

# Java

eclipseの使い方とクラスの基礎

**16** 時間目

# プログラミング言語の種類

解説

プログラミング言語には、「プログラミングパラダイム」と呼ばれる種類があります。

1 手続き型プログラミング

2 オブジェクト指向プログラミング

・・・などなど

補足

プログラミングパラダイムとは、プログラムを記述する際の考え方の種類です。  
その他のプログラミングパラダイムでは、関数型プログラミング（例：Scala）も最近人気が高まっています。

# 手続き型プログラミングとは

## 1 手続き型プログラミング

言語	特徴
C言語、BASIC、perl など	1 行目から最後の行まで、基本的に <b>個別に機能を作成し順番通り</b> に実行する

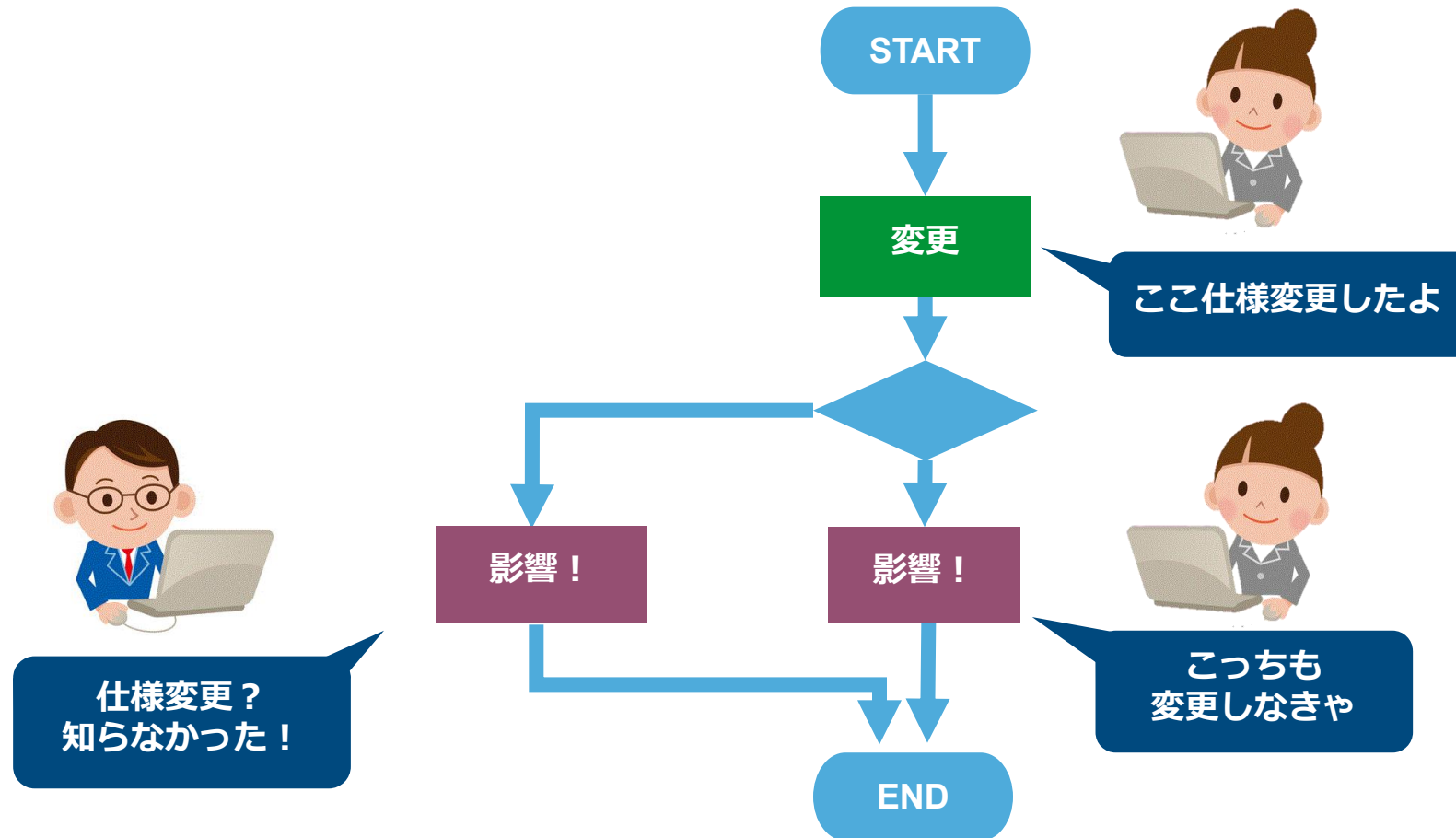
## 2 オブジェクト指向プログラミング

言語	特徴
Java、Ruby、PHP など	機能をオブジェクトと呼ばれる「データと動きのワンセット」単位で1つの部品として分解し、その <b>設計図を何度も流用</b> しながら独立したパーツを作成・ <b>組み合わせて</b> 実行する。

# 手続き型プログラミング

## 解説

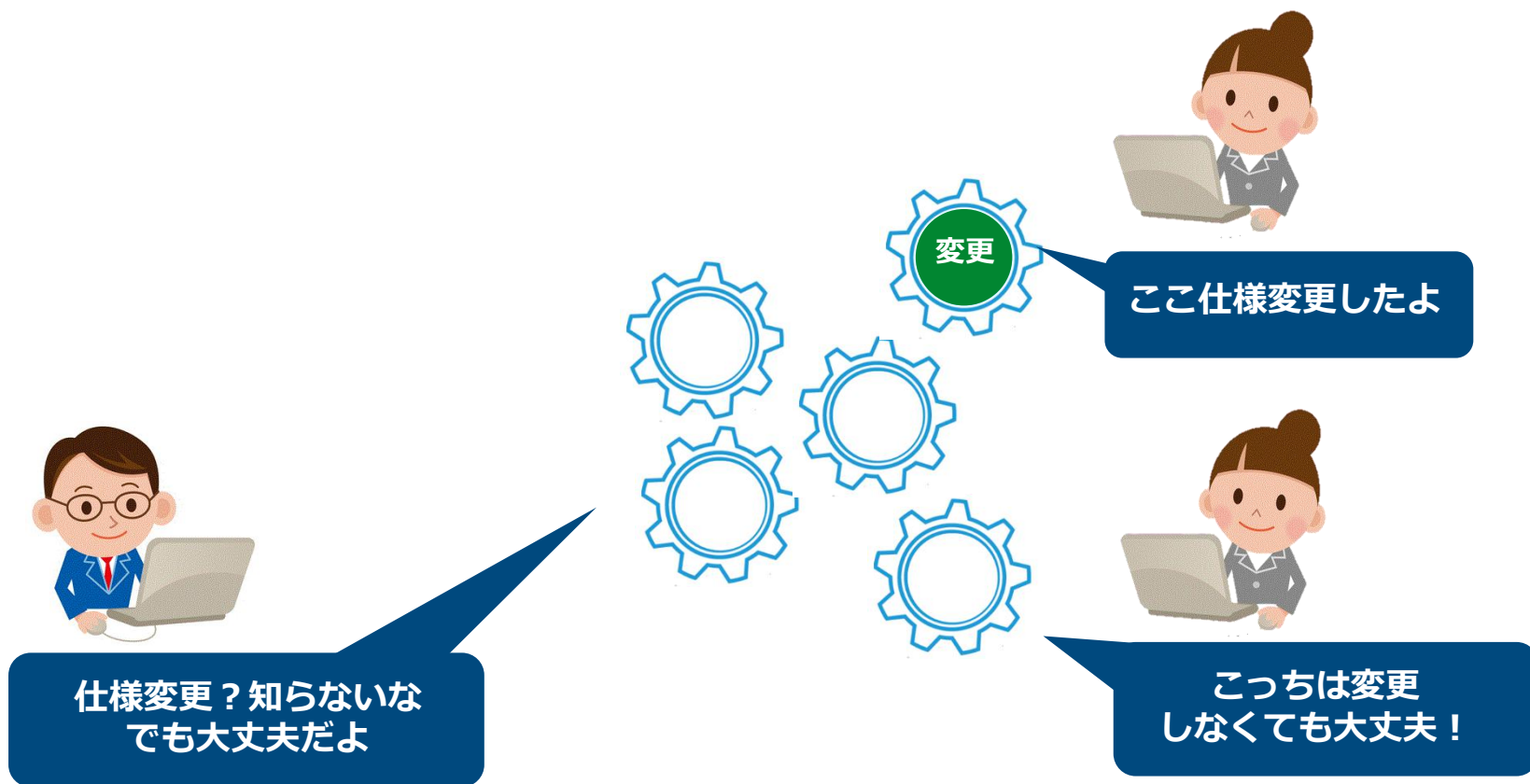
従来の手続き型では、各部品が完全には独立していないため、大勢で開発する場合に連絡ミスなどの混乱が生じたり、仕様変更への対応が広範囲になるなど大規模開発への対応が難しかった。



# オブジェクト指向プログラミングのメリット

## 解説

「部品」に特化したオブジェクト指向を採用することによって、全体（他の人の作業内容）までしっかり把握しなくても部品が作れたり、仕様変更の影響が出にくくなった。

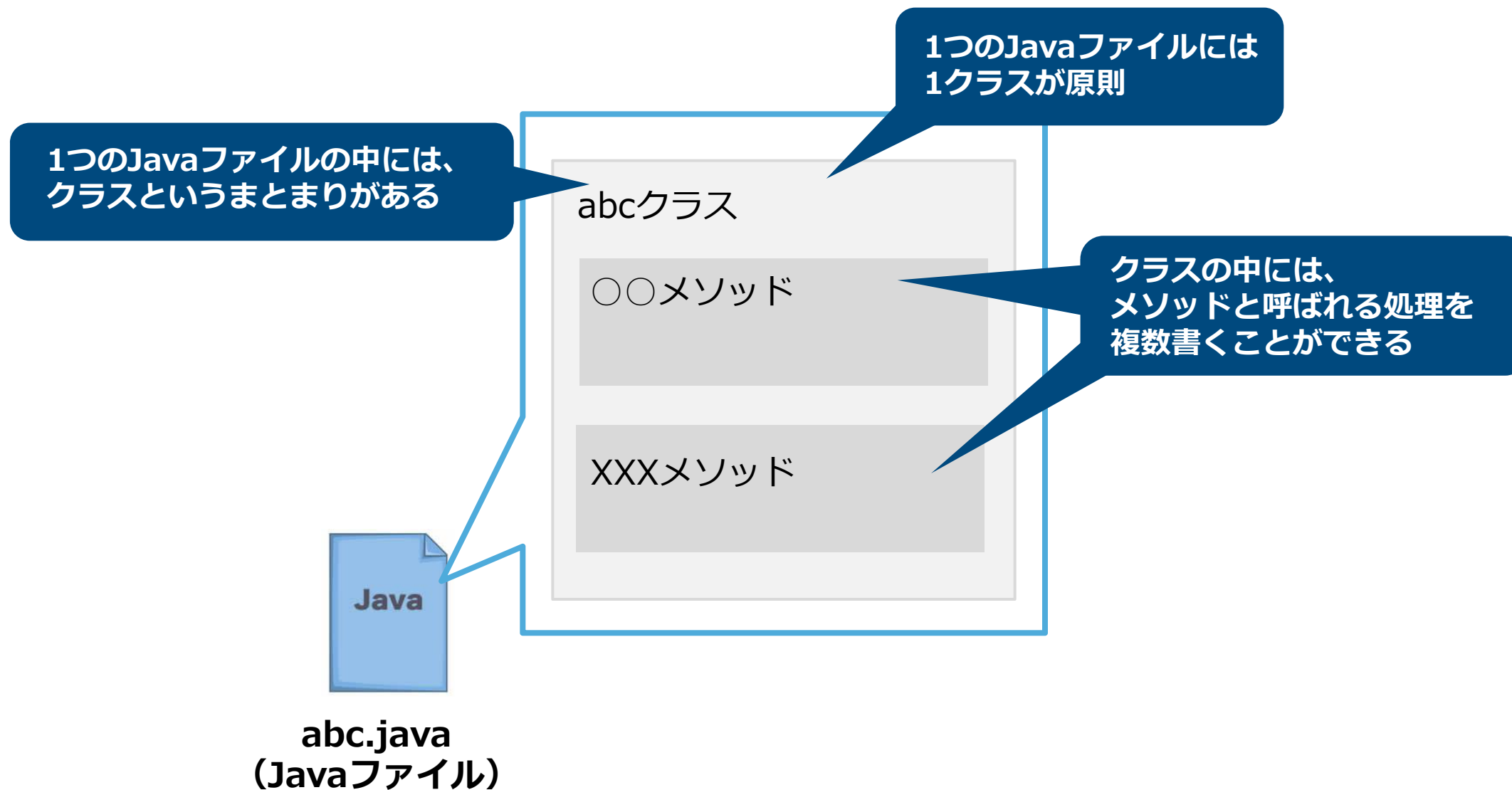


# クラスとメソッド

Javaプログラミングには、下記のルールがある

- 1 1つのJavaファイルの中には、クラスという“まとまり”がある
- 2 そのJavaの“ファイル名”= クラス名
- 3 1つのJavaファイルには1クラスが原則
- 4 クラスの中には、メソッドと呼ばれる処理を複数書くことができる

# Javaファイルとクラスとメソッドのイメージ



# クラスとメソッドの解説

public classという文字は、eclipseを使用してJavaを記述する際に、デフォルト（初期状態）で必ず自動で出てくるモノ。publicという標準的なクラスを作ることができるようになっている。

public クラスの後ろにつく〇〇〇部分が（=自動的に表示される）、クラス名。

この場合『Humanクラス』と言う。

このクラス名と、ファイル名は  
同じにしないダメ。



**Human.java**

```
public class Human {  
  
    public static void main (String[] args) {  
  
    }  
}
```

緑の部分がメソッドとよばれる部分。メソッドとは、クラスを起動した時に実行したい処理の事。（任意の名前を付けられる）

メソッド名を「main」にすると、クラスを起動したときに、一番最初に処理されることになる。



# static void、(String[]args)の意味は理解しないでOK

下記の**緑色**の部分は、現時点では**覚えなくてOK**  
(無視して進めましょう)



Human.java

```
public class Human {  
  
    public static void main (String[] args) {  
  
    }  
}
```

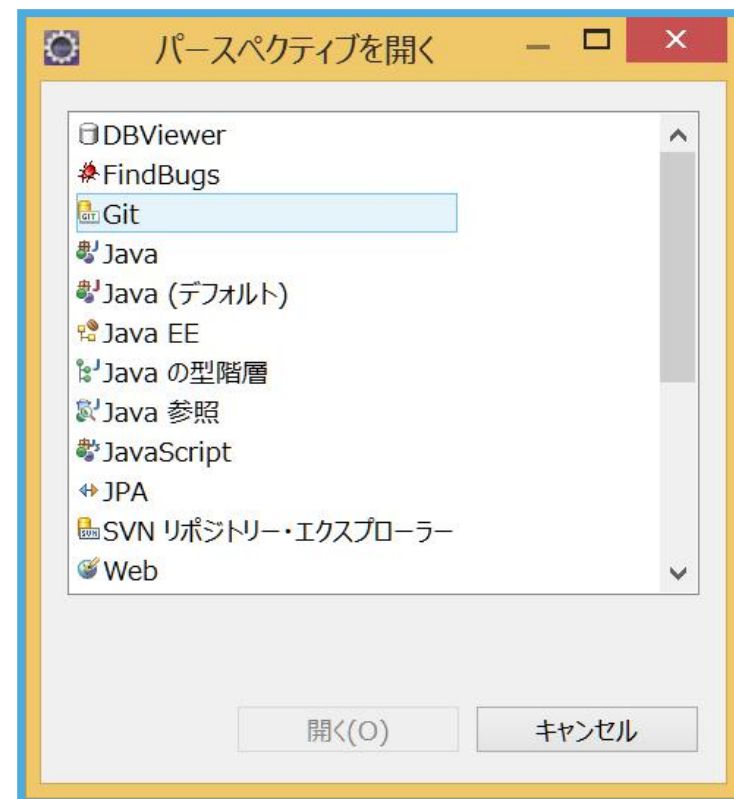
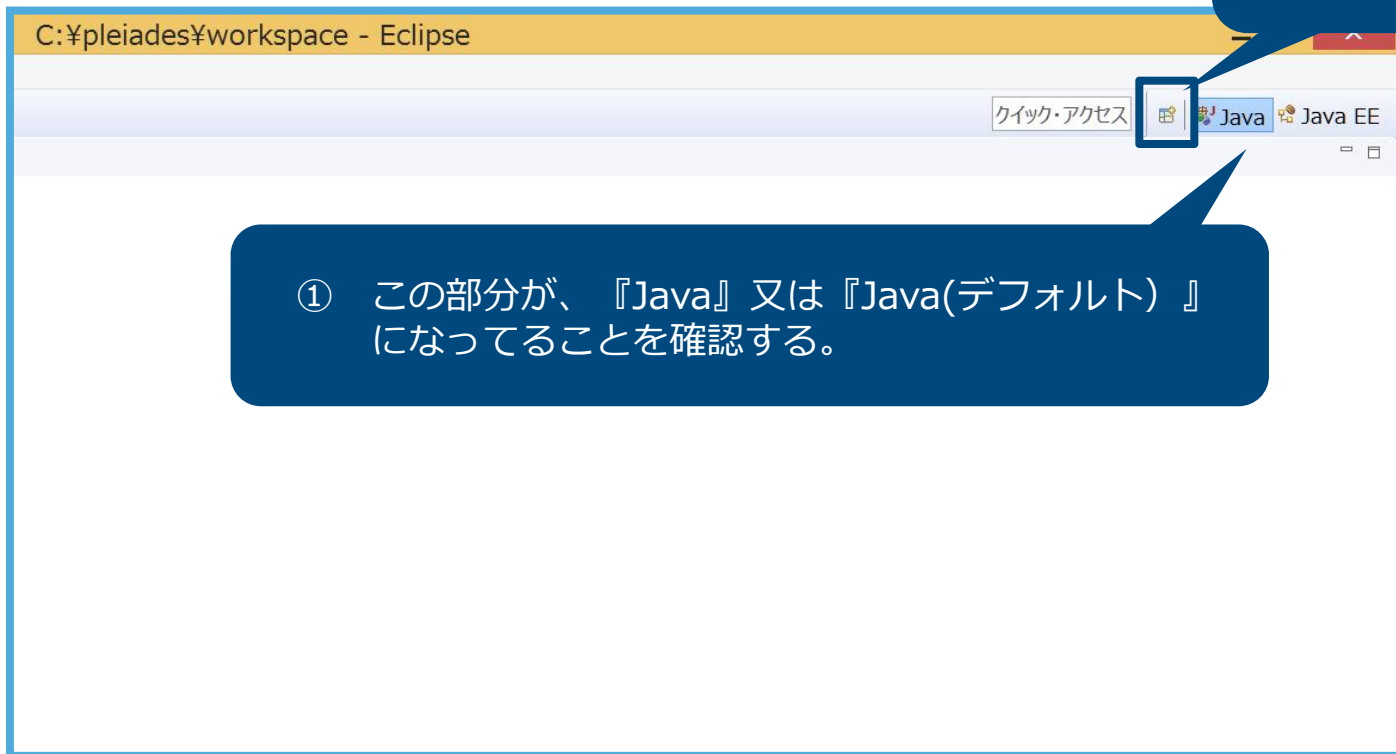
# クラスやメソッドを意識してJavaを書いてみよう



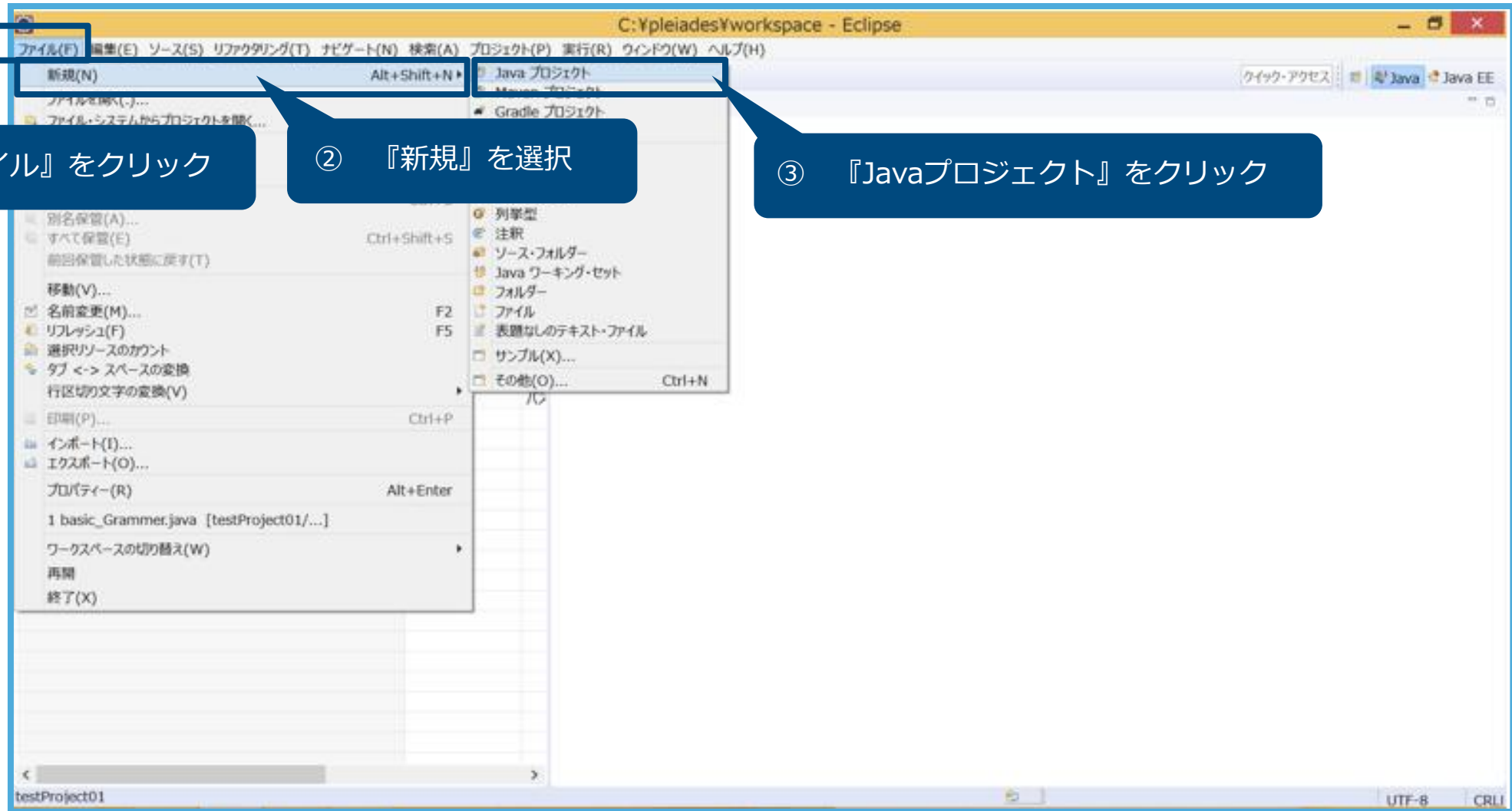
# Java、又は Java(デフォルト)を選択

- ② なっていない場合は、このボタンをクリック。すると、下のような画面が開くので、『Java』又は『Java(デフォルト)』を選択する。

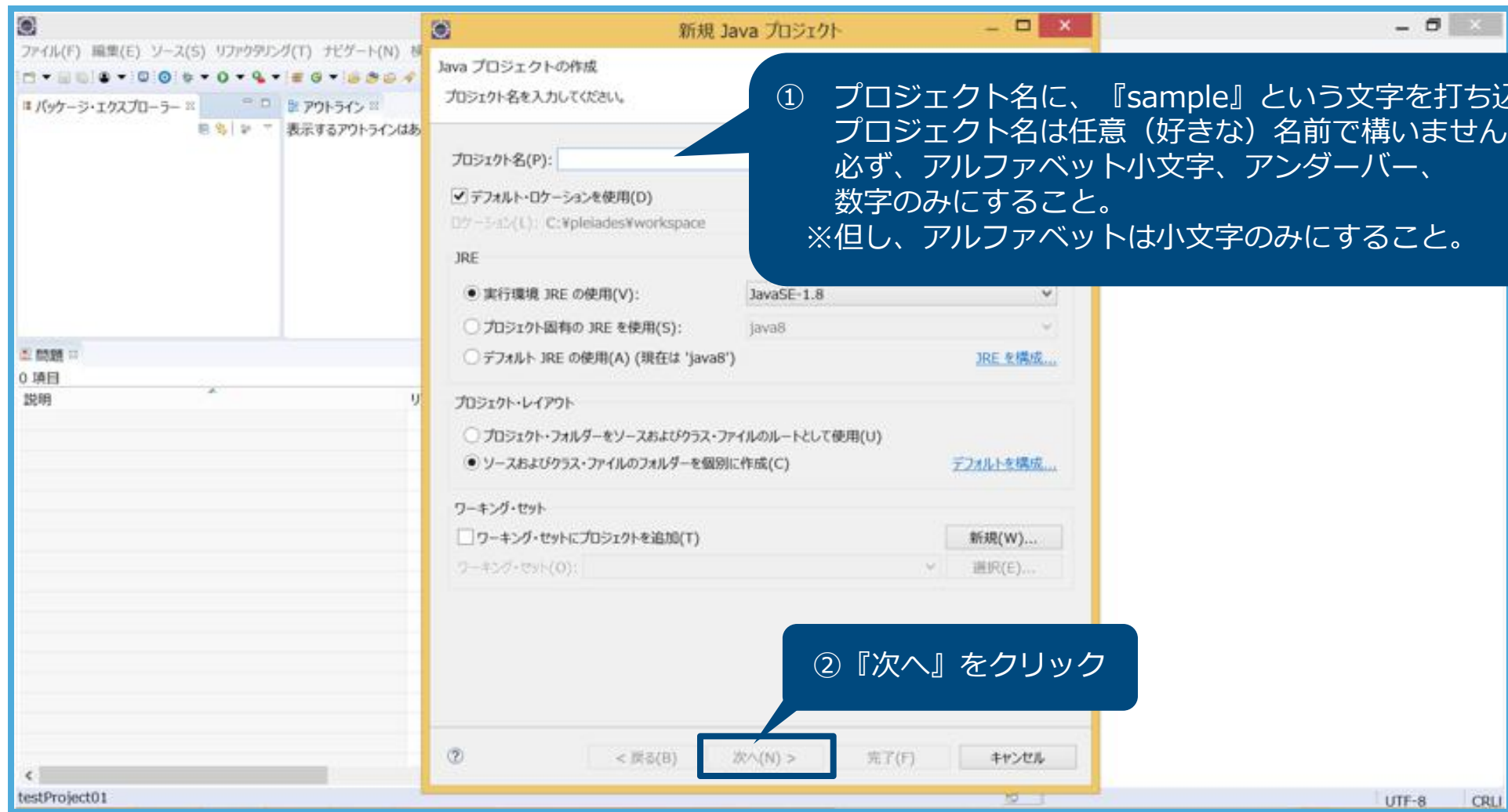
- ① この部分が、『Java』又は『Java(デフォルト)』になっていることを確認する。



# 新規でプロジェクトを作成

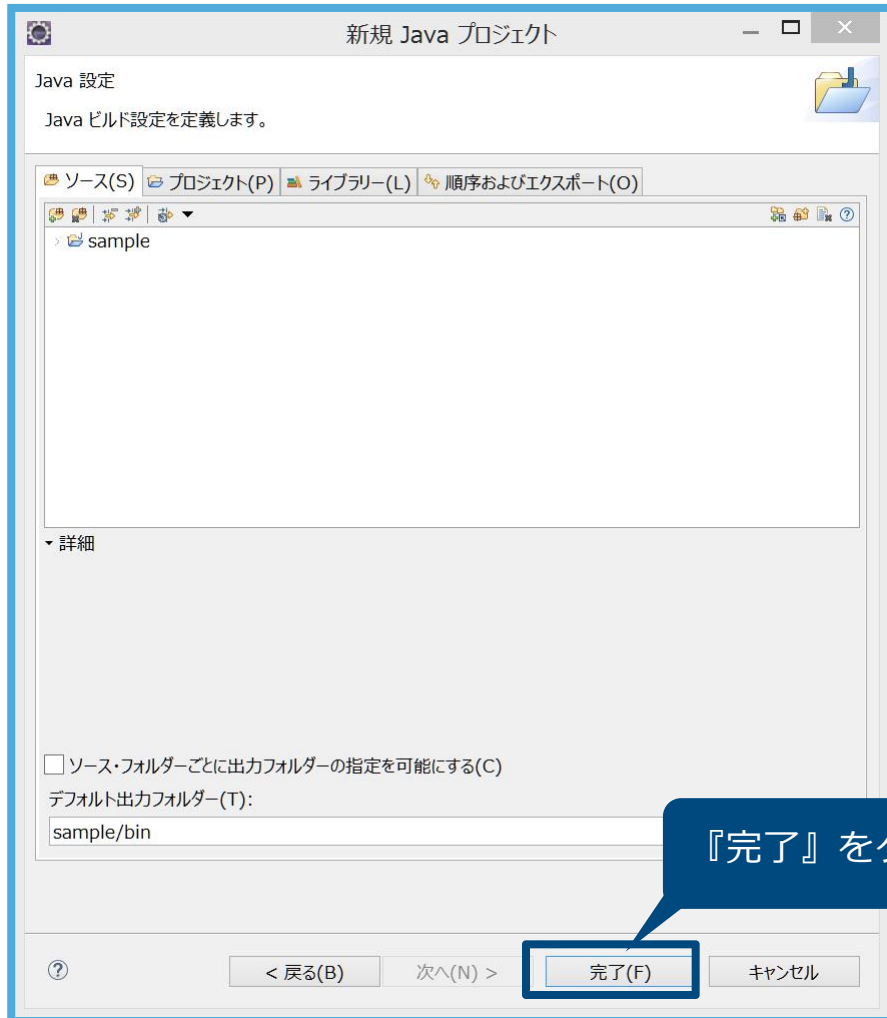


# 新規でプロジェクトを作成



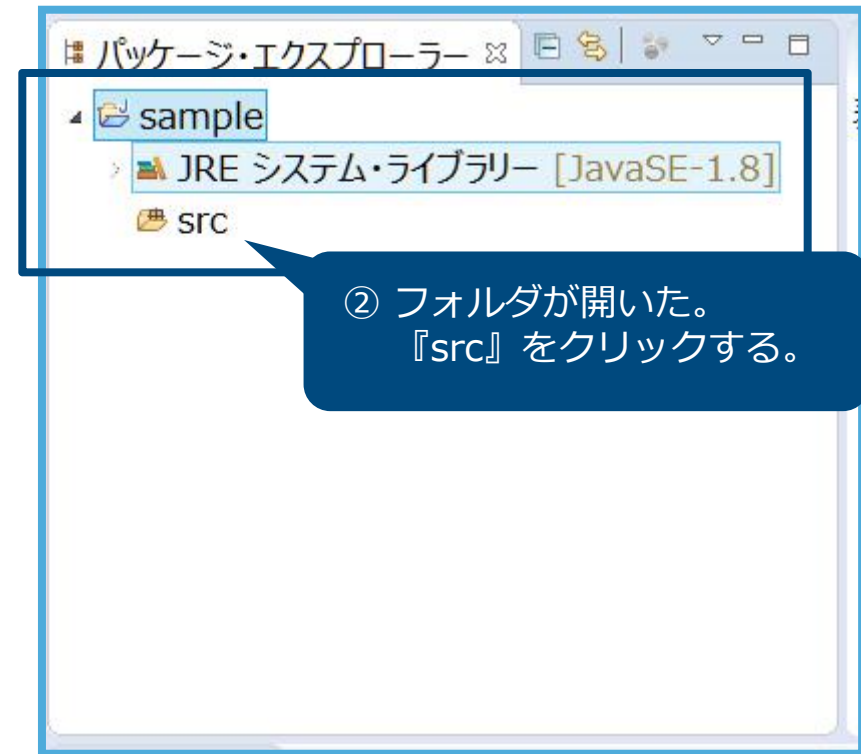
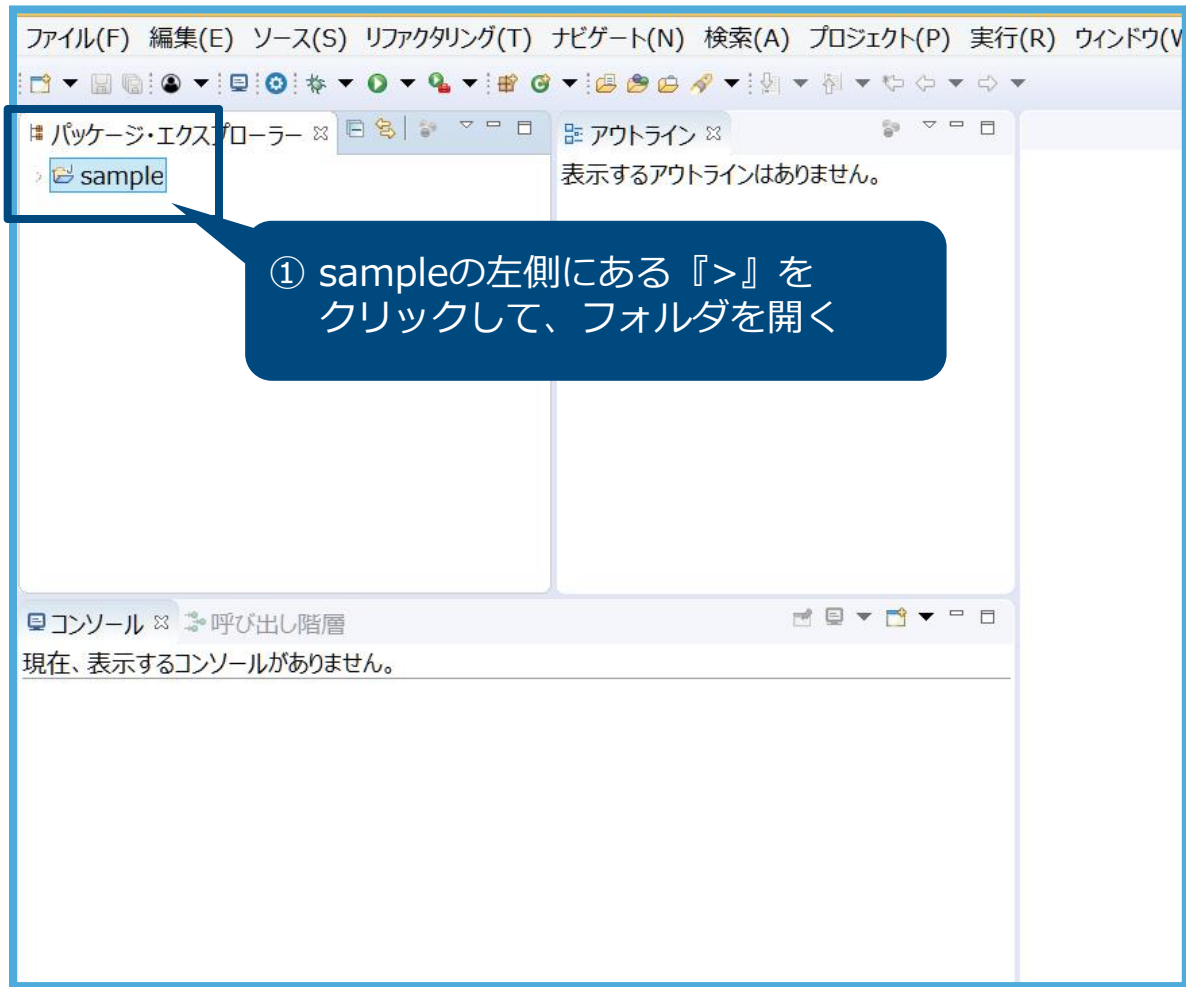
# 新規でプロジェクトを作成

sampleというJavaプロジェクト（フォルダ=ディレクトリ）の中に「src」という名前のフォルダを作りますという確認ページ。



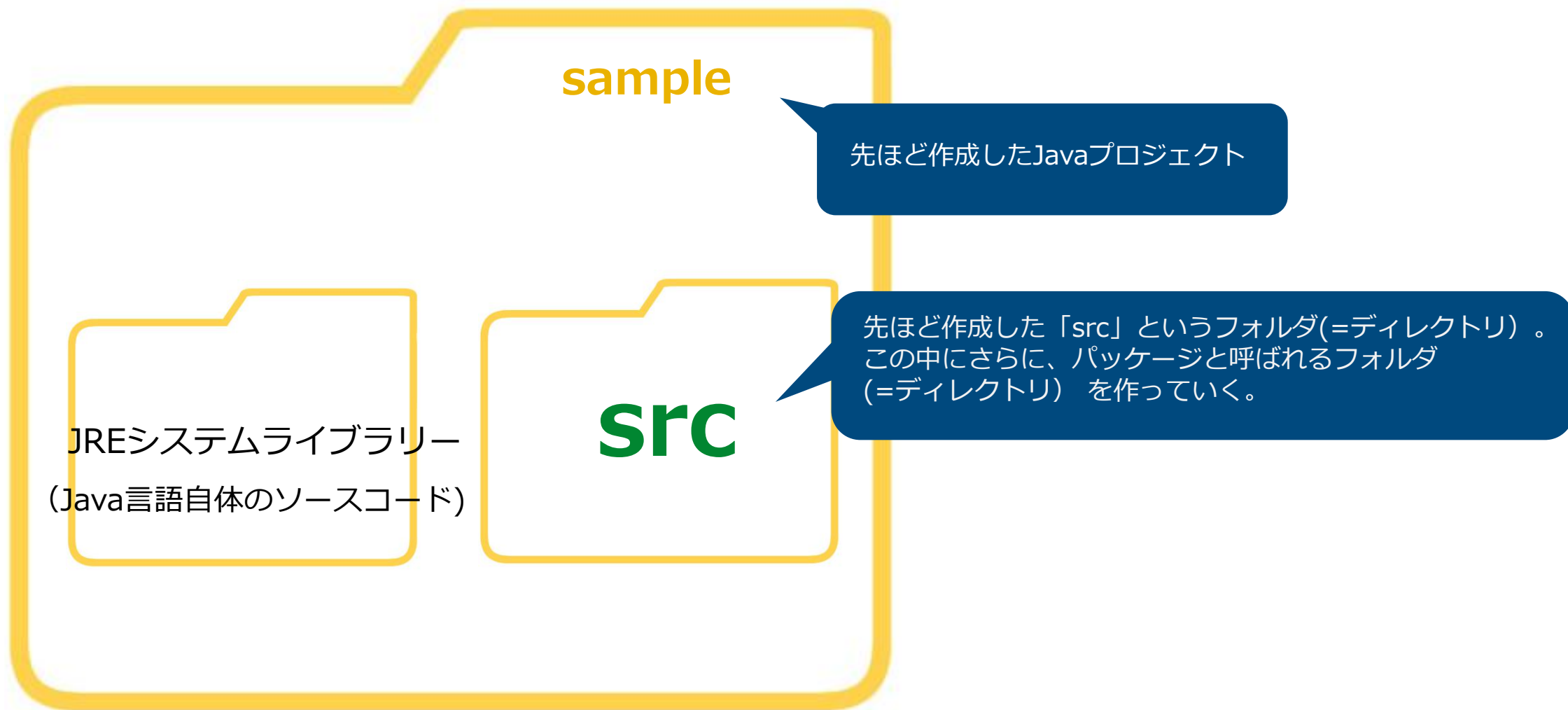
『完了』をクリック

# src（ソース）を選択



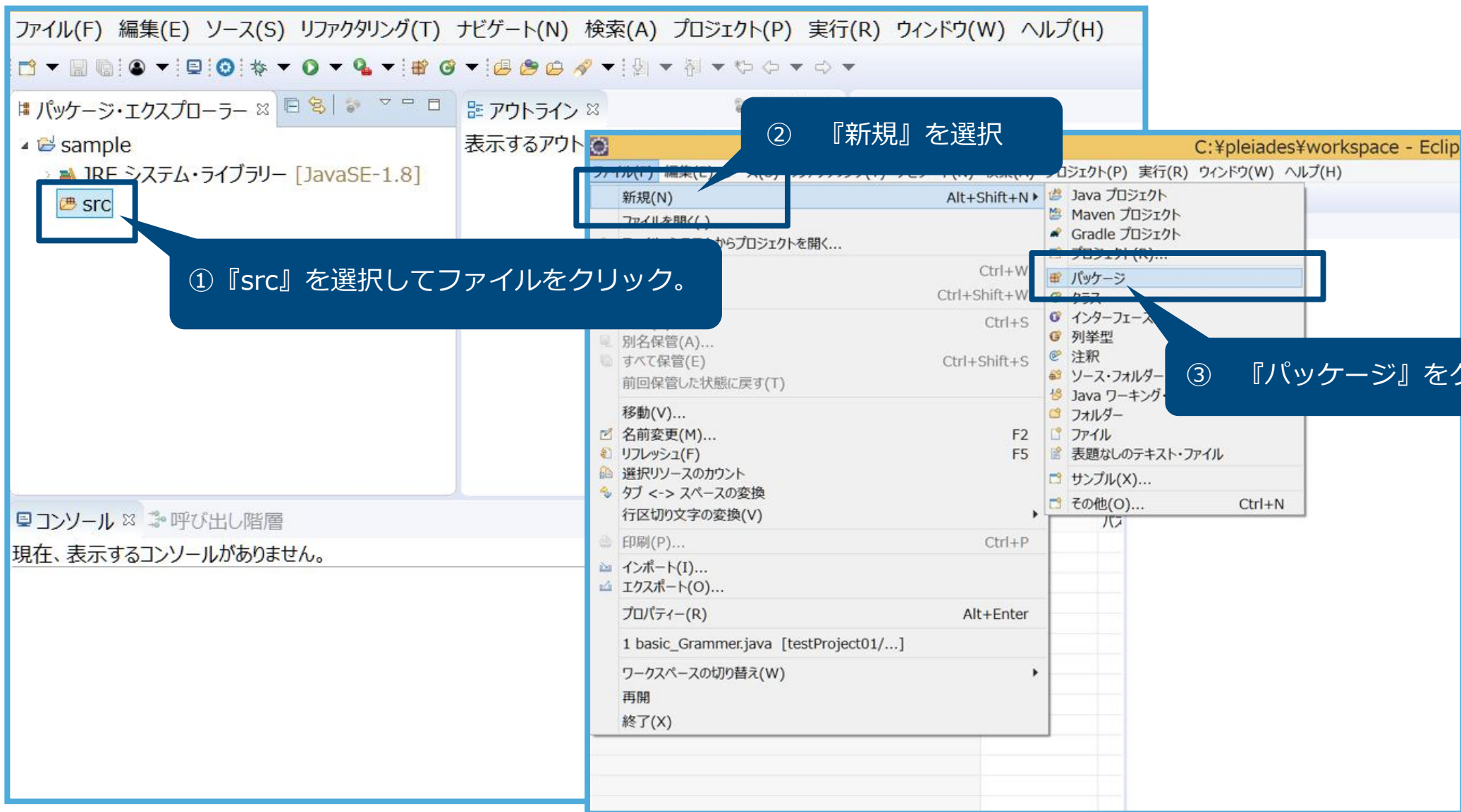


# 現時点での作成したフォルダ構成（イメージ図）





# パッケージを作る

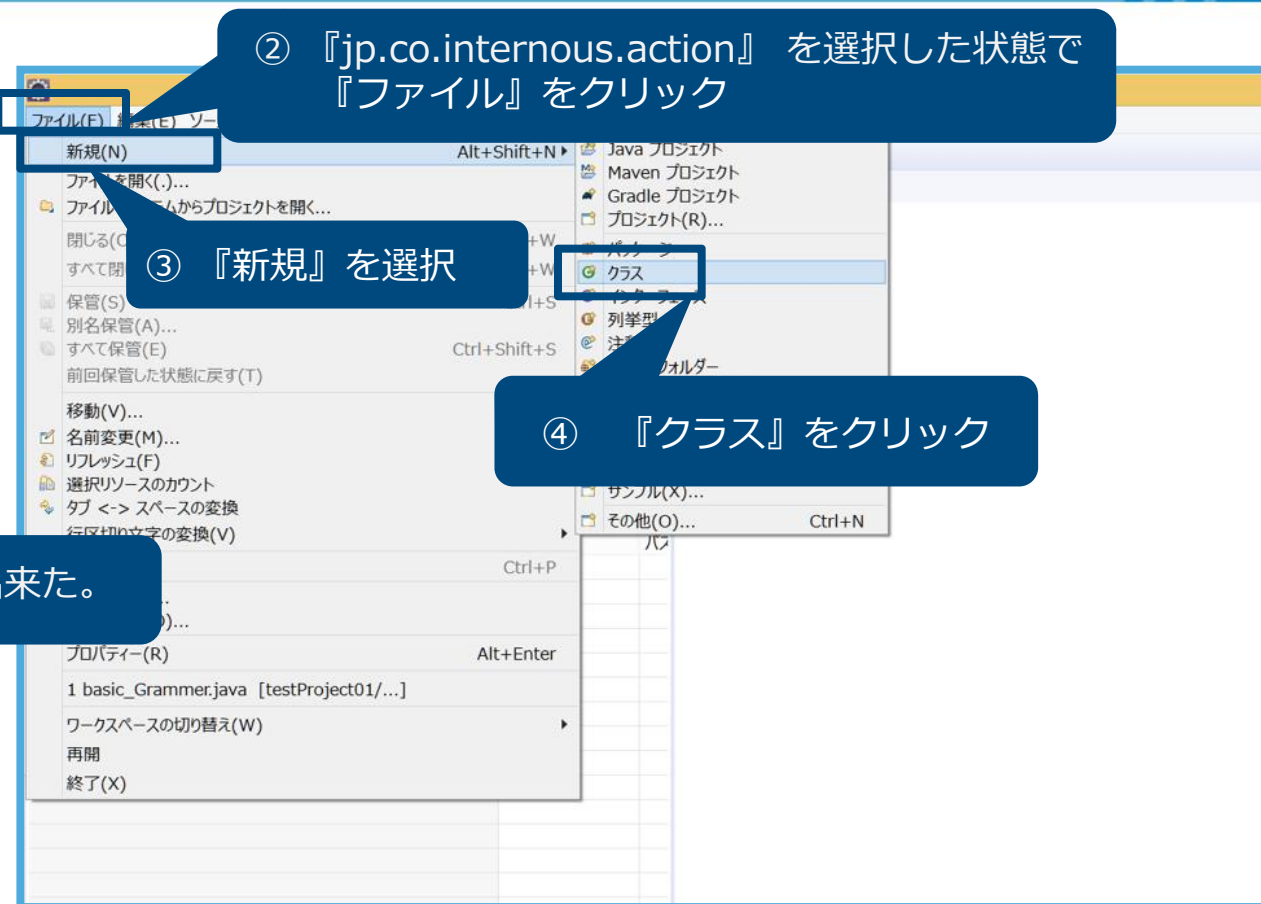
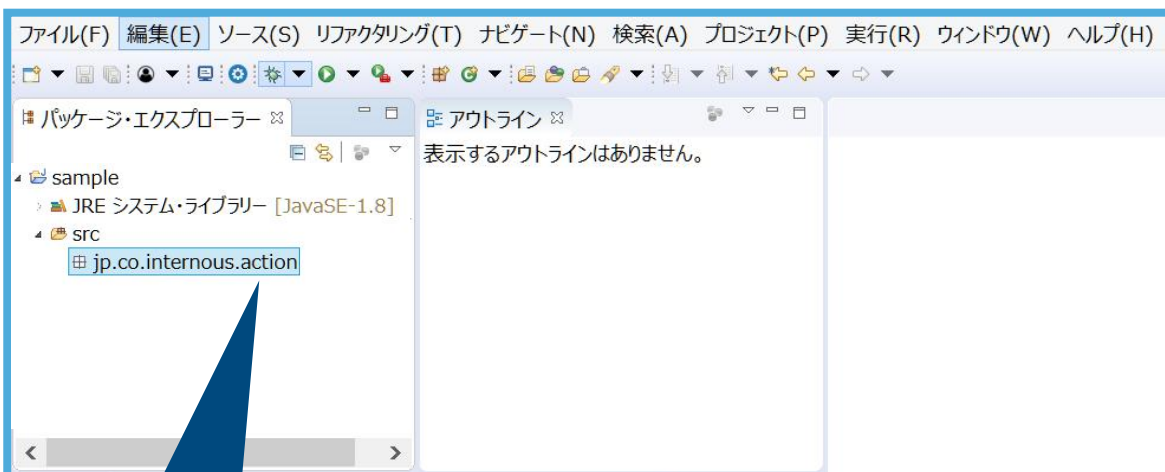


# パッケージを作る

The screenshot shows the '新規 Java パッケージ' (New Java Package) dialog box. It has a title bar with a gear icon and standard window controls. The main area contains the following elements:

- Java パッケージ** (Java Package) section with a gift icon and the instruction: '新規 Java パッケージを作成します。' (Create a new Java package.)
- ソース・フォルダー(D):** (Source folder) text box containing 'sample/src'. A callout bubble points to this section with the text: '① このままで良い' (It's fine as is).
- 名前(M):** (Name) text box containing 'sample'. A large callout bubble points to this field with the text: '② 『jp.co.internous.action』と書く。パッケージ名は、IT業界の慣習として、公開したいURLの逆にするのが一般的。※これやらないと現場で怒られるので注意。internous.co.jpが、当社のURLなので、ここでは、このURLを逆にして、『jp.co.internous.action』と入力する。' (Write 'jp.co.internous.action'. Package names are generally the reverse of the URL you want to publish in the IT industry. ※Be careful, as you will be angry in the field if you don't do this. internous.co.jp is our URL, so here, reverse this URL and enter 'jp.co.internous.action').
- package-info.java を作成する(C)** checkbox, which is unchecked. A callout bubble points to it with the text: '③ チェックを外す' (Uncheck).
- 完了(F)** (Finish) and **キャンセル** (Cancel) buttons at the bottom. A callout bubble points to the '完了(F)' button with the text: '④ 『完了』をクリック' (Click 'Finish').

# クラスを作る



# クラスを作る

新規 Java クラス

Java クラス

新規 Java クラスを作成します。

ソース・フォルダー(D): sample/src 参照(O)...

パッケージ(K): jp.co.internous.action 参照(W)

☐ エンクローシング型(Y):

名前(M):

修飾子: ☒ public(P) ☐ パッケージ(C) ☐ private(V) ☐ protected(V)

☐ abstract(T) ☐ final(L) ☐ static(C)

スーパークラス(S): java.lang.Object

インターフェース(I):

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(U)

☒ 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)

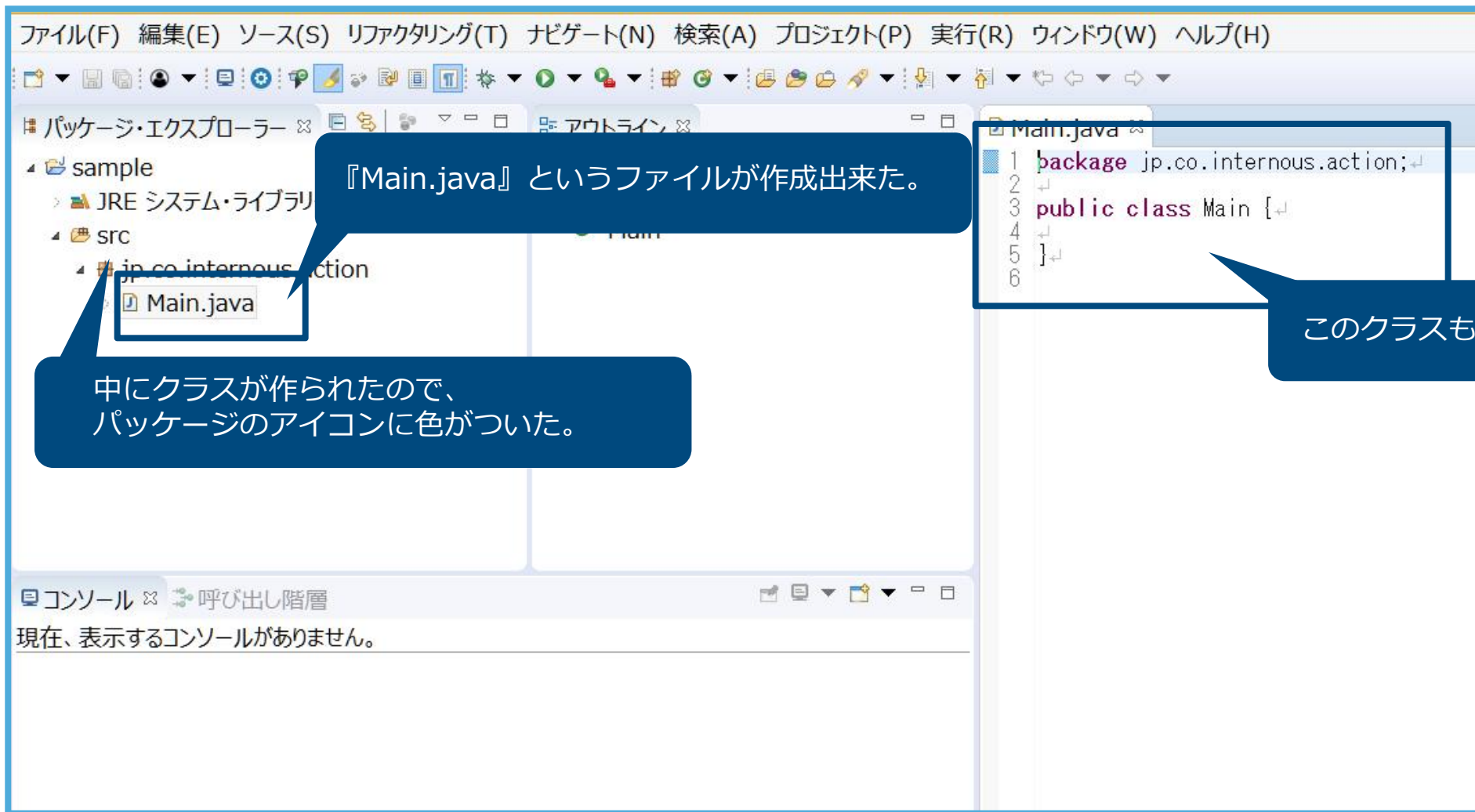
完了(F) キャンセル

① 『Main』 と入力。

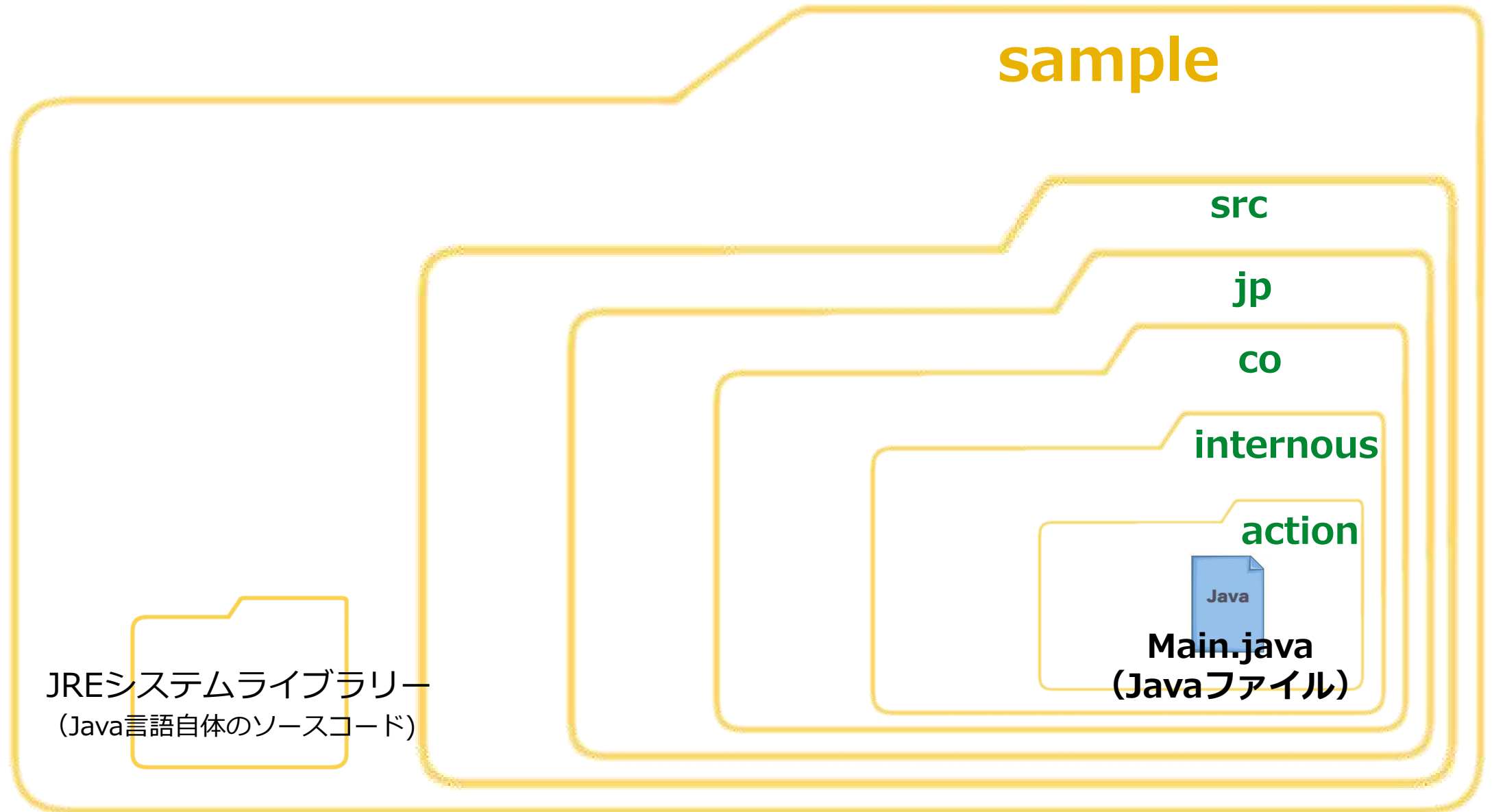
※クラス名の頭文字は必ず大文字にすること。  
頭文字を小文字にするとアラートがあがり、さらに  
現場でも怒られるので注意。

② 『完了』 をクリック

# クラスを作る



# 現時点での作成したフォルダ構成（イメージ図）





# パッケージを作る

この『Main.java』がどのフォルダ（=ディレクトリ）にあるかを示している。

The screenshot shows an IDE with three main panels. On the left, the 'Package Explorer' shows a project named 'sample' with a 'src' folder containing a package 'jp.co.internous.action' and a file 'Main.java'. In the center, the 'Outline' view shows the package 'jp.co.internous.action' and the class 'Main'. On the right, the 'Main.java' file is open, showing the following code:

```
package jp.co.internous.action;  
  
public class Main {  
    // ...  
}
```

Green lines connect the package 'jp.co.internous.action' in the Outline view to the package declaration in the code and the package 'jp.co.internous.action' in the Package Explorer. A blue line connects the file 'Main.java' in the Package Explorer to the class declaration in the code. A blue callout box points to the class declaration in the code.

『Main.java』というファイル名が、  
そのまま『public class Main』とクラス名になっている。

コンソール 呼び出し階層  
現在、表示するコンソールがありません。

# 可読性の高いコード（readableコード/リーダブルコード）

## 解説

チームで作業をするには、可読性（読みやすさ）の高いコードを書く必要があります。  
可読性の高いコードの事を、readable（リーダブル）コードと言います。

Main.java

```
1 package jp.co.internous.action;
2
3 public class Main {
4
5 }
6
```

- ① デフォルト（初期状態）で、4行目にカーソルが合わせてある。  
ここでキーボードの「ENTER」を押す。

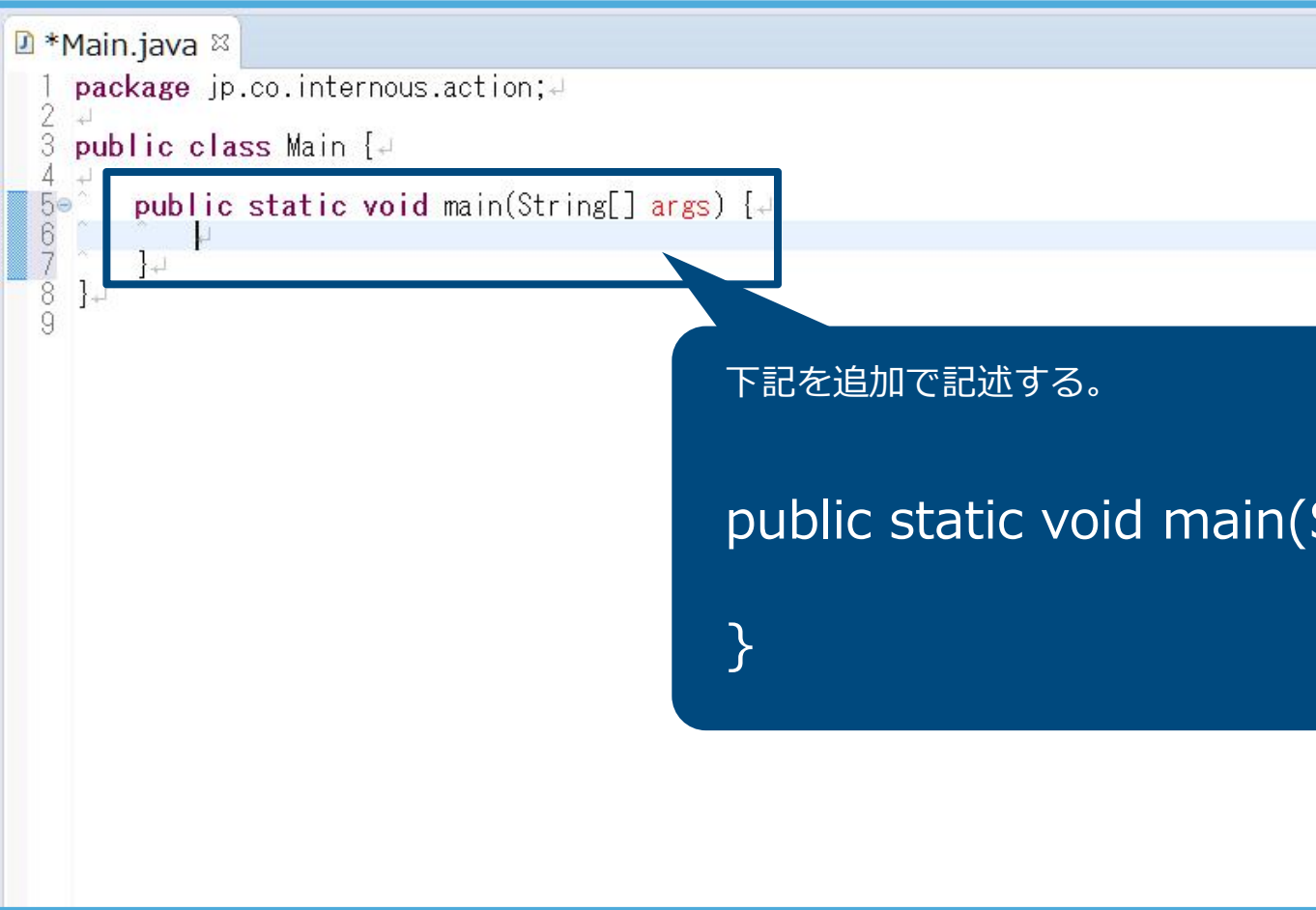
\*Main.java

```
1 package jp.co.internous.action;
2
3 public class Main {
4
5
6 }
7
```

- ② 「ENTER」を押すと、5行目の左側にスペースが空いた場所にカーソルが移動する。  
この改行とスペース空けた書き方を「階層（又はインデント）を意識した書き方」と言う。



# public static void main(String[] args){} を追加で記述



```
*Main.java
1 package jp.co.internous.action;
2
3 public class Main {
4
5     public static void main(String[] args) {
6     }
7
8 }
9
```

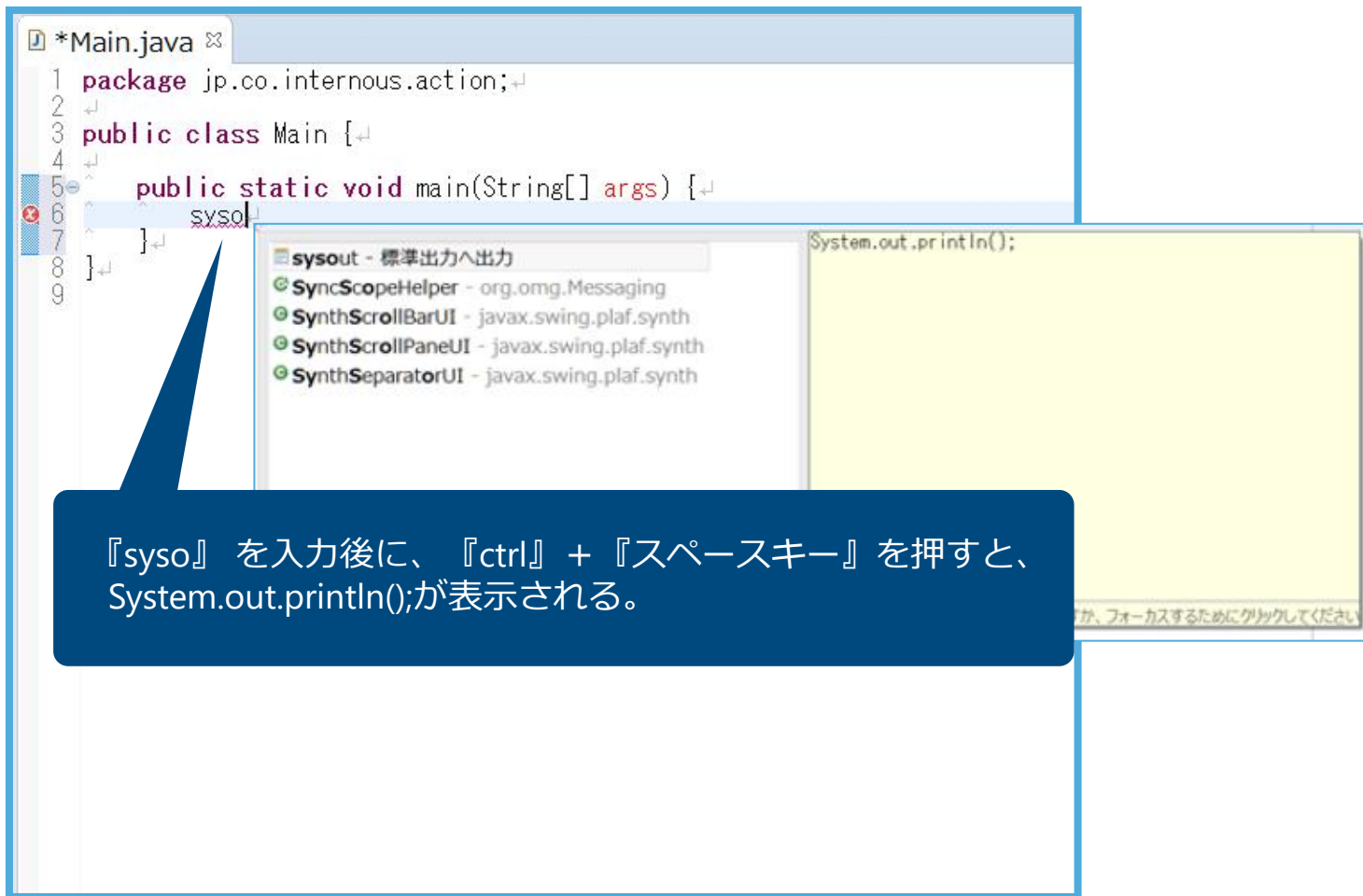
下記を追加で記述する。

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

# System.out.println();のショートカットキー

解説

『System.out.println();』を表示させる為には、下記のショートカットを活用すると便利です。  
『syso』を入力後に『ctrl』+『スペースキー』を押すと、下記の画面が表示されるので、ここで『ENTER』を押す。



# Hello Worldを記述

『\*』が表示されているうちは、保存されていないという意味

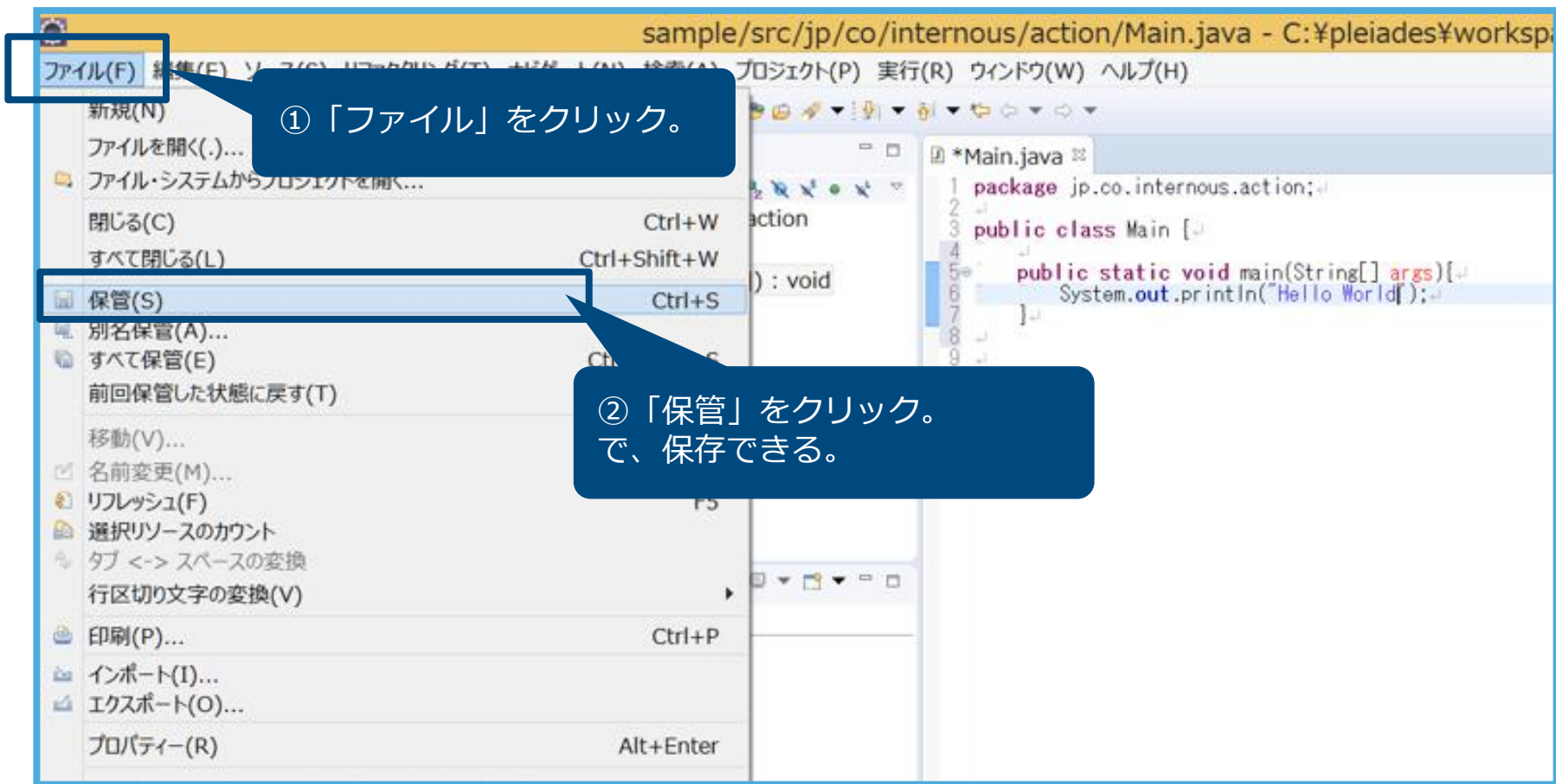
\*Main.java

```
1 package jp.co.internous.action;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         System.out.println("Hello World");
7     }
8
9 }
10
```

下記を追加で記述する。

```
System.out.println("Hello World");
```

# 保存



# 実行する

ファイル(F) 編集(E) ソース(S) リファクタリング(T) 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)

① 『Main.java』を右クリック。

② 『実行』を選択

③ 『Javaアプリケーション』をクリック。

```
1 package jp.co.internous.action;
2
3 public class Main {
4
5     public static void main(String[] args){
6         System.out.println("Hello World");
7     }
8 }
9
10
11
```

現在、表示するコンソールがありません。

# 実行結果

