Java の環境構築

Pleiades All In One Eclipse のインストール

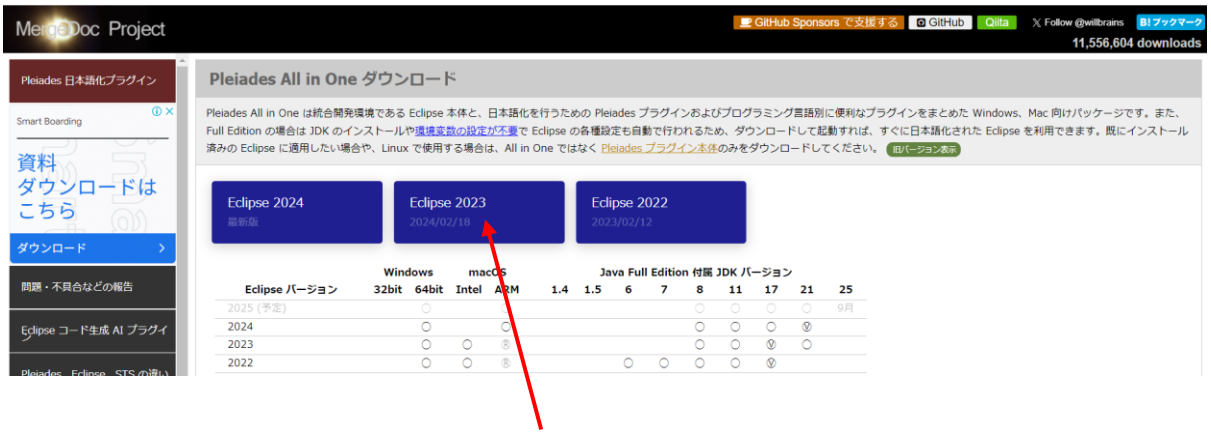
Pleiades All in One は統合開発環境である Eclipse 本体と、日本語化を行うための Pleiades プラグインおよびプログラミング言語別に便利なプラグインを加え、さらに Web アプリケーションサーバ（Apache Tomcat）をセットした Windows、Mac 向けパッケージである。

そのため、このソフトウエ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

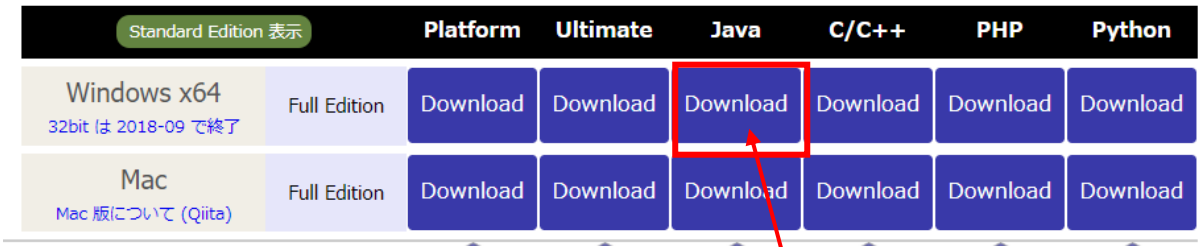
下記 URL へアクセスする

> https://willbrains.jp/



上記のような画面となるので、Eclipse2023 を選択する

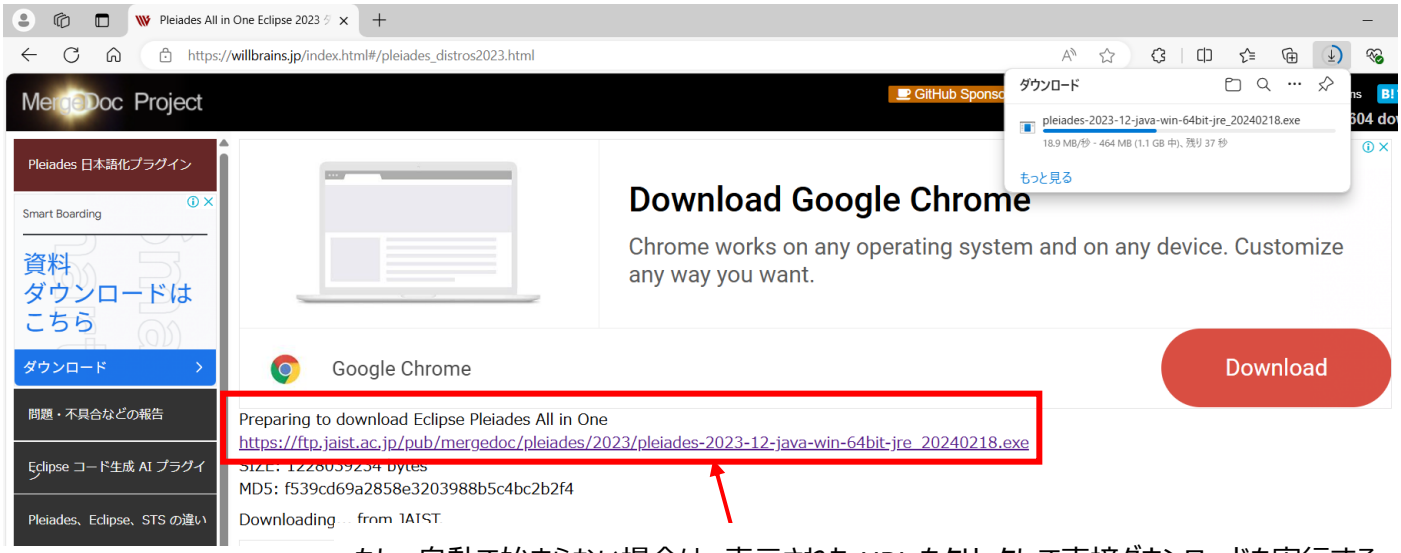
※Eclipse2023 の方が安定しているため



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

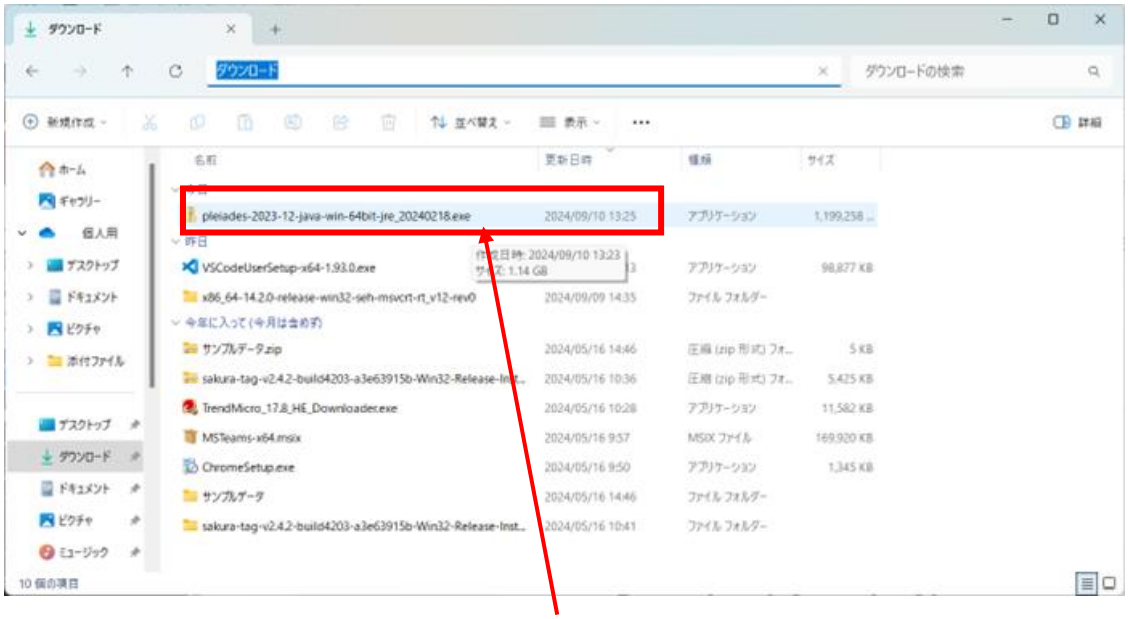
自動でダウンロードが始まる。



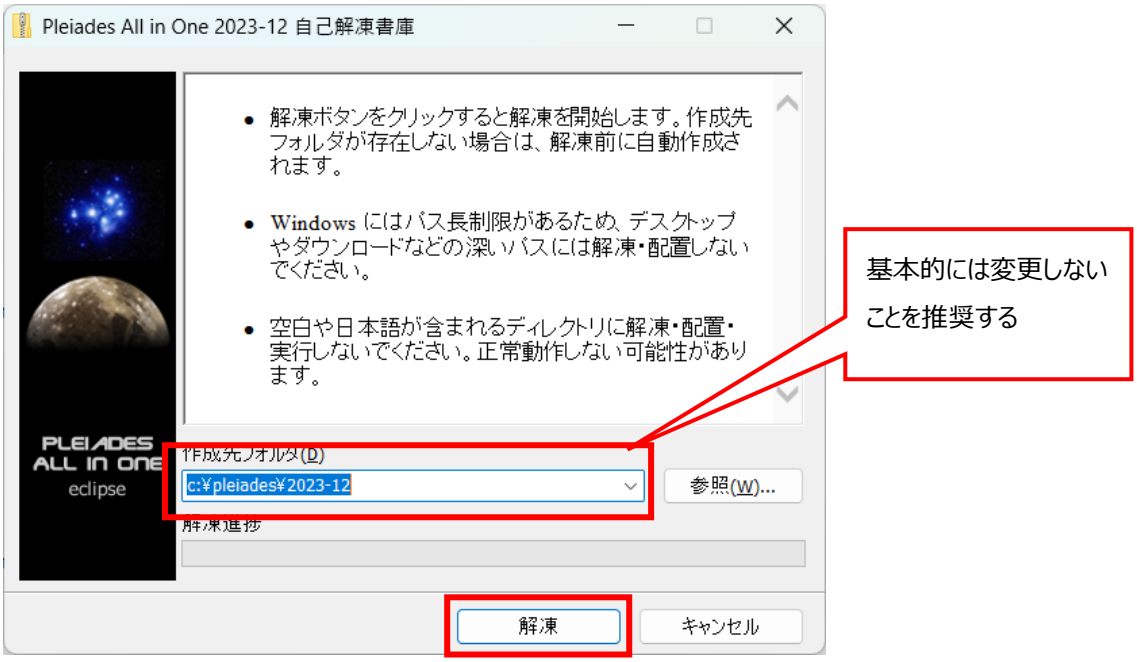
もし、自動で始まらない場合は、表示された URL をクリックして直接ダウンロードを実行する

1.2 Pleiades のインストール

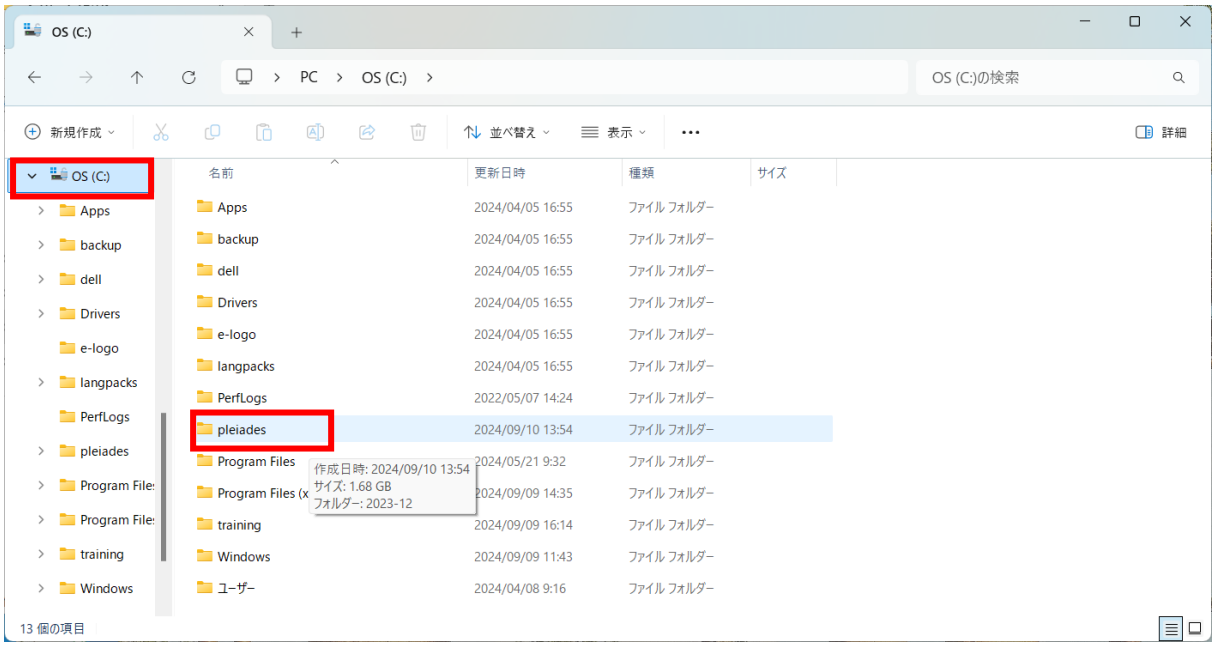
① ダウンロードした exe ファイルを実行する



② インストール先を指定し「解凍」ボタンをクリックする

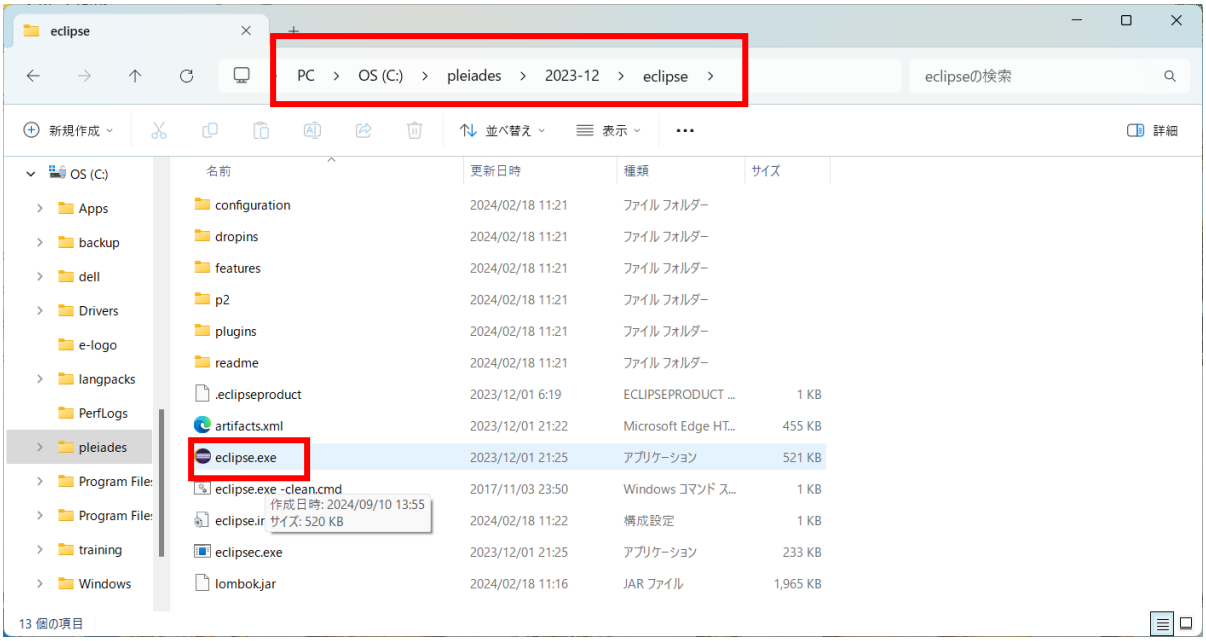


③ フォルダが作成されることを確認する



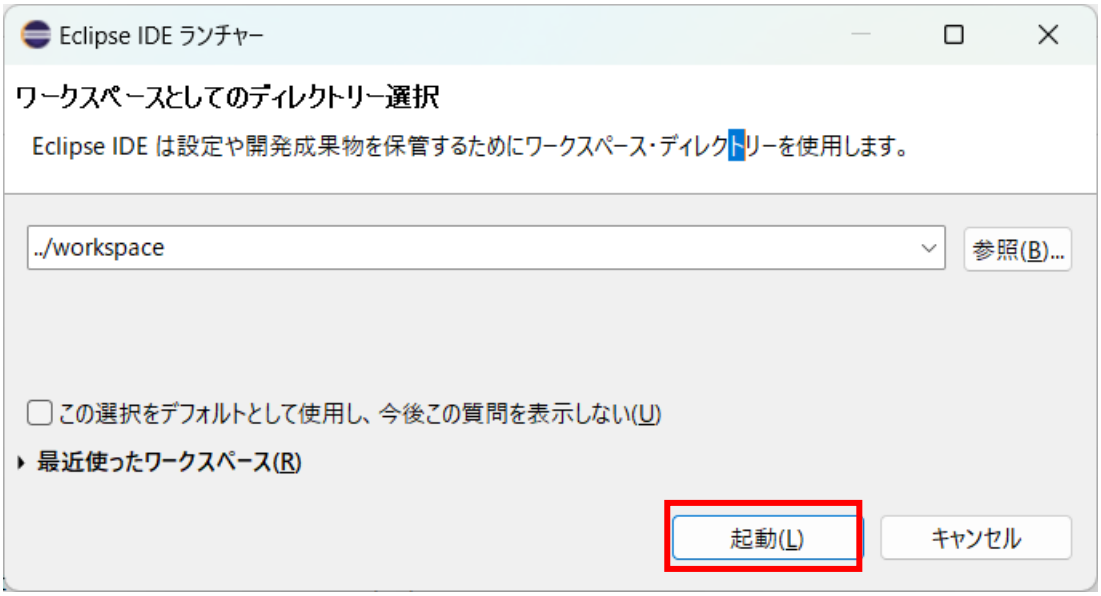
②で指定した場所に pleiades フォルダが作成されていることを確認する

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）

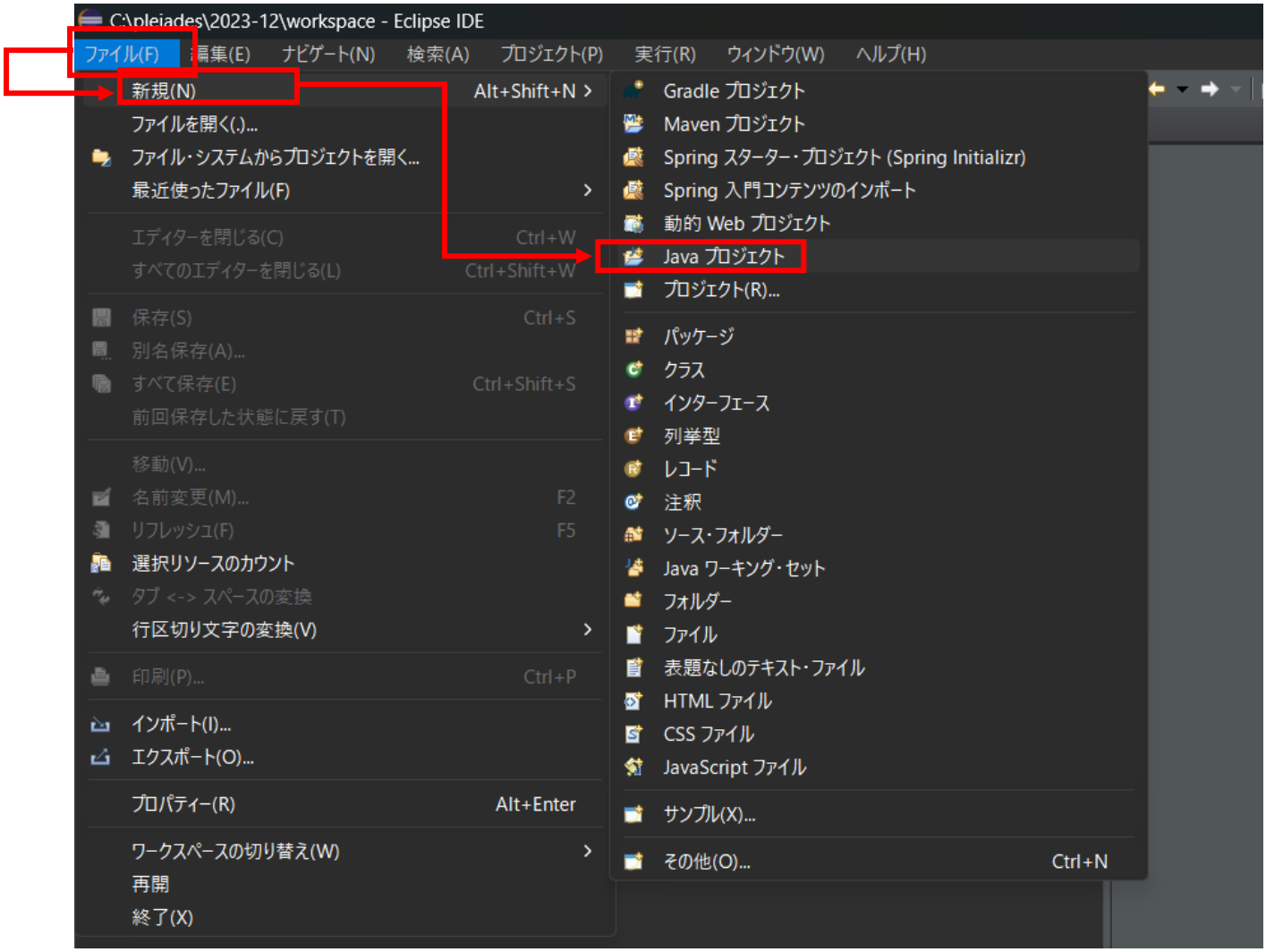
Java アプリケーションの開発方法

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

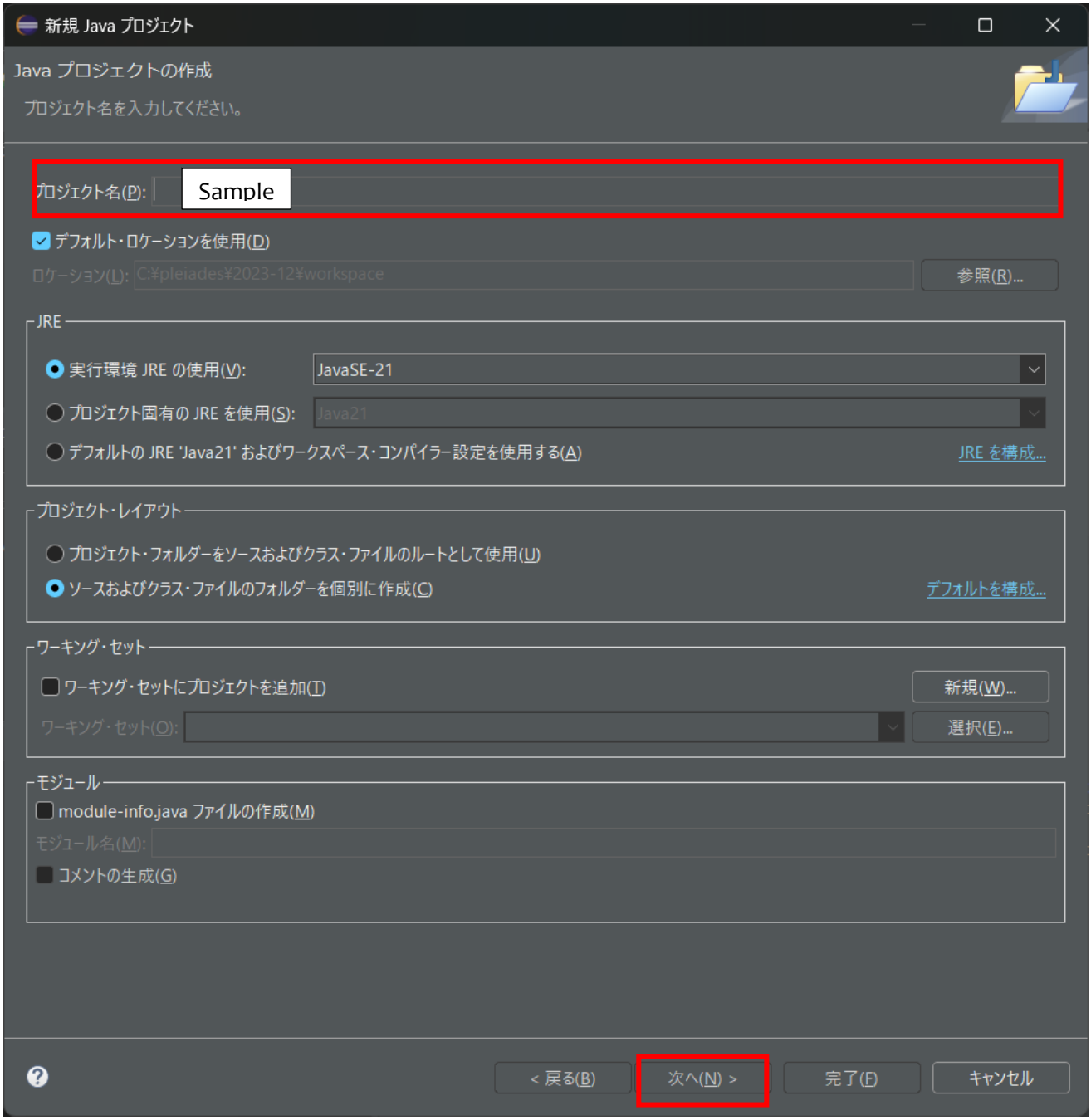
1.1 プロジェクト作成

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

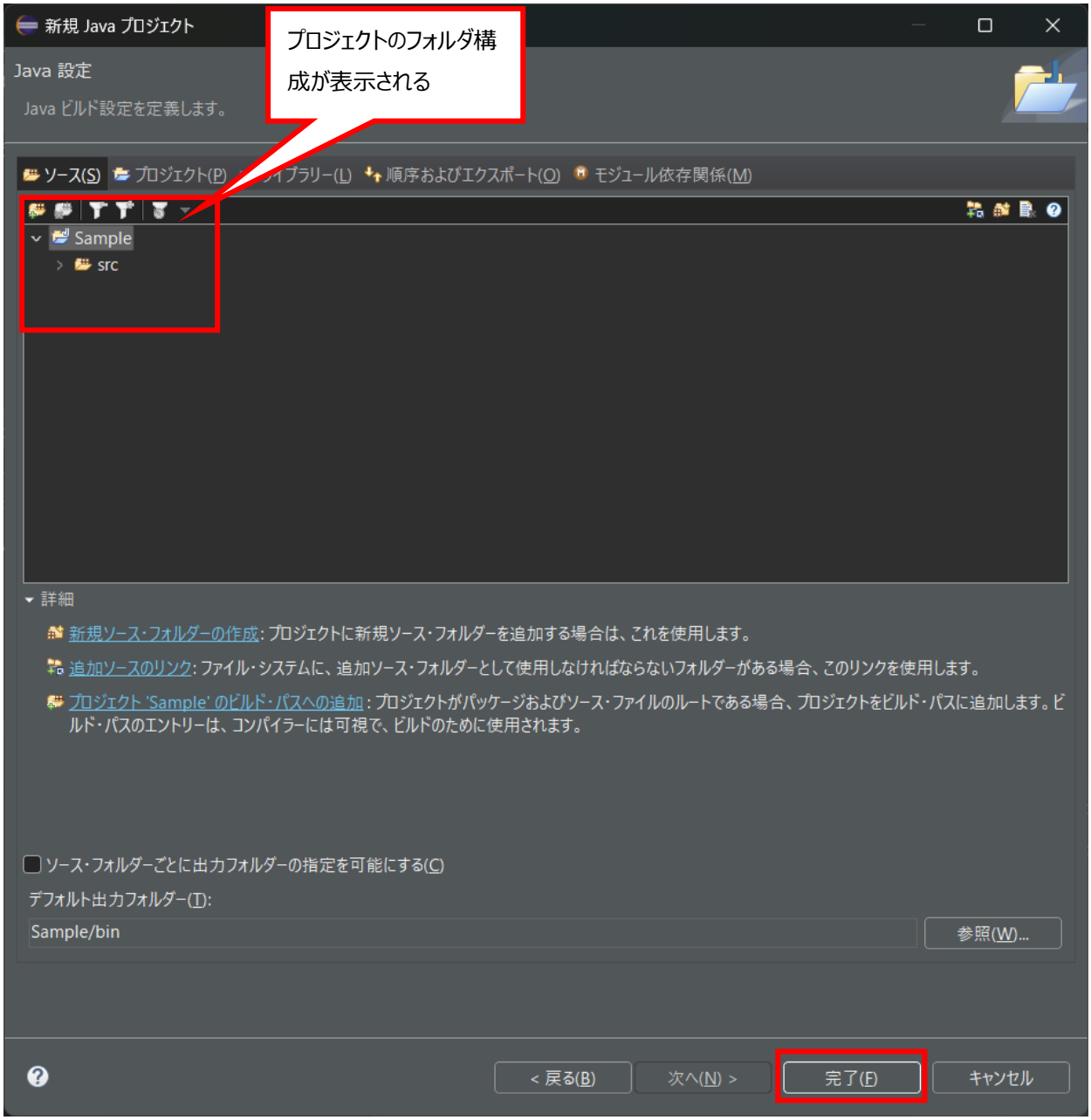
① 「ファイル」⇒「新規」⇒「Java プロジェクト」の順にクリックする



② プロジェクトの名前を入力する

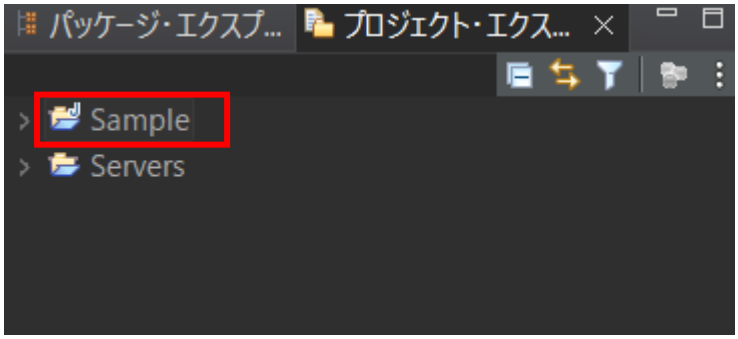


プロジェクト名に「Sample」と入力し「次へ」ボタンを押下する



「完了」ボタンを押下する

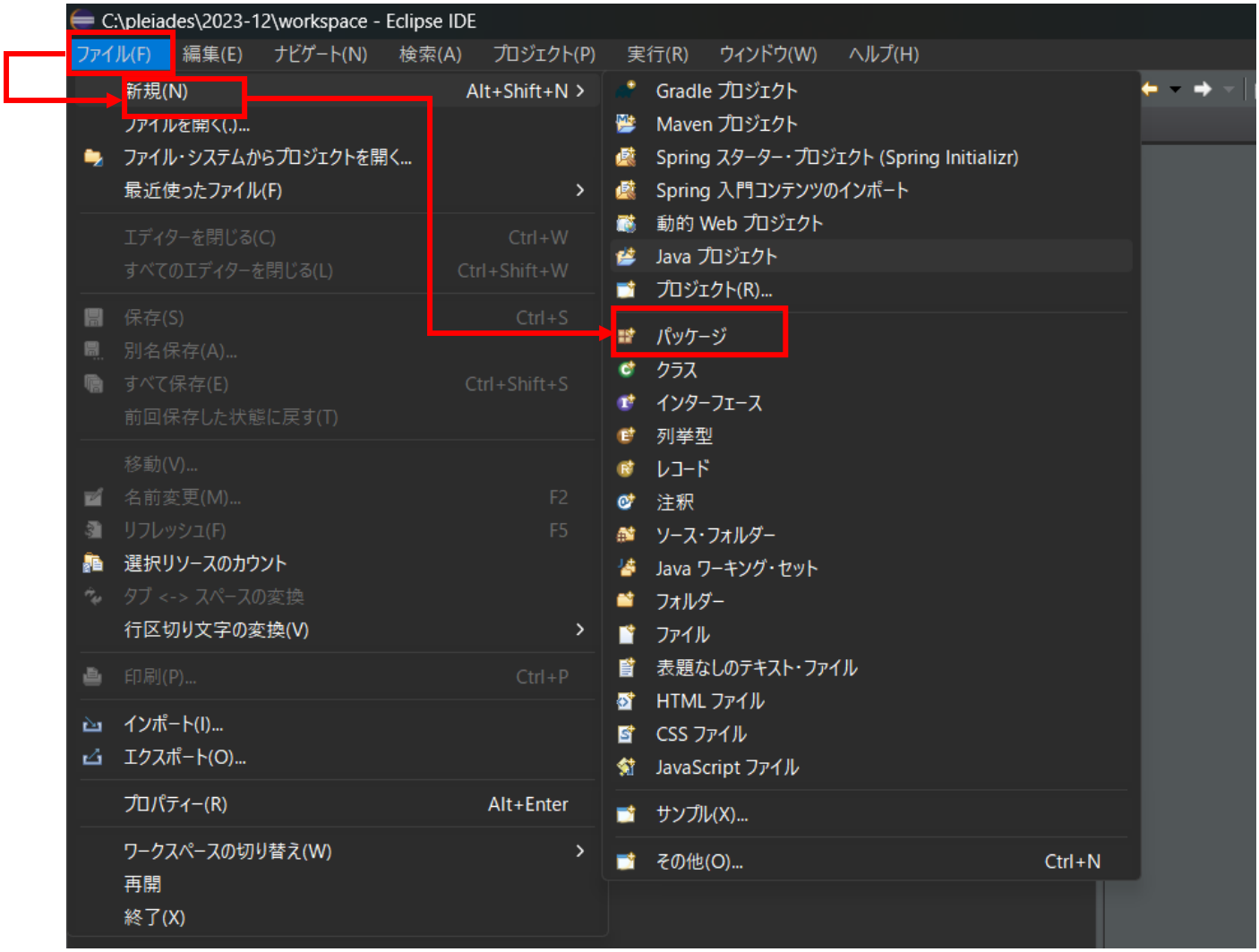
③　プロジェクトが作成され、プロジェクトエクスプローラーに追加される



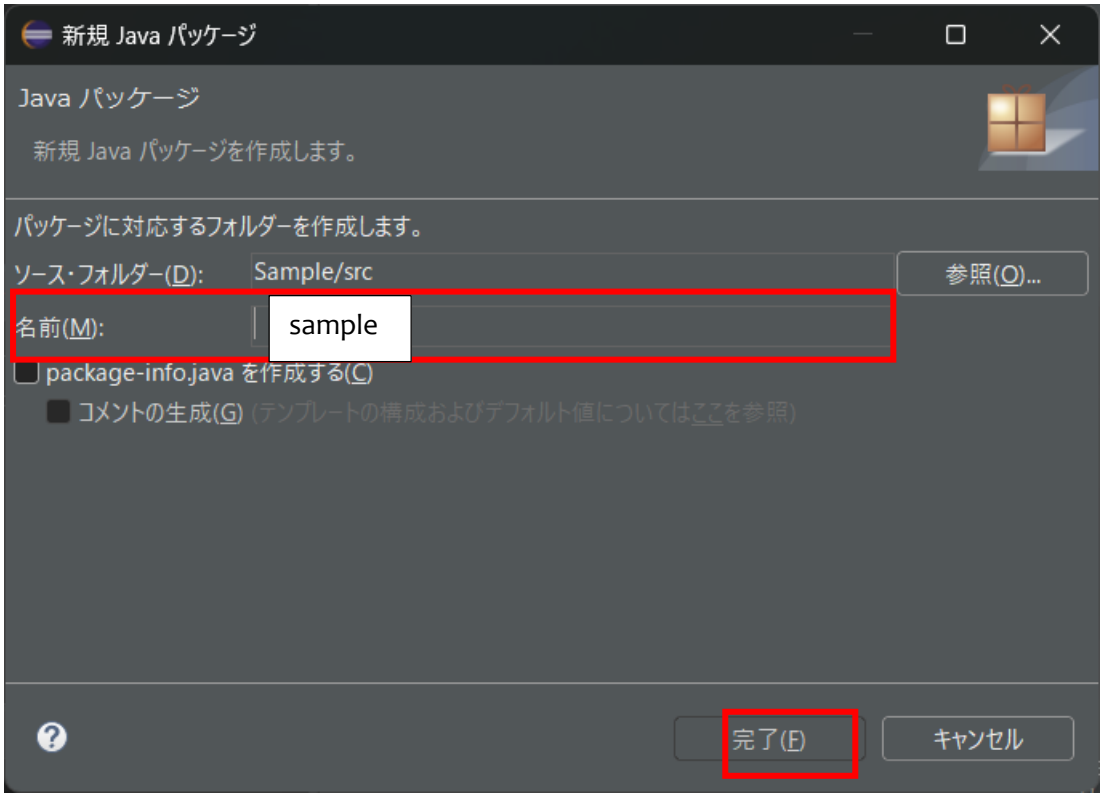
1.2 パッケージ作成

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

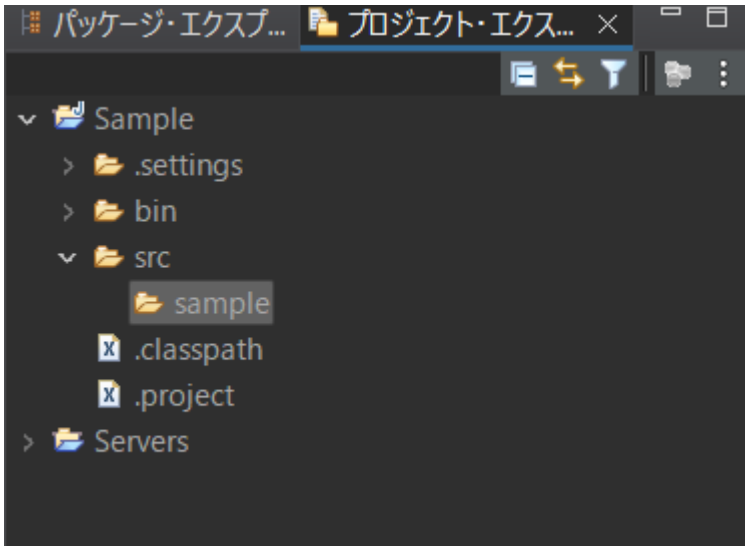


② パッケージ名を入力する



名前に「sample」と入力し「完了」ボタンを押下する

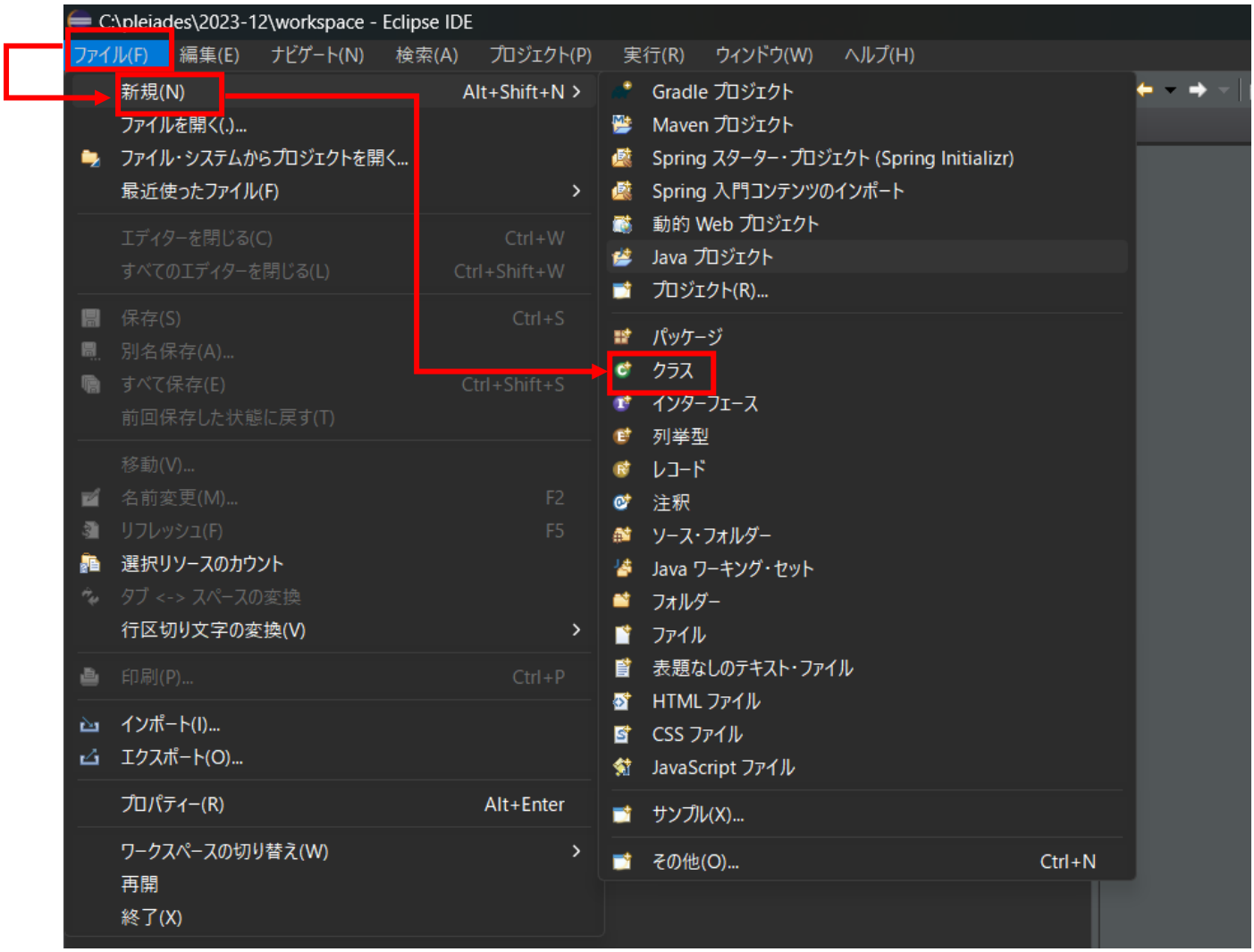
③　プロジェクトの src 直下にパッケージが作成される



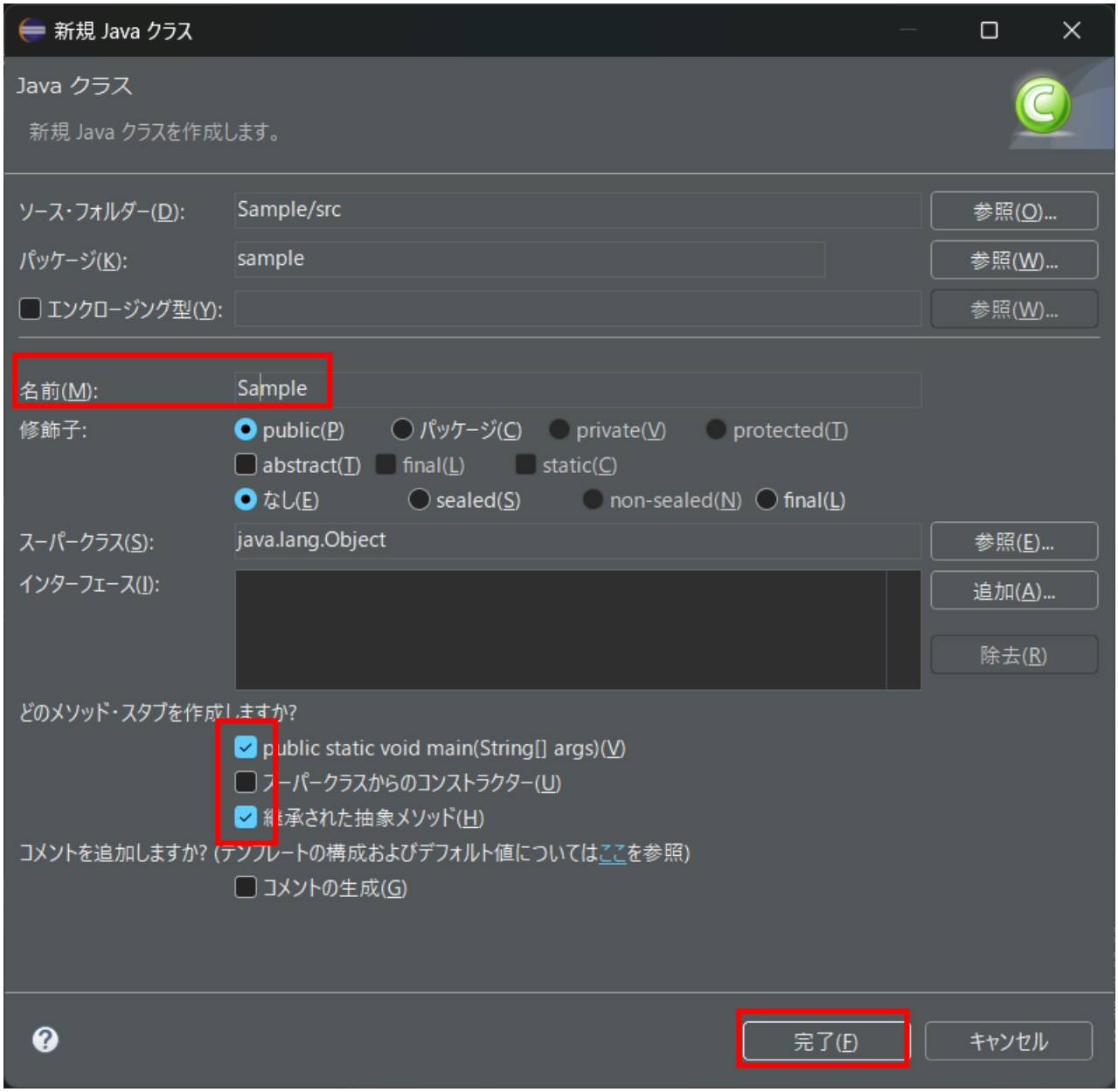
1.3　クラス作成

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



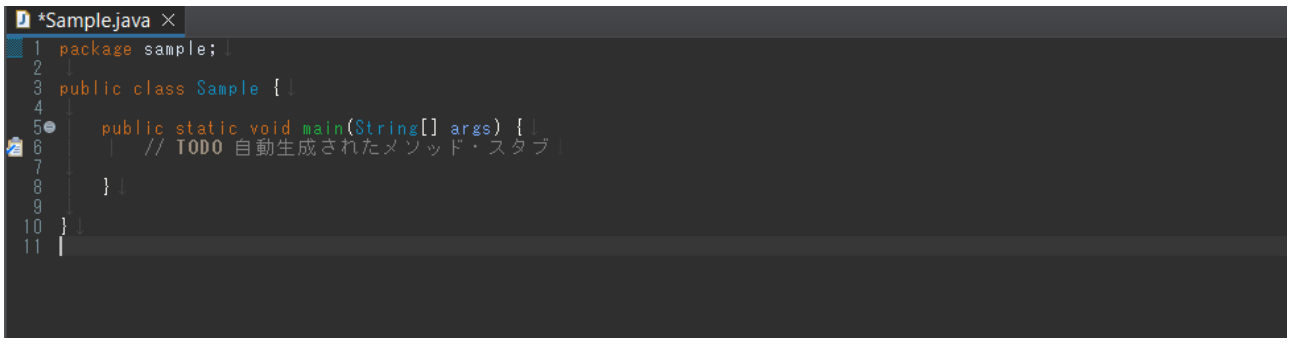
クラス名を Sample と入力する

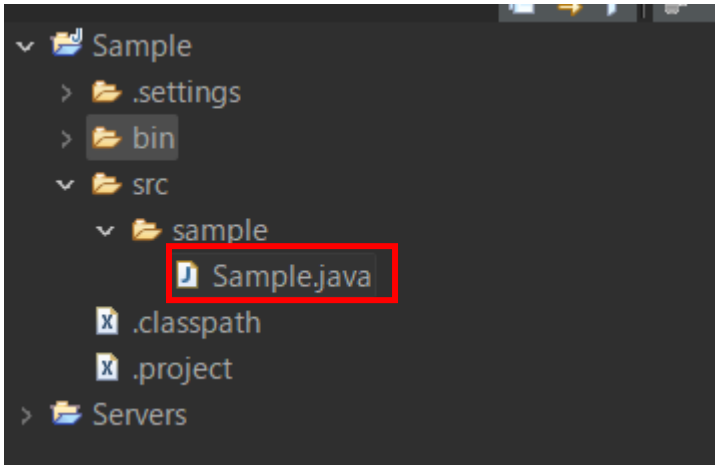
public static void main(Staing[] args)にチェックを入れると Java プログラム作成時に、必要な main メソッドが自動で生成される

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





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");

   }

}
```

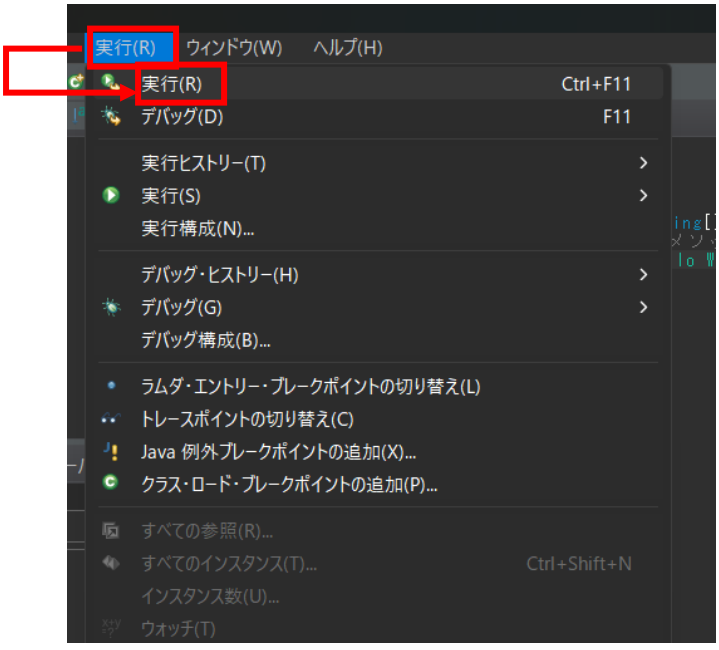
この行を追加する

ソース編集後、ファイルを保存する

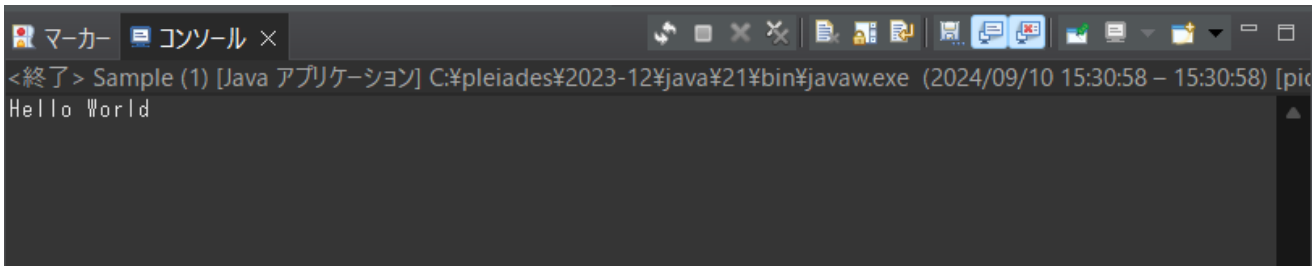
1.5 実行

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



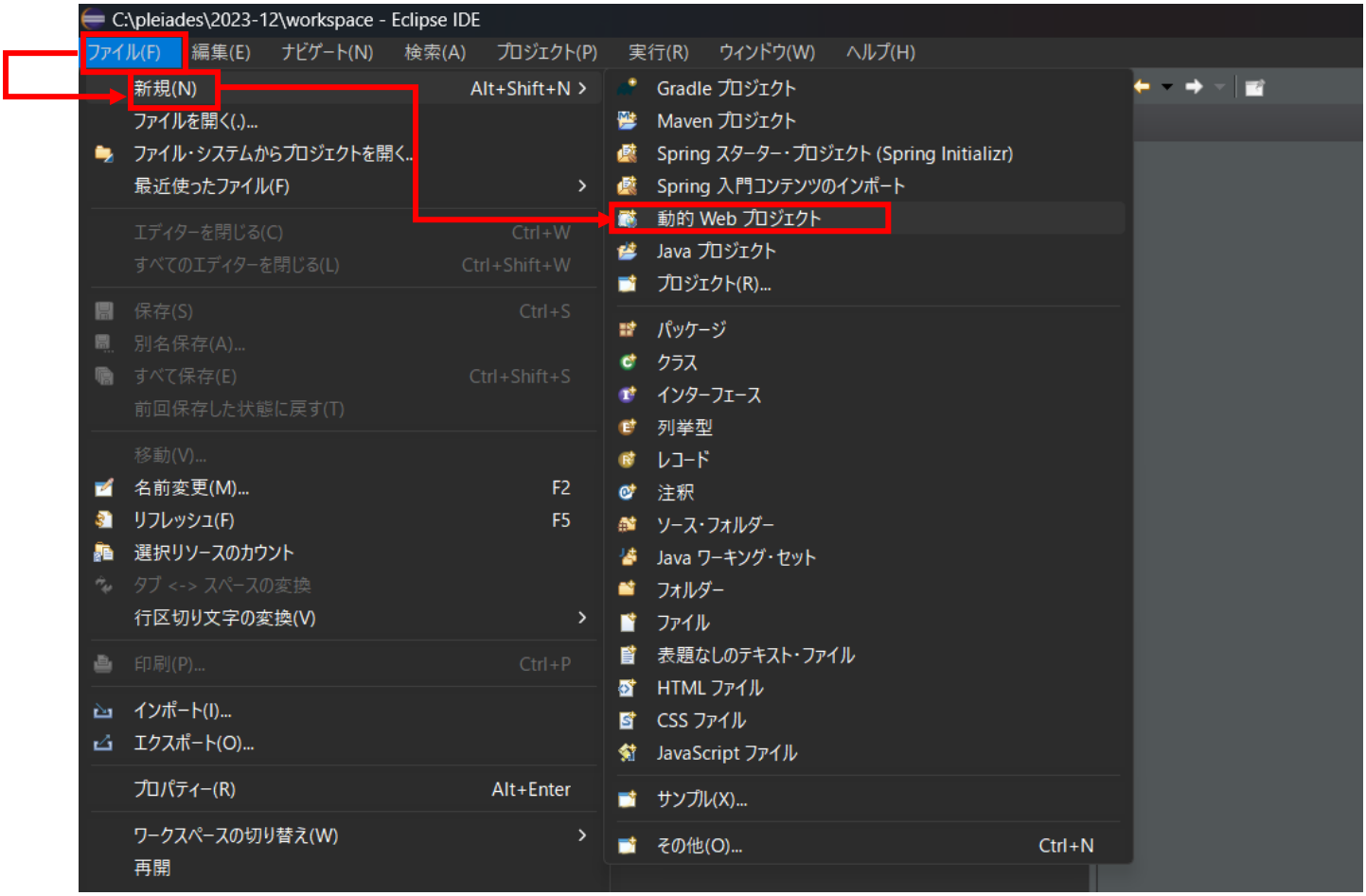
Java サーブレットでの Web アプリケーション開発方法

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

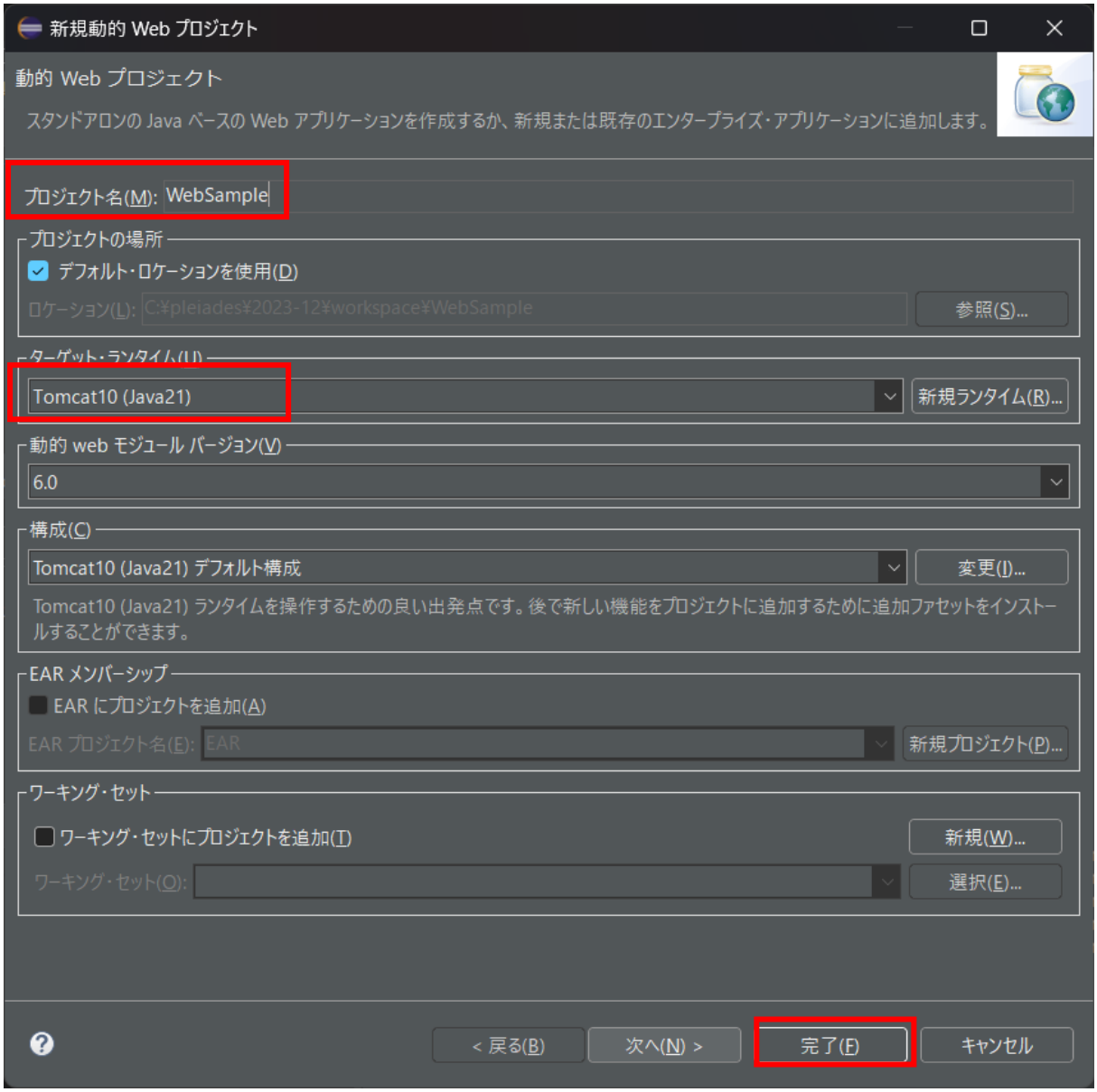
① 「ファイル」⇒「新規」⇒「動的 Web プロジェクト」を選択



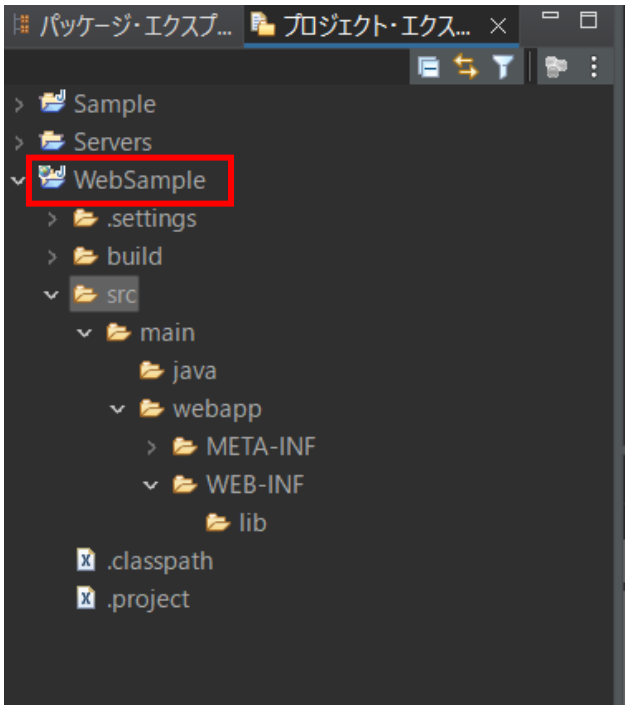
② プロジェクト名、ターゲット・ランタイムを選択

プロジェクト名：WebSample

ターゲット・タイムライン：Tomcat10(Java21)を選択

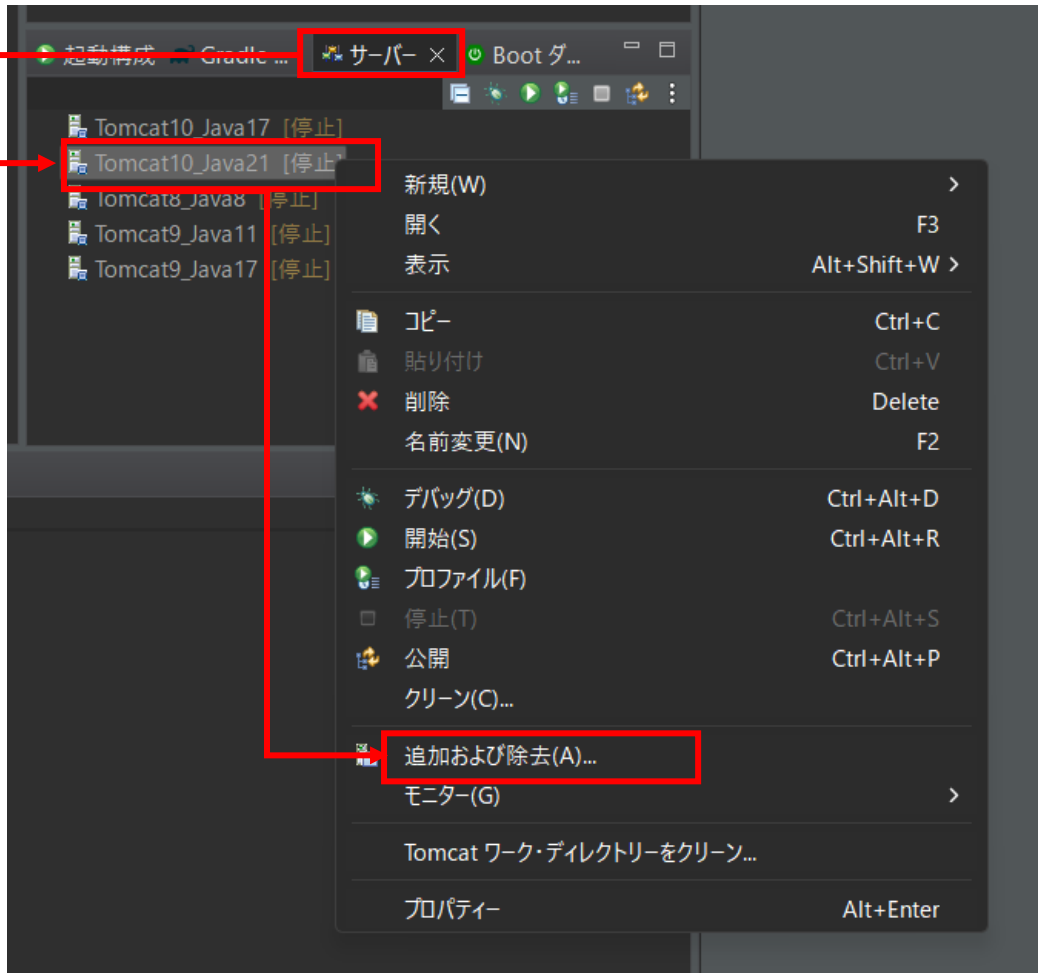


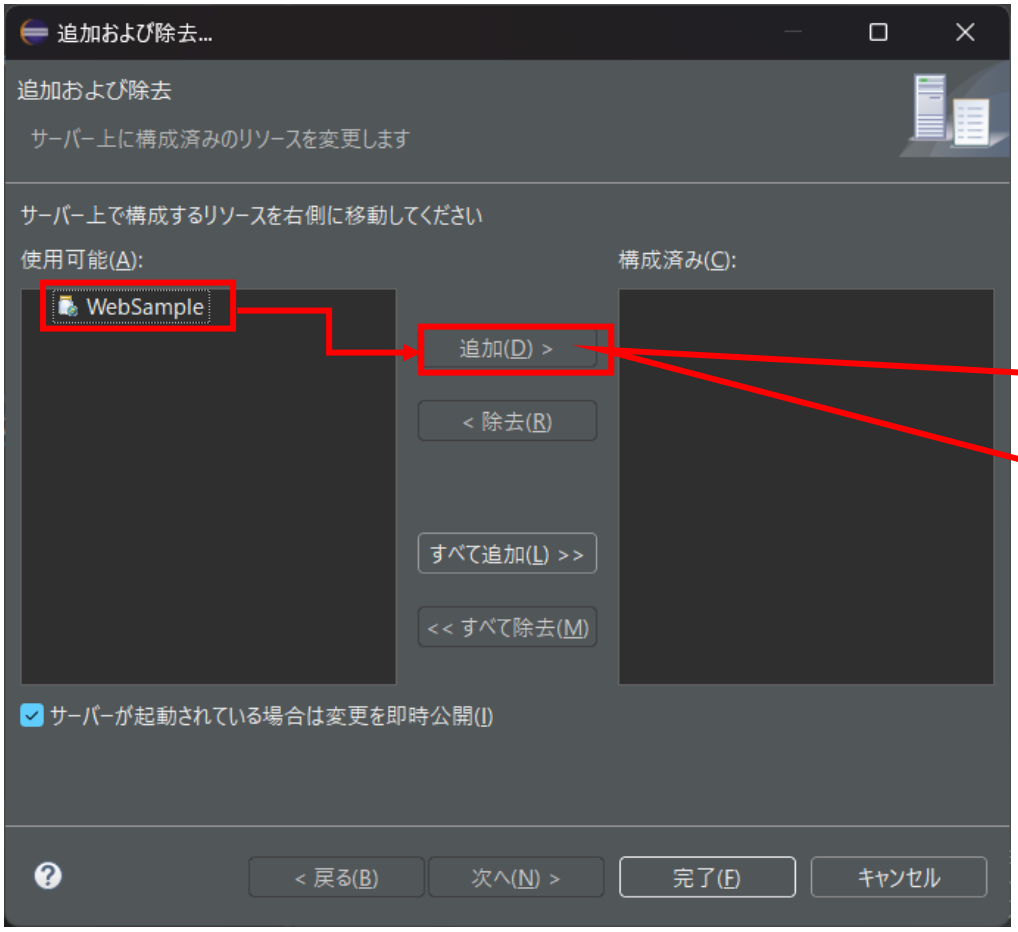
③　完了後、プロジェクトが作成できていることをプロジェクトエクスプローラーで確認する



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

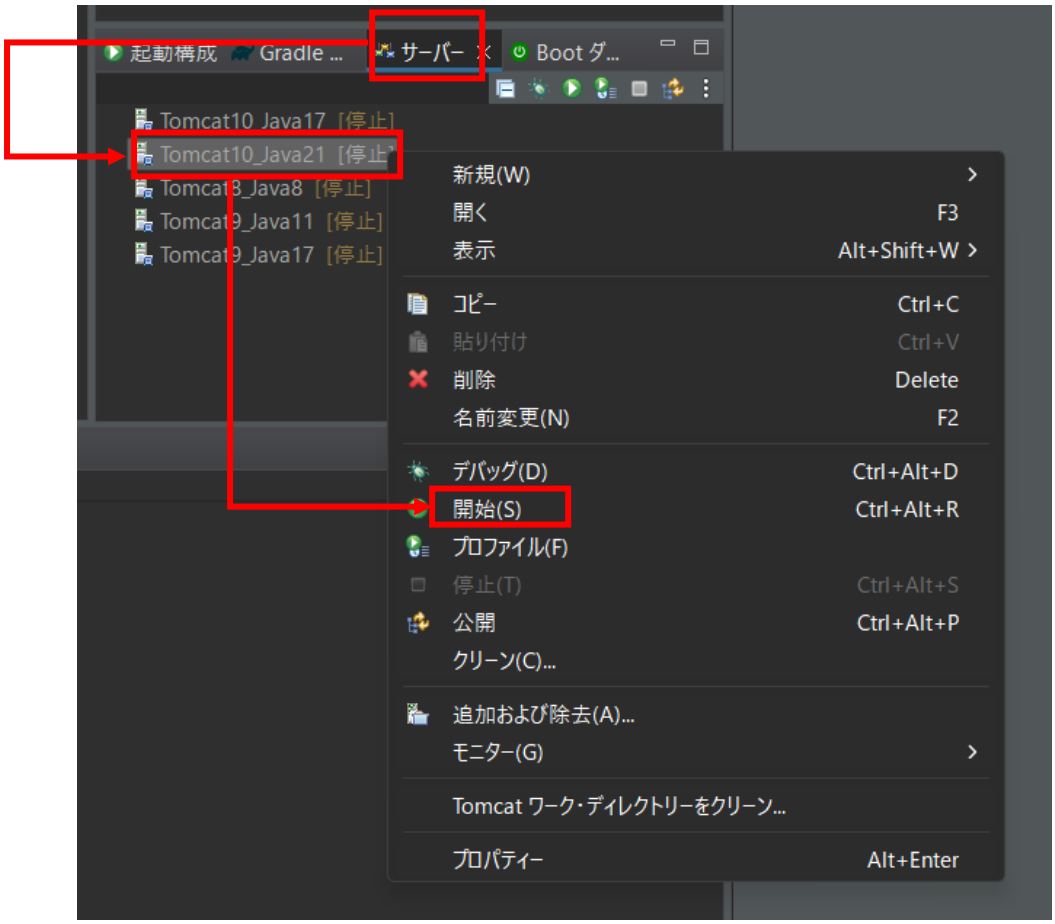


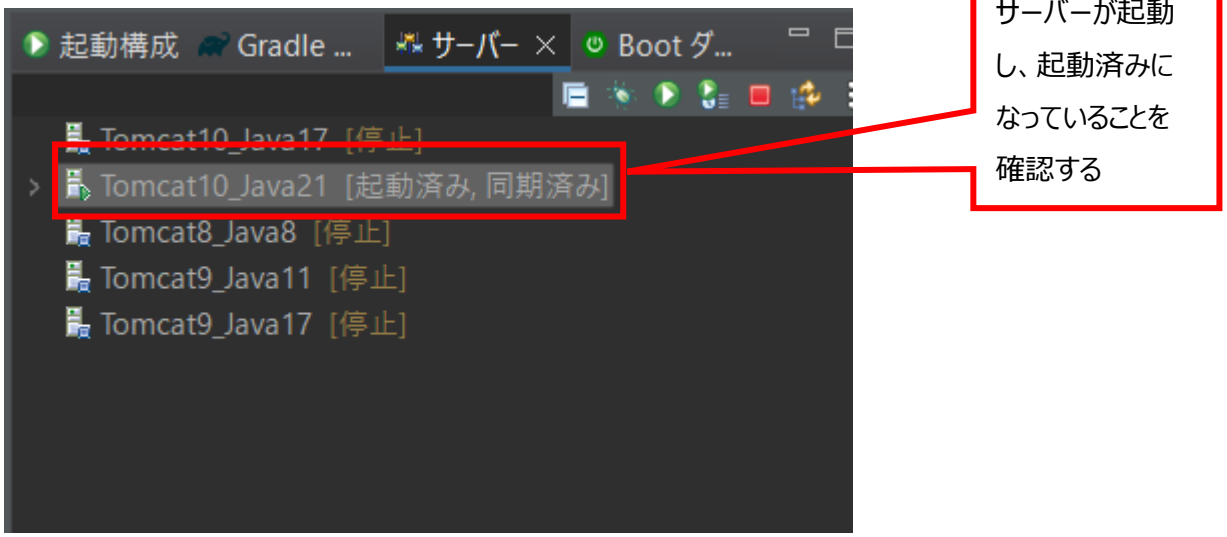


「使用可能」に表示されている動的 Web プロジェクトを選択し、「追加」ボタンを押下し、「構成済み」へ移動する

構成済み」へ移動できたことを確認後、「完了」ボタンを押下する

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

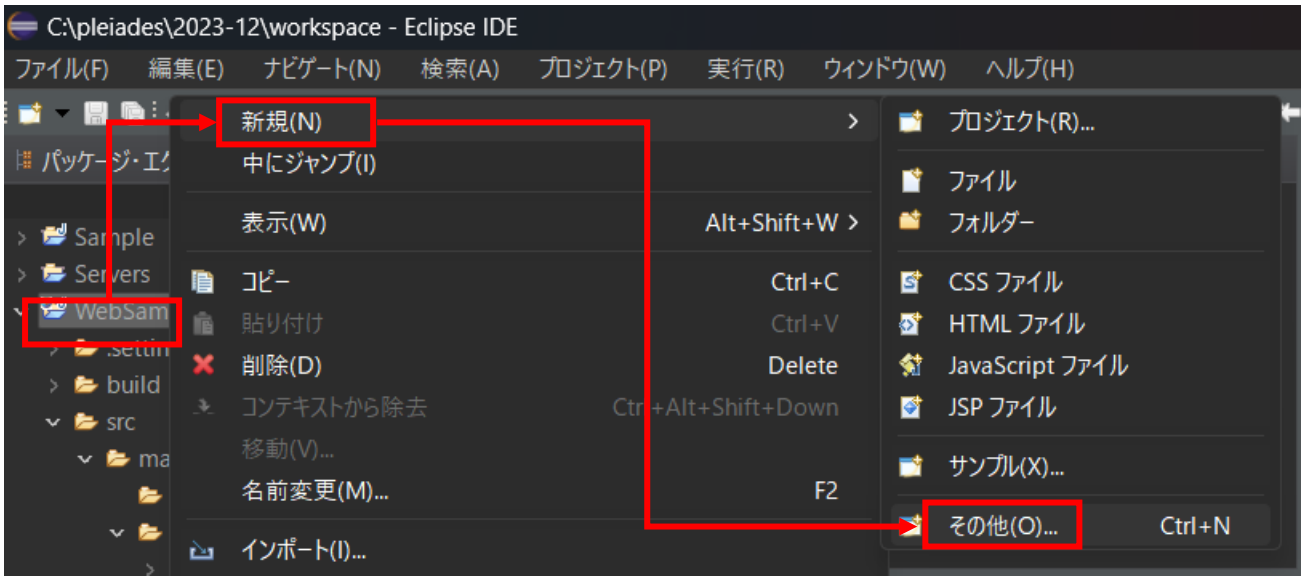




1.2 サーブレットクラスの作成

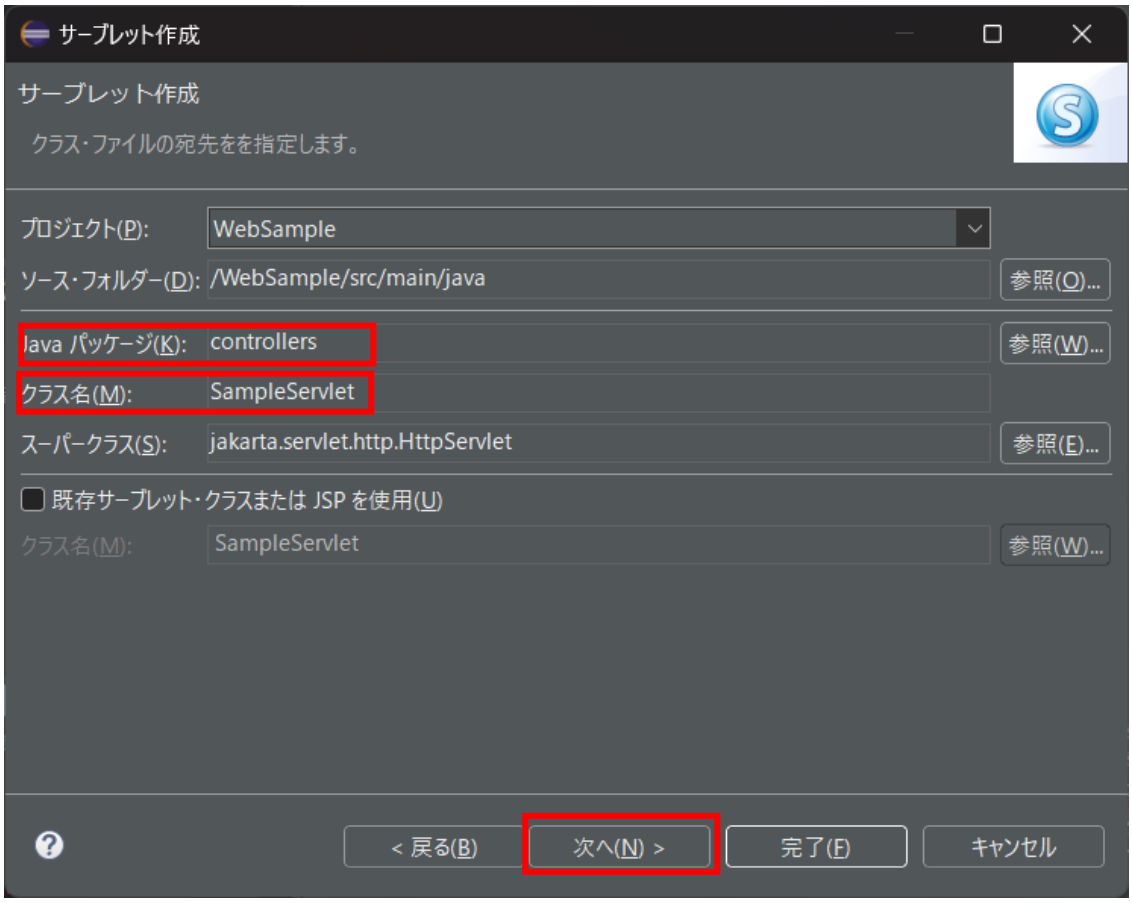
※サーブレットは Java を使いサーバサイドプログラムを作成する技術のこと

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

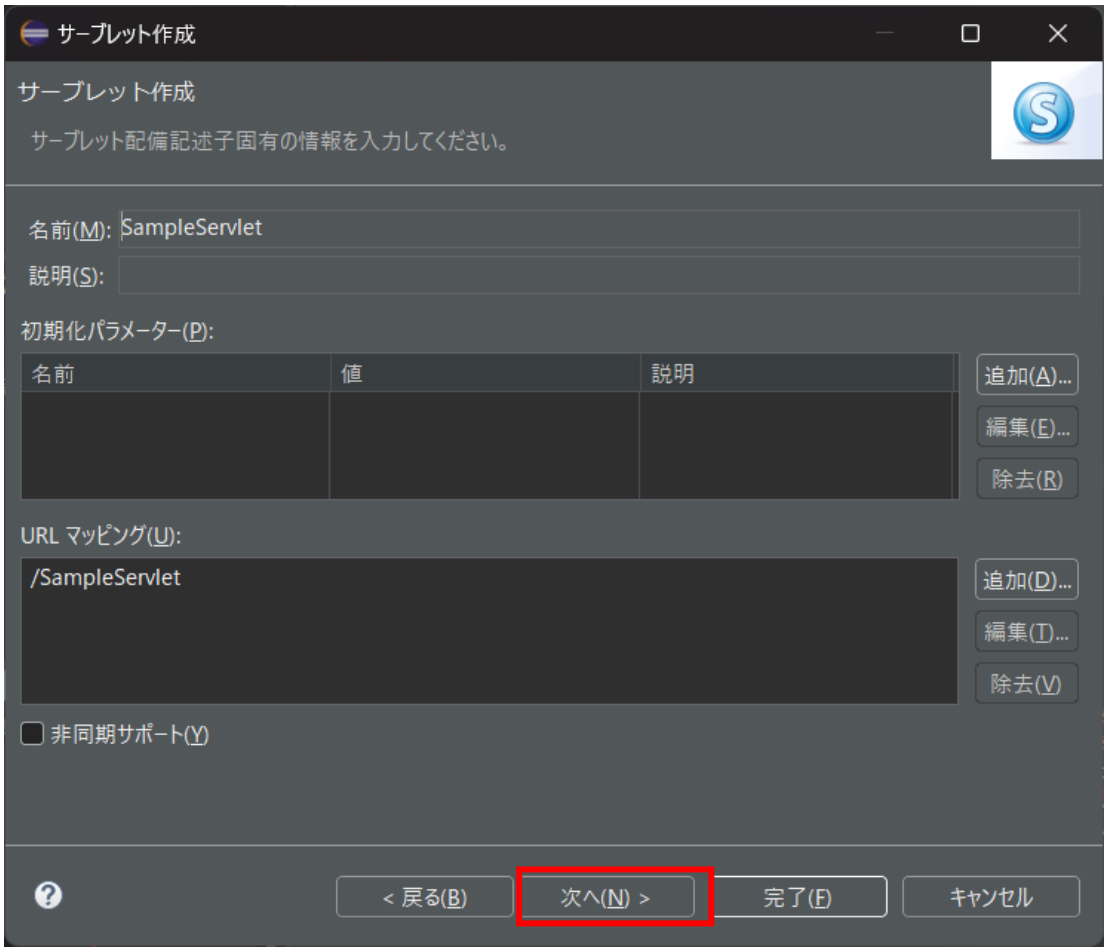


② 「ウィザードを選択」画面において、「Web」⇒「サーブレット」を選択し「次へ」ボタンを押下する

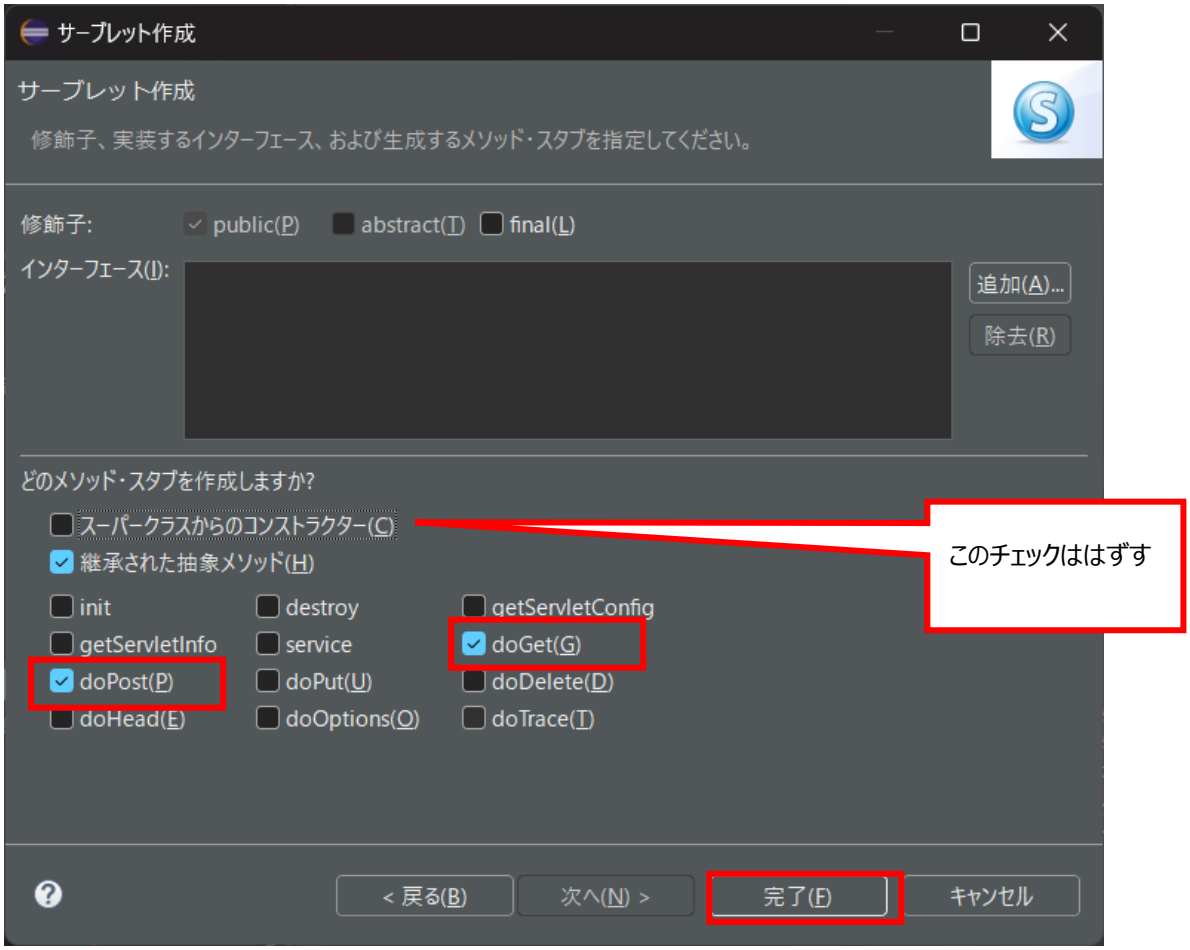
③ 「サーブレット作成」画面において、Java パッケージ名、クラス名を入力し「次へ」ボタンを押下する

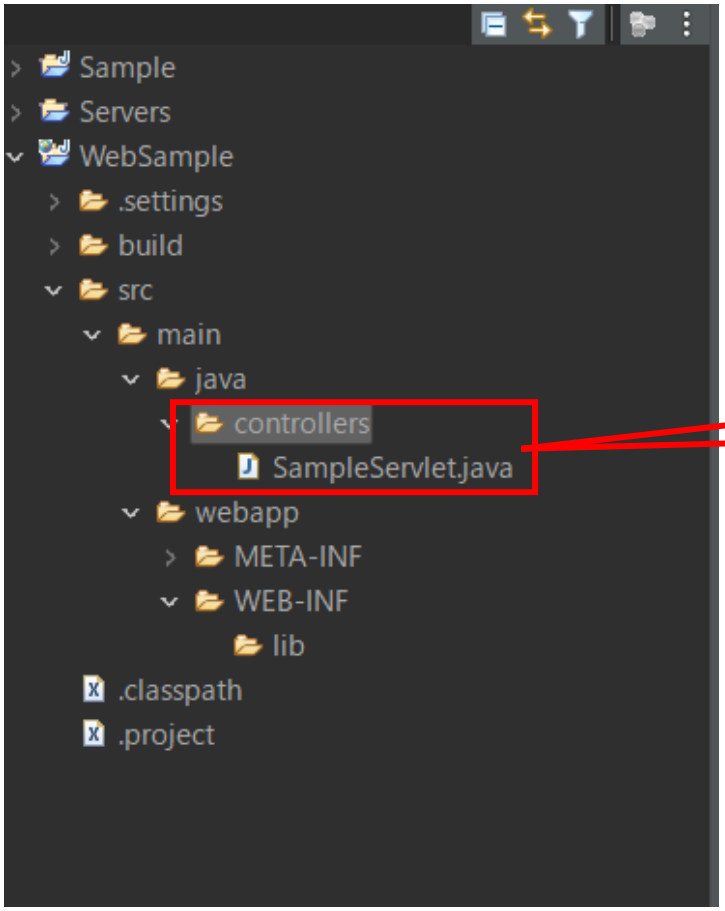


④　サーブレットの名前や URL パターンなどのサーブレットクラスに関する情報を設定し「次へ」ボタンを押下する



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





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

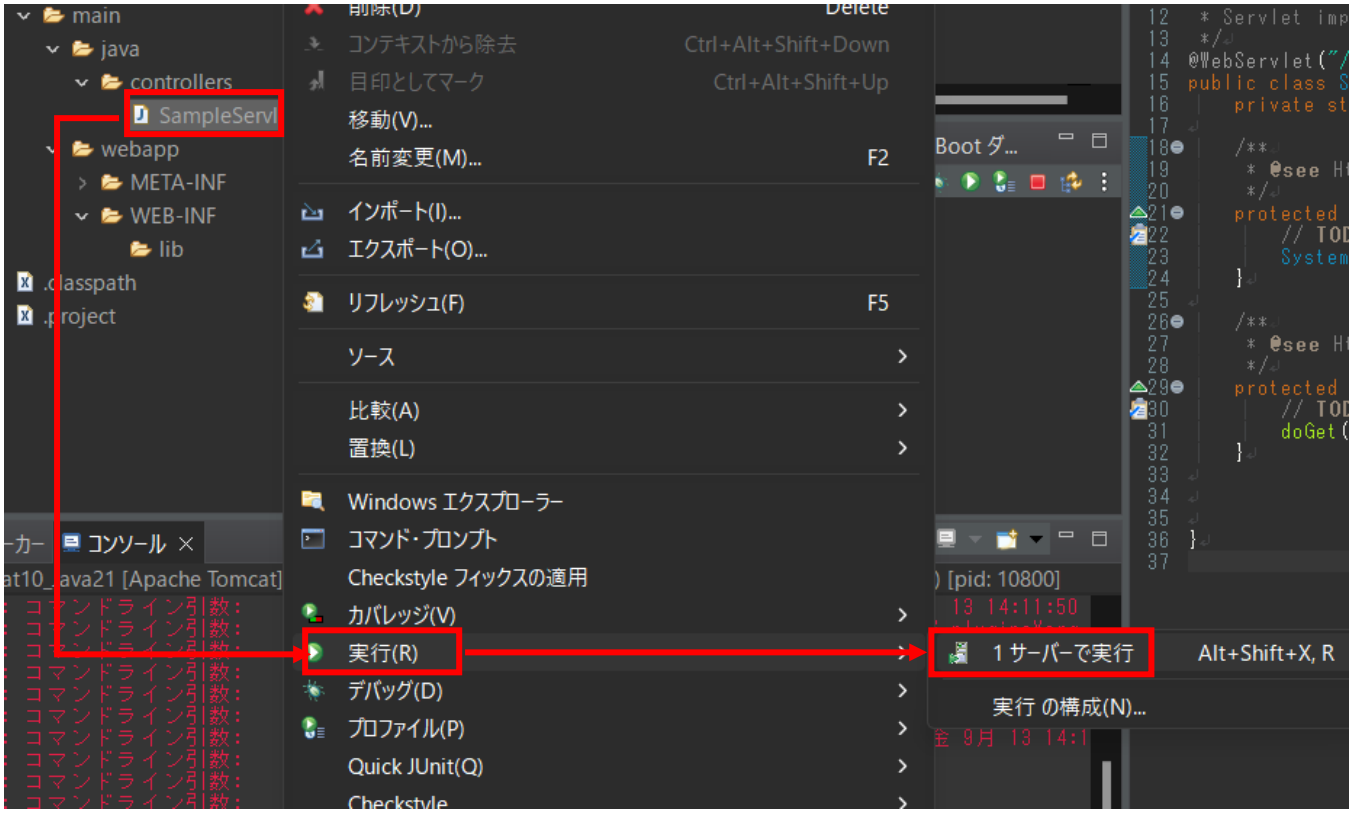
```
/**
 * Servlet implementation class SampleServlet
 */
@WebServlet("/SampleServlet")
public class SampleServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        // TODO Auto-generated method stub
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("Hello Java World");
        out.println("</html>");
    }
```

※この行は削除しない

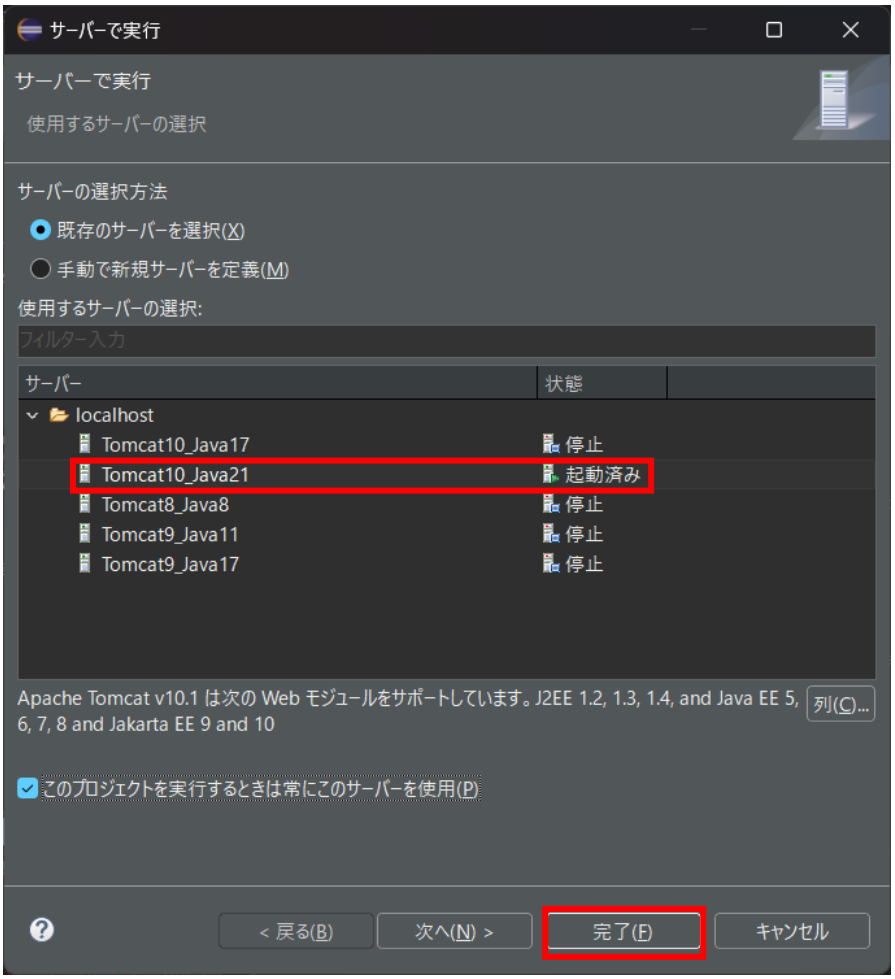
@WebServlet アノテーションといい、URL パターンを設定している

1.3 サーブレットクラスの実行

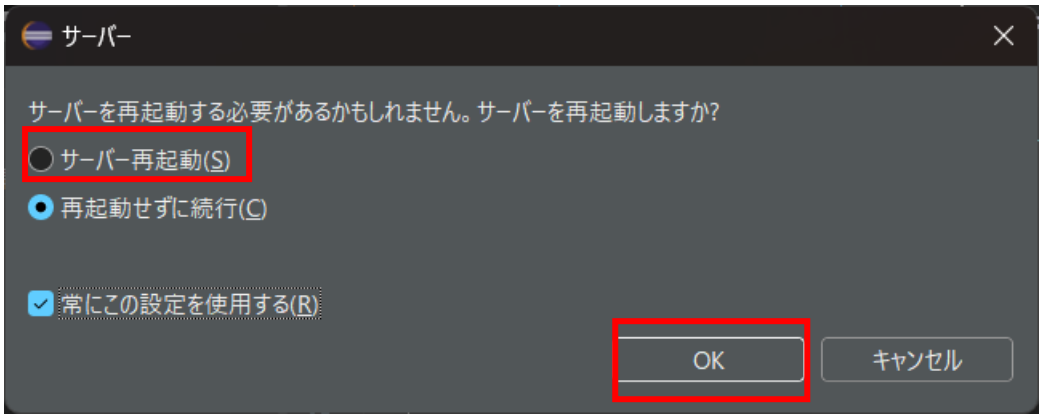
① 実行するサーブレットクラスを選択し右クリック⇒「実行」⇒「サーバで実行」を選択する



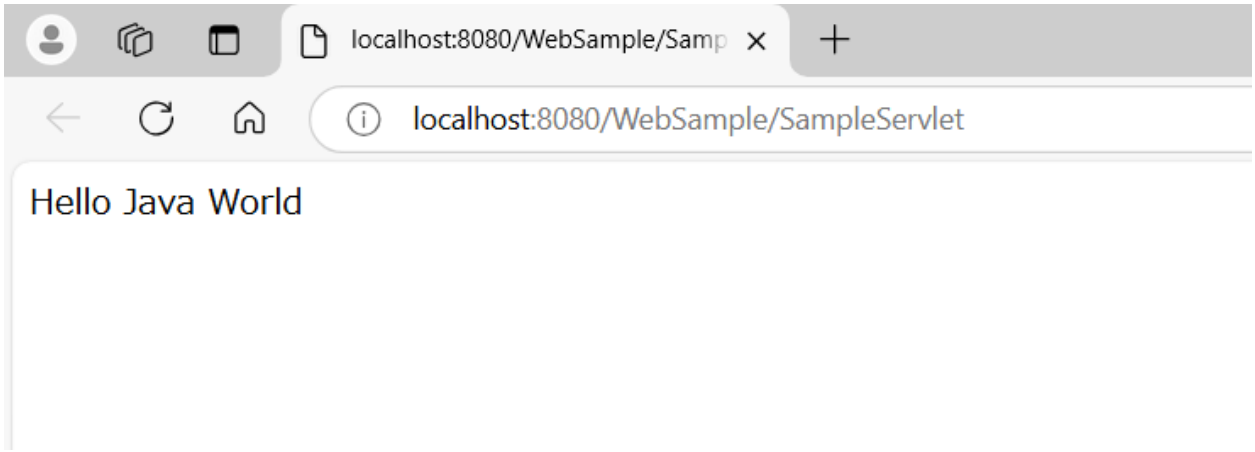
② 実行するサーバを選択し完了ボタンを押下する



③ サーバを再起動する必要のあるときは、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



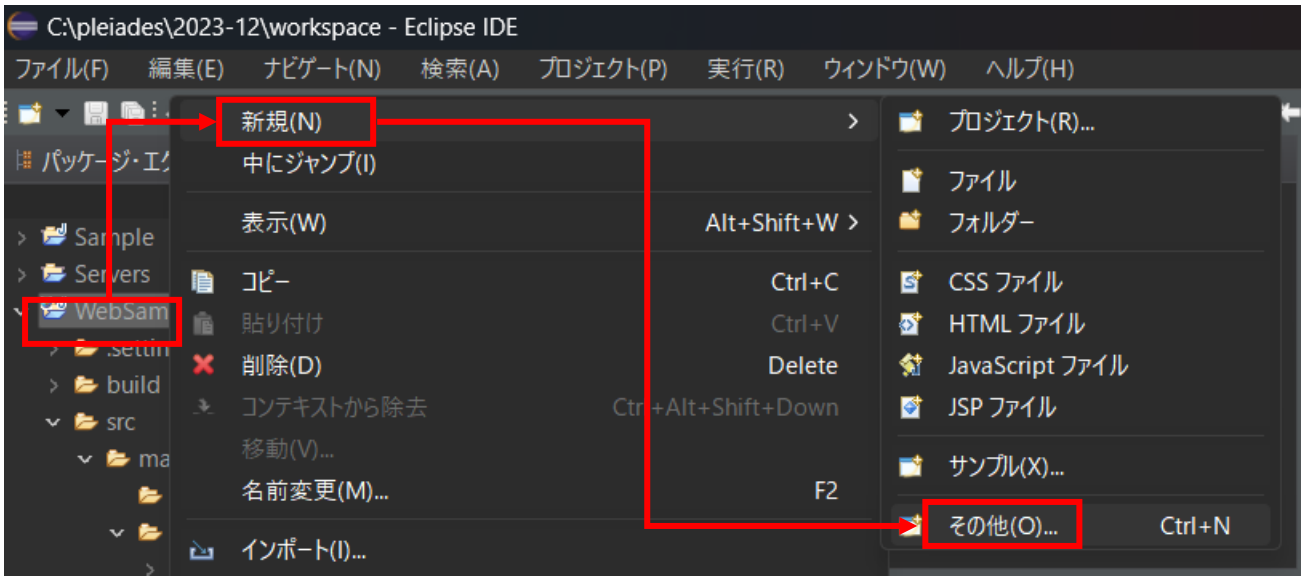
④ ブラウザに実行結果が表示される



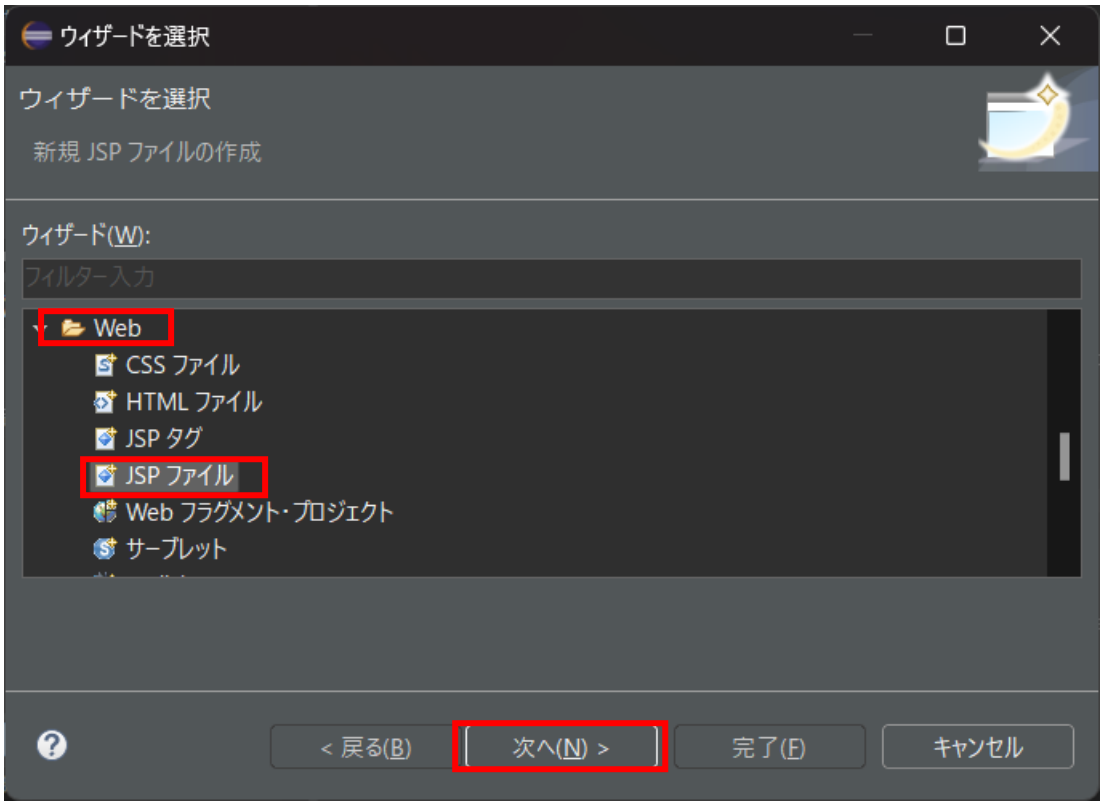
1.4 JSP ファイルの作成

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

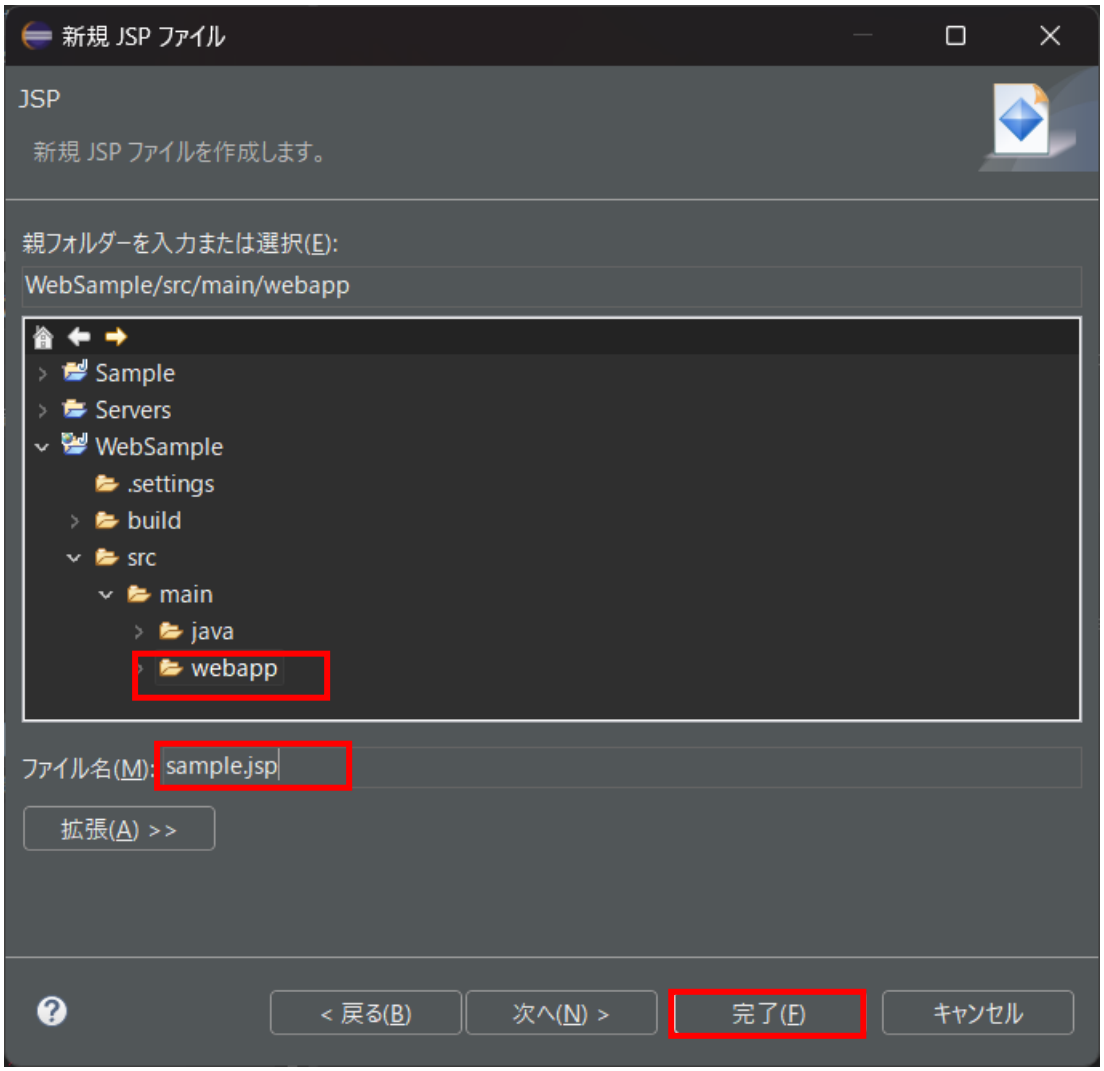
① 動的 Web プロジェクトを選択し、右クリック⇒「新規」⇒「その他」を選択



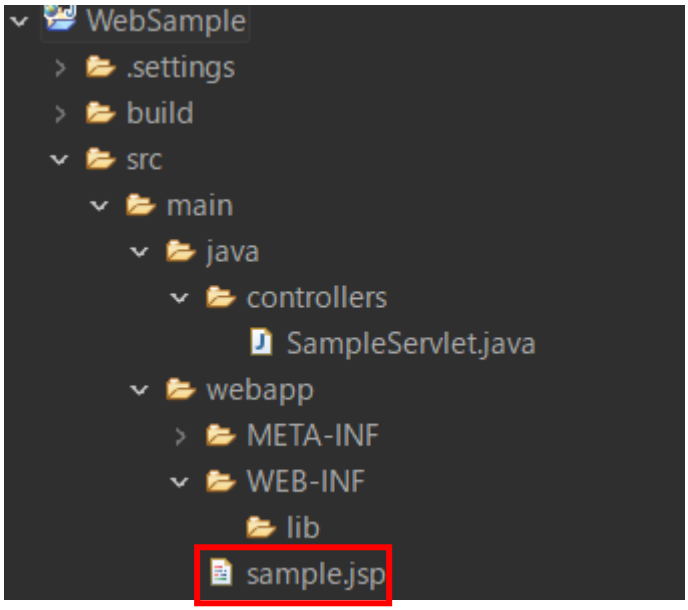
② 「Web」⇒「JSP ファイル」を選択し「次へ」ボタンを押下する



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 Servl
    // 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 サーブレットの実行を参照