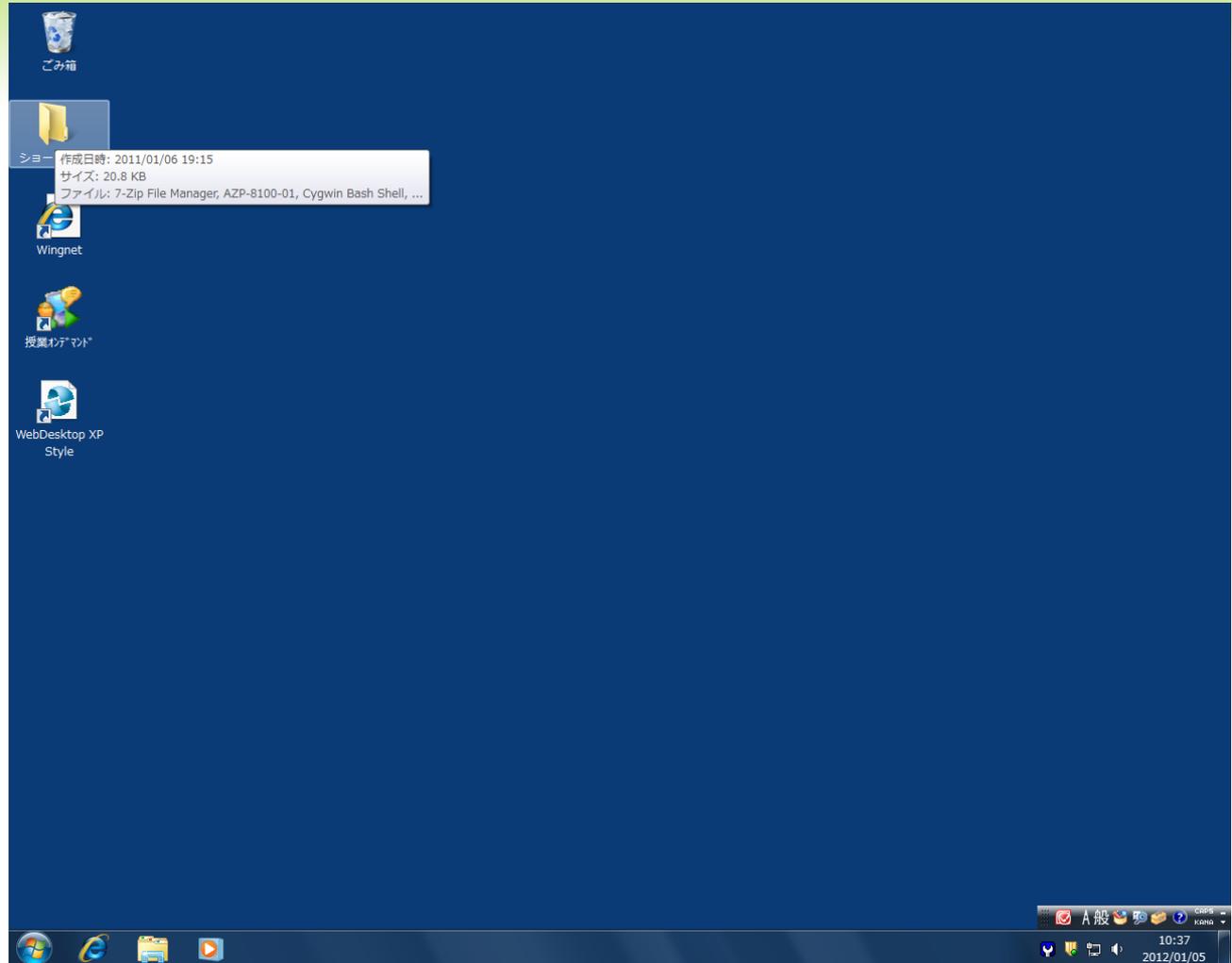


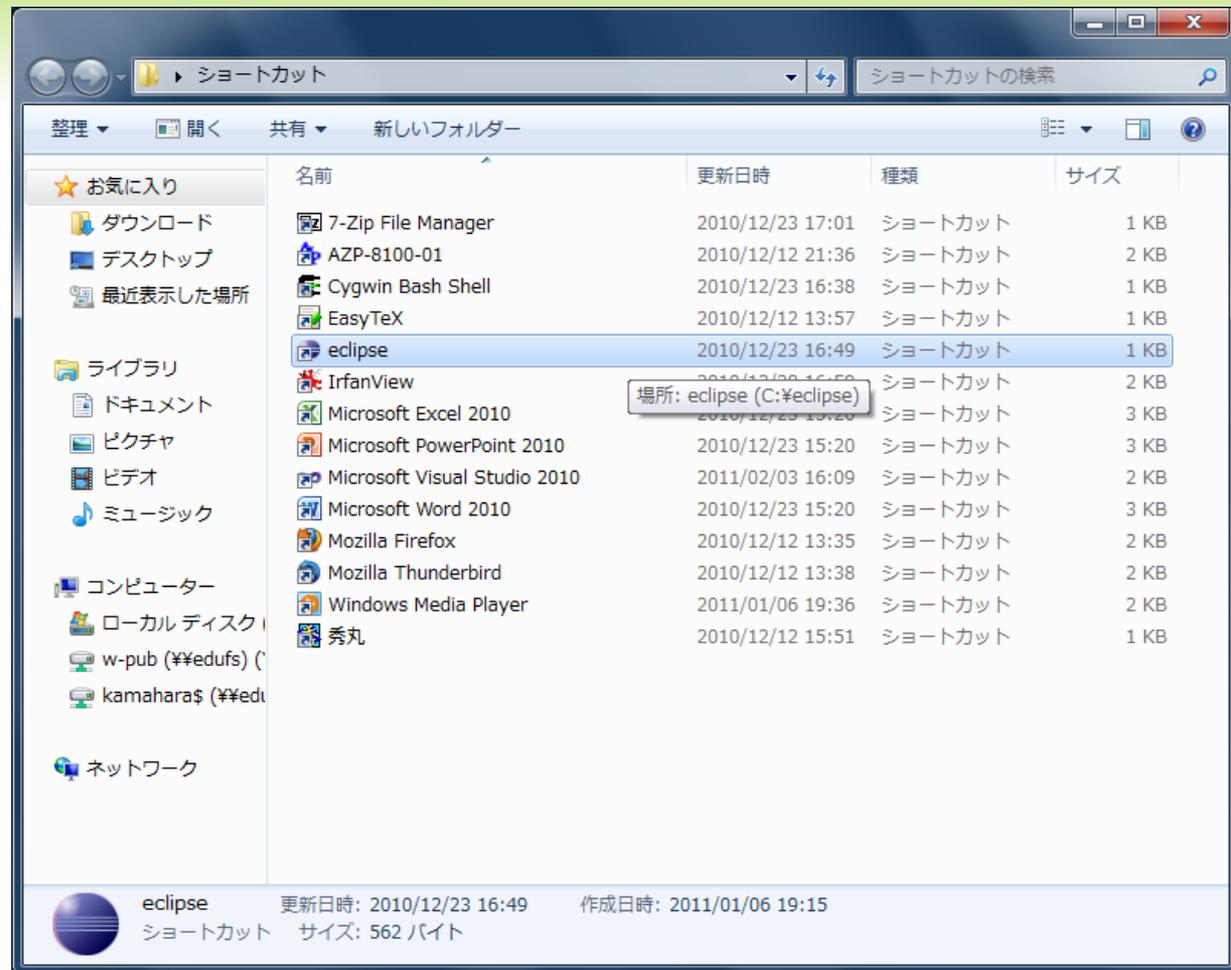
情報処理演習 8

鎌原淳三 kamahara@port.kobe-u.ac.jp

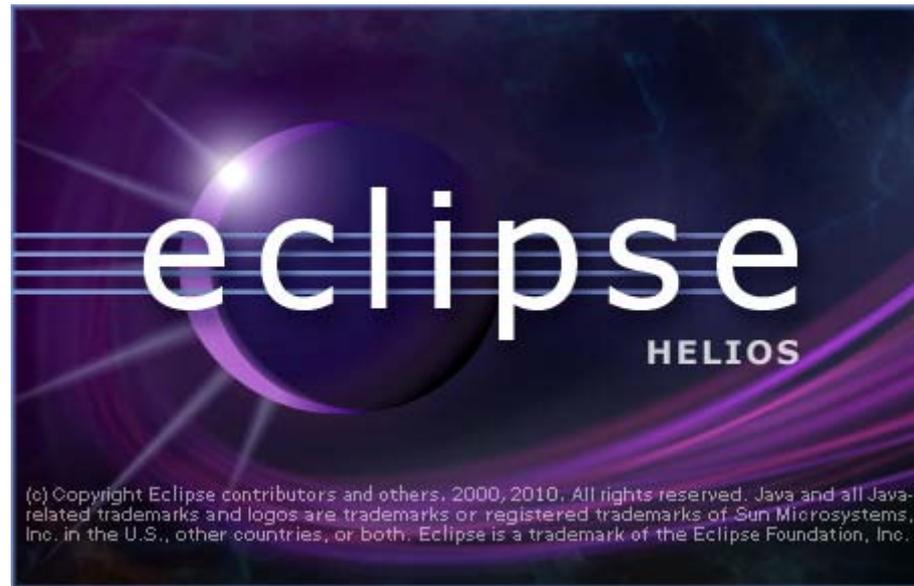
ECLIPSEを使う - ショットカットを開く



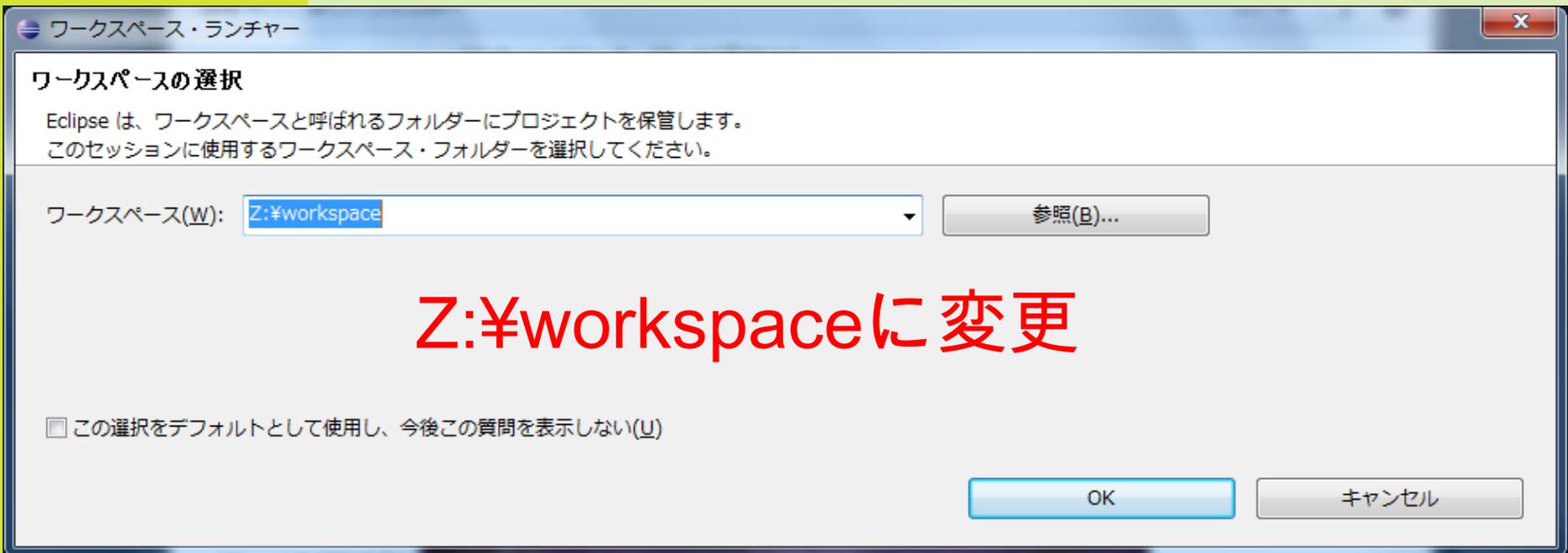
ECLIPSEをダブルクリック



起動中

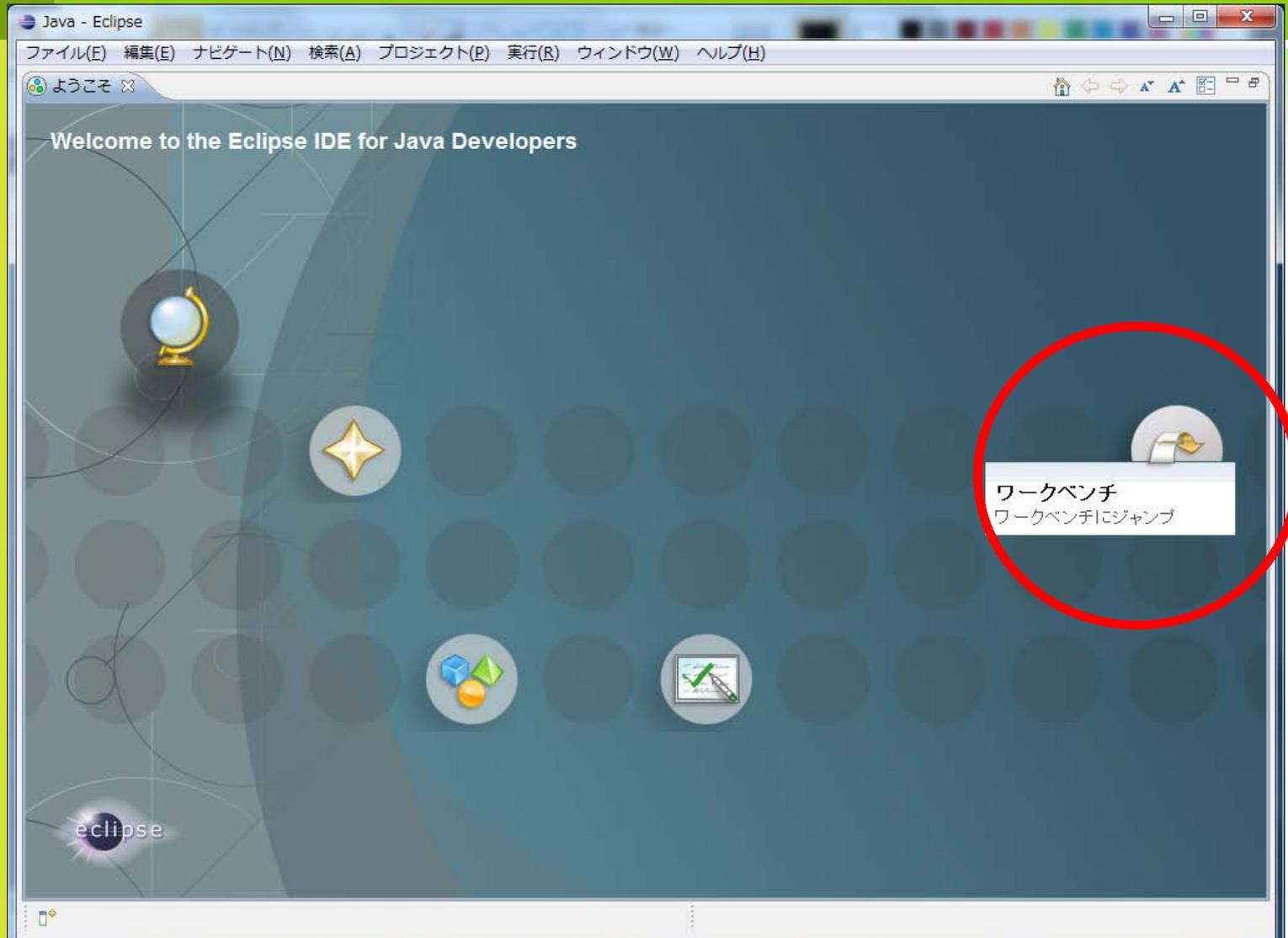


ワークスペースを指定する

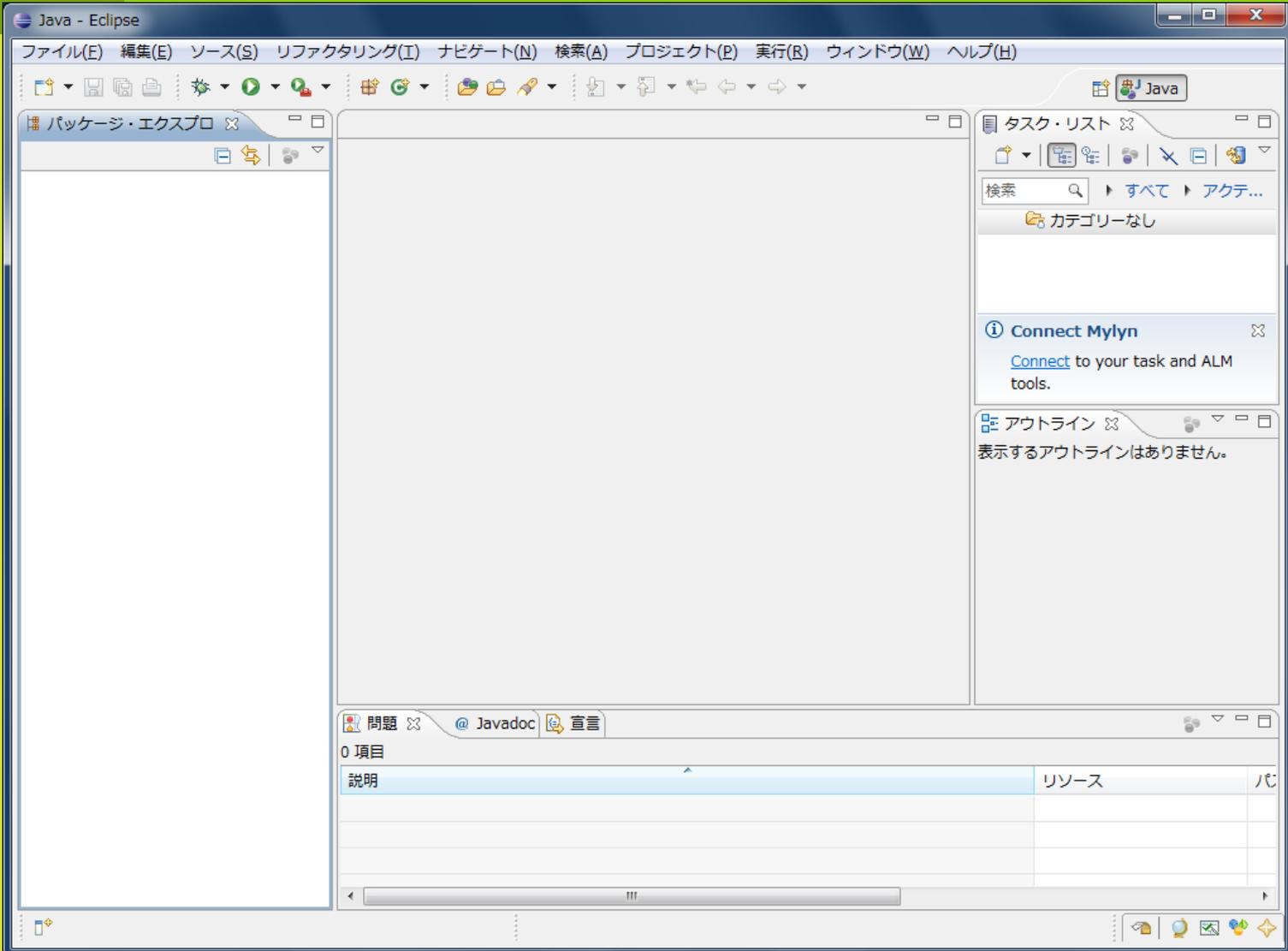


Z:¥workspaceに変更

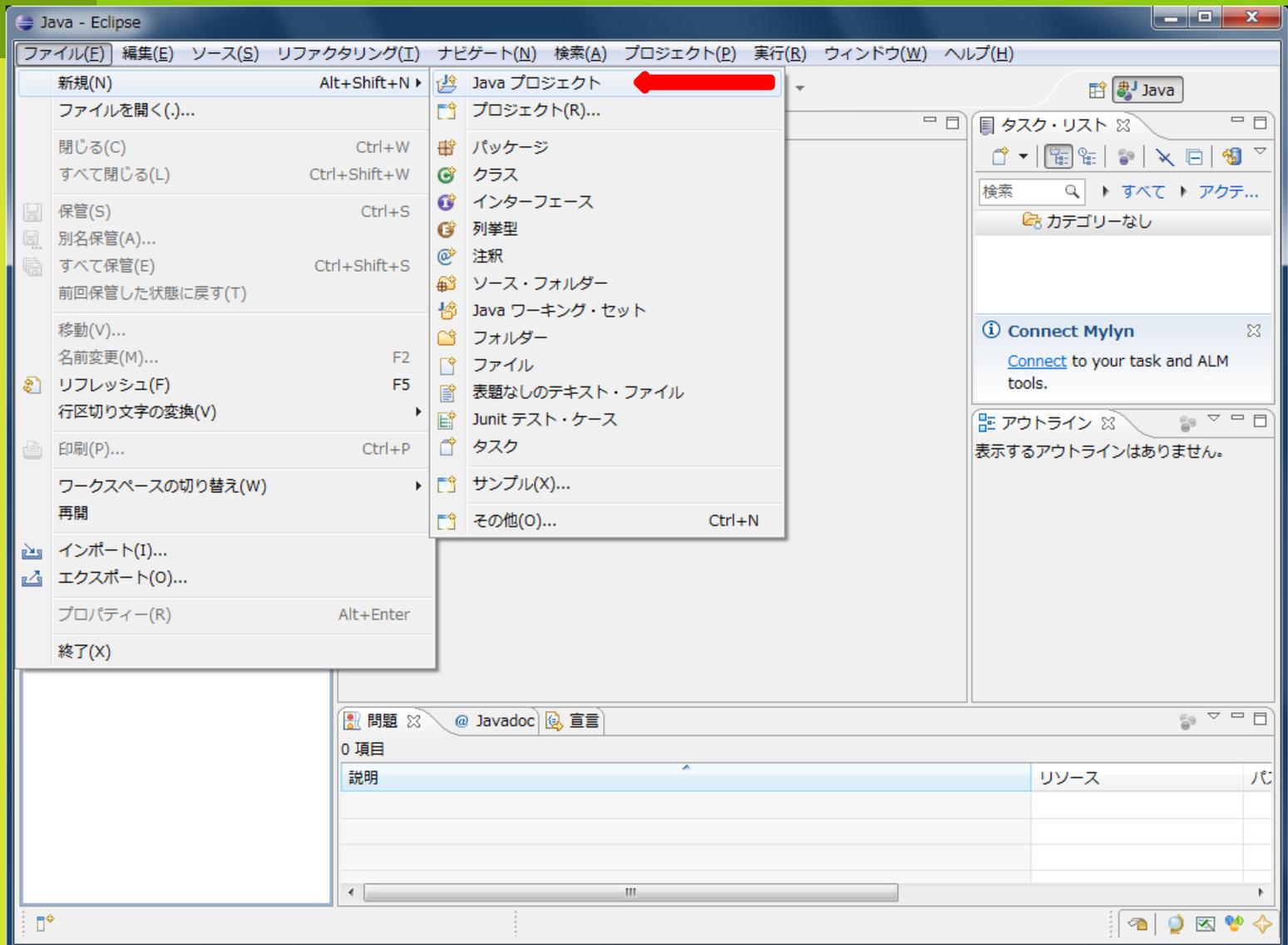
ワークベンチにジャンプ をクリック



コンピュータを学ぶの データが残っているかも



プロジェクトを作る



プロジェクト名を入力 "TESTPROJECT"

新規 Java プロジェクト

Java プロジェクトの作成

Java プロジェクトをワークスペースまたは外部ロケーションに作成します。

プロジェクト名(P):

デフォルト・ロケーションを使用(D)

ロケーション(L): [参照\(B\)...](#)

JRE

実行環境 JRE の使用(V):

プロジェクト固有の JRE を使用(S):

デフォルト JRE の使用(A) (現在は 'jre6') [JRE を構成...](#)

プロジェクト・レイアウト

プロジェクト・フォルダーをソースおよびクラス・ファイルのルートとして使用(U)

ソースおよびクラス・ファイルのフォルダーを個別に作成(C) [デフォルトを構成...](#)

ワーキング・セット

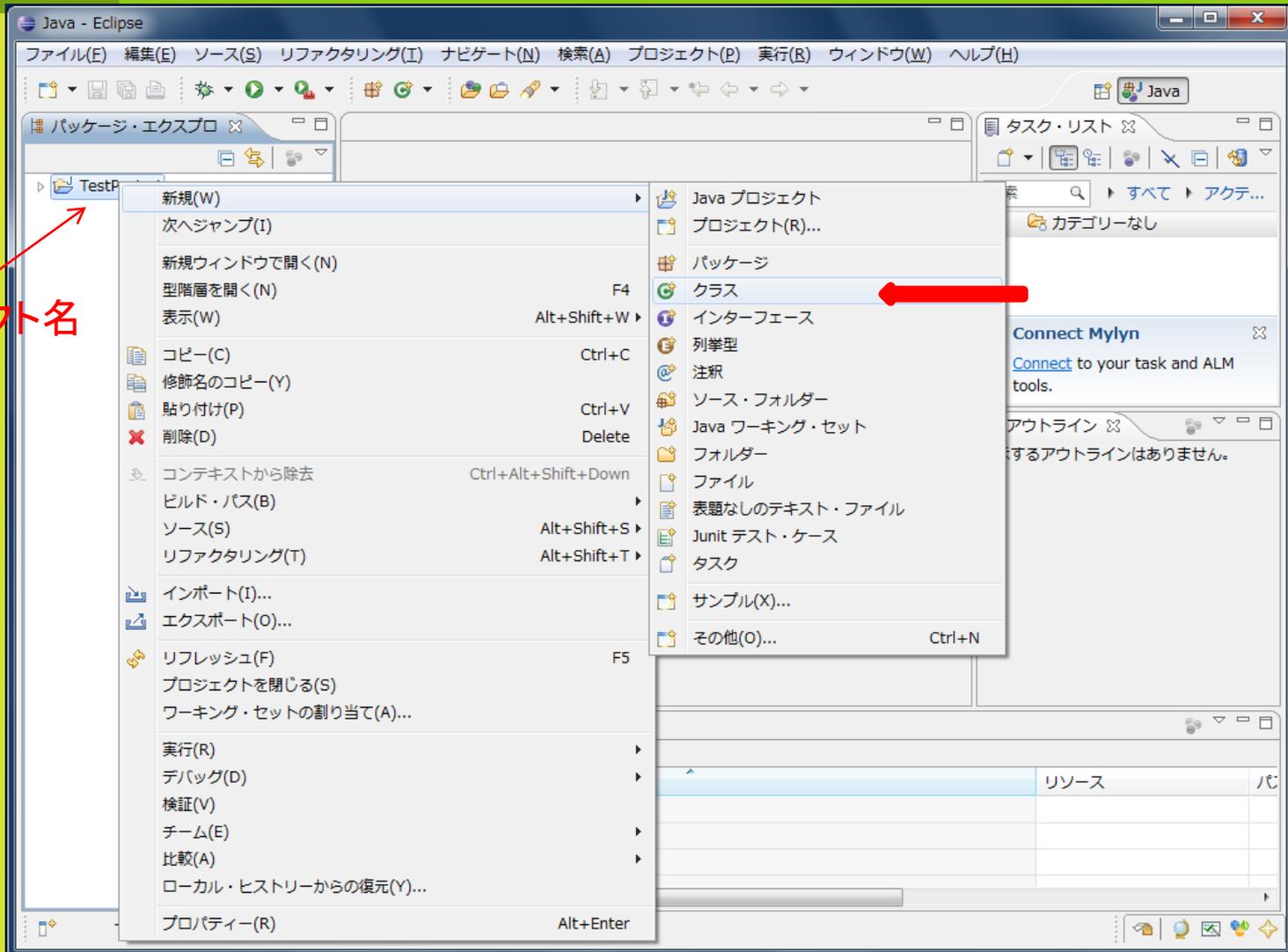
ワーキング・セットにプロジェクトを追加(I)

ワーキング・セット(Q): [選択\(E\)...](#)

[?<](#) [< 戻る\(B\)](#) [次へ\(N\) >](#) [完了\(F\)](#) [キャンセル](#)

新規クラスを作成 プロジェクト名上で右クリック

プロジェクト名



クラス名を入力 "SAMPLE1"

新規 Java クラス

Java クラス

⚠ デフォルト・パッケージの使用は推奨されません。

ソース・フォルダー(D): TestProject/src 参照(O)...

パッケージ(K): (デフォルト) 参照(W)...

エンクロージング型(Y): 参照(W)...

名前(M): ① Sample1

修飾子: public(P) default(U) private(V) protected(I) ②

abstract(I) final(L) static(C)

スーパークラス(S): java.lang.Object 参照(E)...

インターフェース(I): 追加(A)...

除去(R)

どのメソッド・スタブを作成しますか?

public static void main(String[] args)(V)

スーパークラスからのコンストラクター(C)

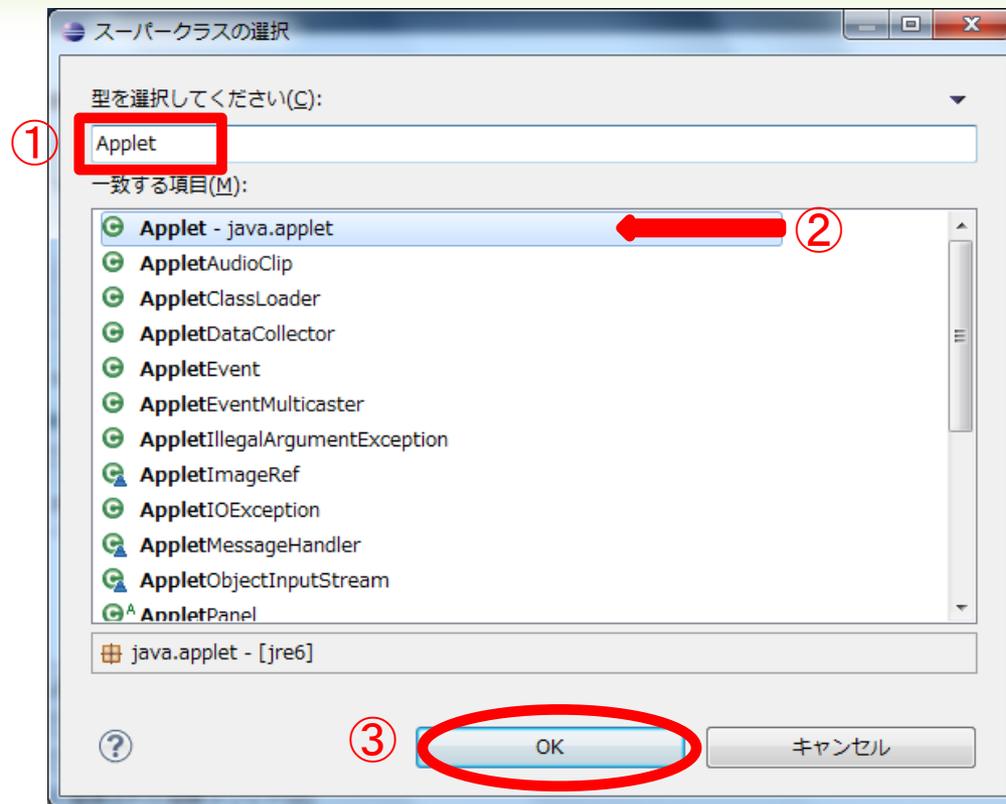
継承された抽象メソッド(H)

コメントを追加しますか? (テンプレートの構成およびデフォルト値については[ここ](#)を参照)

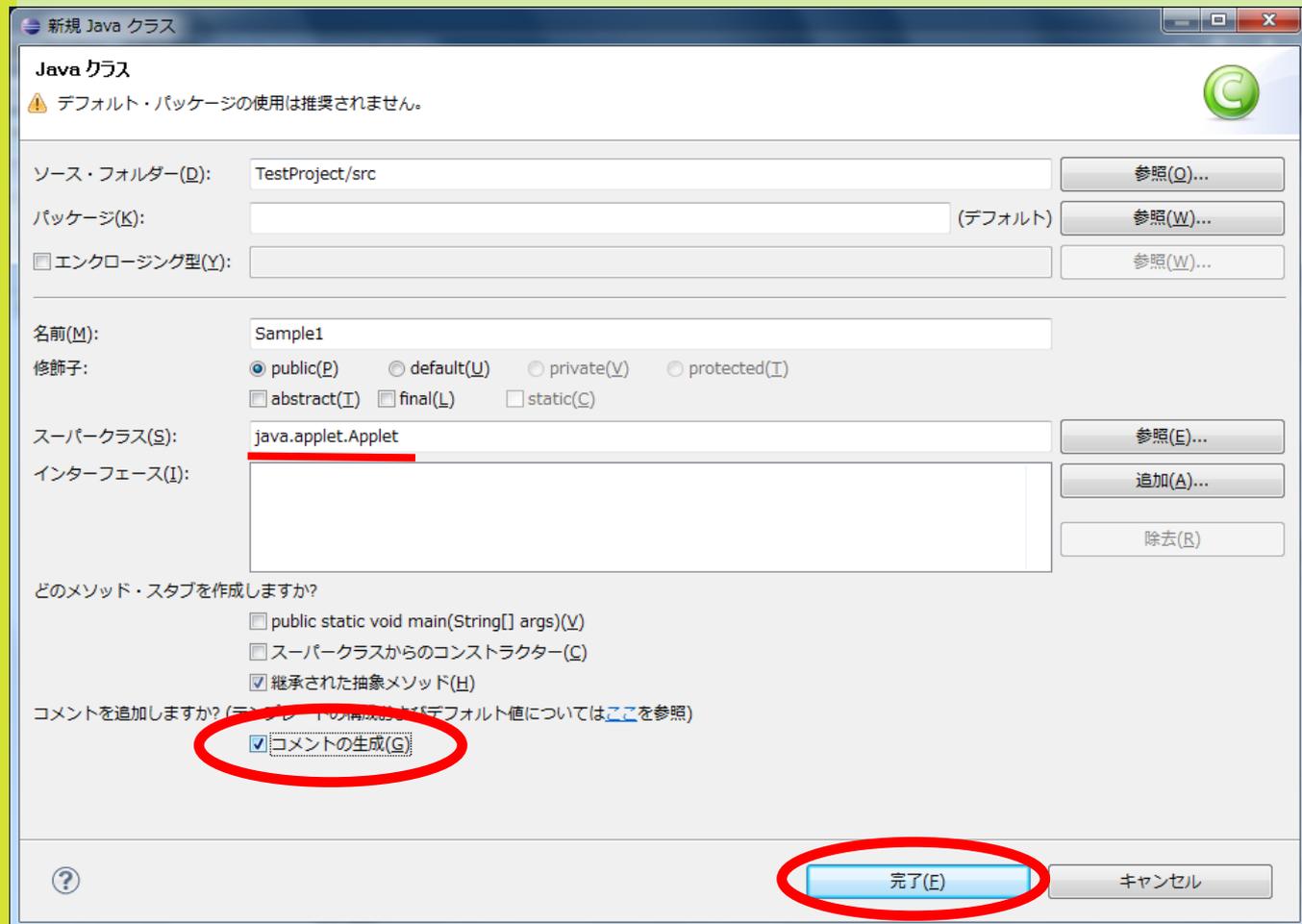
コメントの生成(G)

? 完了(F) キャンセル

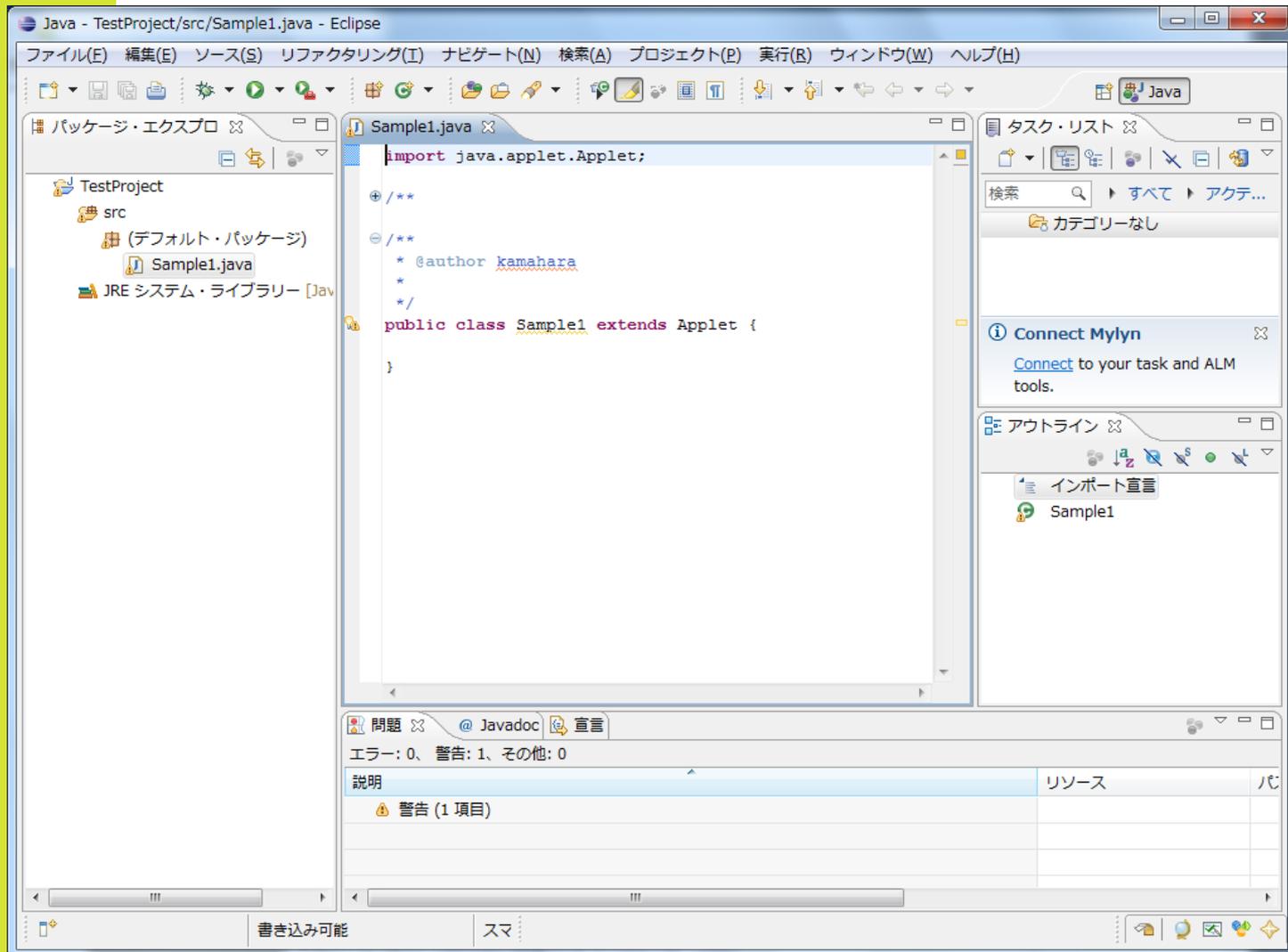
スーパークラスを選択 “APPLET”を入力



スーパークラスが変わっている



生成された画面



APPLET

- ◎ ウェブ上に表示ができるJavaのプログラム

APPLETは自動的に開始される

```
import java.applet.Applet;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
    }
}
```

入力する

init()関数は初期設定をする関数

説明	リソース	バ
警告 (1 項目)		

アプレットに文字表示を行うプログラム

- ◎ Appletには、「部品」を貼り付けることで画面を作成する
- ◎ 文字表示などを行うのはLabel
- ◎ Labelでは文字だけでなくアイコンも表示できる

新しいLABELを作る

```
import java.applet.Applet;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
        Label lb = new Label("0");
    }
}
```

クリックしてちょっと待つ

```
Label lb = new Label("0");
```

問題 @ Javadoc 宣言
エラー: 0、警告: 1、その他: 0
説明 リソース パ
警告 (1 項目)

書き込み可能 スマ

LABEL を使えるようにインポートする

The screenshot shows the Eclipse IDE interface. The main editor displays the following code in `Sample1.java`:

```
import java.applet.Applet;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
        this.add(new Label("0"));
    }
}
```

A tooltip is visible over the `Label` class name in the `init()` method. The tooltip contains the following options:

- Label をインポートします (java.awt)
- クラス 'Label' を作成します
- LabelUI に変更します (javax.swing.plaf)
- LabelView に変更します (javax.swing.text)
- Level に変更します (java.util.logging)
- ファイル内の名前変更 (Ctrl+2, R)
- プロジェクト・セットアップの修正...

A red arrow points to the first option, "Label をインポートします (java.awt)". To the right of the tooltip, the text "ダブルクリックする" (Double-click) is written in red.

At the bottom of the IDE, the Error Console shows two error messages:

- エラー (2 項目)
- Label を型に解決できません
- Label を型に解決できません

The bottom status bar shows the text "Label...きません" and "書き込み可能".

JAVA.AWT.LABELをIMPORT

The screenshot shows the Eclipse IDE interface. The main editor displays the following Java code:

```
import java.applet.Applet;
import java.awt.Label;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
        Label lb = new Label("0");
    }
}
```

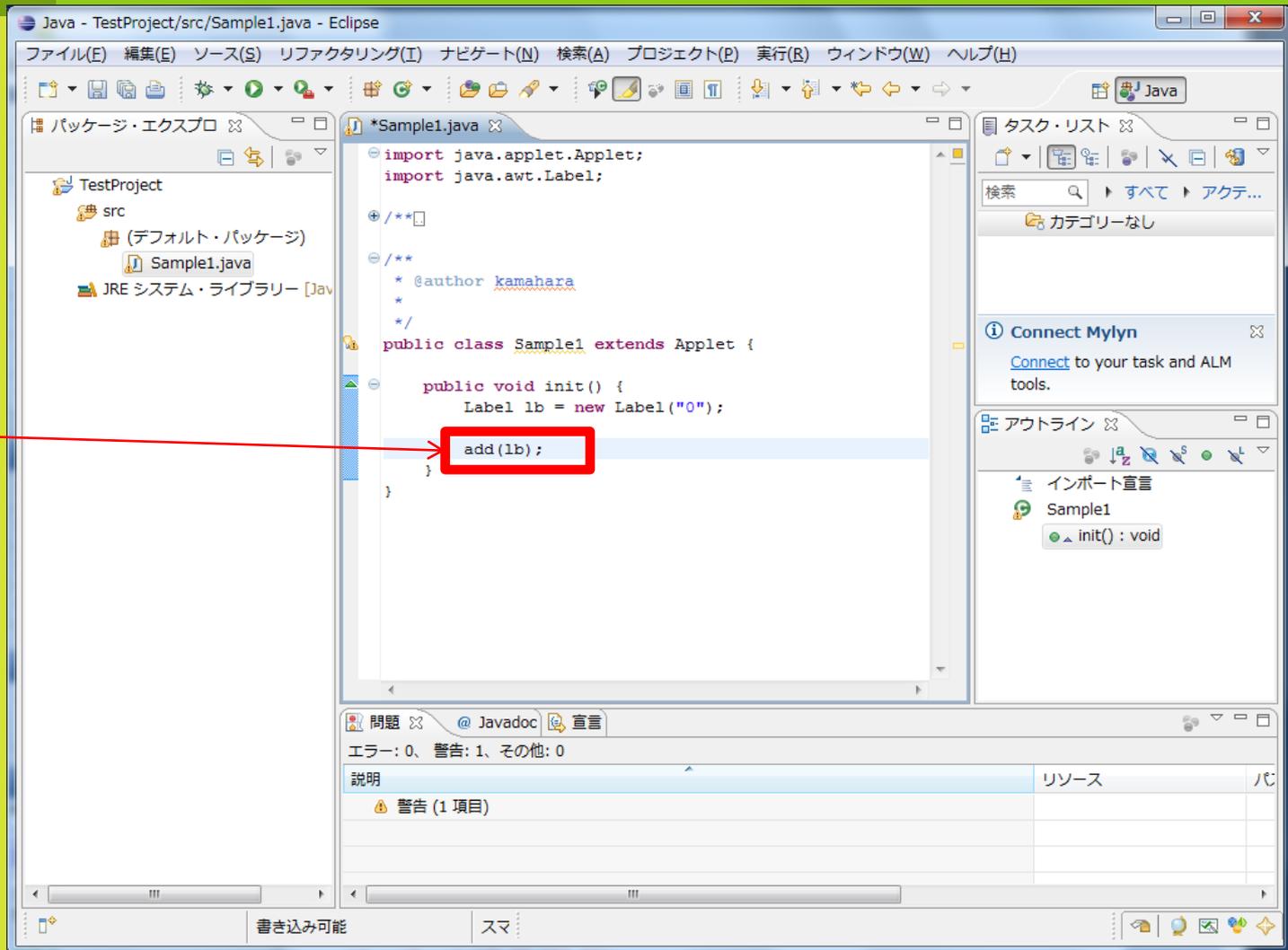
A red box highlights the `import java.awt.Label;` line, with the Japanese text "ここに追加された" (Added here) written in red above it. The IDE's status bar at the bottom shows two errors:

- エラー (2 項目)
- Label を型に解決できません
- Label を型に解決できません

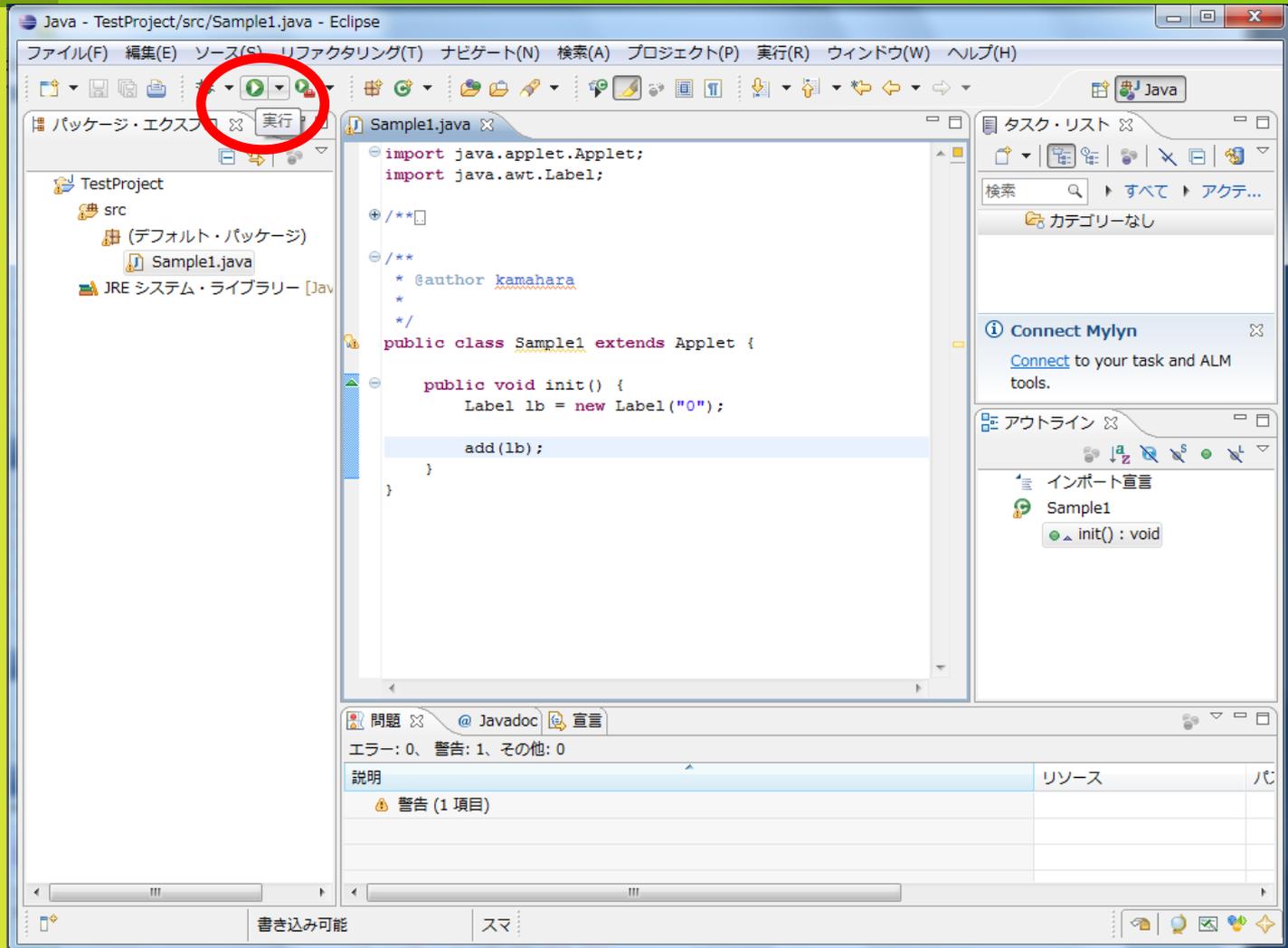
The error messages are linked to the `Label` class in the code. The right-hand side of the IDE shows the 'Task List' and 'Outline' views, and the bottom status bar indicates '書き込み可能' (Writeable).

ADD()でLABELをアプレットに貼り付ける

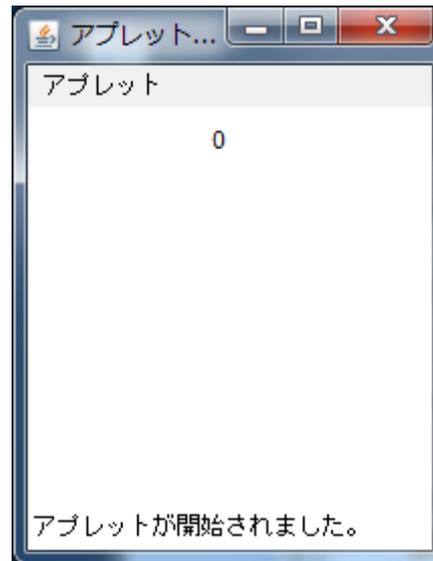
add(lbl);



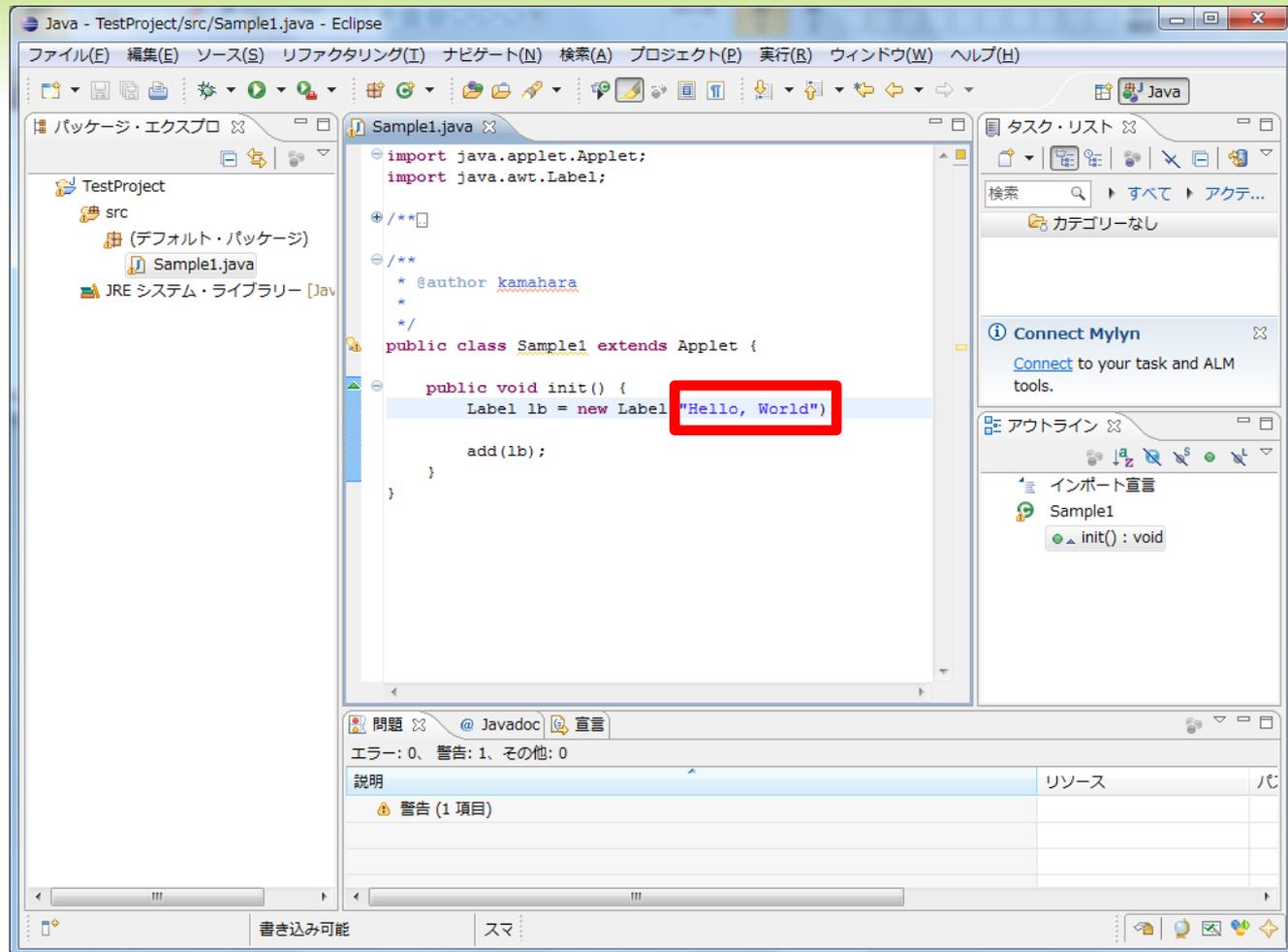
実行ボタンで実行



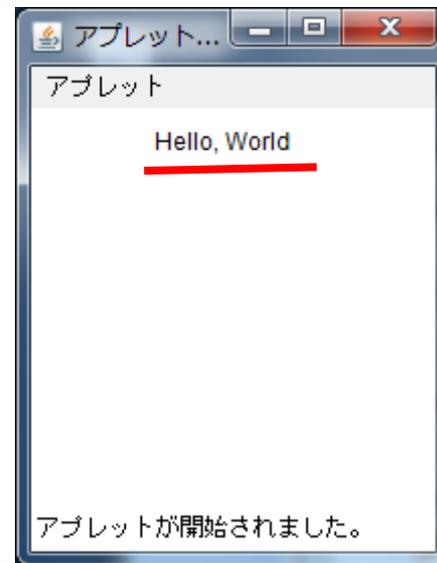
アプレットが表示



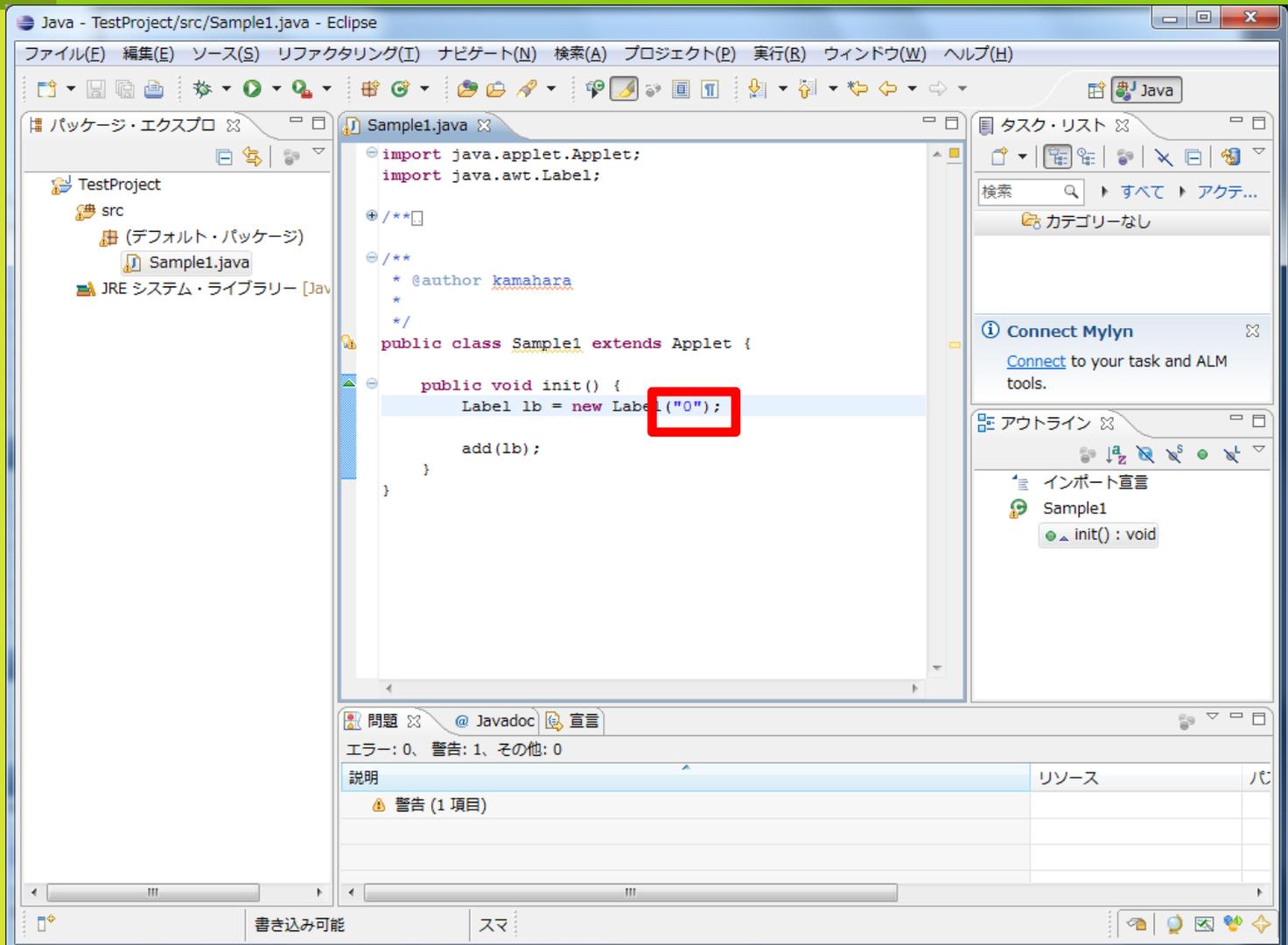
0を文字にしてみる “HELLO, WORLD”



表示されるか確認



0に戻る



Java - TestProject/src/Sample1.java - Eclipse

ファイル(E) 編集(E) ソース(S) リファクタリング(I) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)

パッケージ・エクスプロー

TestProject

- src
- (デフォルト・パッケージ)
- Sample1.java
- JRE システム・ライブラリー [Java]

```
import java.applet.Applet;
import java.awt.Label;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
        Label lb = new Label("0");

        add(lb);
    }
}
```

タスク・リスト

検索 すべて アクテ...

カテゴリーなし

Connect Mylyn

Connect to your task and ALM tools.

アウトライン

- インポート宣言
- Sample1
 - init() : void

問題 @ Javadoc 宣言

エラー: 0、警告: 1、その他: 0

説明	リソース	バ
警告 (1 項目)		

書き込み可能 スマ

ボタンを追加する

- ◎ 反応してラベルを書き換えるボタンを作る

BUTTONを作成

②
クリックしてちょっと待つ

```
import java.applet.Applet;
import java.awt.Label;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
        Label lb = new Label("0");
        Button bt1 = new Button("1");
        add(lb);
    }
}
```

①

Button bt1= new Button("1");

問題 @ Javadoc 宣言
エラー: 0、警告: 1、その他: 0
説明 リソース バ
警告 (1 項目)

BUTTONをインポート

The screenshot shows the Eclipse IDE interface. The main editor window displays the code for `Sample1.java`, which includes imports for `java.applet.Applet` and `java.awt.Label`. The code defines a class `Sample1` that extends `Applet` and has an `init()` method that creates a `Label` and a `Button`. The `Button` line is highlighted in blue. A context menu is open over the `Button` line, listing various options for importing or creating the class. The first option, `Button` をインポートします (java.awt), is highlighted with a red arrow. Red text `ダブルクリックする` is overlaid on the menu, indicating that a double-click on this option will import the class. The Package Explorer on the left shows the project structure, and the Outline view on the right shows the class structure.

```
import java.applet.Applet;
import java.awt.Label;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
        Label lb = new Label("0");
        Button bt1 = new Button("1");
    }
}
```

ダブルクリックする

- Button をインポートします (java.awt)
- クラス 'Button' を作成します
- インターフェース 'Button' を作成します
- ButtonBorder を変更します (javax.swing.plaf.basic.BasicButtonBorder)
- ButtonBorder を変更します (javax.swing.plaf.metal.MetalButtonBorder)
- ButtonGroup を変更します (javax.swing)
- ButtonModel を変更します (javax.swing)
- ButtonPeer を変更します (java.awt.peer)
- ButtonUI を変更します (javax.swing.plaf)
- 列挙 'Button' を作成します
- ファイル内の名前変更 (Ctrl+2, R)

Java - TestProject/src/Sample1.java - Eclipse

ファイル(E) 編集(E) ソース(S) リファクタリング(I) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)

パッケージ・エクスプロ TestProject
src
(デフォルト・パッケージ)
Sample1.java
JRE システム・ライブラリー [Java]

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Label;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

    public void init() {
        Label lb = new Label("0");
        Button bt1 = new Button("1");
        add(lb);
    }
}
```

ここに追加された

タスク・リスト
検索 すべて アクテ...
カテゴリなし

Connect Mylyn
Connect to your task and ALM tools.

アウトライン
インポート宣言
Sample1
init() : void

問題 @ Javadoc 宣言
エラー: 0、警告: 1、その他: 0

説明	リソース	バ
警告 (1 項目)		

Butt...きません 書き込み可能 スマ...

ボタンを画面に追加

add(bt1);

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Label;

/**
 * @author kamahara
 */
public class Sample1 extends Applet {

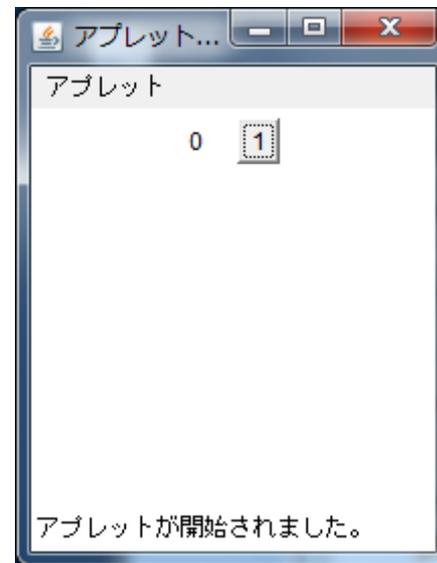
    public void init() {
        Label lb = new Label("0");
        Button bt1 = new Button("1");

        add(bt1);
    }
}
```

エラー: 0、警告: 1、その他: 0

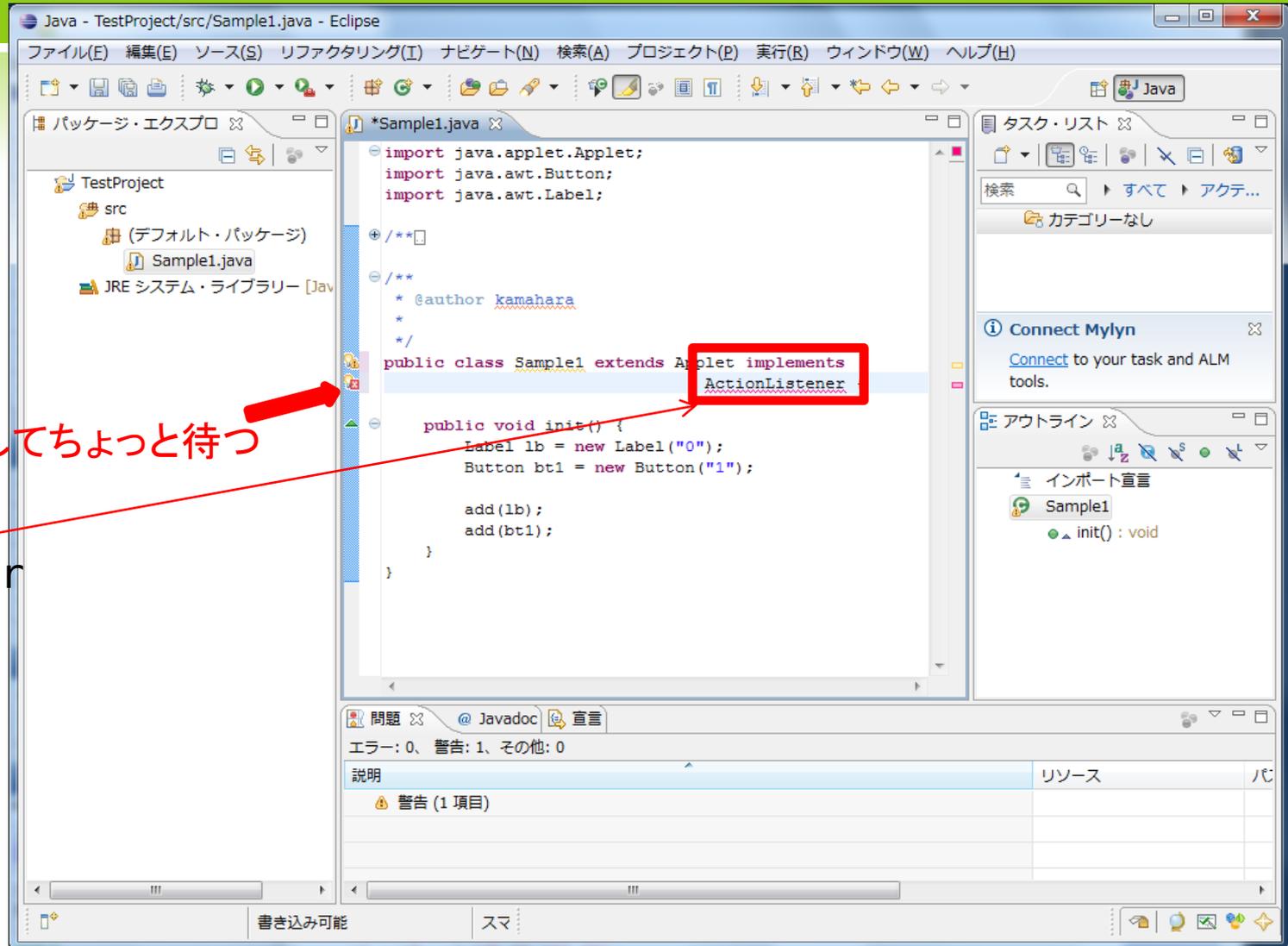
説明	リソース	バ
警告 (1 項目)		

実行してみる



ただし、ボタンを押しても反応しない

反応するよう ACTIONLISTENERを実装



②

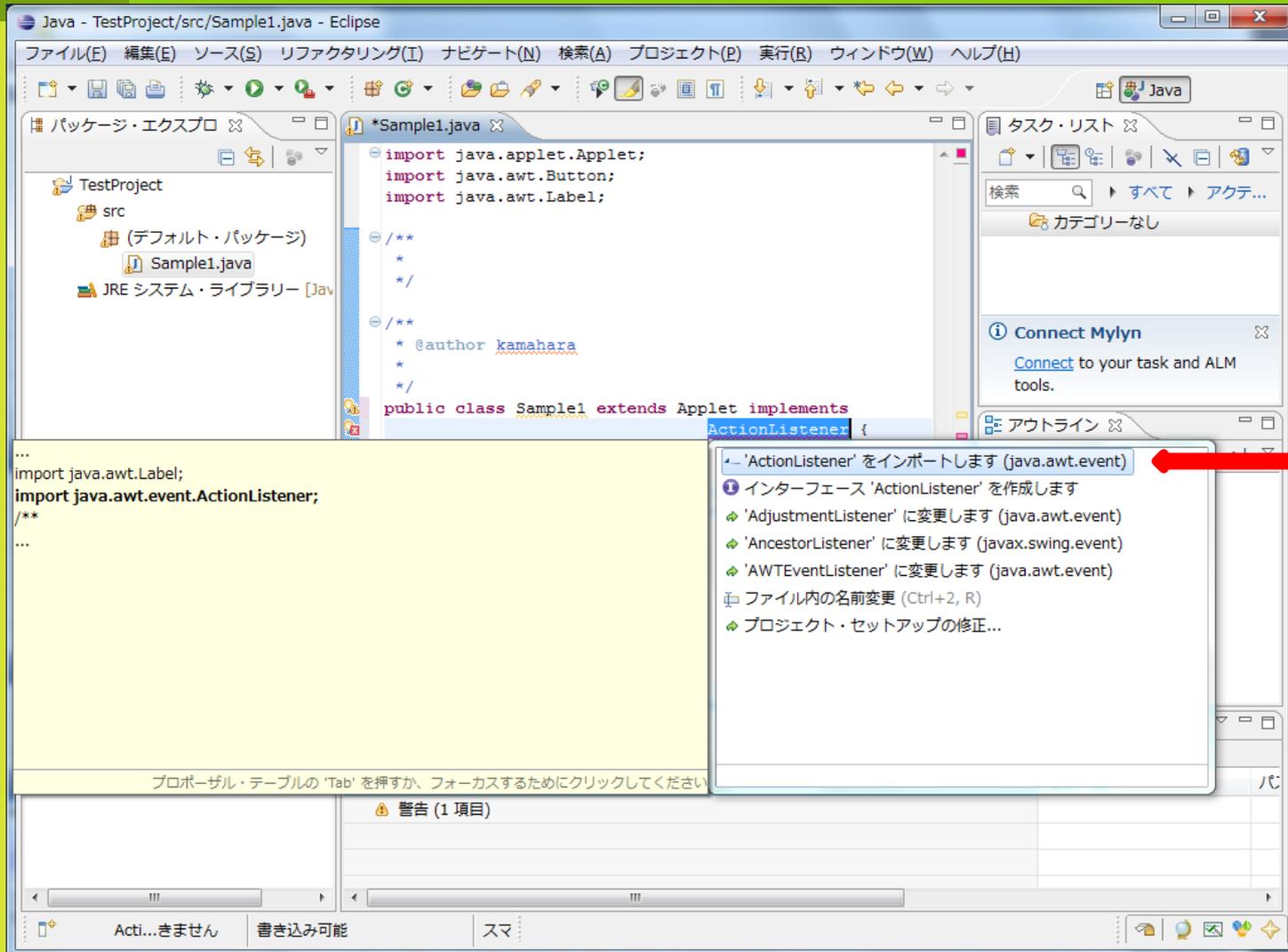
クリックしてちょっと待つ

implements
ActionListener

を追加

①

ACTIONLISTENERもインポート



もう一度白抜きの赤をクリック

The screenshot shows the Eclipse IDE interface. The main editor displays the following Java code:

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Label;
import java.awt.event.ActionListener;

/**
 *
 */

/**
 * @author kamahara
 *
 */

この行に複数マーカーがあります
- シリアライズ可能クラス Sample1 は long 型の static final serialVersionUID フィールドを宣言していません
- 型 Sample1 は継承された抽象メソッド ActionListener.actionPerformed(ActionEvent) を実装する必要があります
button_bt1 = new Button("1");

add(bt1);
add(bt1);
}
```

A red arrow points to a red icon in the left margin of the editor, which is the warning icon. A tooltip is visible over this icon, containing the text: "この行に複数マーカーがあります" (This line has multiple markers) and two bullet points: "- シリアライズ可能クラス Sample1 は long 型の static final serialVersionUID フィールドを宣言していません" and "- 型 Sample1 は継承された抽象メソッド ActionListener.actionPerformed(ActionEvent) を実装する必要があります".

On the left side of the IDE, the Package Explorer shows the project structure: TestProject, src, (デフォルト・パッケージ), Sample1.java, and JRE システム・ライブラリー [Java].

On the right side, the Task List and Outline views are visible. The Outline view shows the class hierarchy for Sample1, including the init() method.

At the bottom, the Problems view shows a warning: "警告 (1 項目)".

クリックしてちょっと待つ

Java - TestProject/src/Sample1.java - Eclipse

ファイル(E) 編集(E) ソース(S) リファクタリング(I) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)

パッケージ・エクスプロー

- TestProject
 - src
 - (デフォルト・パッケージ)
 - Sample1.java
 - JRE システム・ライブラリー [Java]

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Label;
import java.awt.event.ActionListener;

/**
 *
 */

/**
 * @author kamahara
 */

public class Sample1 extends Applet implements
```

タスク・リスト

検索

Connect Mylyn

アウトライン

インポート宣言

Sample1

- init() : void

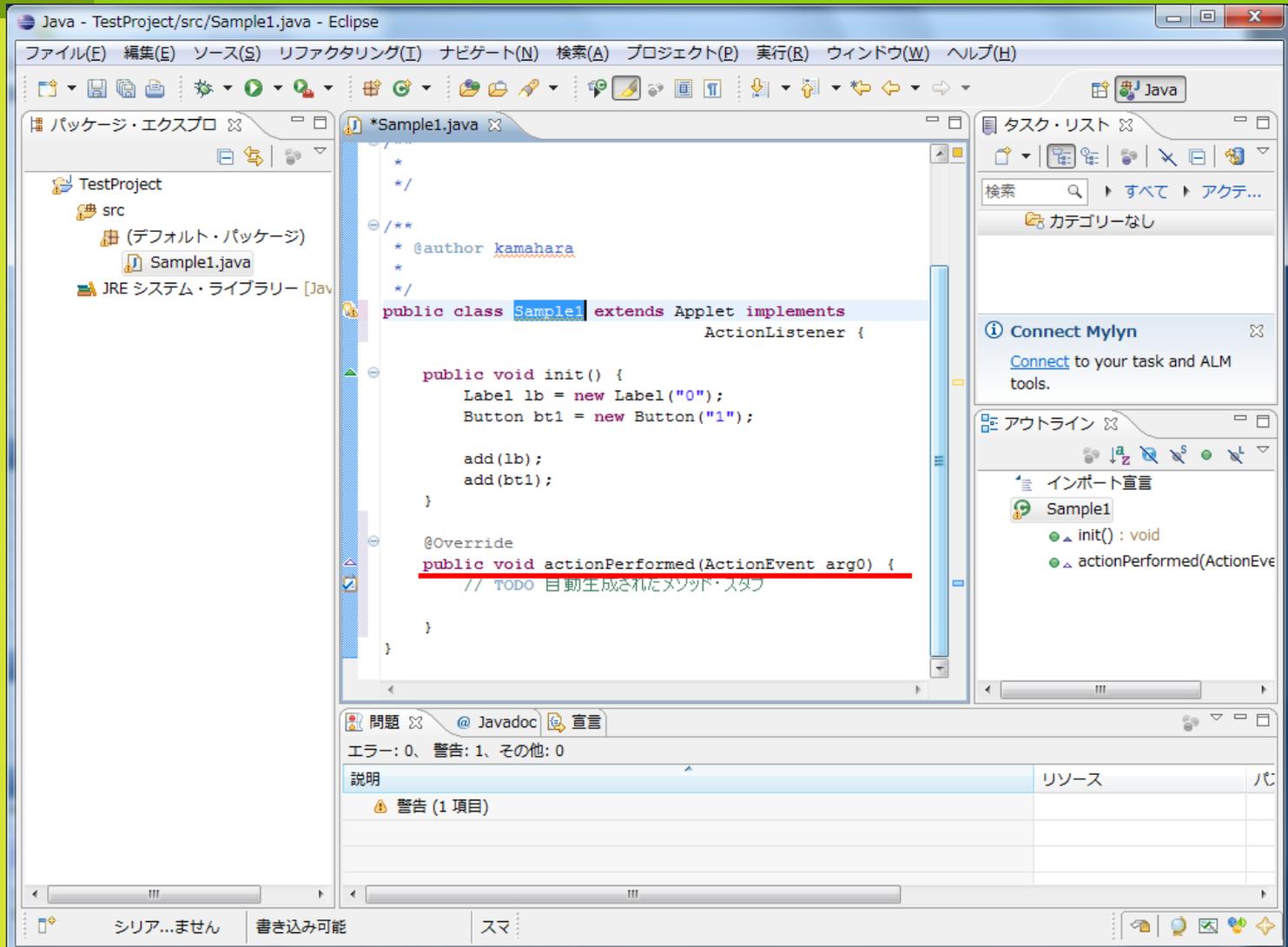
警告 (1 項目)

シリア...ません 書き込み可能 スマ...

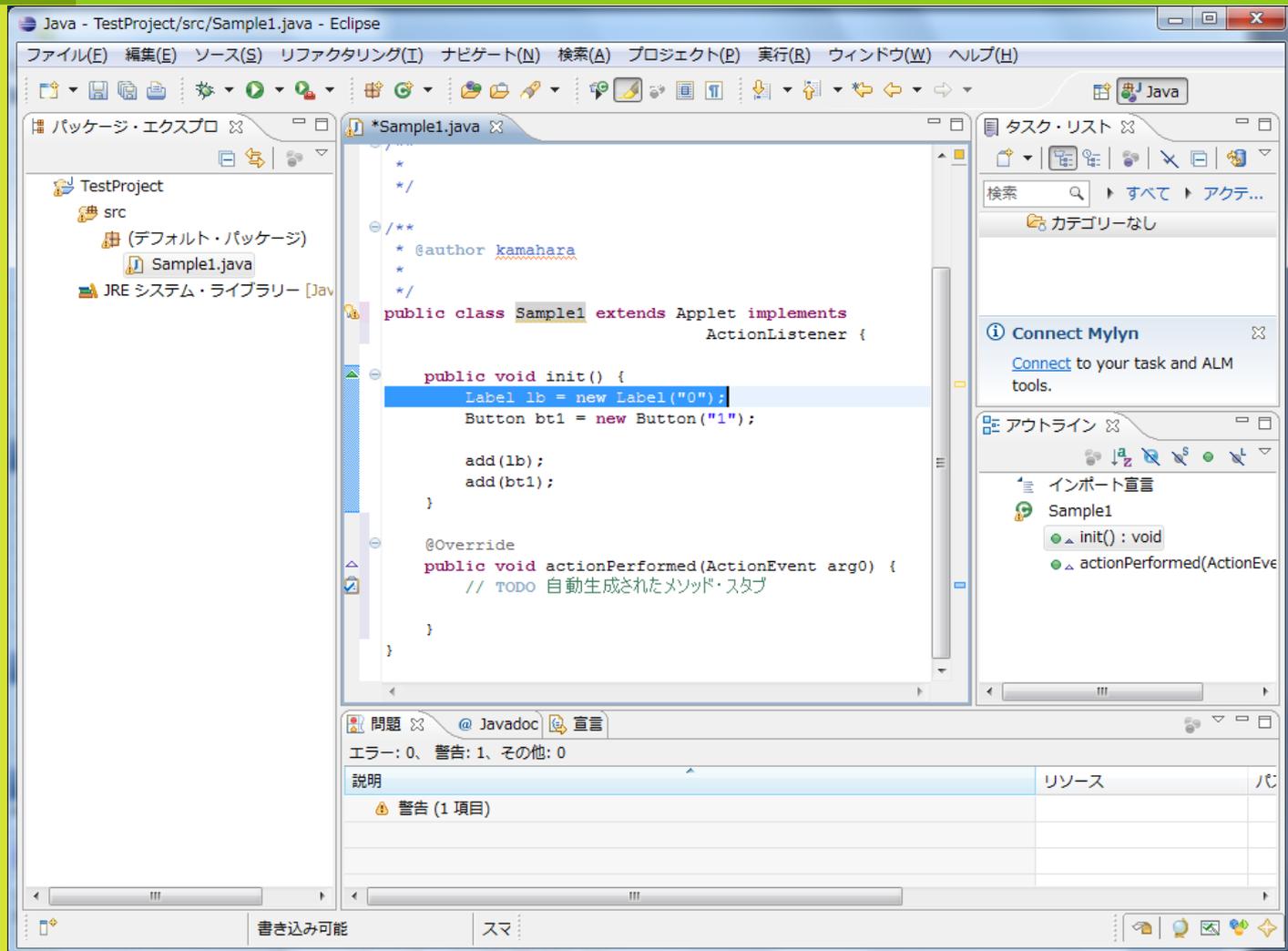
実装されていないメソッドの追加

- + デフォルト・シリアル・バージョン ID の追加
- + 生成シリアル・バージョン ID の追加
- + 型 'Sample1' を abstract にします
- ファイル内の名前変更 (Ctrl+2, R)
- ワークスペース内の名前変更 (Alt+Shift+R)
- @ SuppressWarnings 'serial' を 'Sample1' に追加します

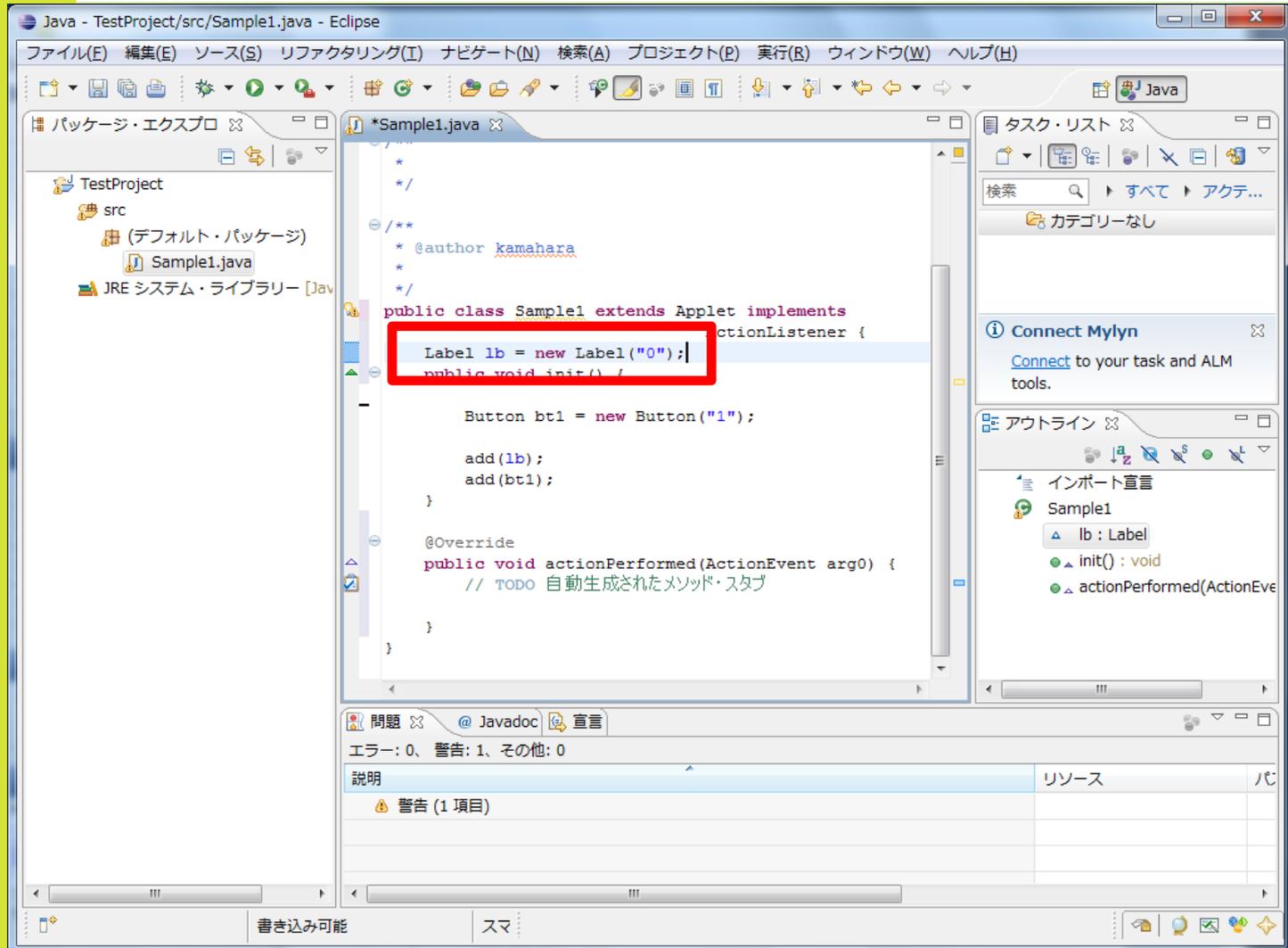
ACTIONPERFORMED関数が自動的に追加



LABELの定義の位置を移動



カットして貼り付け



反応に対する処理を追加

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'TestProject' containing 'src' and 'Sample1.java'.
- Editor:** Displays the code for 'Sample1.java'. The code includes:

```
public class Sample1 extends Applet implements ActionListener {  
    Label lb = new Label("0");  
    public void init() {  
        Button bt1 = new Button("1");  
        add(lb);  
        add(bt1);  
        bt1.addActionListener(this);  
    }  
    @Override  
    public void actionPerformed(ActionEvent arg0) {  
        // TODO 自動生成されたメソッド・スタブ  
        lb.setText(arg0.getActionCommand());  
    }  
}
```
- Annotations:** Two red boxes highlight the lines `bt1.addActionListener(this);` and `lb.setText(arg0.getActionCommand());`. Red arrows point from external text to these lines.
- Task List:** Shows 'Connect Mylyn' and 'アウトライン' (Outline) panels.
- Outline:** Lists 'インポート宣言' (Import Declaration) and 'Sample1' with members 'lb : Label', 'init() : void', and 'actionPerformed(ActionEvent)'.

説明	リソース	バ
警告 (1 項目)		
- Problems View:** Shows 'エラー: 0, 警告: 1, その他: 0' and a table with one warning entry.

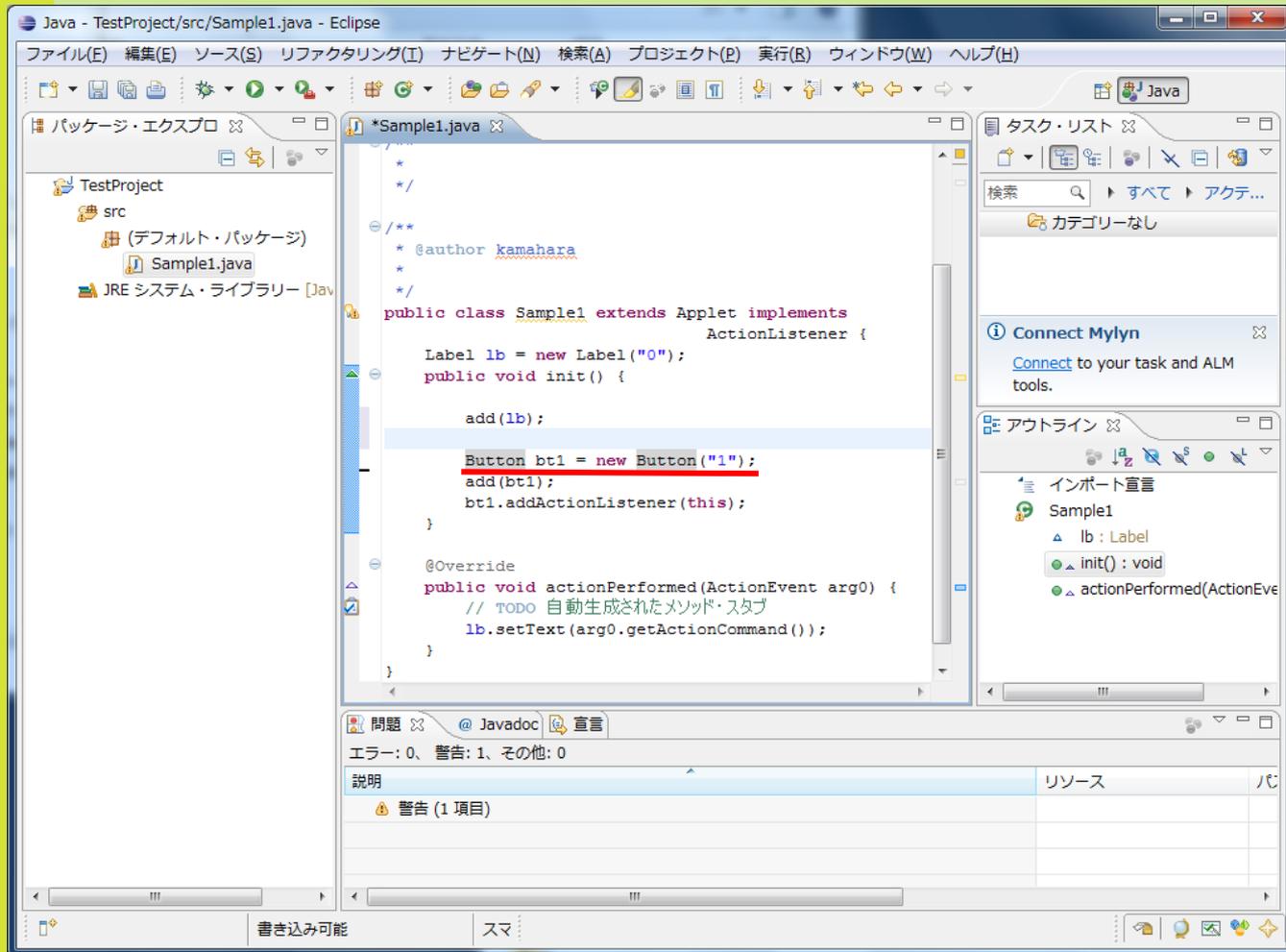
bt1.addActionListener(this);

lb.setText(arg0.getActionCommand());

ボタンを押すと文字が書き変わることを確認



ボタンを生成する位置を ちょっと変更



課題

- ◎ 押すとラベルが0から9まで書き変わる10個のボタンをfor文を使って生成し、アプレット上に表示されるようにしなさい

課題の提出

- ◎ Z:¥workspace¥TestProject¥Sample1.javaを添付ファイルとして提出すること。ファイル中にコメントして学籍番号が入っていること
- ◎ 件名：情報処理演習 8 学籍番号 名前
- ◎ 宛先：kamahara@port.kobe-u.ac.jp