

Java の環境構築

Pleiades All In One Eclipse のインストール

Pleiades All in One は統合開発環境である Eclipse 本体と、日本語化を行うための Pleiades プラグインおよびプログラミング言語別に便利なプラグインを加え、さらに Web アプリケーションサーバ（Apache Tomcat）をセットした Windows、Mac 向けパッケージである。

そのため、このソフトウェアをインストールするだけで、開発環境が準備できる

1.1 Pleiades All In One Eclipse のダウンロード

下記 URL へアクセスする

<https://willbrains.jp/>

Eclipse バージョン	Windows		macOS		Java Full Edition 付属 JDK バージョン								
	32bit	64bit	Intel	ARM	1.4	1.5	6	7	8	11	17	21	25
2025 (予定)													9月
2024													
2023													
2022													

上記のような画面となるので、Eclipse2023 を選択する

※Eclipse2023 の方が安定しているため

	Platform	Ultimate	Java	C/C++	PHP	Python
Windows x64 32bit は 2018-09 で終了	Full Edition	Download	Download	Download	Download	Download
Mac Mac 版について (Qiita)	Full Edition	Download	Download	Download	Download	Download

いくつかの開発言語に対応したパッケージがあるが、ここでは

Java の Full Edition を選択する

自動でダウンロードが始まる。

MergeDoc Project

Download Google Chrome

Chrome works on any operating system and on any device. Customize any way you want.

Google Chrome

Download

Preparing to download Eclipse Pleiades All in One
https://ftp.jaist.ac.jp/pub/mergedoc/pleiades/2023/pleiades-2023-12-java-win-64bit-jre_20240218.exe
SIZE: 1226059254 bytes
MD5: f539cd69a2858e3203988b5c4bc2b2f4
Downloading ... from IAIST

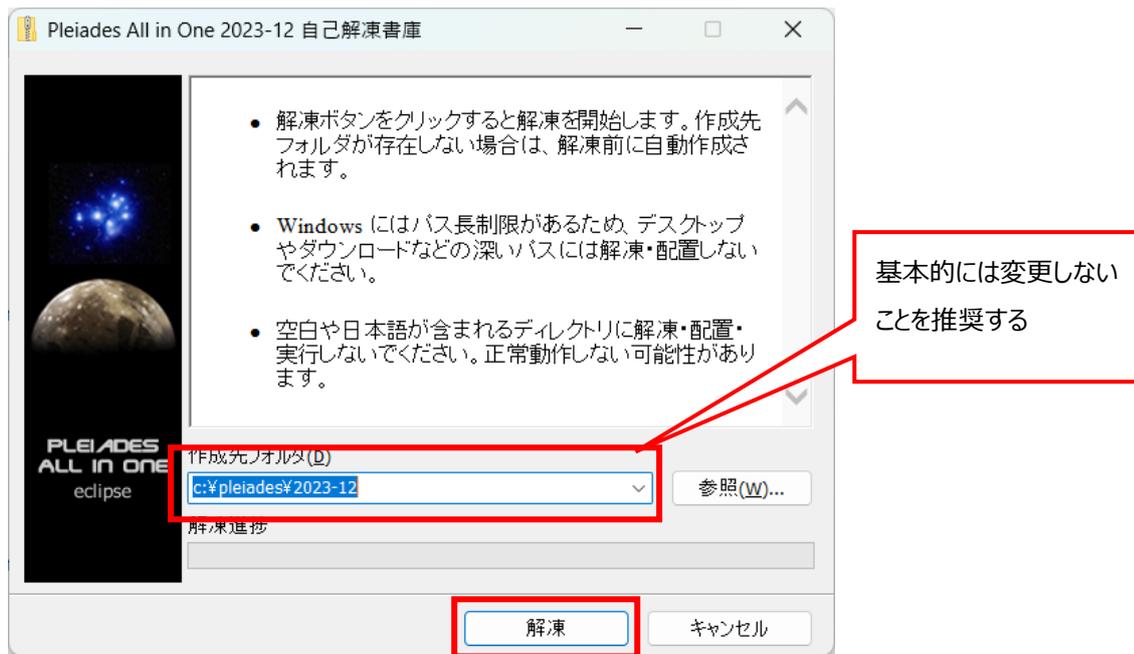
もし、自動で始まらない場合は、表示された URL をクリックして直接ダウンロードを実行する

1.2 Pleiades のインストール

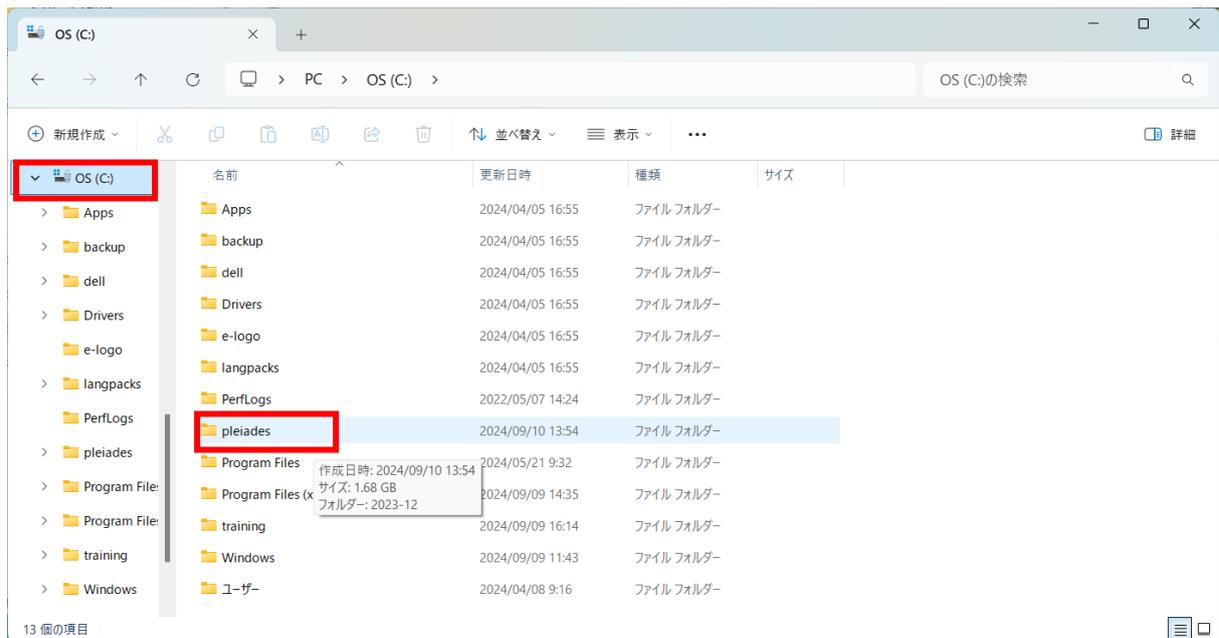
- ① ダウンロードした exe ファイルを実行する

名前	更新日時	種類	サイズ
pleiades-2023-12-java-win-64bit-jre_20240218.exe	2024/09/10 13:25	アプリケーション	1,199,258 ..
VSCoUserSetup-x64-1.93.0.exe	作成日時: 2024/09/10 13:23 サイズ: 1.14 GB	アプリケーション	98,877 KB
x86_64-14.2.0-release-win32-def-msvcrt-rt_v12-rev0	2024/09/09 14:35	ファイル フォルダ	
今年に入って (今月は含めず)			
サンプルデータ.zip	2024/05/16 14:46	圧縮 (zip 形式) フォ...	5 KB
sakura-tag-v2.4.2-build4203-a3e63915b-Win32-Release-Inst...	2024/05/16 10:36	圧縮 (zip 形式) フォ...	5,425 KB
TrendMicro_17&_HE_Download.exe	2024/05/16 10:28	アプリケーション	11,582 KB
MSTeams-x64.msi	2024/05/16 9:57	MSIX ファイル	169,920 KB
ChromeSetup.exe	2024/05/16 9:50	アプリケーション	1,345 KB
サンプルデータ	2024/05/16 14:46	ファイル フォルダ	
sakura-tag-v2.4.2-build4203-a3e63915b-Win32-Release-Inst...	2024/05/16 10:41	ファイル フォルダ	

② インストール先を指定し「解凍」ボタンをクリックする

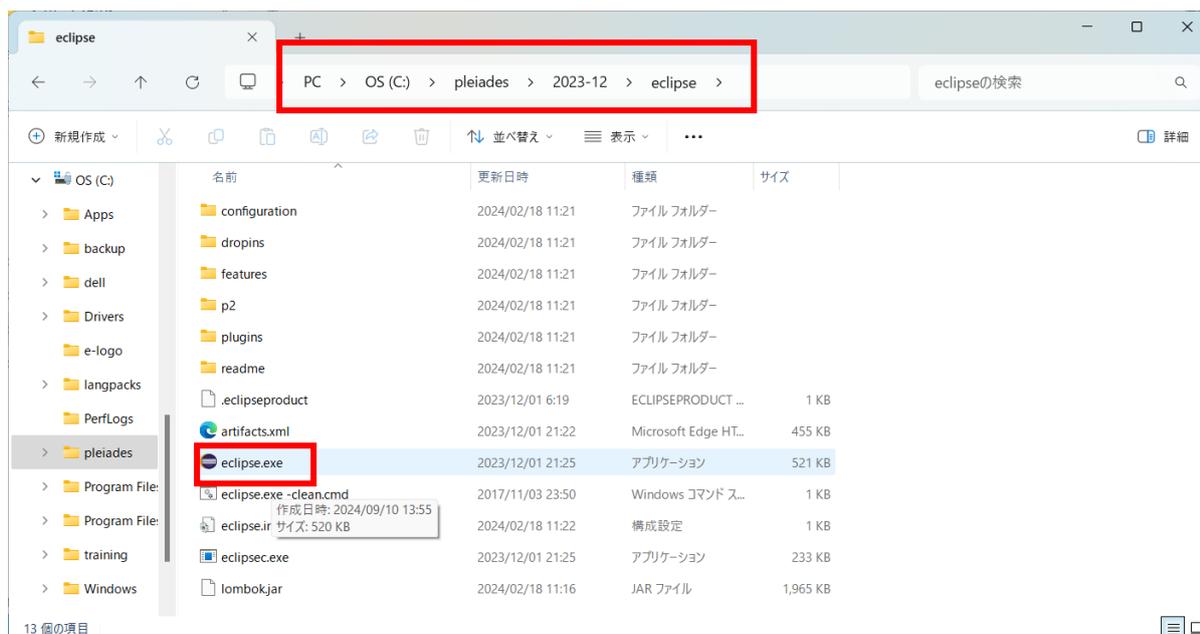


③ フォルダが作成されることを確認する



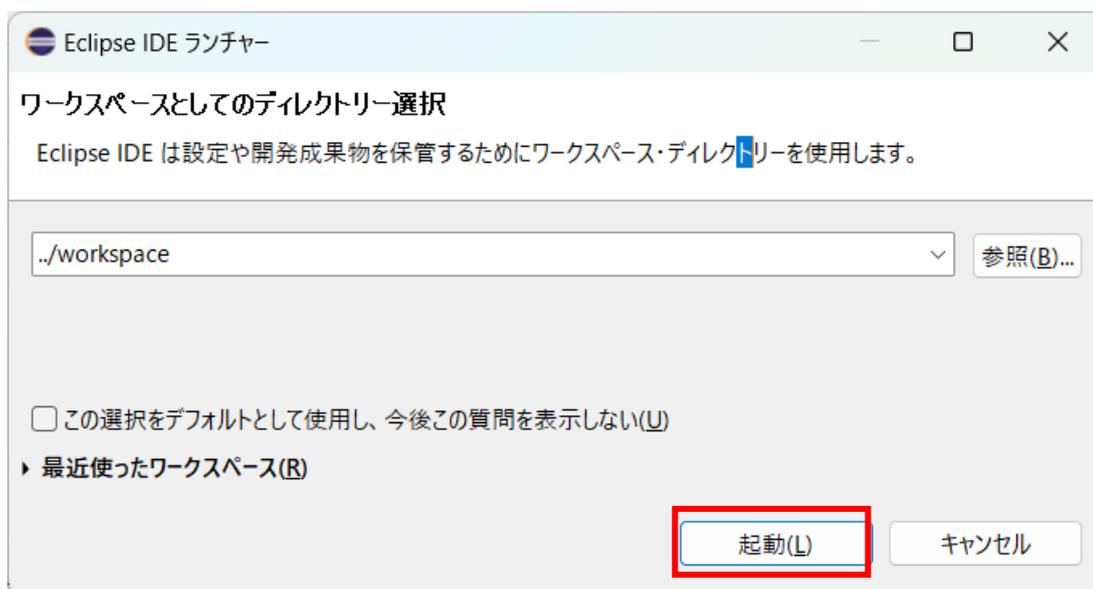
②で指定した場所に pleiades フォルダが作成されていることを確認する

1.3 Eclipse の起動



- ① C:\pleiades\2023-12\eclipse フォルダの eclipse.exe を実行する

作成したソースコードなどのファイルを保存する場所を設定する画面が表示されるので、「起動」ボタンを押下する



- ② C:\pleiades\2023-12\workspace フォルダが作成される
- ③ Eclipse の画面が表示されればインストール完了

1.4 パッチ適用

Eclipse の WTP プラグインの不具合を修正する必要がある。

下記 URL を参考にしパッチを適用する

[Eclipse プラグインの修正 \(Windows\)](#)

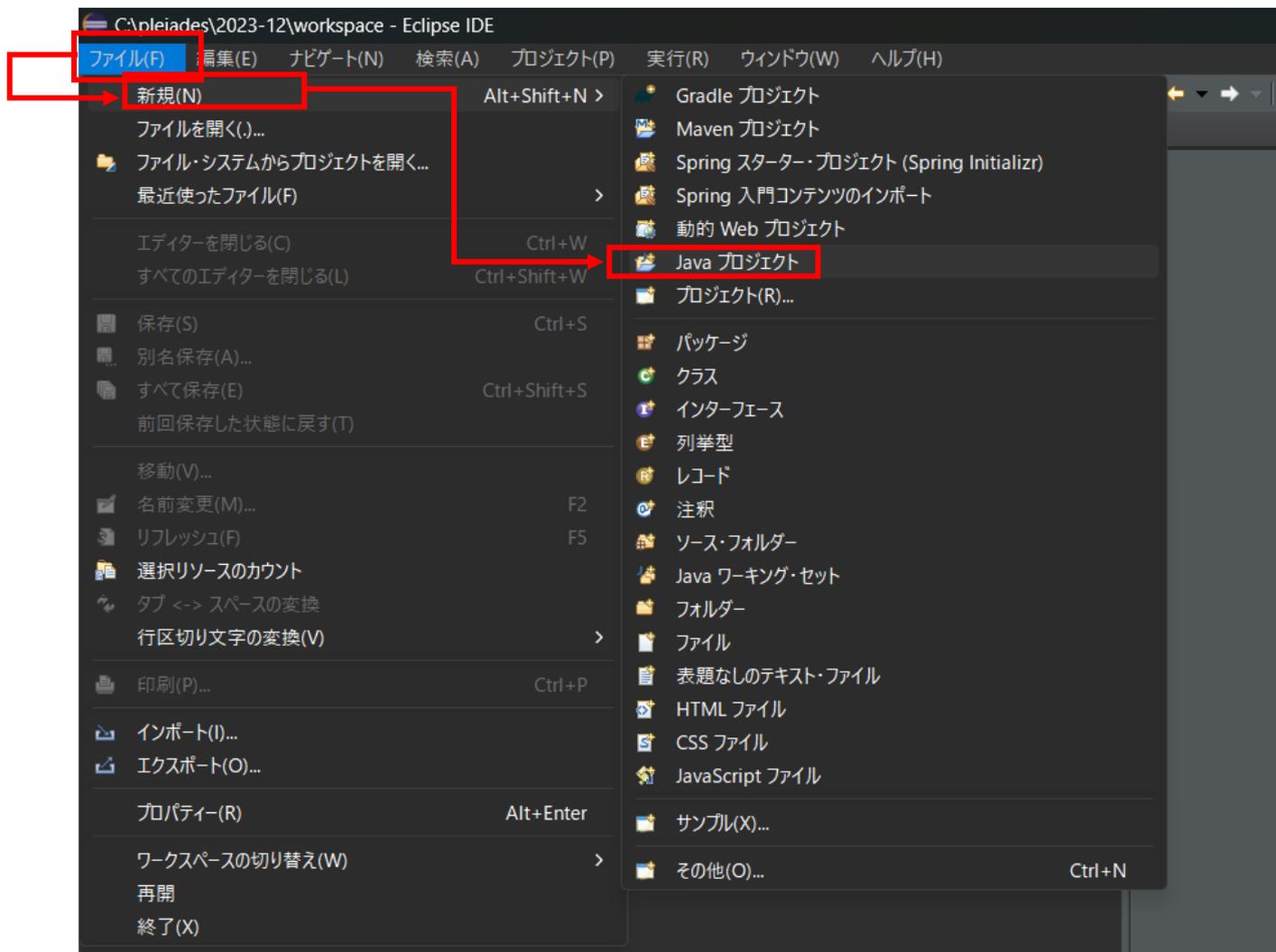
Java アプリケーションの開発方法

Eclipse をインストール後、コマンドラインに「Hello World」を表示するまでの流れを説明する

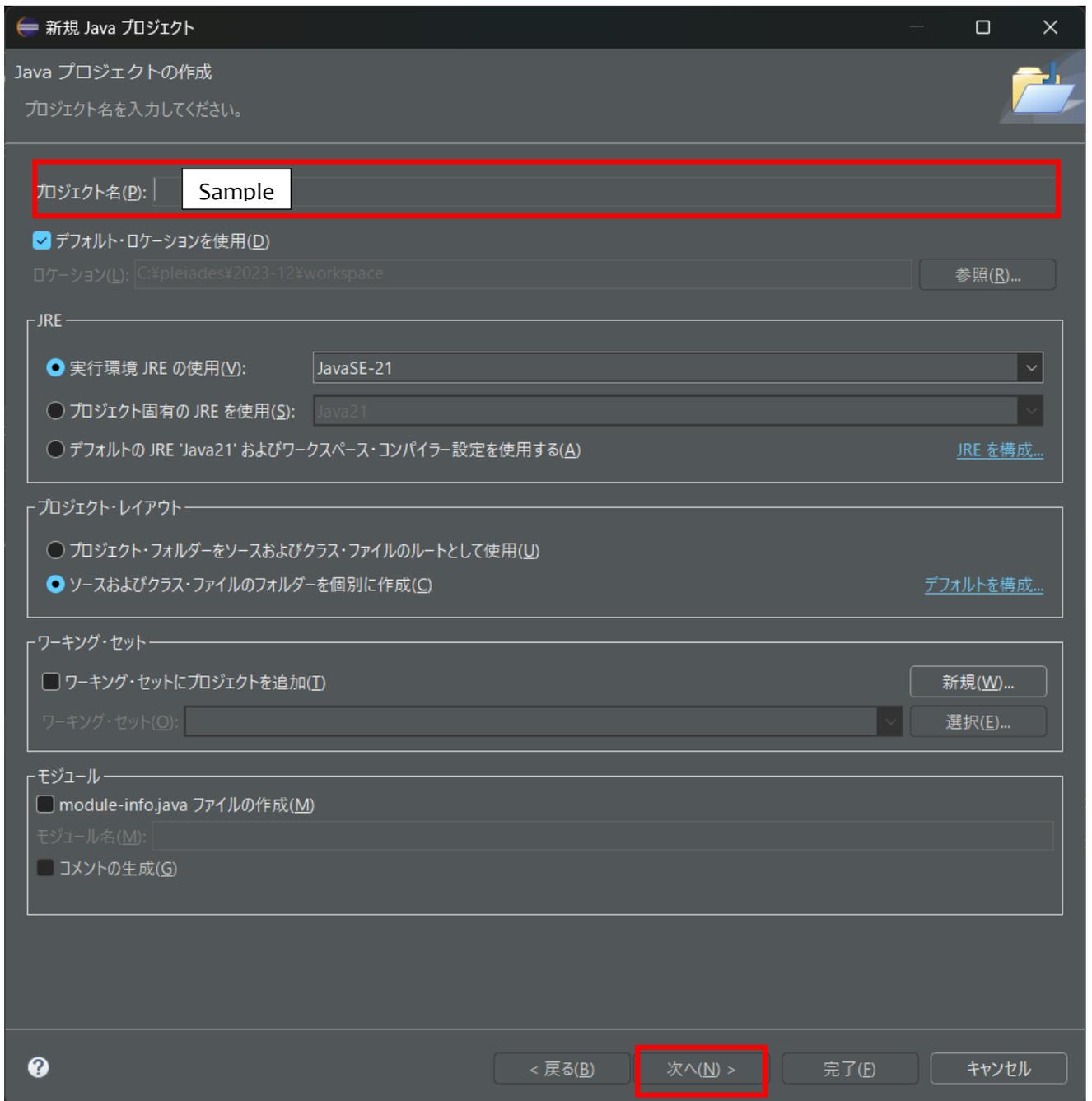
1.1 プロジェクト作成

プロジェクトとは、プログラム一式をまとめて管理するもので、プロジェクトの中にソースファイルやライブラリなどを格納していく

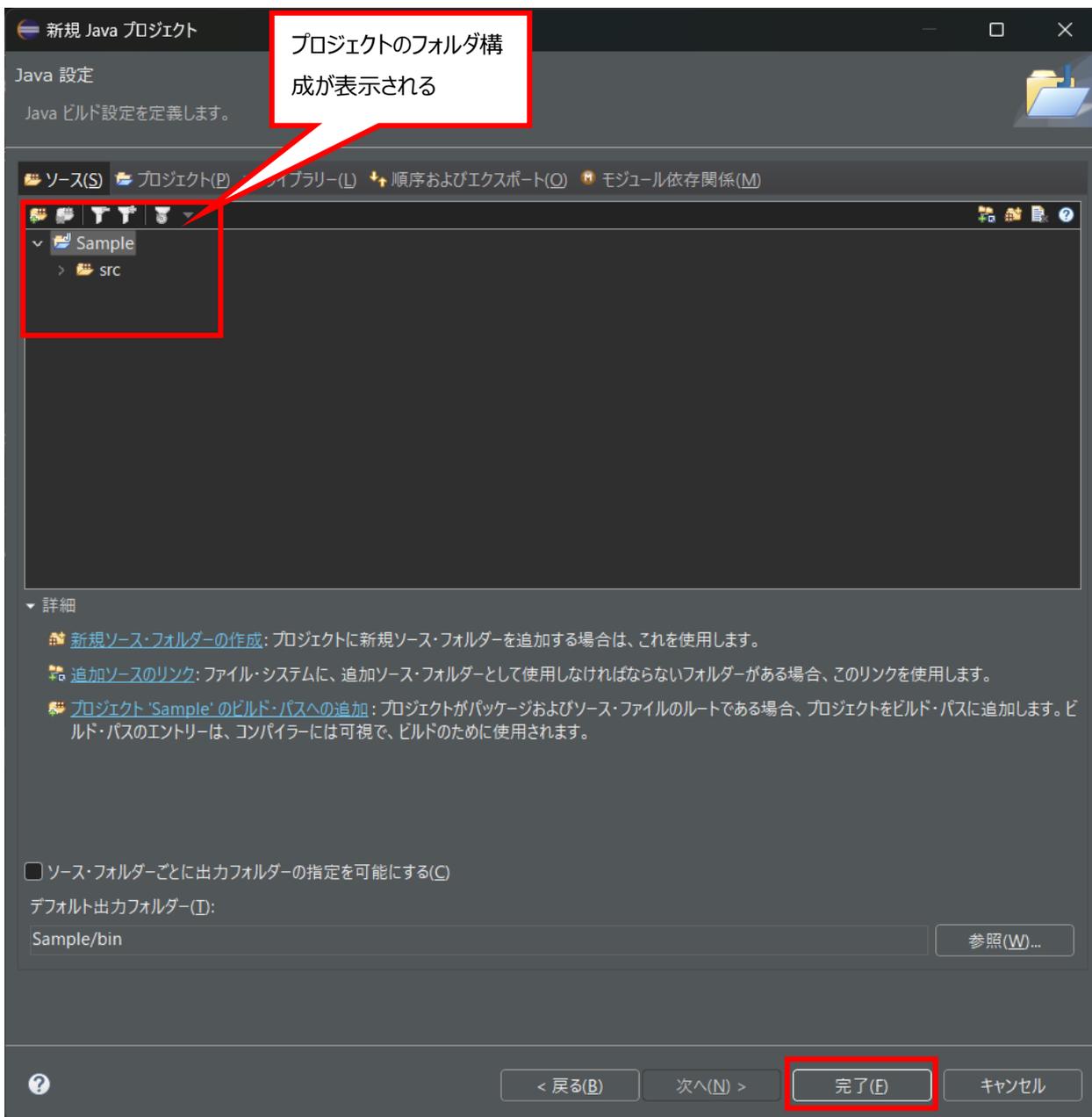
- ① 「ファイル」⇒「新規」⇒「Java プロジェクト」の順にクリックする



- ② プロジェクトの名前を入力する

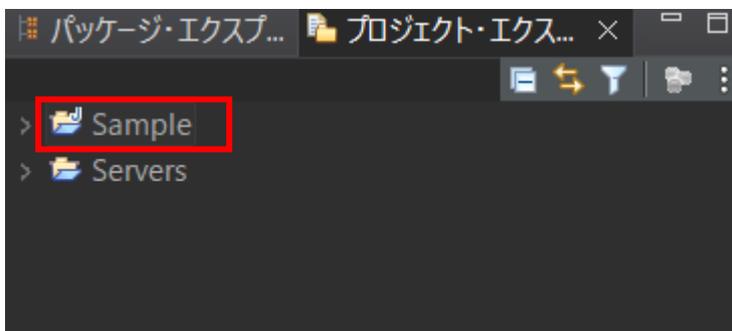


プロジェクト名に「Sample」と入力し「次へ」ボタンを押下する



「完了」ボタンを押下する

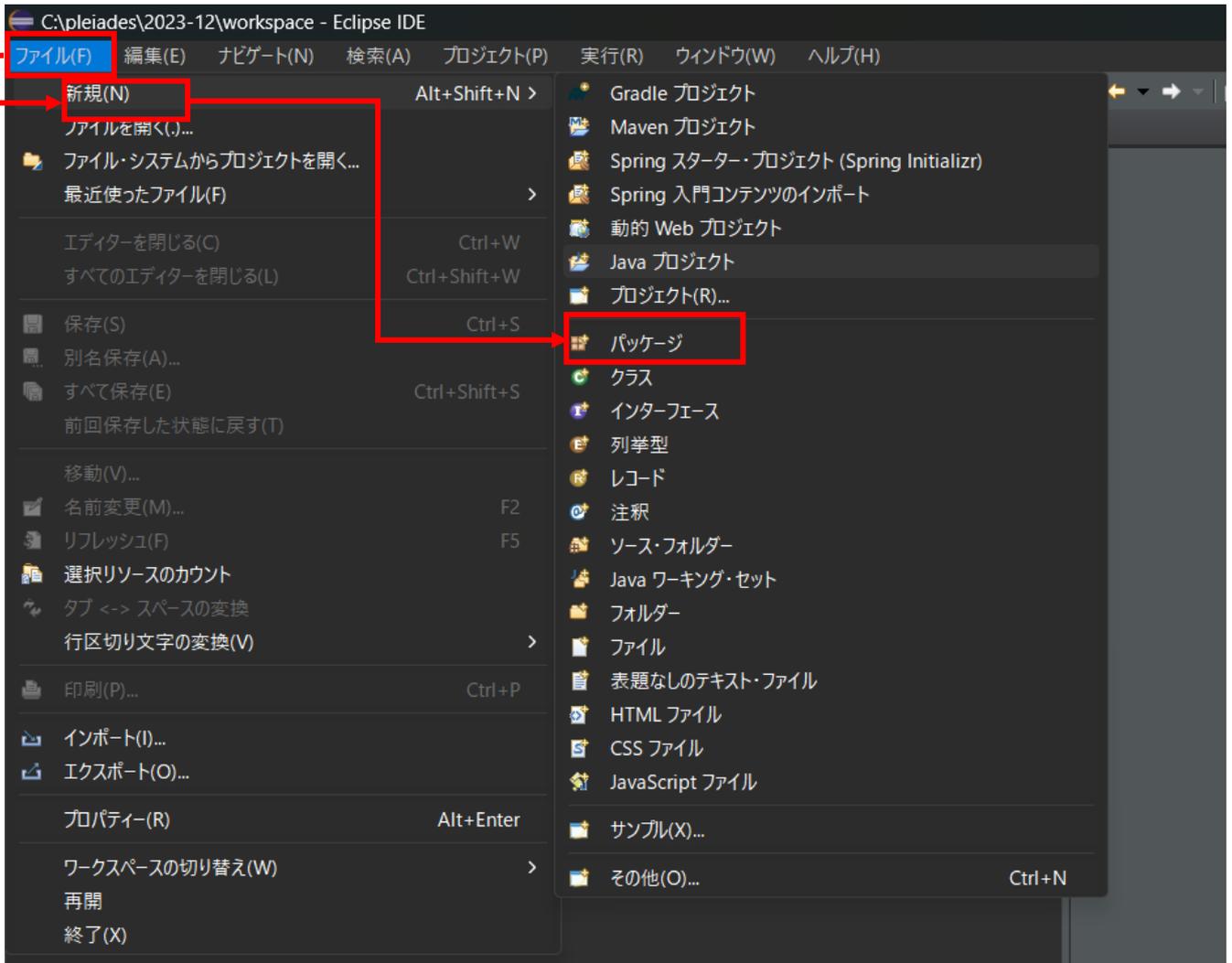
- ③ プロジェクトが作成され、プロジェクトエクスプローラーに追加される



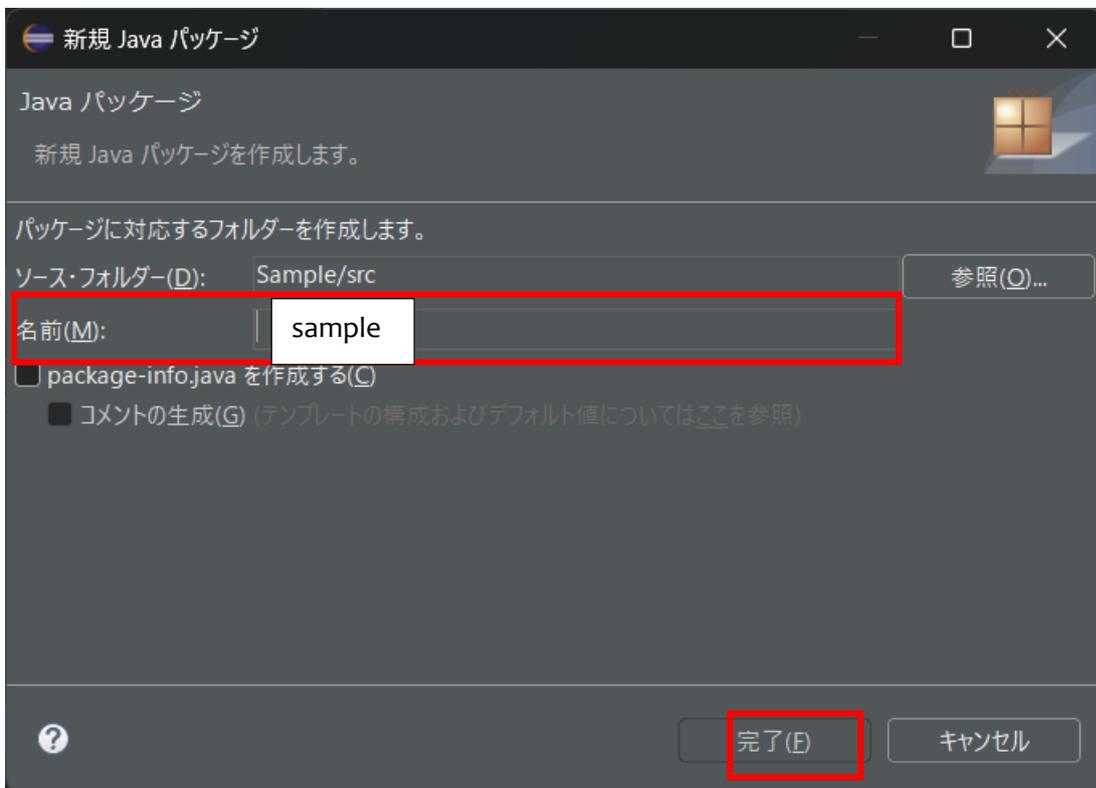
1.2 パッケージ作成

ソースファイルなどを格納するためのパッケージを作成する

- ① 「ファイル」⇒「新規」⇒「パッケージ」の順に選択する

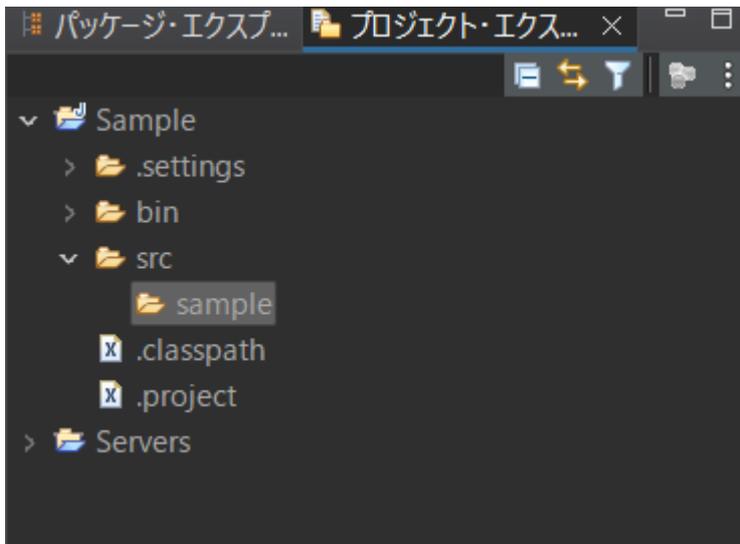


② パッケージ名を入力する



名前に「sample」と入力し「完了」ボタンを押下する

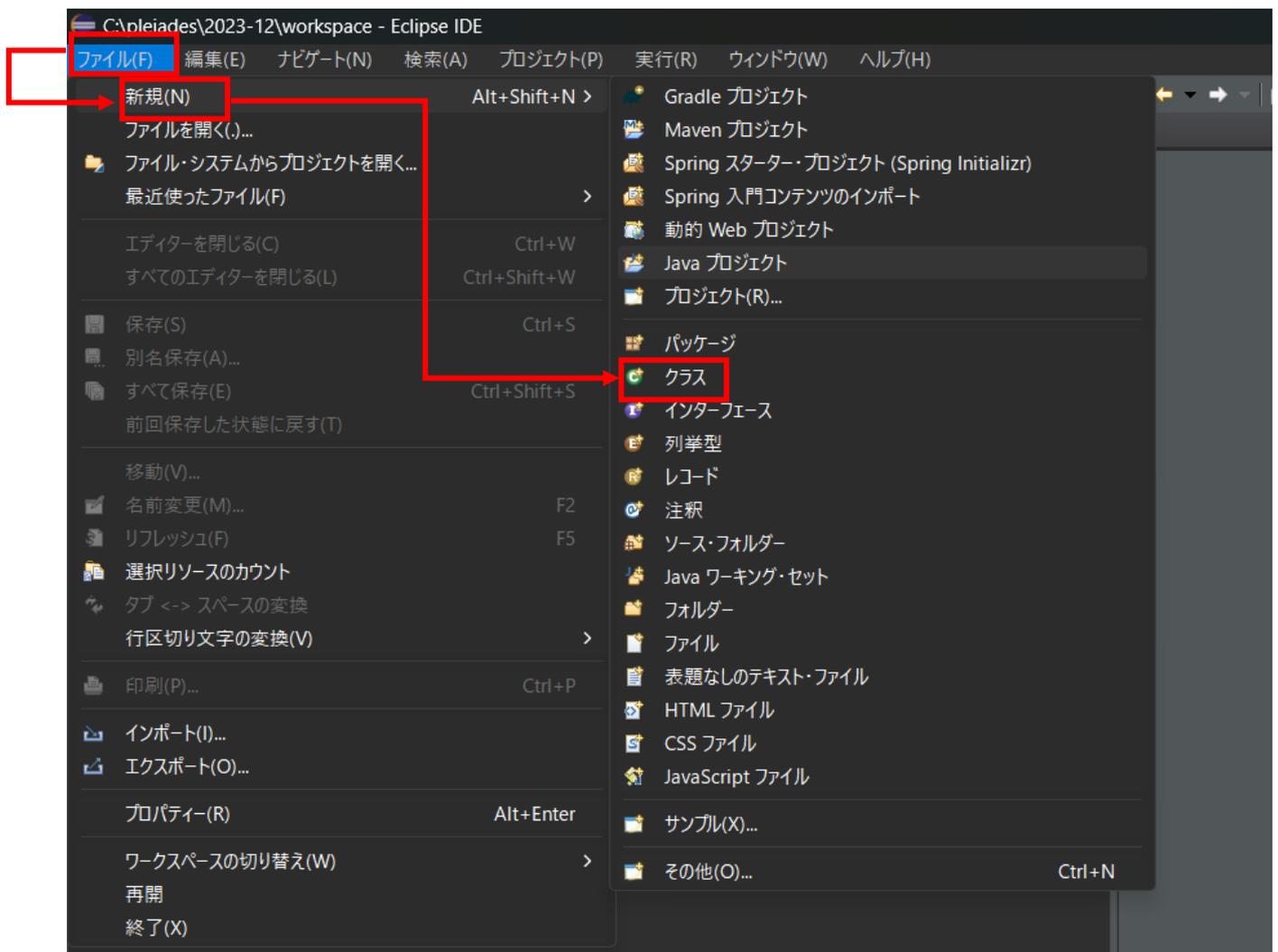
- ③ プロジェクトの src 直下にパッケージが作成される



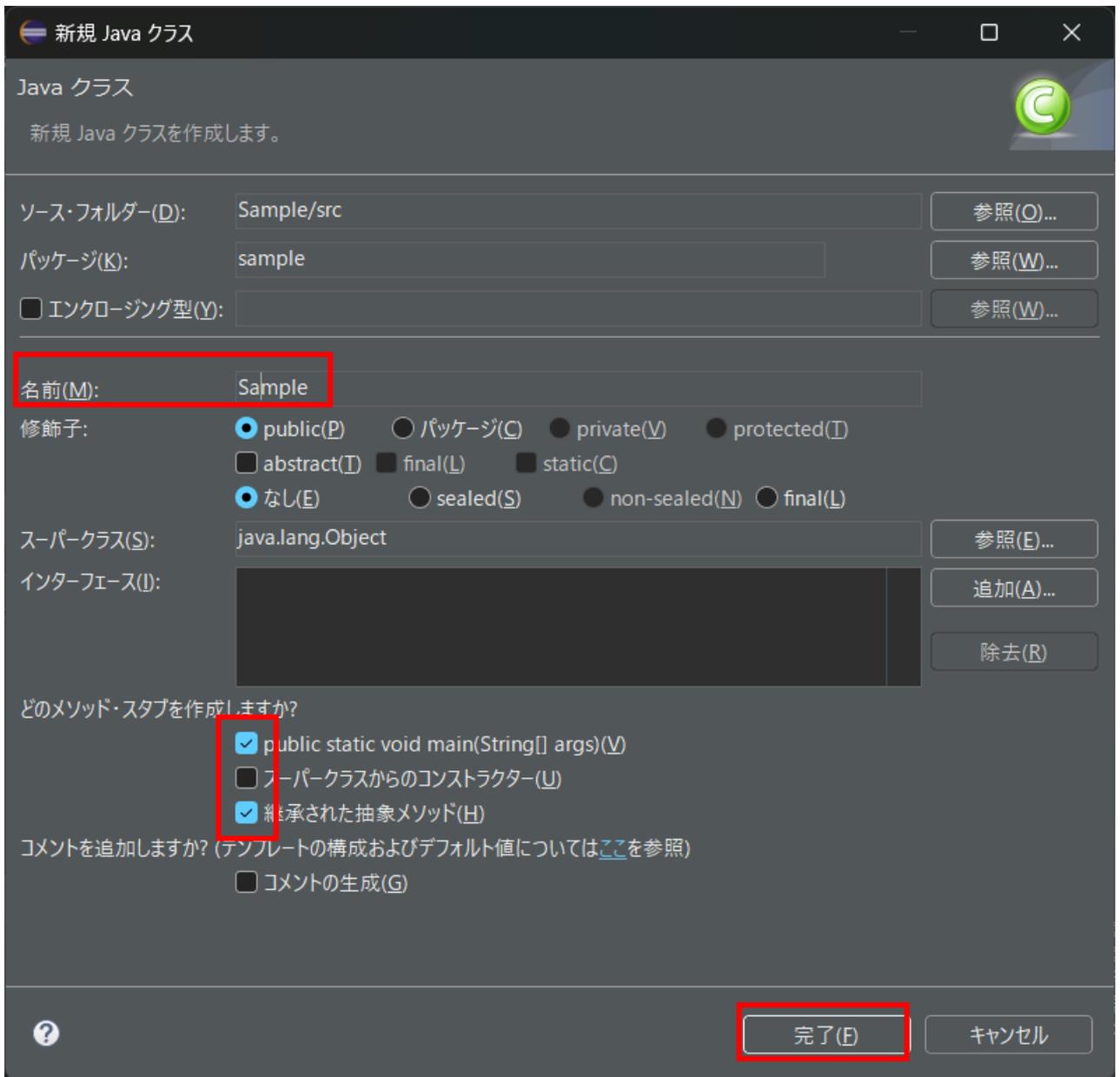
1.3 クラス作成

1.2 で作成したパッケージの下にクラスを作成する

- ① パッケージを選択した状態で、「ファイル」⇒「新規」⇒「クラス」の順に選択する



- ② クラス名などの情報を入力する



クラス名を Sample と入力する

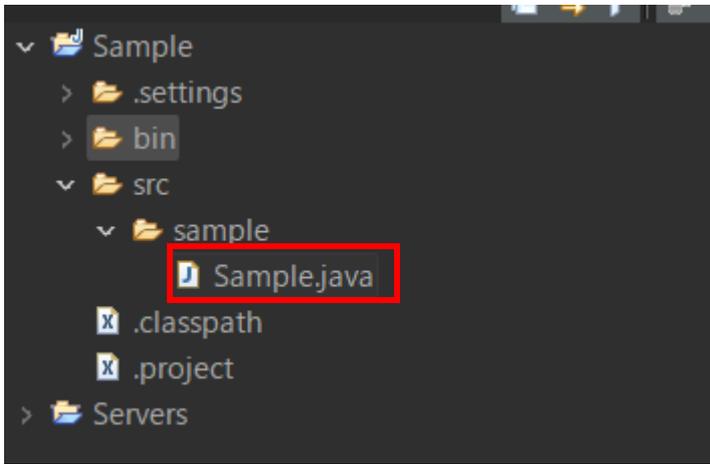
public static void main(Staing[] args)にチェックを入れると Java プログラム作成時に、必要な main メソッドが自動で生成される

継承された抽象メソッドはデフォルトでチェックが入るので、そのままチェックを付けておく

一通り入力後「完了」ボタンを押下する

③ Sample.java ファイルが生成される

```
*Sample.java ×
1 package sample;
2
3 public class Sample {
4
5     public static void main(String[] args) {
6         // TODO 自動生成されたメソッド・スタブ
7     }
8 }
9
10
11
```



1.4 プログラムの作成

Sample.java にソースを追加する

```
package sample;

public class Sample {

    public static void main(String[] args) {

        // TODO 自動生成されたメソッド・スタブ

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

    }

}
```

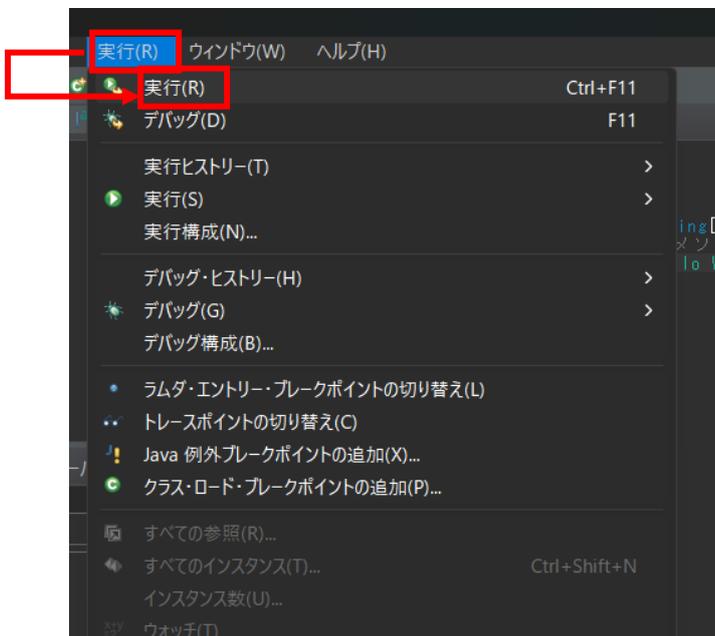
この行を追加する

ソース編集後、ファイルを保存する

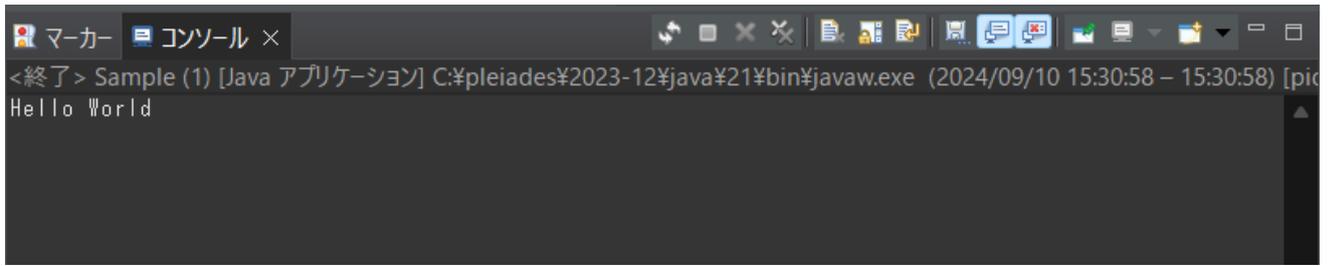
1.5 実行

① 実行する

「実行」⇒「実行」の順にクリックする



② コンソールに Hello World と表示されることを確認する



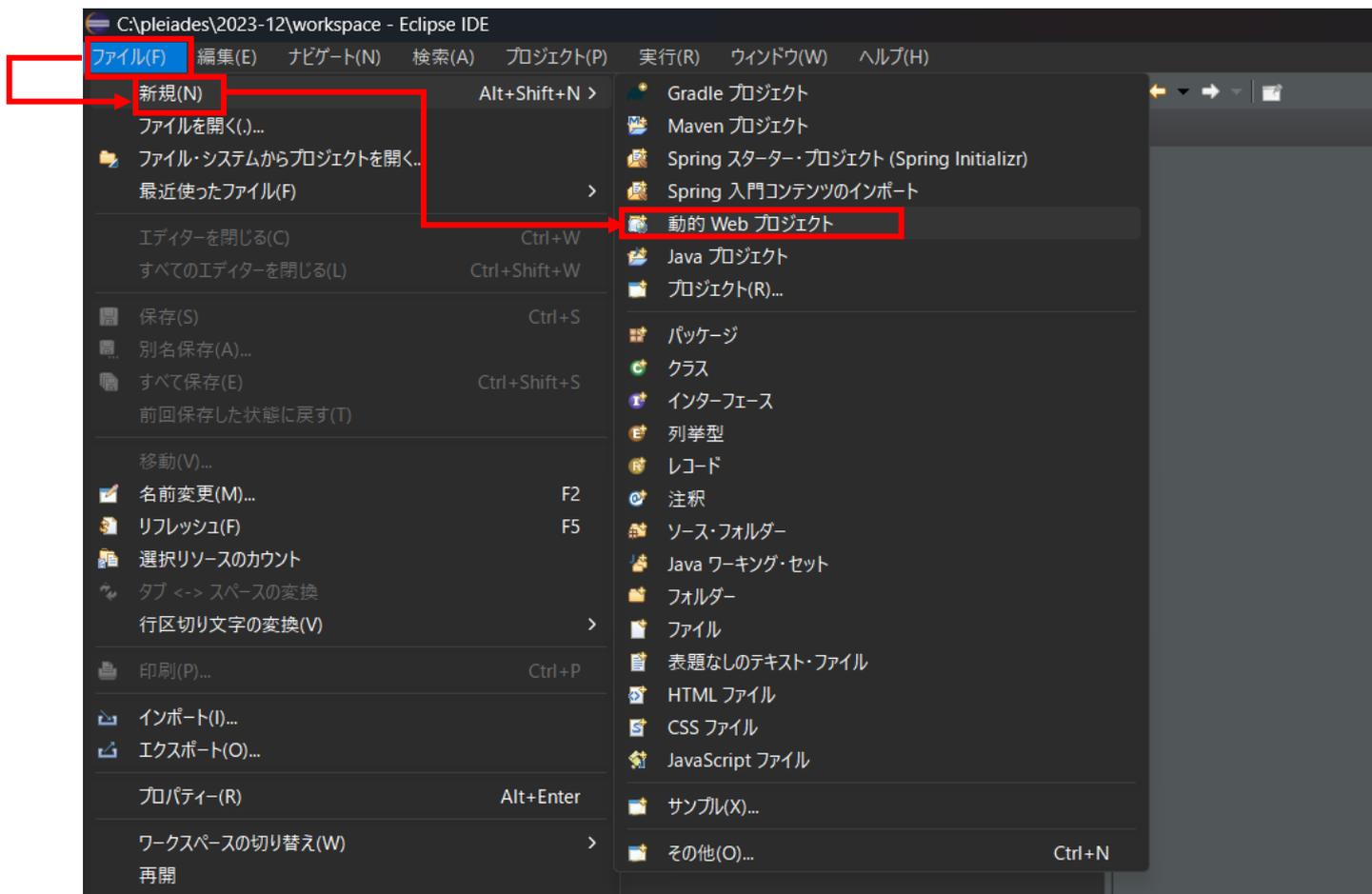
Java サブレットでの Web アプリケーション開発方法

Eclipse を起動し、Web プロジェクトを作成し起動するまでの手順を説明する

1.1 動的 Web プロジェクトの作成

※動的 Web プロジェクトとは、Web アプリケーションのためのプロジェクトのこと

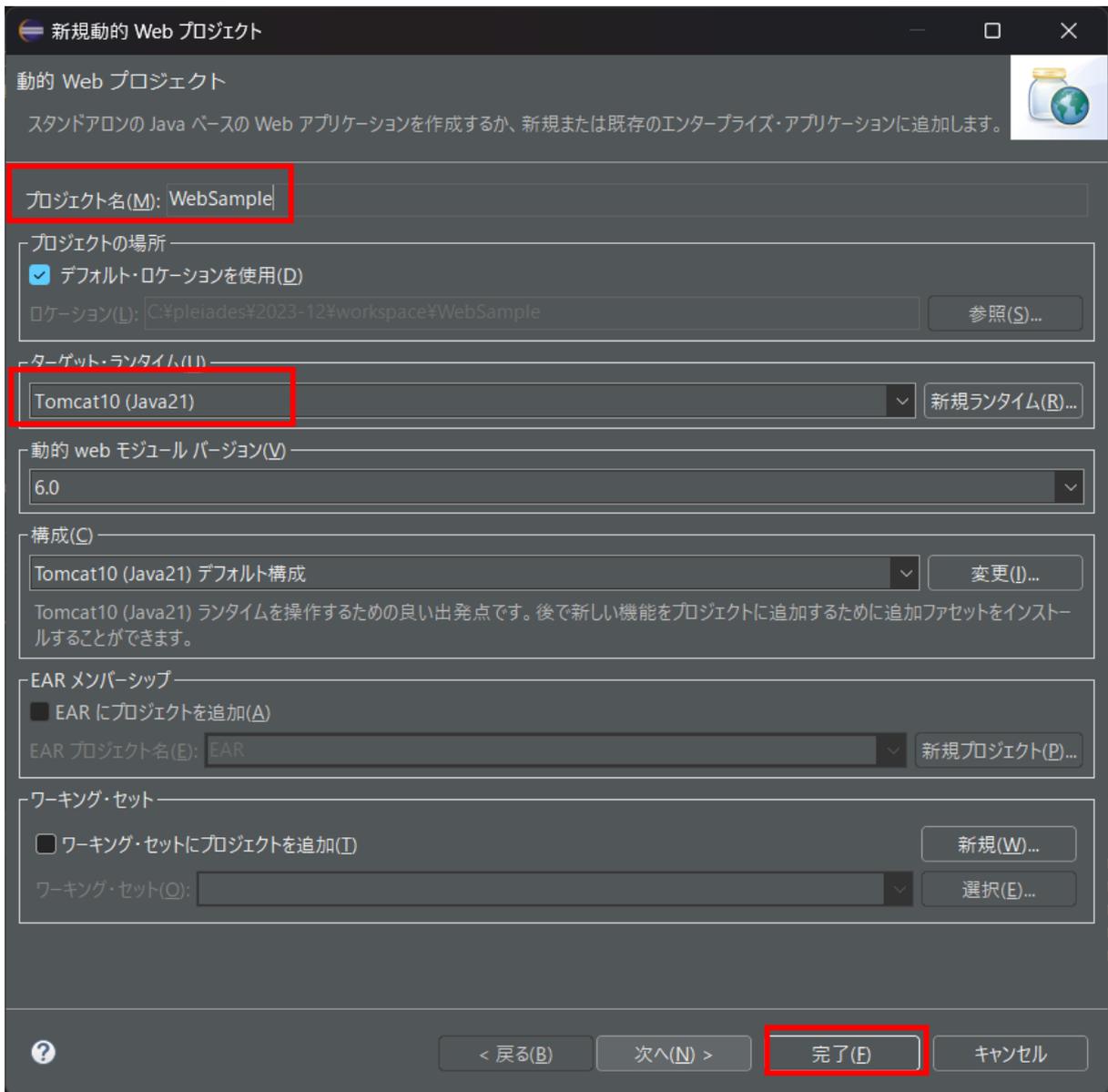
- ① 「ファイル」⇒「新規」⇒「動的 Web プロジェクト」を選択



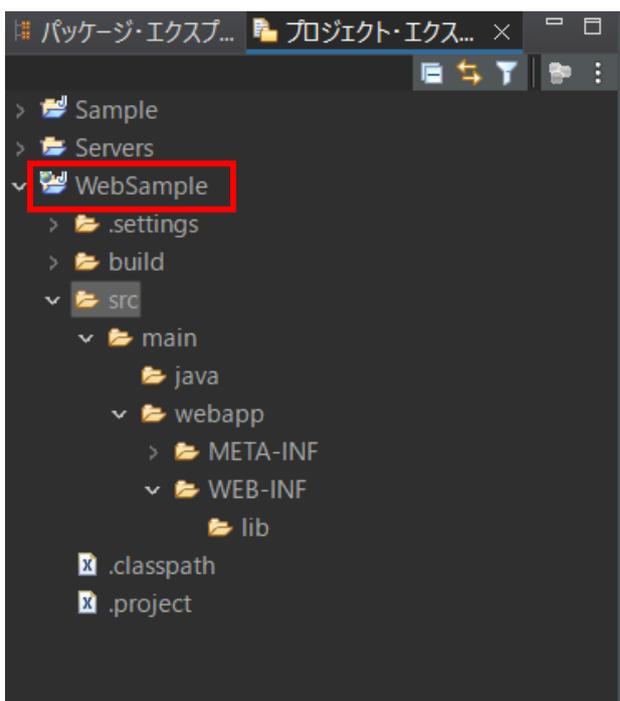
- ② プロジェクト名、ターゲット・ランタイムを選択

プロジェクト名 : WebSample

ターゲット・タイムライン : Tomcat10(Java21)を選択

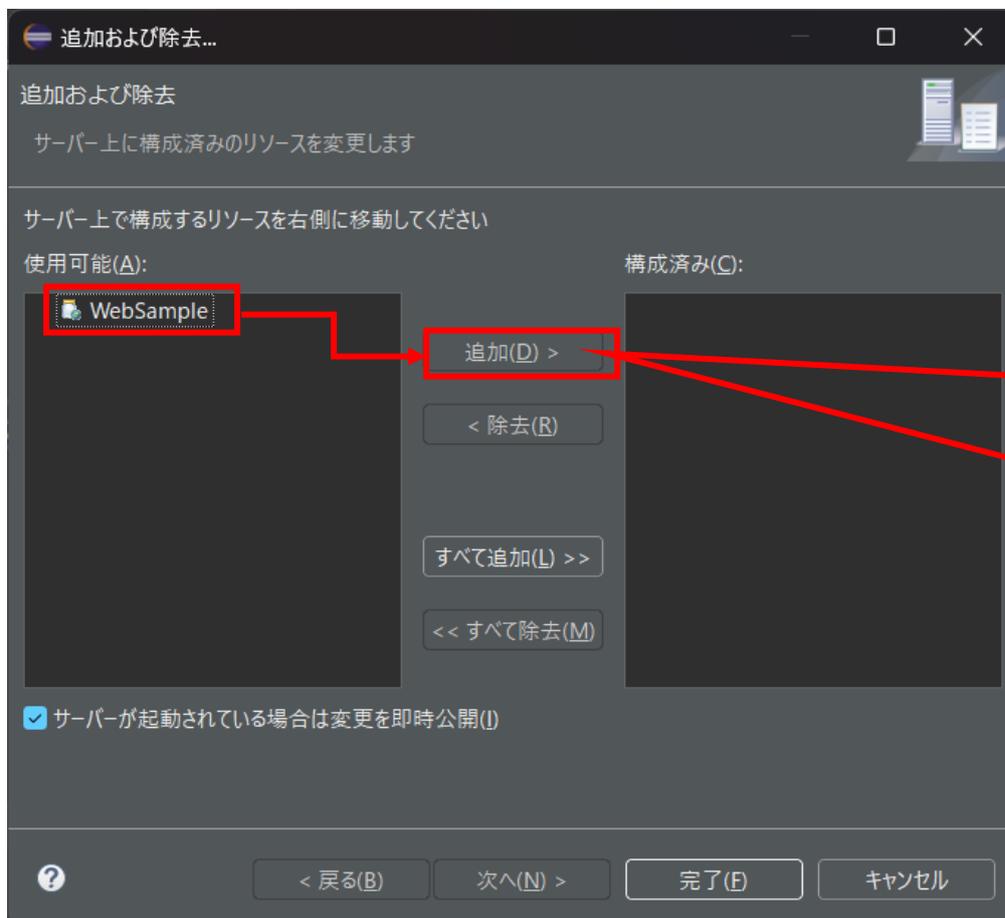
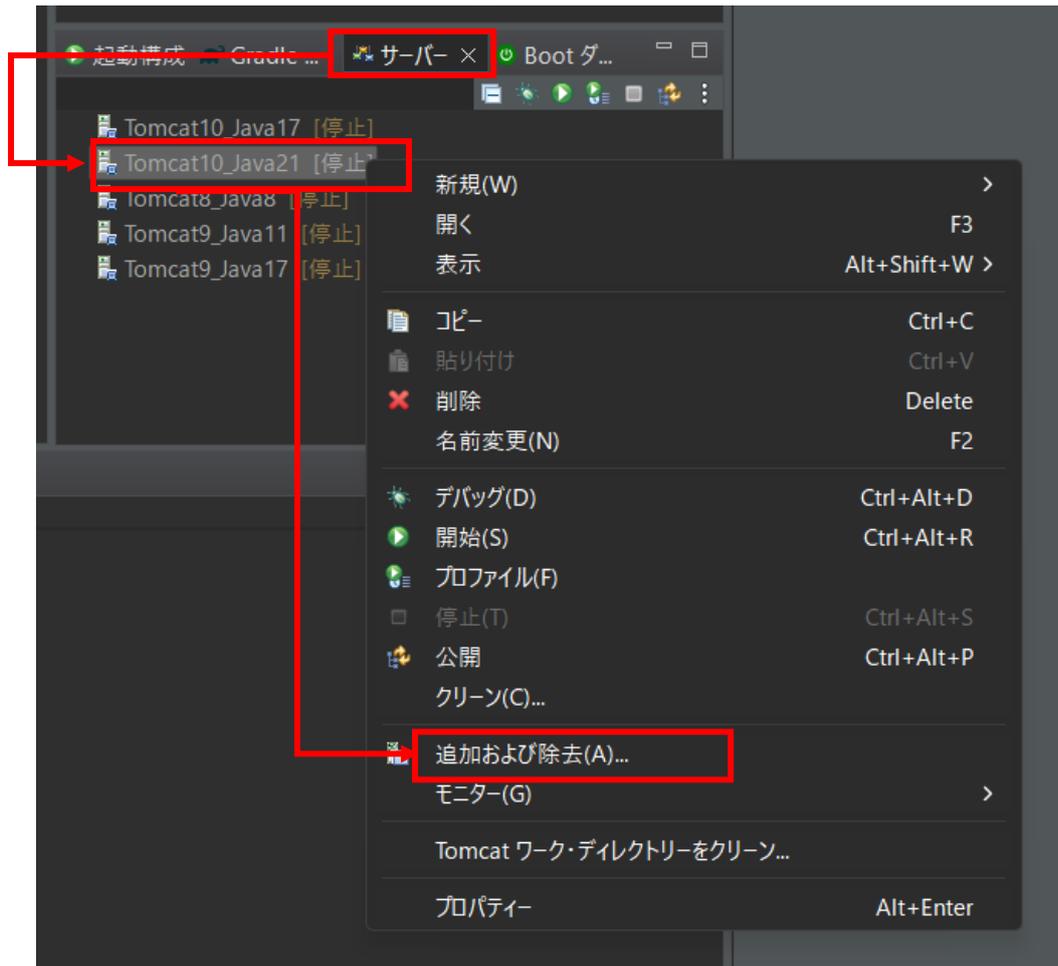


- ③ 完了後、プロジェクトが作成できていることをプロジェクトエクスプローラーで確認する

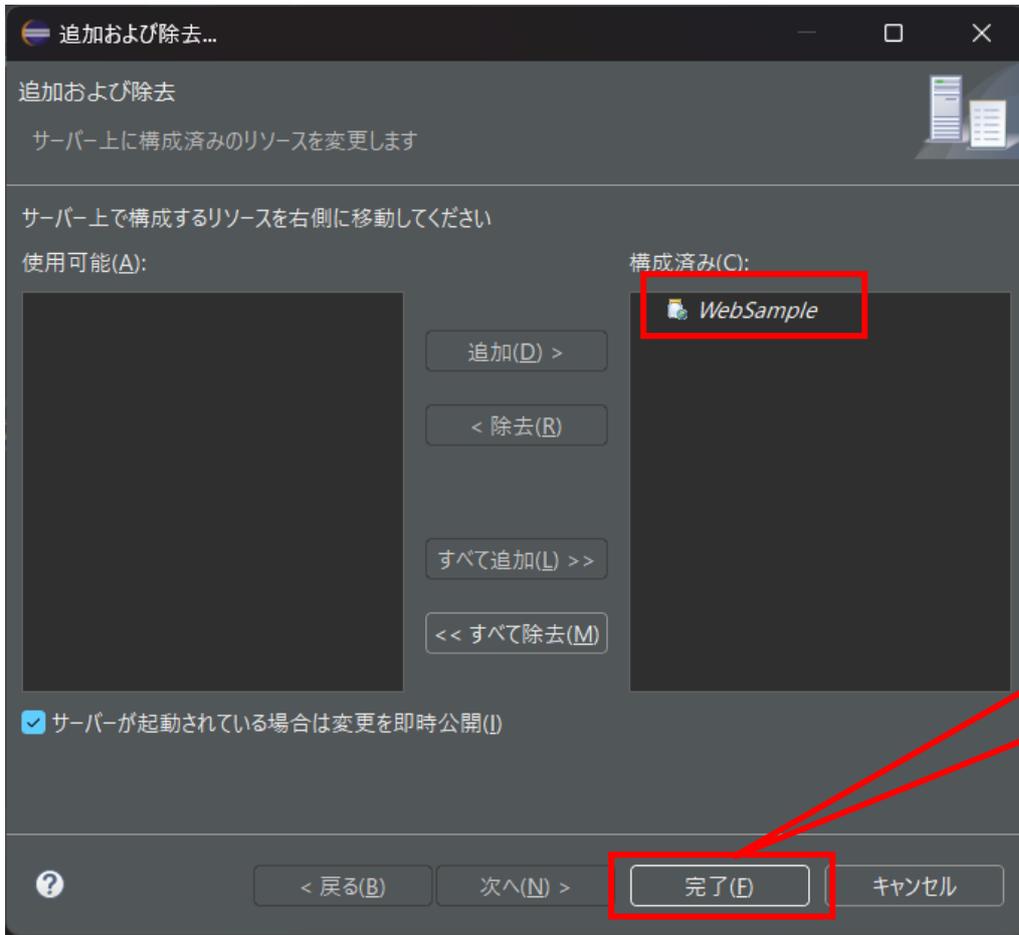


④ 動的 Web プロジェクトをサーバーへ追加

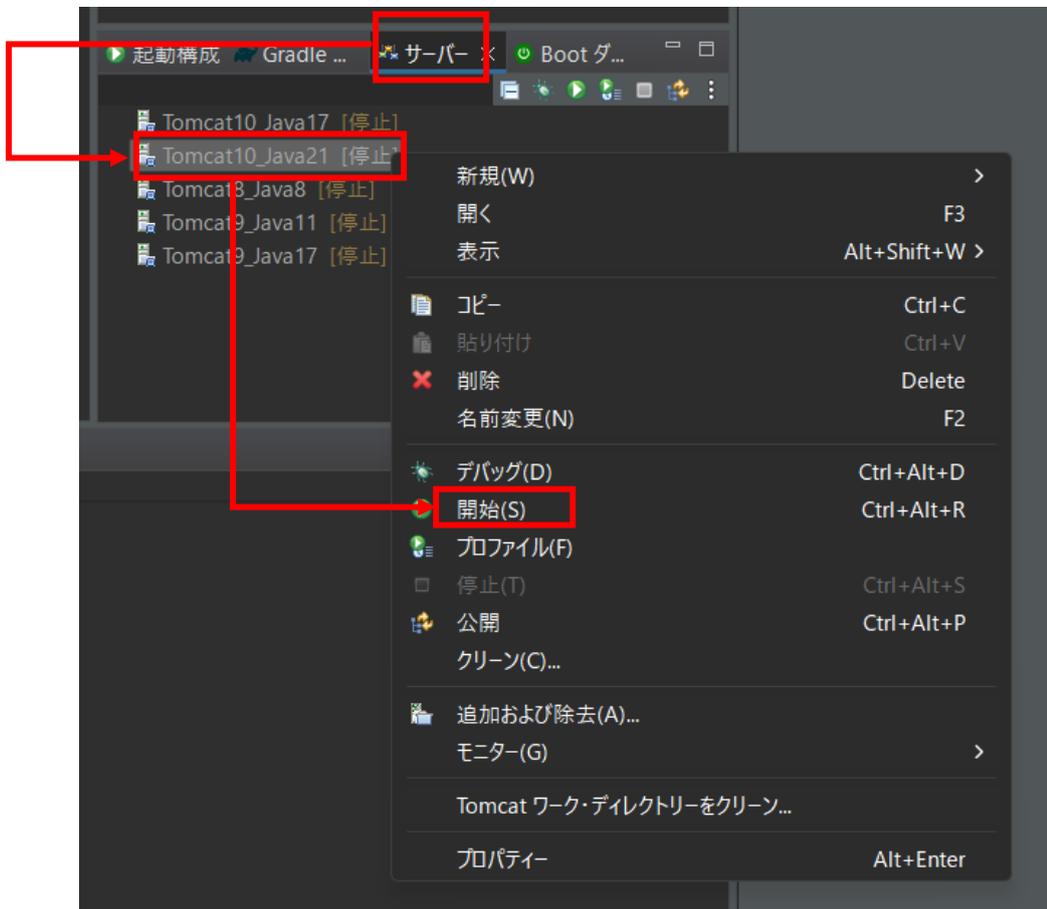
「サーバー」ビューを選択⇒「Tomcat10_Java21」を選択⇒右クリック⇒「追加および除去」を選択

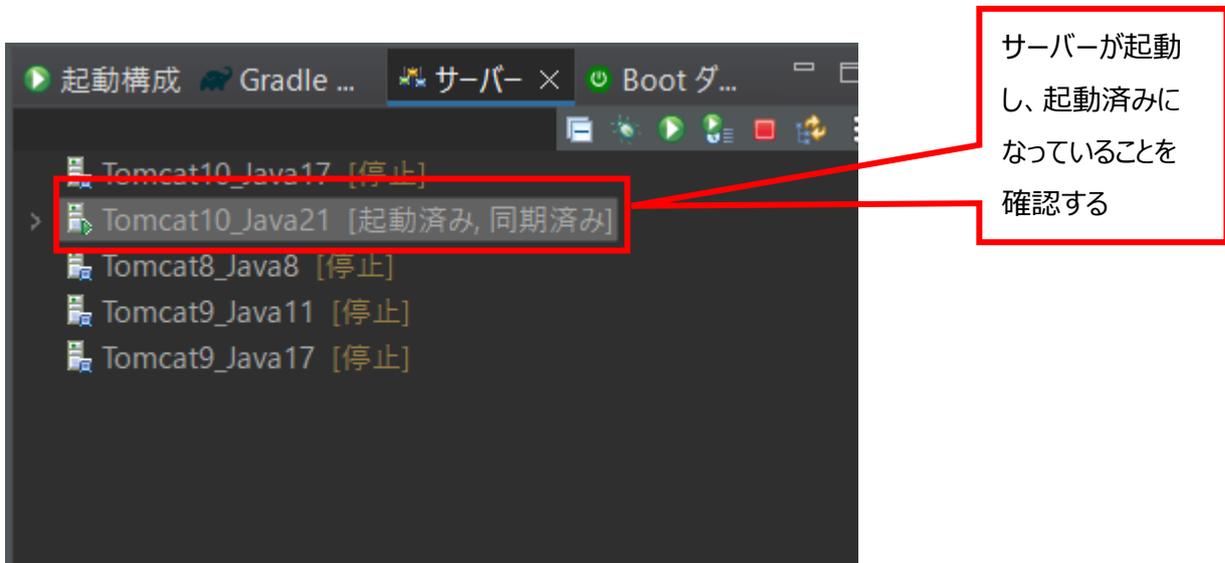


「使用可能」に表示されている動的 Web プロジェクトを選択し、「追加」ボタンを押下し、「構成済み」へ移動する



⑤ 「サーバー」ビューを選択⇒「Tomcat10_Java21」を選択⇒右クリック⇒「開始」を選択

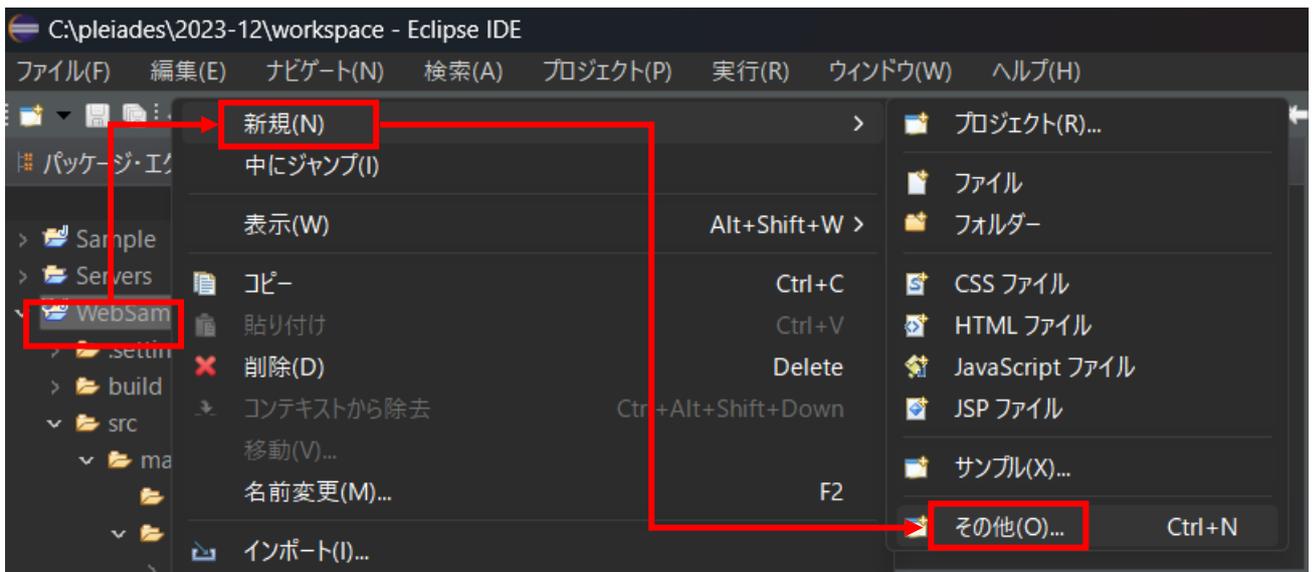




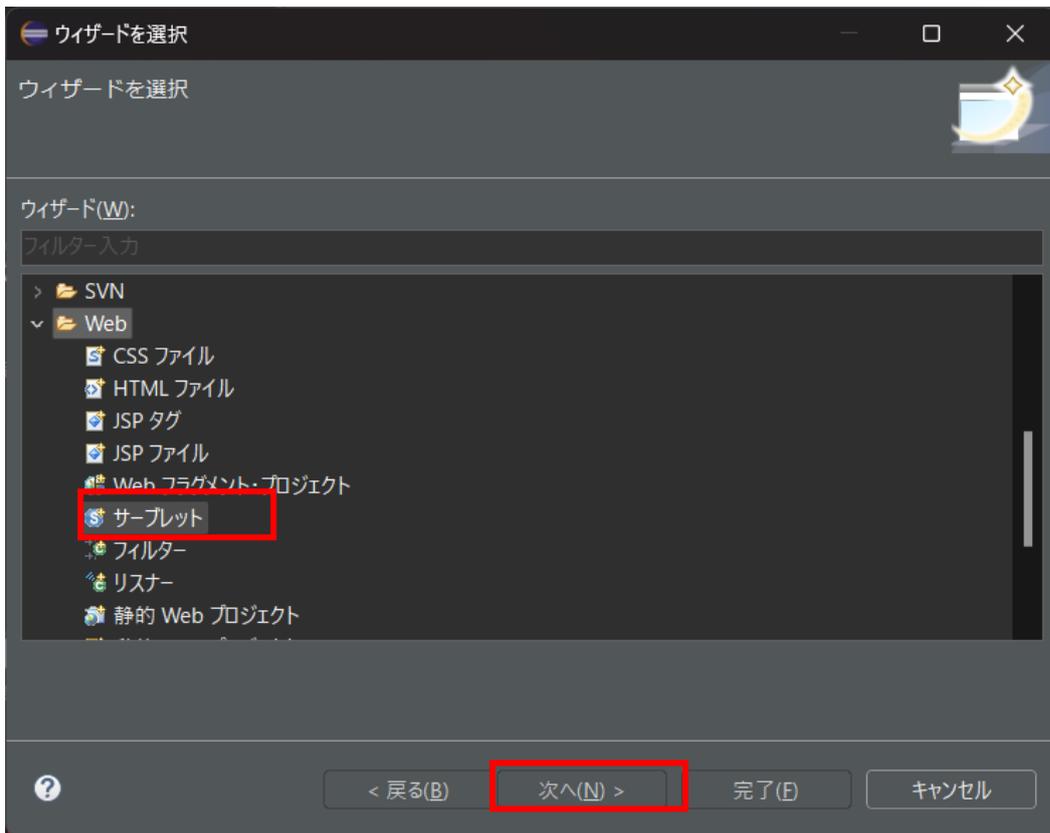
1.2 サーブレットクラスの作成

※サーブレットは Java を使いサーバサイドプログラムを作成する技術のこと

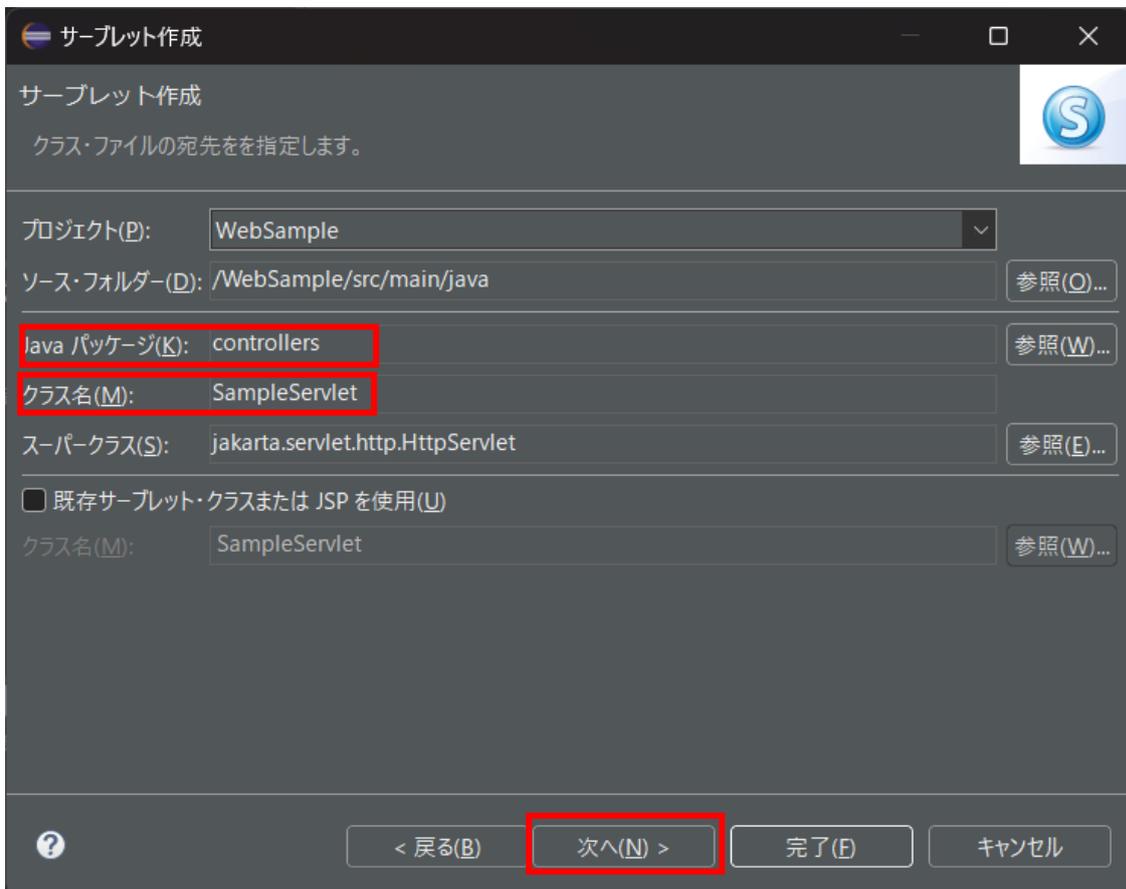
- ① サーブレットクラスを作成する動的 Web プロジェクトを選択し、右クリック⇒「新規」⇒「その他」を選択



- ② 「ウィザードを選択」画面において、「Web」⇒「サーブレット」を選択し「次へ」ボタンを押下する



- ③ 「サーブレット作成」画面において、Java パッケージ名、クラス名を入力し「次へ」ボタンを押下する



- ④ サブレットの名前や URL パターンなどのサブレットクラスに関する情報を設定し「次へ」ボタンを押下する

サブレット作成

サブレット配備記述子固有の情報を入力してください。

名前(M): SampleServlet

説明(S):

初期化パラメーター(P):

名前	値	説明
----	---	----

追加(A)...
編集(E)...
除去(R)

URL マッピング(U):

/SampleServlet

追加(D)...
編集(I)...
除去(V)

非同期サポート(Y)

< 戻る(B) **次へ(N) >** 完了(F) キャンセル

- ⑤ doGet、または doPost の必要なメソッドにチェックをいれ、「完了」ボタンを押下する

サブレット作成

修飾子、実装するインターフェース、および生成するメソッド・スタブを指定してください。

修飾子: public(P) abstract(I) final(L)

インターフェース(I):

追加(A)...
除去(R)

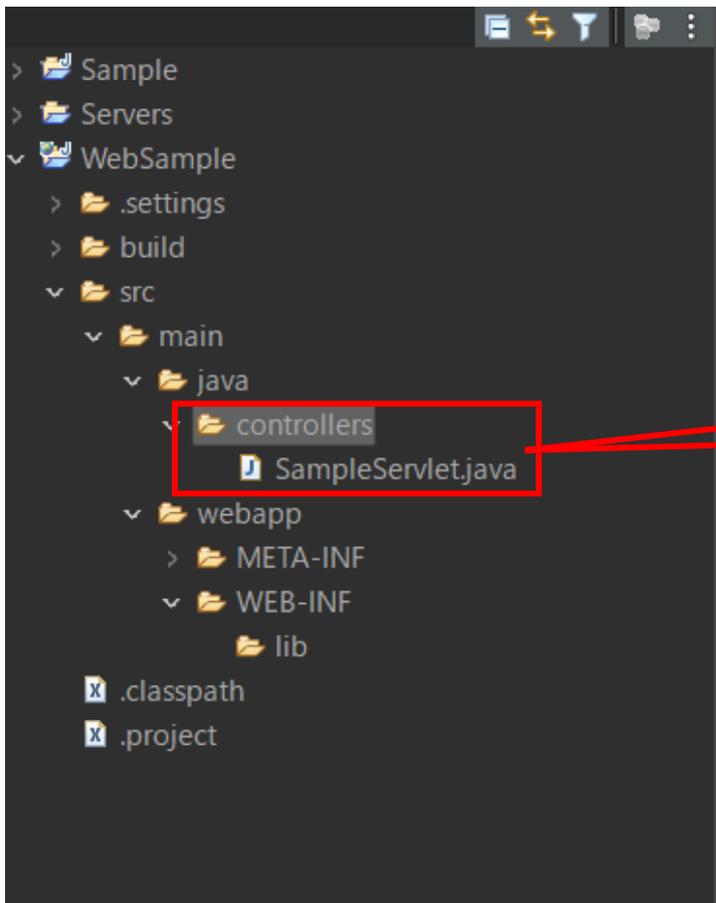
どのメソッド・スタブを作成しますか?

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)

<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet(G)
<input checked="" type="checkbox"/> doPost(P)	<input type="checkbox"/> doPut(U)	<input type="checkbox"/> doDelete(D)
<input type="checkbox"/> doHead(E)	<input type="checkbox"/> doOptions(O)	<input type="checkbox"/> doTrace(T)

このチェックははずす

< 戻る(B) 次へ(N) > **完了(F)** キャンセル



サーブレットクラスが
作成される

⑥ ソースコードを編集する

「src/main/java/controllers」内にサーブレットクラスが作成され、エディタには作成したサーブレットクラスの内容が表示される。⑤でチェックをつけたメソッドが自動で作成されている。

ここでは、doGet()関数を以下のように修正する

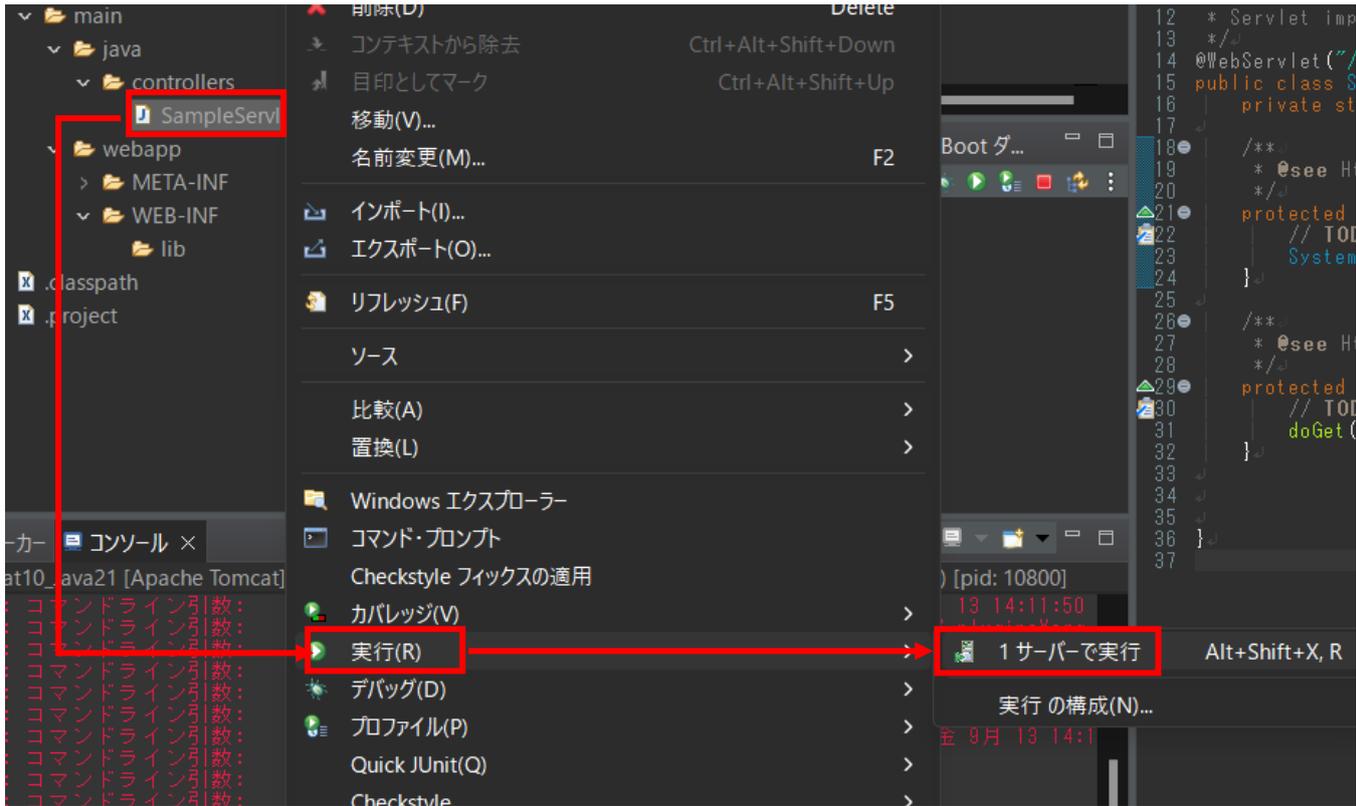
```
1  /**
2   * Servlet implementation class SampleServlet
3   */
4  @WebServlet("/SampleServlet")
5  public class SampleServlet extends HttpServlet {
6      private static final long serialVersionUID = 1L;
7
8      /**
9       * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
10     */
11     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
12         // TODO Auto-generated method stub
13         PrintWriter out = response.getWriter();
14         out.println("<html>");
15         out.println("Hello Java World");
16         out.println("</html>");
17     }
18 }
```

※この行は削除しない

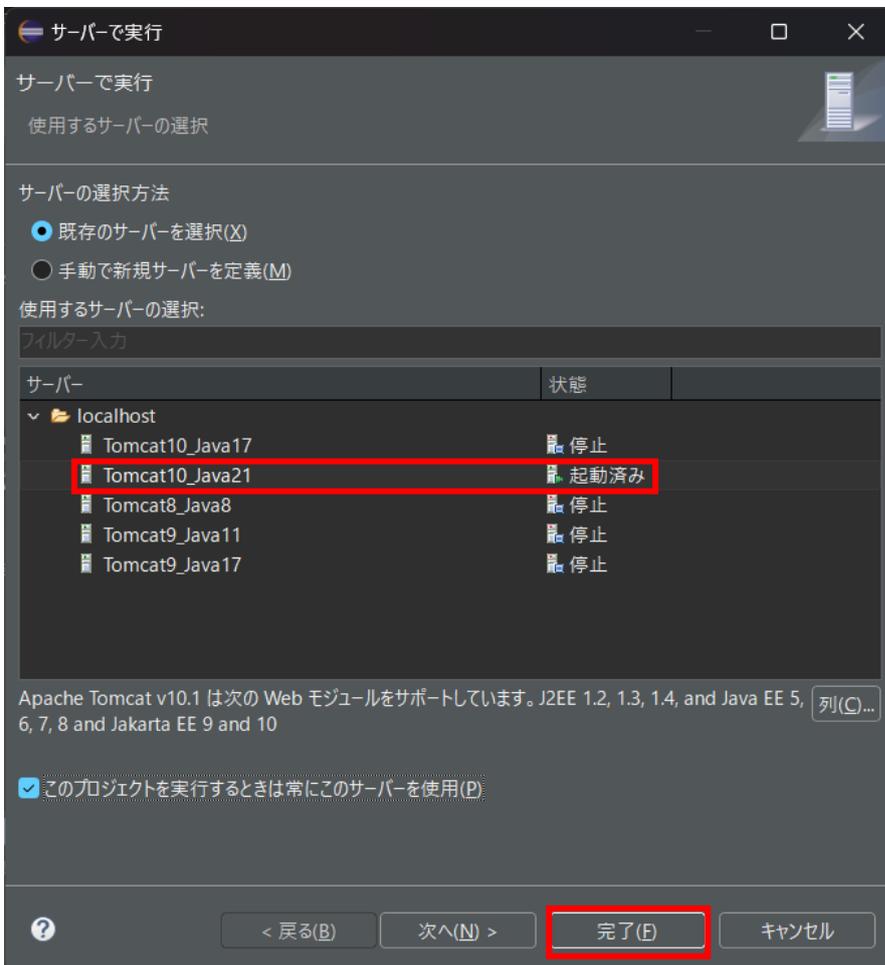
@WebServlet アノテーションといい、URL パターンを設定している

1.3 サブレットクラスの実行

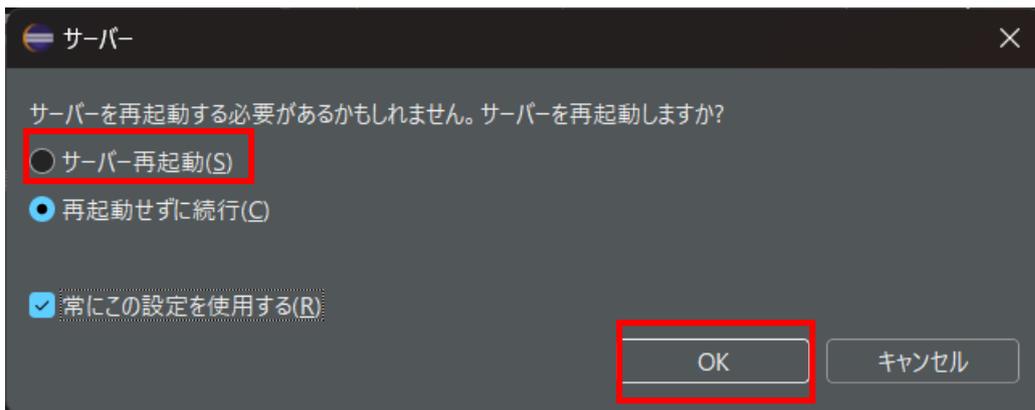
- ① 実行するサブレットクラスを選択し右クリック⇒「実行」⇒「サーバで実行」を選択する



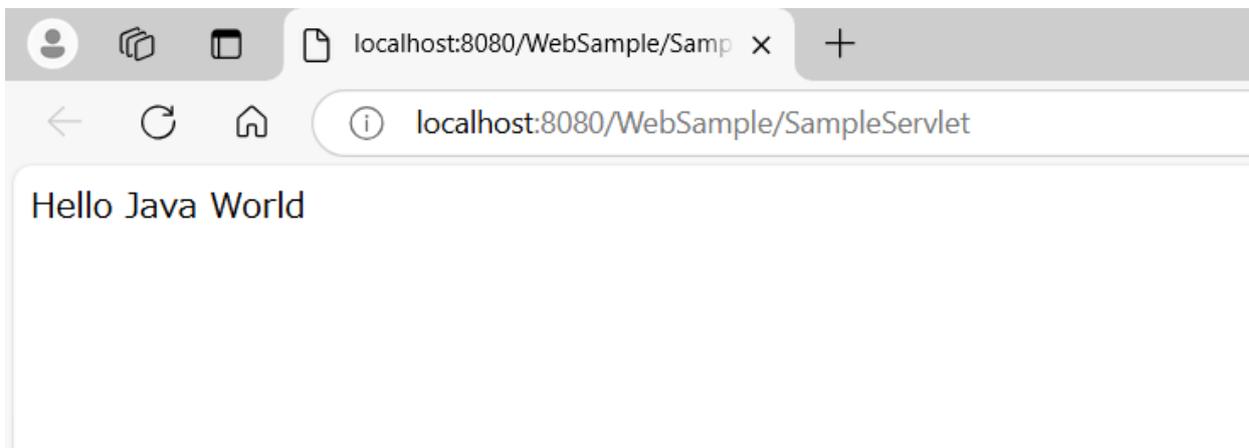
- ② 実行するサーバを選択し完了ボタンを押下する



- ③ サーバを再起動する必要があるときは、以下の画面が表示されるので、「サーバ再起動」を選択し OK ボタンを押下する



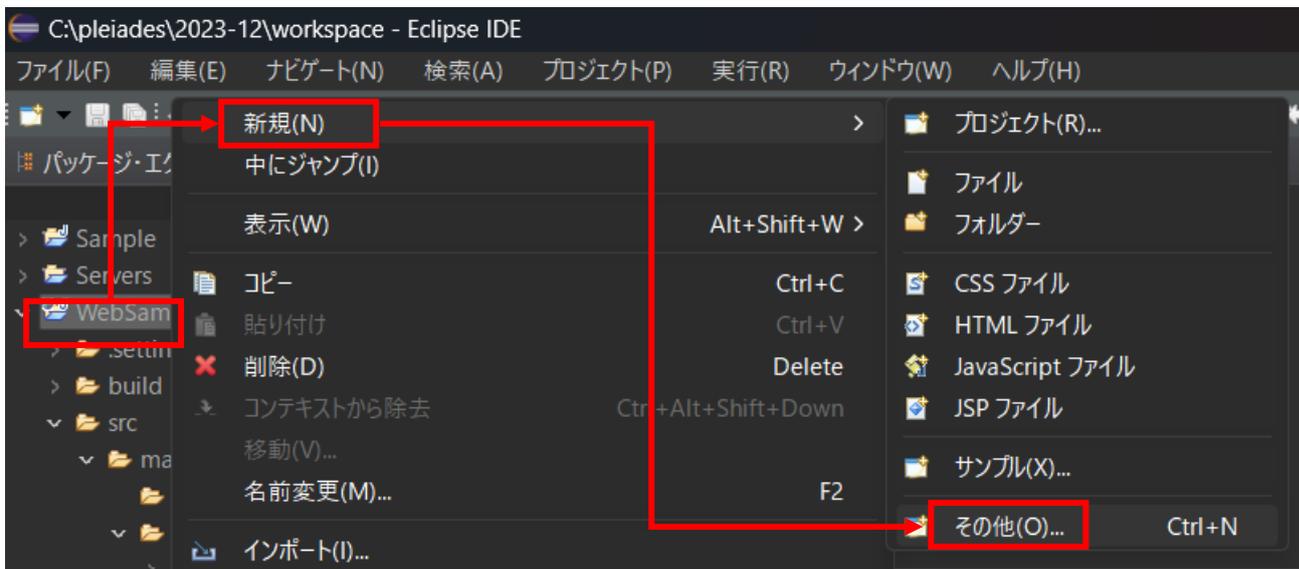
- ④ ブラウザに実行結果が表示される



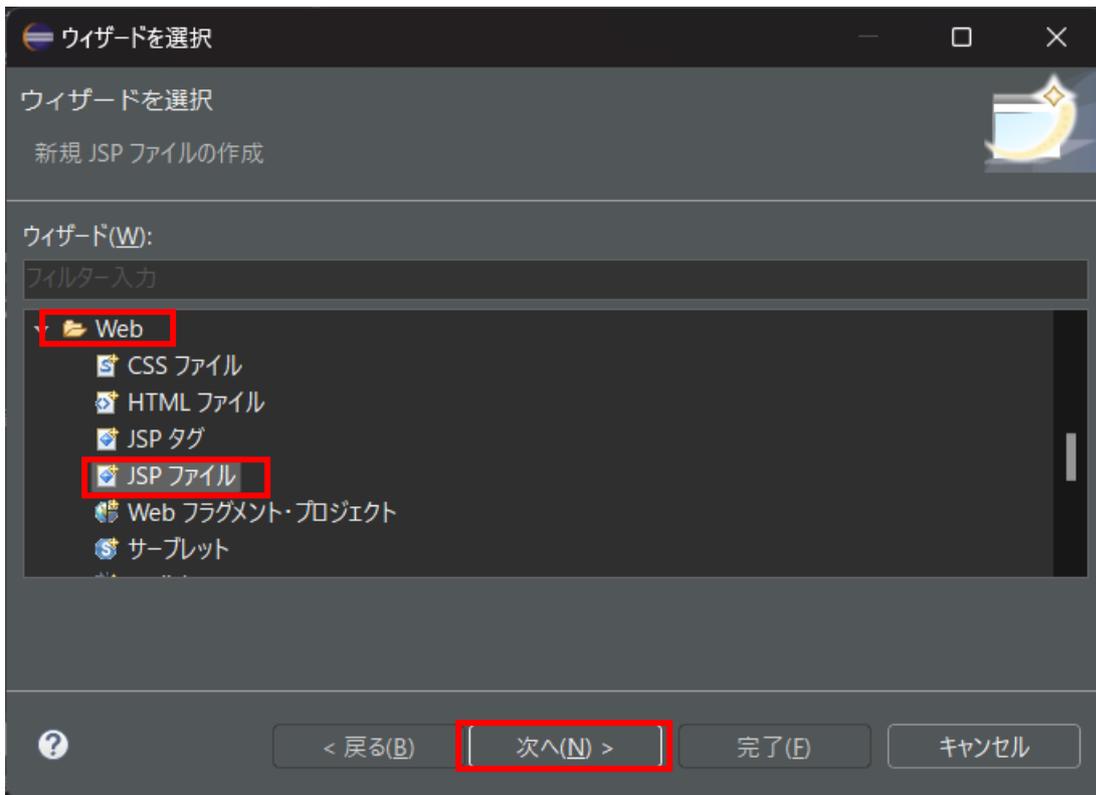
1.4 JSP ファイルの作成

JSP ファイルは、HTML と Java のコードで構成されたファイルのこと

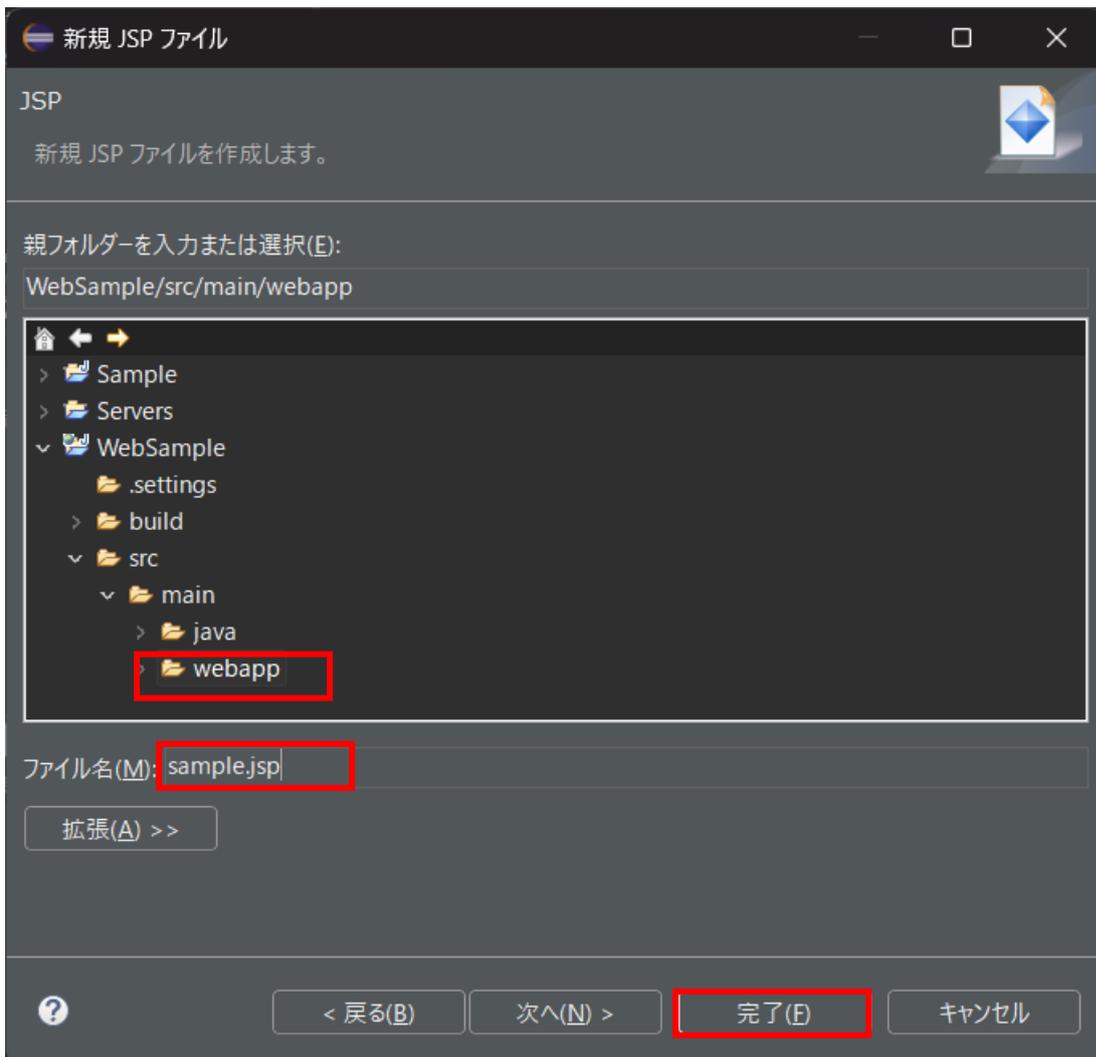
- ① 動的 Web プロジェクトを選択し、右クリック⇒「新規」⇒「その他」を選択



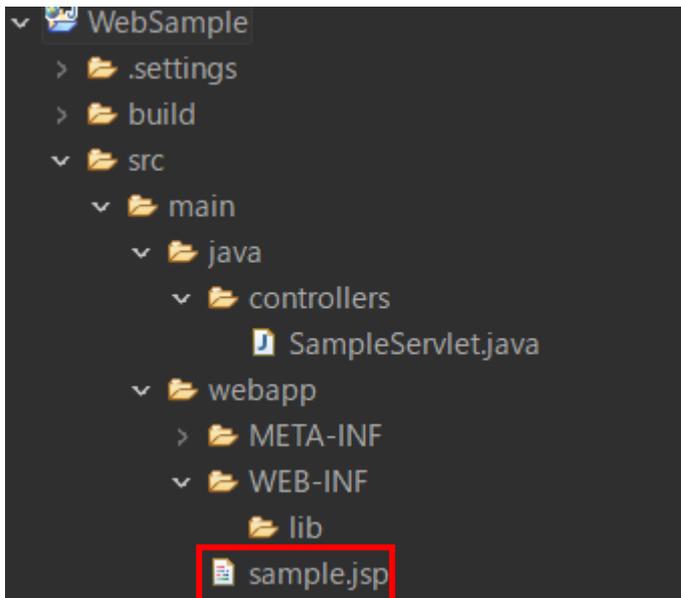
- ② 「Web」⇒「JSP ファイル」を選択し「次へ」ボタンを押下する



- ③ ファイル名を入力する。保存場所は「src/main/webapp/」フォルダとし、「完了」ボタンを押下する



④ ファイルが作成できる



⑤ ソースコードを編集する

■ SampleServlet.java

サーブレットクラスから作成した sample.jsp ファイルを呼び出すように doGet()関数を修正する

```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    // TODO Auto-generated method stub
    RequestDispatcher dispatcher = request.getRequestDispatcher("sample.jsp");
    dispatcher.forward(request, response);
}
```

■ sample.jsp

HTML を修正し、画面に「Hello Java World」と表示する

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
Hello Java World
</body>
</html>
```

※実行は 1.3 サブレットの実行を参照