Rocket software

Visual COBOL チュートリアル

COBOL プログラムを JVM バイトコードにコンパイルして利用する

1　目的

Visual COBOL の COBOL コンパイラーは、ネイティブコード生成とは別に JVM バイトコードや CIL コードと呼ばれる.NET クラスへダイレクトに変換する機能を持っています。この機能を利用することにより複雑な計算ロジックや精度を維持するような COBOL が得意とする分野に対して既存の COBOL コードを再利用することが可能になります。生成されたクラスファイルは Java や.NET 言語を駆使するプログラマーからはあたかも既存の Java や.NET 言語で作成されたクラスファイルと同等に呼び出すことができます。

このドキュメントでは、簡単な COBOL プログラムの作成と、その後、Eclipse IDE 用の Visual COBOL を使った COBOL コードをダイレクトに JVM クラスとして生成し、Java プロジェクトからそれを利用する方法について説明します。また、作成した Java プロジェクトおよび COBOL JVM プロジェクトの内容を再構成して、JAR ファイルを作成し、Java プロジェクトにて参照、実行できるようにします。これにより COBOL コンポーネントを利用した Java アプリケーションを任意の環境で利用できるようになります。

2　前提

本チュートリアルは、下記の環境を前提に作成されています。

- 開発環境

  Windows 11

  Visual COBOL 11.0 Patch Update 01 for Eclipse

- チュートリアル用サンプルプログラム

  下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダーに解凍しておいてください。

  サンプルプログラムのダウンロード

内容

3 チュートリアル手順

3.1 Windows クライアントでの開発準備作業

1) Visual COBOL for Eclipse を起動します。

① [スタート] メニュー > [Rocket Visual COBOL] > [Visual COBOL for Eclipse] を選択します。

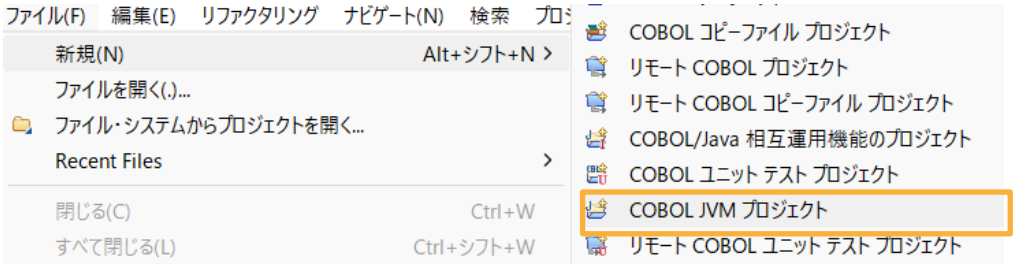
② ワークスペースの選択画面は任意のワークスペースを指定して、[起動(L)] をクリックします。

③ ようこそ画面は閉じてください。

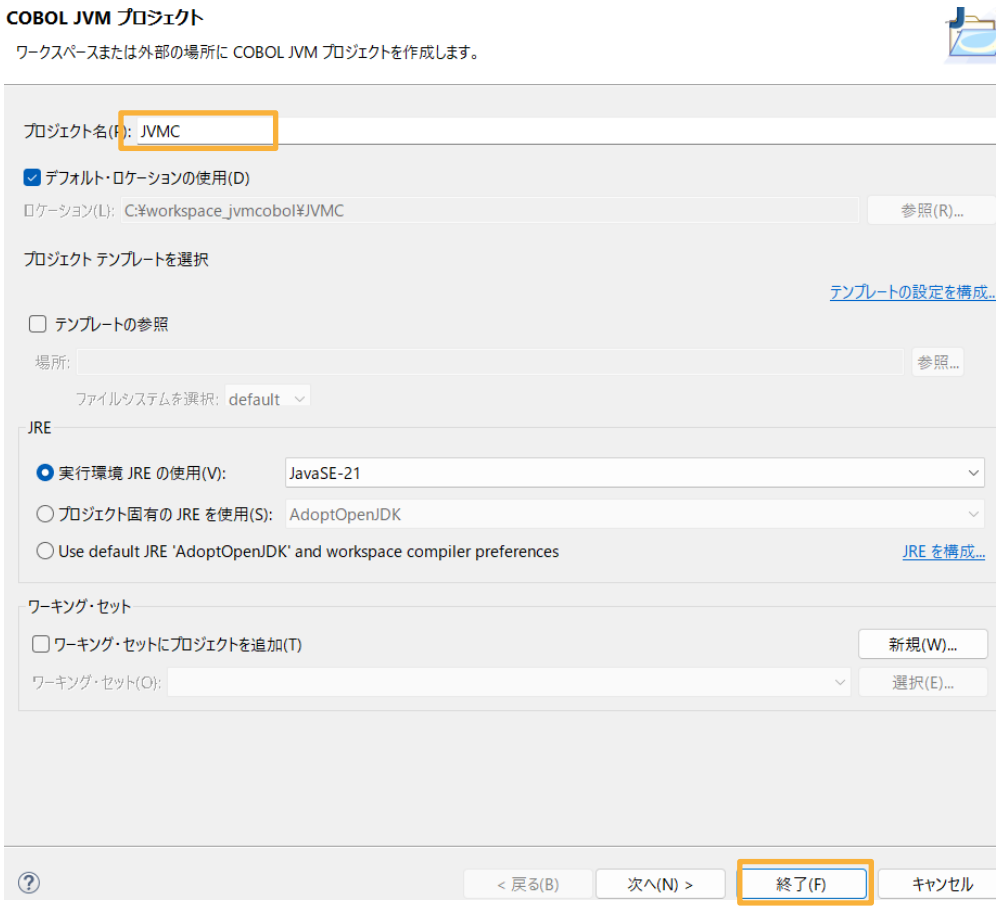
3.2 JVM COBOL プロジェクトの作成

1) COBOL JVM プロジェクトを作成します。

① [ファイル(F)]メニュー > [新規(N)] > [COBOL JVM プロジェクト] を選択します。

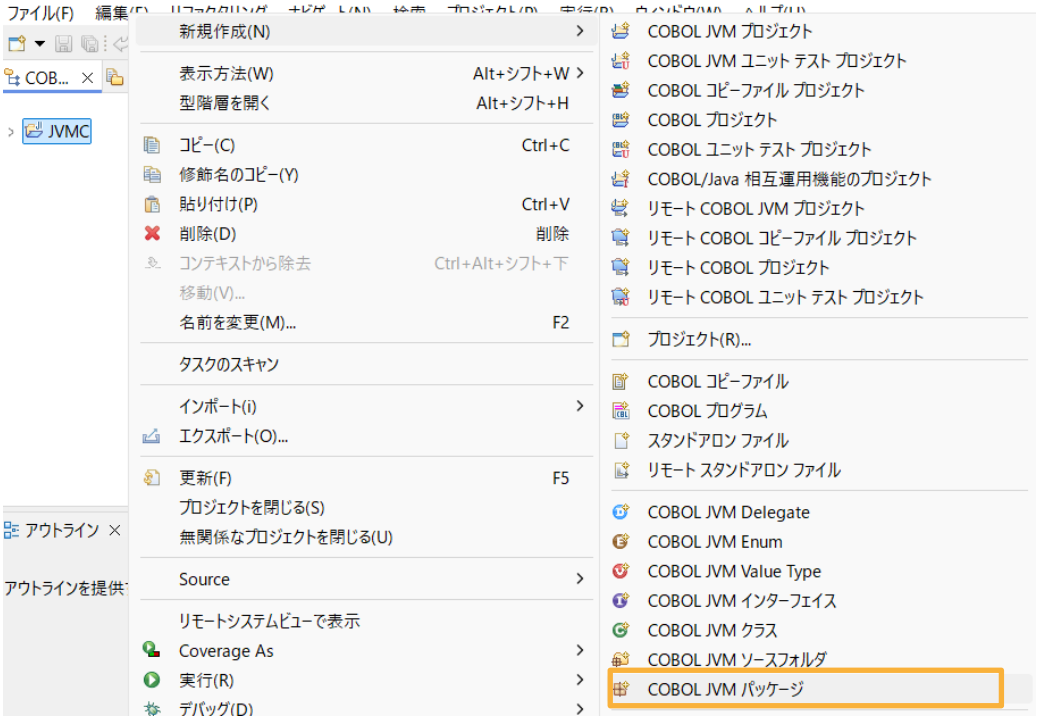


② プロジェクト名に “JVMC” を入力し、[終了(F)] をクリックします。

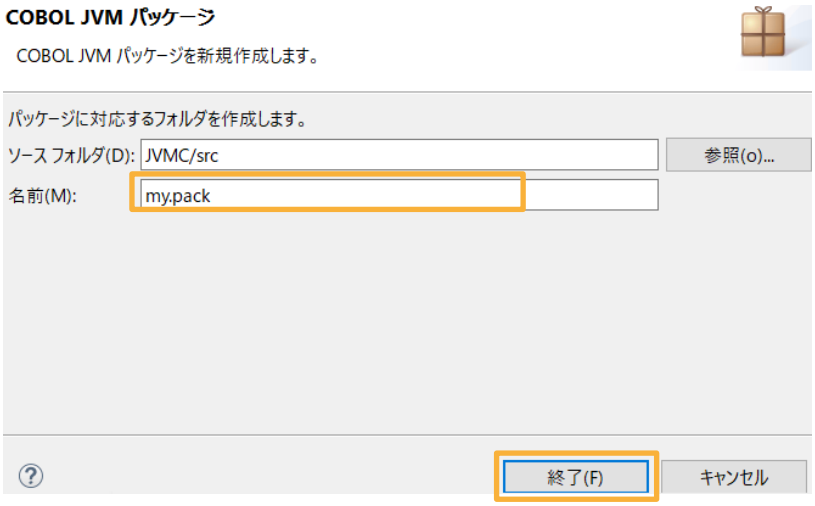


2) COBOL JVM パッケージを作成、設定変更します。

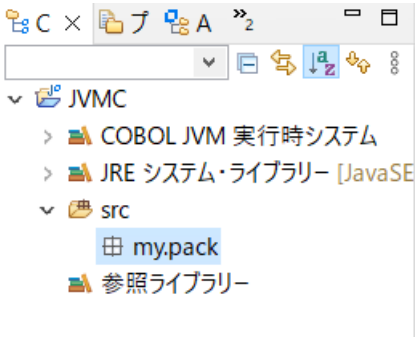
① 「JVMC」プロジェクトを右クリックし、コンテクストメニューから、[新規作成(N)] > [COBOL JVM パッケージ] を選択します。



② COBOL JVM パッケージ作成ダイアログが表示されるので名前に "my.pack" を入力して [終了(F)] をクリックします。

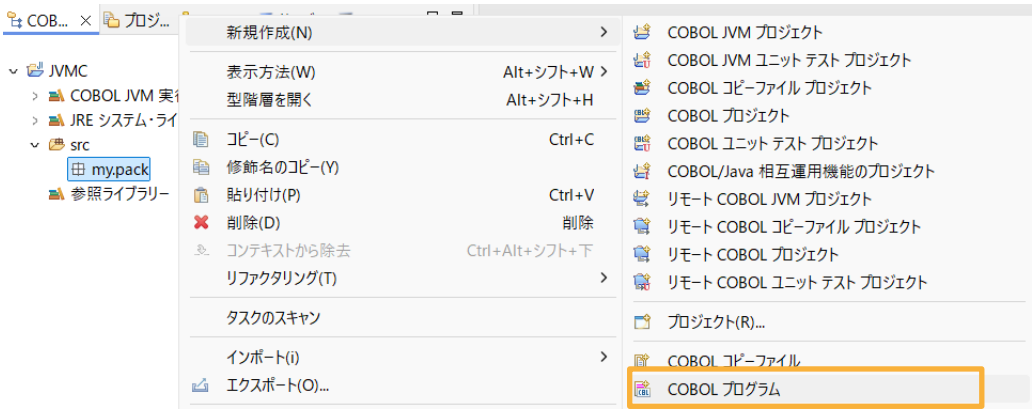


「src」の配下に「my.pack」パッケージが作成されます。

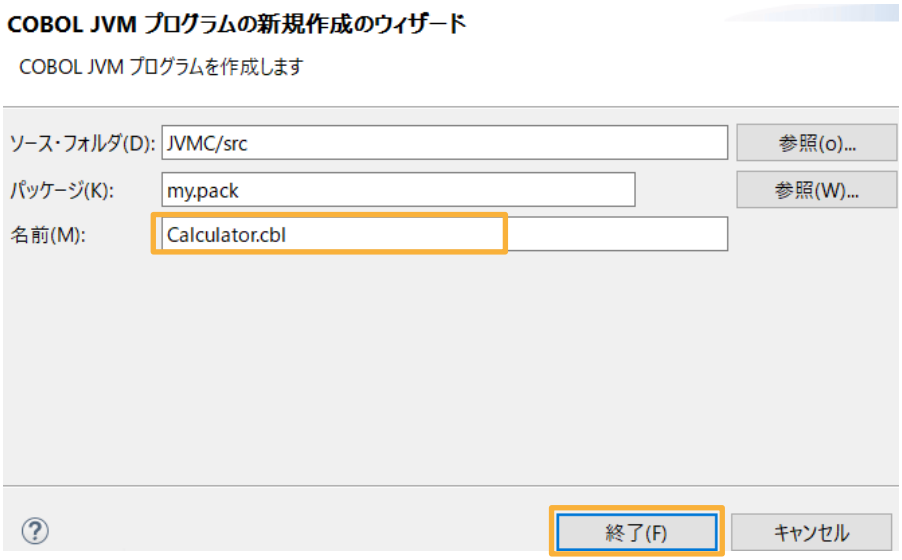


3) COBOL プログラムを作成します。

① 作成された「my.pack」パッケージを右クリックし、コンテクストメニューから [新規作成(N)] > [COBOL プログラム] を選択します。



② 「COBOL JVM プログラムの新規作成のウィザード」が表示されます。 [名前] に "Calculator.cbl" を入力し、[終了(F)] をクリックします。



テンプレートの「Calculator.cbl」が展開されます。

③　ダウンロードしたサンプルプログラムから「Caluculator.cbl」をメモ帳等でオープンし、内容を全てコピー＆ペーストします。

COBOL コードの説明）

$set ilnamespace "my.pack"　←Java のパッケージ化と同等の機能

$set ilsmartlinkage "my.pack"　←linkage section に記述されている変数に対応する Java クラスを生成

$set ilcutprefix “lnk-“ ←上記 ilsmartlinkage で生成されるクラス・変数名は、デフォルトでは変数名となるが、本指令により名称から「lnk-」を削除

```
linkage section.
 01 args
     03 lnk-arg1              pic 9(5) comp-3.
     03 lnk-arg2              pic 9(5) comp-3.
    03 lnk-sum               pic 9(5) comp-3.
```

ここでは Args というクラスファイルが自動的に生成される。「Lnk-」をカットする指令が指定されているため、Java からは「lnk-」を除いた変数名でアクセスができる。

```
procedure division using args.
  add lnk-arg1 to lnk-arg2 giving lnk-sum.
```

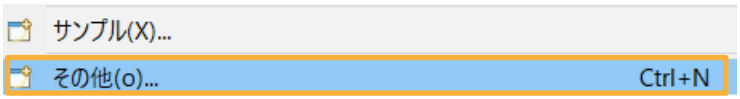
Linkage section の変数を引き継いで加算処理が行われる。

④　CTRL+S を押してファイルを保存するとコンパイルが行われます。

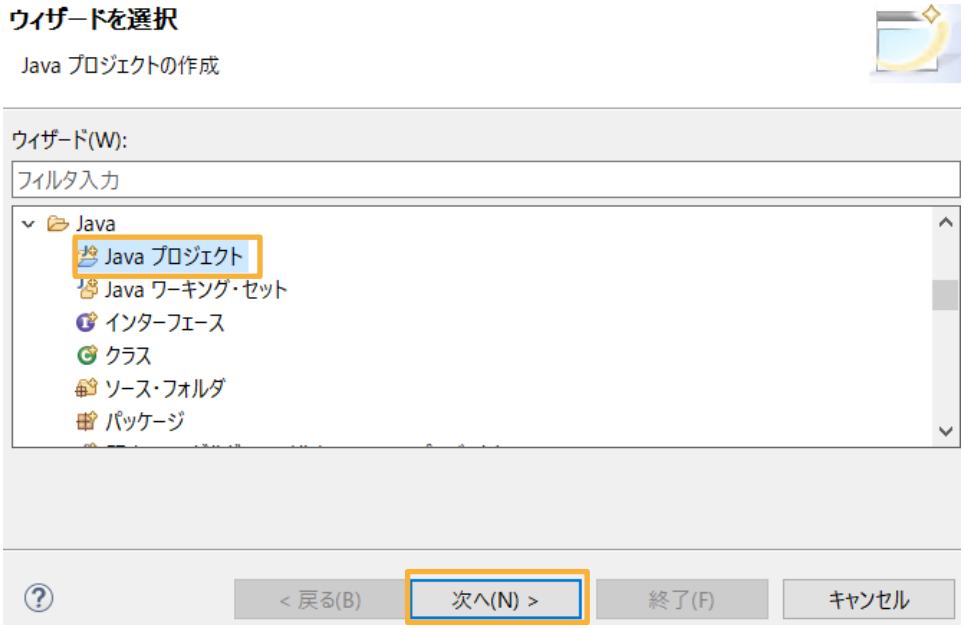
3.3　Java プロジェクトの作成

1)　呼び出し元の Java プロジェクトを作成します。

①　COBOL エクスプローラーにて、Eclipse メニューより、[ファイル(F)] > [新規(N)] > [その他] を選択します。



②　[Java] > [Java プロジェクト] を選択し、[次へ(N)] をクリックします。

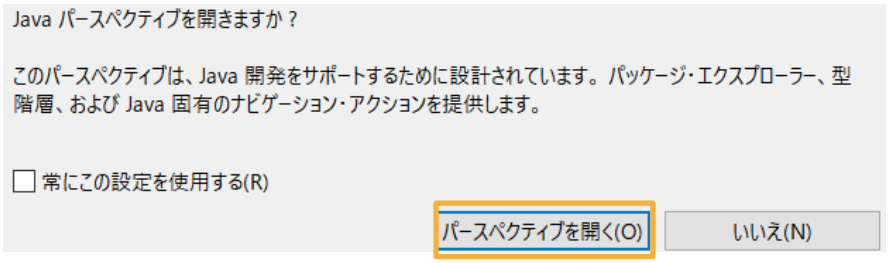


③ 「Java プロジェクトの作成」ウィザードが表示されるので、以下の設定を行ったうえで、[終了(F)] をクリックします。

プロジェクト名： “CALC”

Module-info.java を作成： チェックを外す

パースペクティブの切り替えのダイアログが表示された場合は、[パースペクティブを開く(O)] をクリックします。

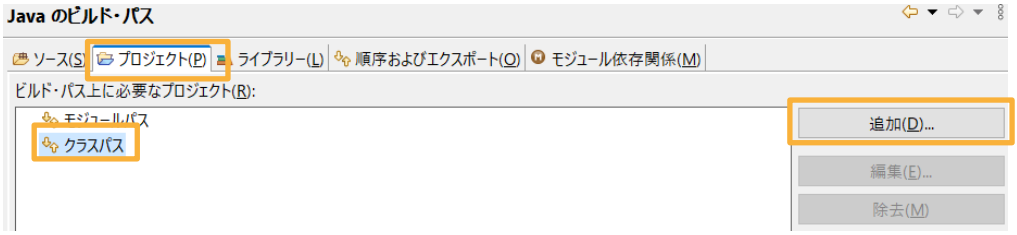


2) プロパティ情報を更新します。

① 「CALC」プロジェクト上で右クリックして、コンテクストメニューから[プロパティ(R)]を選択します。

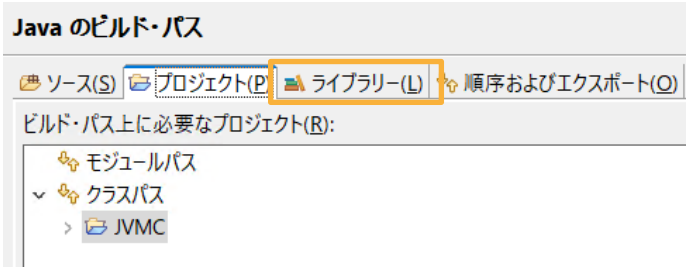
② [Java のビルドパス] を選択して、[プロジェクト(P)] タブをクリックします。

③ [クラスパス] を選択し、[追加(D)] をクリックします。



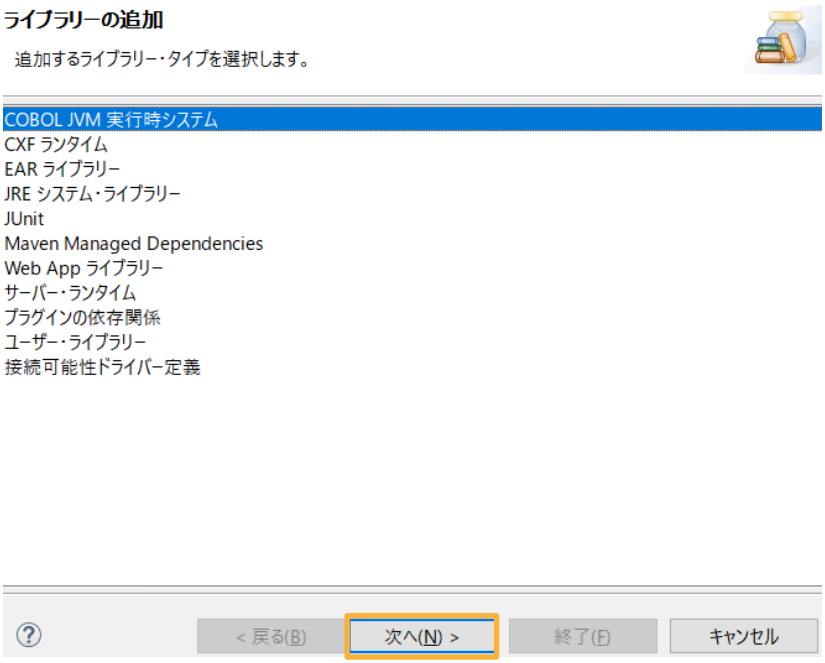
④ JVMC プロジェクトにチェックを入れて、[OK] をクリックします。

⑤ 次にライブラリー(L)タブをクリックします。

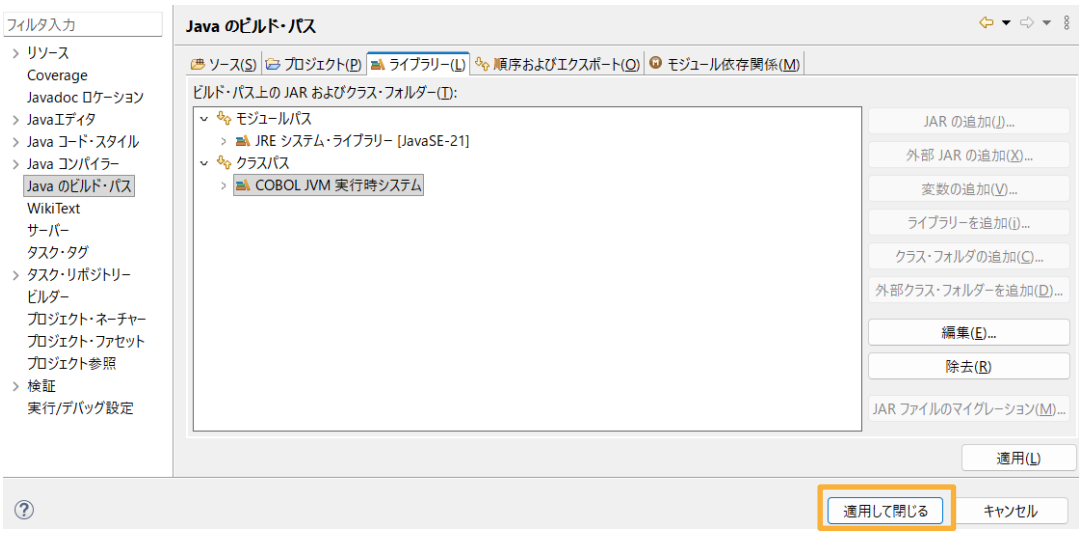


⑥ [クラスパス] を選択し、[ライブラリーを追加(i)] をクリックします。

⑦ 「COBOL JVM 実行時システム」を選択し、[次へ(N)] をクリックし、次の画面でそのまま [終了(F)] をクリックします。

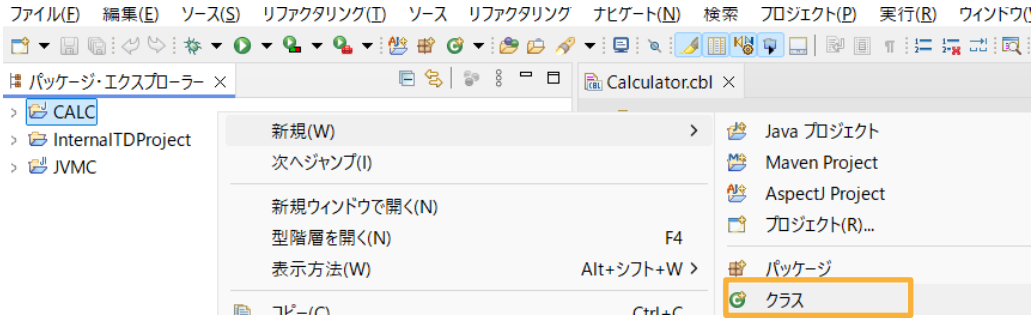


⑧ [適用して閉じる] をクリックします。



3) Main メソッドを含んだ Java アプリケーションを作成します。

① 「CALC」プロジェクトを選択し、マウスの右クリックでコンテクストメニューを開き、[新規(W)] > [クラス] を選択します。



② 新規 Java クラス作成のためのウィザード画面が表示されるので、以下の入力を行い、[終了(F)] をクリックします。

パッケージ： “com.calc”

名前： “MainClass”

③ 雛型の「MainClass.java」が展開されます。ダウンロードしたファイルから MainClass.java をメモ帳等でオープンし、このソースコードにコピー＆ペーストを行います。

④ CTRL+S を押してファイルを保存するとコンパイルが行われます。

Java コードの説明）

Calculator calc = new Calculator(); ← Calculator クラス本体のインスタンス作成

Args arguments = new Args(); ←Linkage section に定義した変数へアクセスするためのクラス

arguments.setArg1(4); ←Linkage section に定義した変数へアクセスする setter メソッド

arguments.setArg2(2); ← 同上

calc.Calculator(arguments); ←計算を行うメソッドの呼び出し

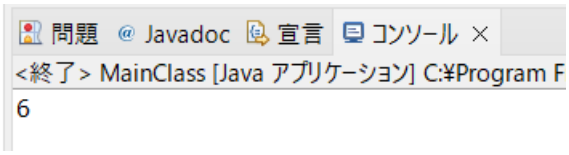
System.out.println(arguments.getSum()); ←Linkage section に定義した変数へアクセスする getter メソッドにて値を取得し、コンソールに表示

3.4 作成した Java アプリケーションの実行

1) Java アプリケーションを実行します。

① 「CALC」プロジェクトを右クリックし、コンテクストメニューから [実行(R)] > [Java アプリケーション] を選択します。

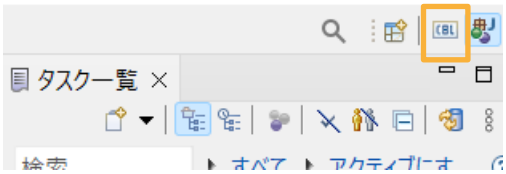
② コンソールに計算結果の 6 が出力されます。



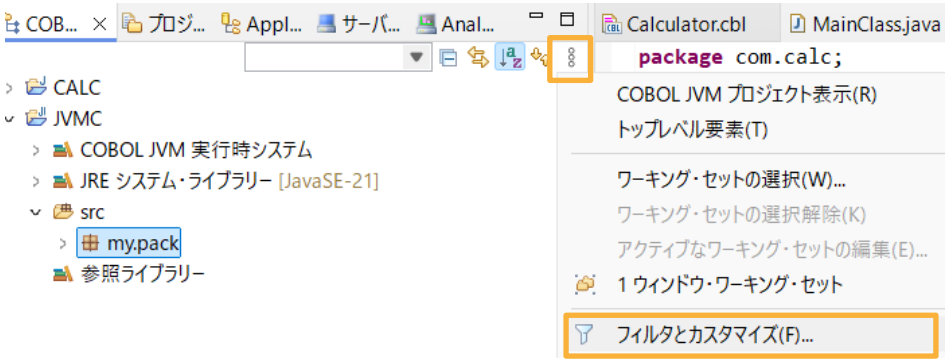
3.5 COBOL パースペクティブのカスタマイズ

1) COBOL パースペクティブをカスタマイズします。

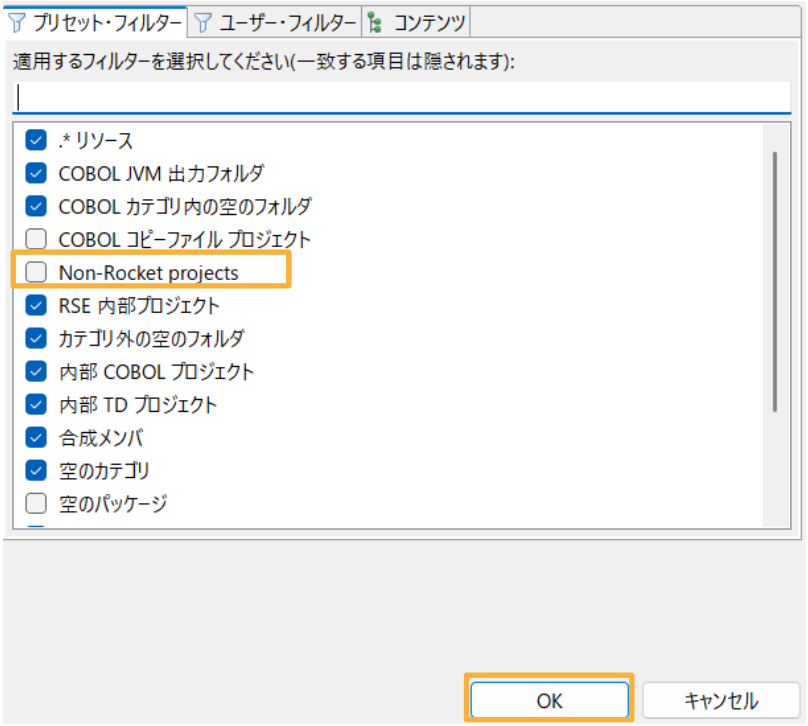
① 画面右上のアイコンをクリックして、パースペクティブを COBOL に変更します。



② COBOL エクスプローラーの設定アイコンをクリックし、 [フィルタとカスタマイズ(F)…] を選択します。



③ [Non-Rocket projects] にチェックされている場合はチェックを外したうえで [OK] をクリックします。



3.6 JVM COBOL のパッケージ化

1) COBOL JVM プロジェクト「JVMC」から JAR ファイルを出力するように設定します。

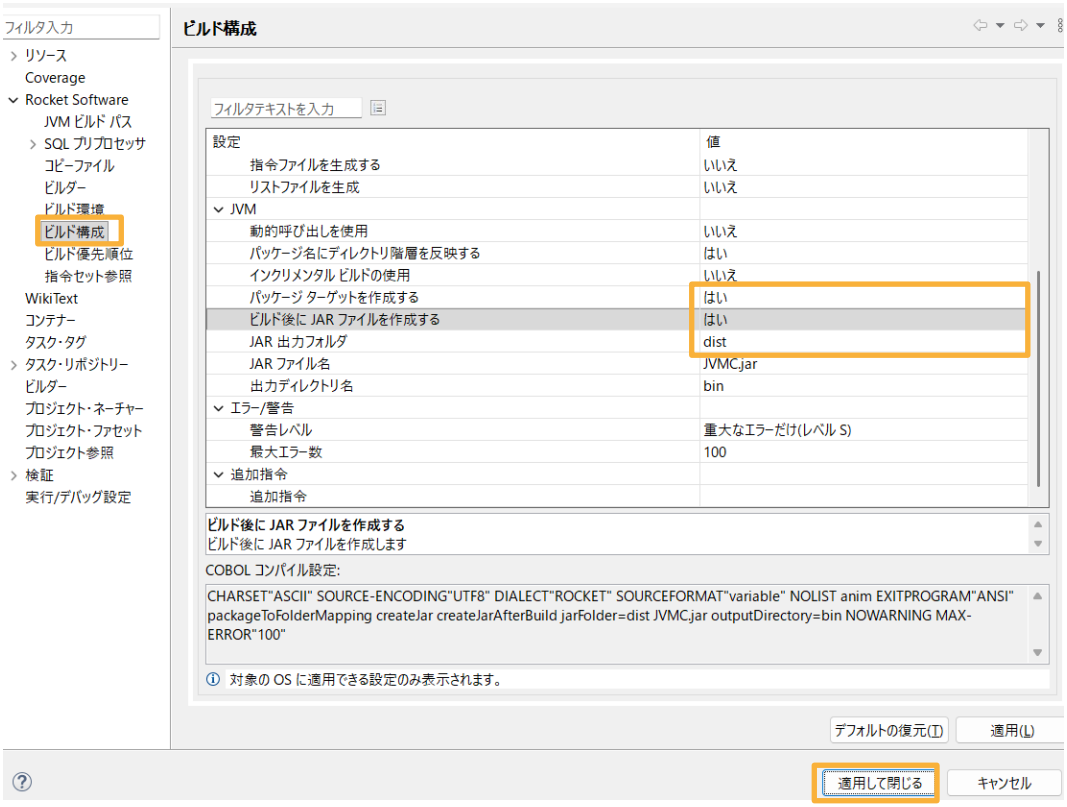
① 「JVMC」プロジェクト上で右クリックし [プロパティ] を選択します。

② [Rocket Software] > [ビルド構成] を選択し、JVM セクションは以下の項目に以下の設定を行ったうえで、[適用して閉じる] をクリックします。

パッケージターゲットを作成する： 「はい」を選択

ビルド後に JAR ファイルを作成する： 「はい」を選択

JAR 出力フォルダ： “dist”



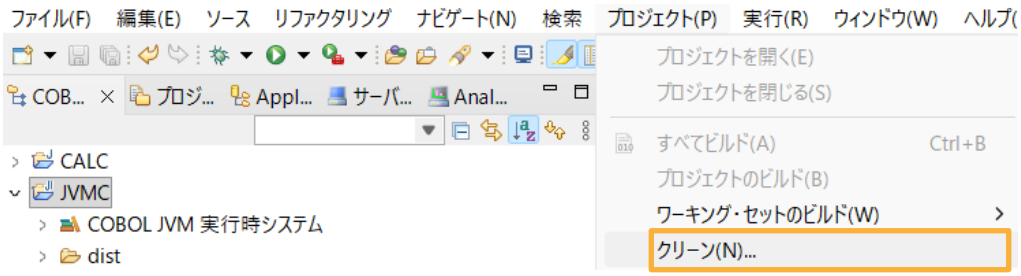
3.7 プロジェクトのビルド

1) JAR ファイルをビルドします。

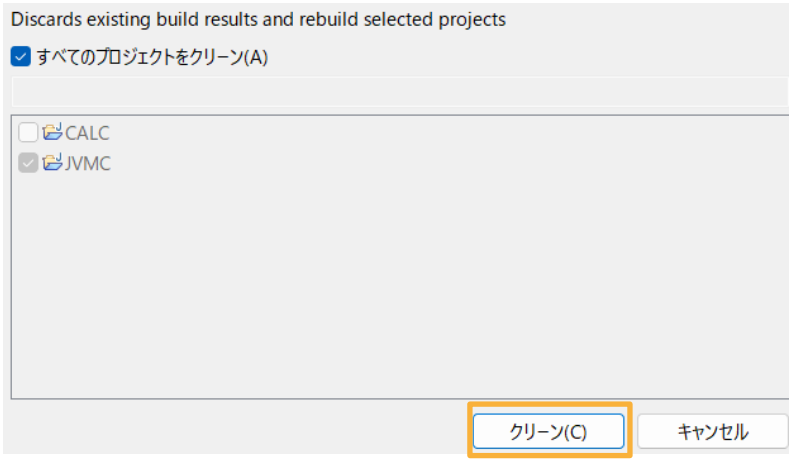
自動ビルドが有効でばない場合は、s 下記の作業を行います。

① COBOL エクスプローラーにて、「JVMC」プロジェクトを選択します。

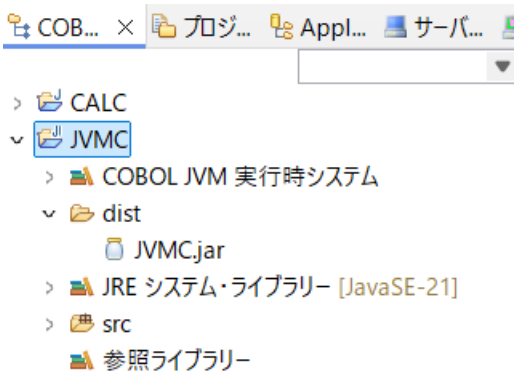
② Eclipse メニューより、[プロジェクト(P)] > [クリーン(N)]を選択します。



③ そのまま、[クリーン] をクリックします。



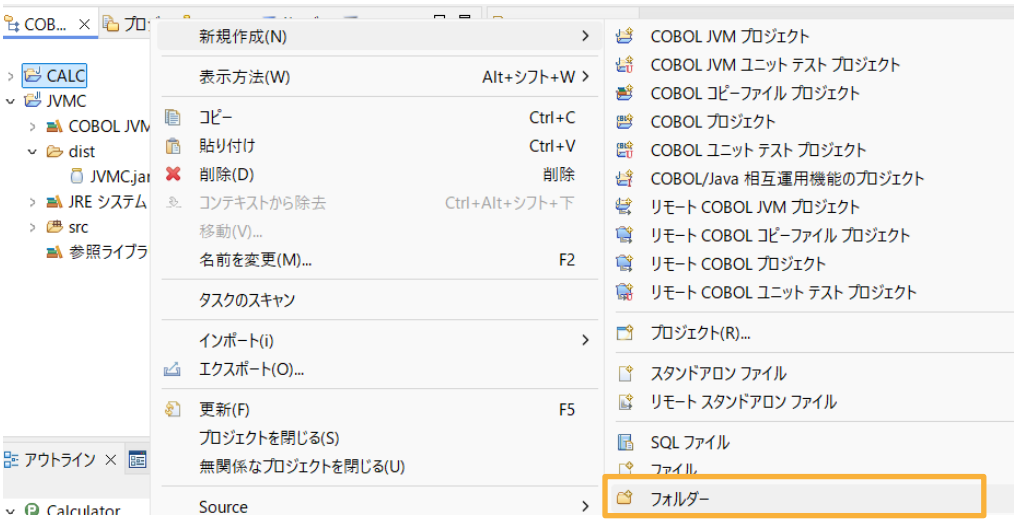
④ ビルドが終了すると「dist」フォルダーに JVMC.jar ファイルが作成されます。



3.8 パッケージ化されたファイルのコピー

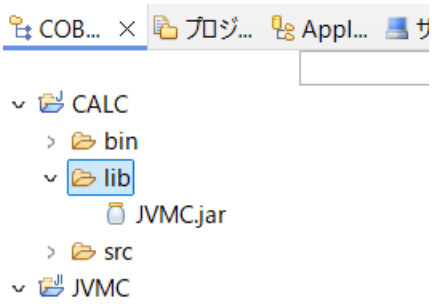
1) 作成した jar ファイルを Java のプロジェクトに設定します。

① 「CALC」プロジェクトを右クリックし、コンテクストメニューから [新規作成]> [フォルダー] を選択します。



② フォルダ名に “lib” を入力し、[終了(F)] をクリックします。

③ 「JVMC」プロジェクトにある「JVMC.jar」を右クリックで選択し、コンテクストメニューから [コピー] を選択し、その後、「CALC」プロジェクト配下の「lib」フォルダー上にて右クリックし、コンテクストメニューから「貼り付け」を選択します。ファイルが「lib」フォルダーにコピーされます。



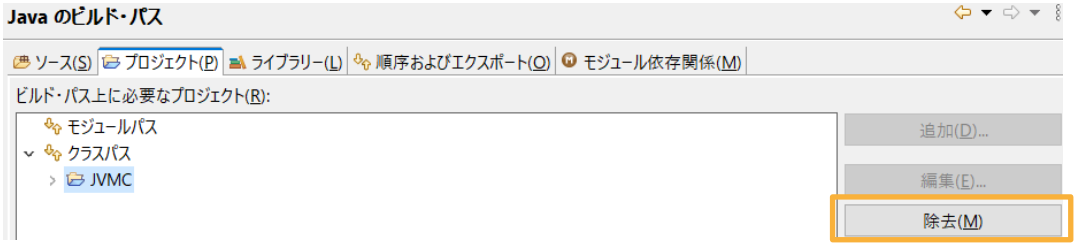
3.9 パッケージ化された Jar ファイルを使用するように Java プロジェクトを変更

1) Java プロジェクトの変更を行います。

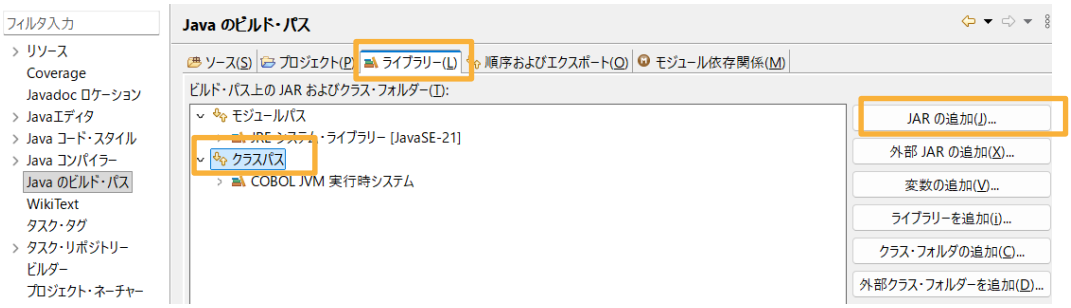
① コンテクストメニューにて「CALC」 プロジェクトを右クリックし、コンテクストメニューから [プロパティ(R)] を選択します。

② Java ビルドパスをクリックします。

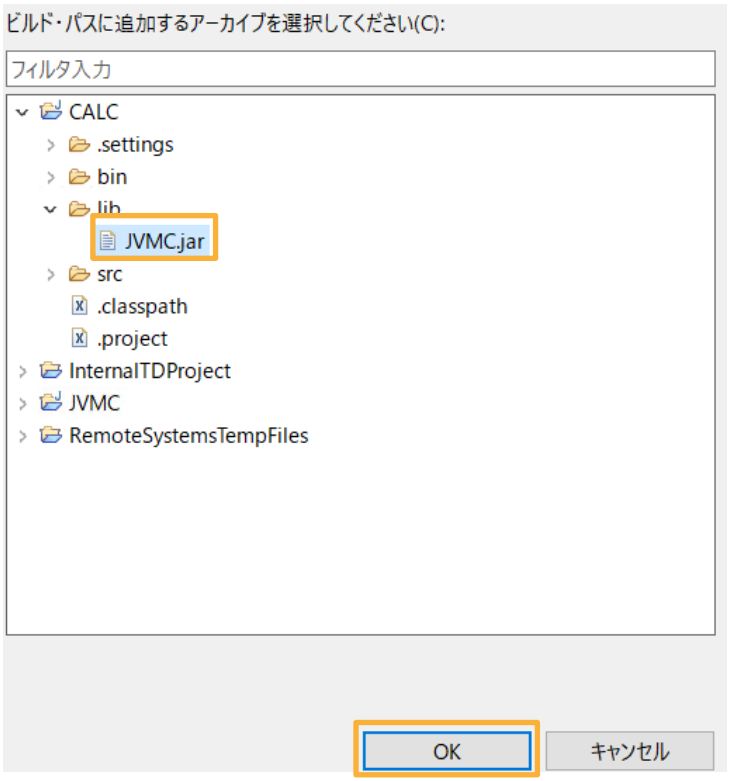
③ [プロジェクト(P)] タブより「JVMC」を選択し、[除去(M)]をクリックします。



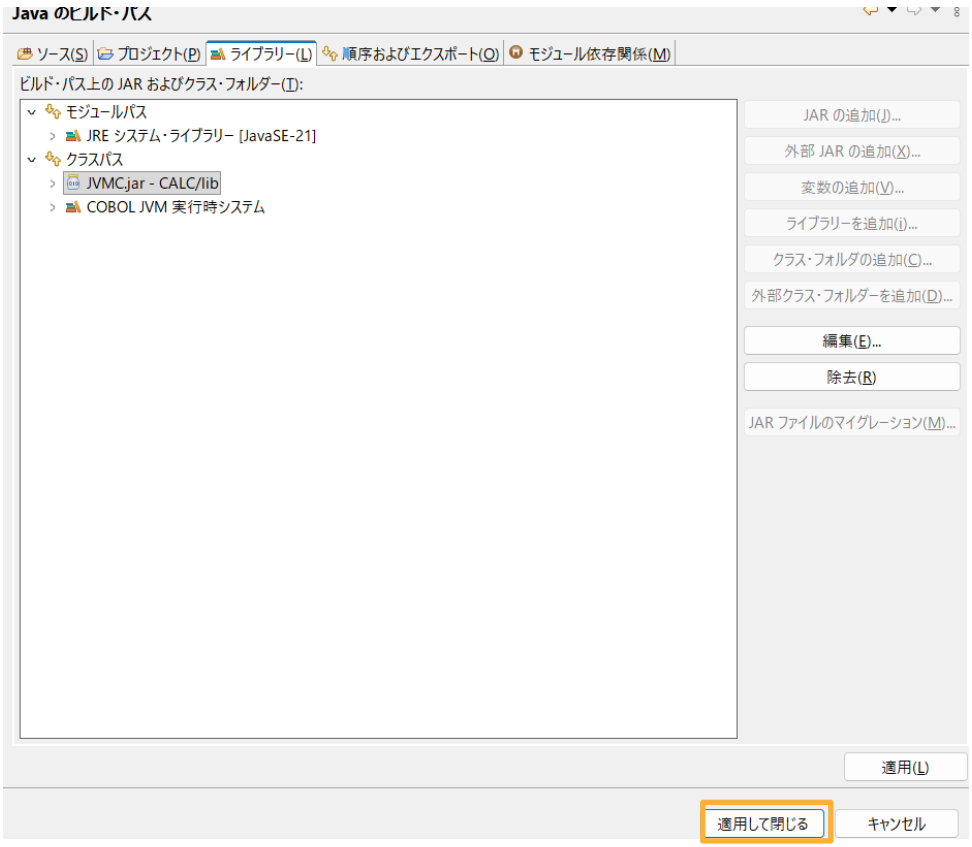
④ 次に[ライブラリー(L)] タブを選択したうえで、クラスパスを選択後、[JAR の追加(J)…] をクリックします。



⑤ 「CALC」プロジェクトを展開し、「lib」フォルダーから CALC プロジェクト配下の 「lib/JVMC.jar」を選択し、[OK] をクリックします。



⑥ [適用して閉じる] をクリックします。



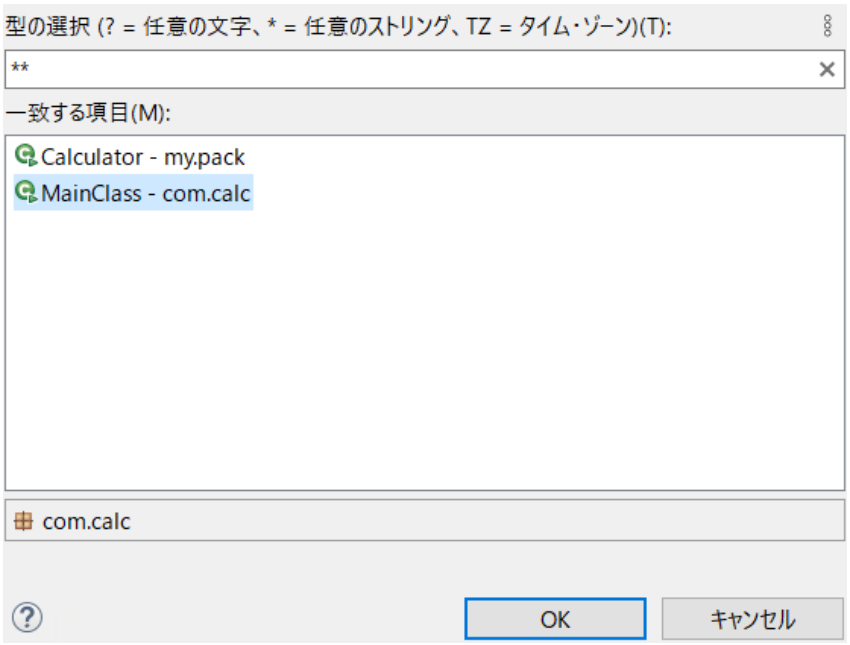
3.10 Java プロジェクトの実行

1) Java アプリケーションを実行します。

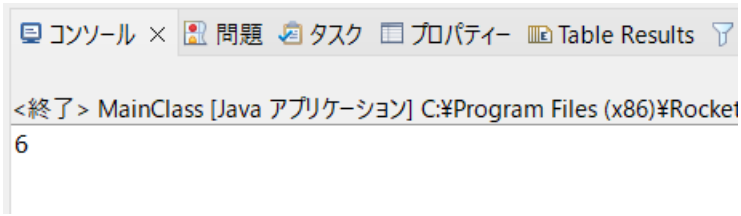
① 「CALC」プロジェクトを右クリックし、コンテクストメニューから [実行(R)] > [Java アプリケーション] を選択します。

② Java アプリケーションの選択ダイアログが表示されます。

③ 「MainClass – com.calc」を選択して、[OK] をクリックします。



さきほど同様、コンソールに結果が出力されます。



免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。 ソースコード、説明、その他すべてについて、無謬性は保障されません。

ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。

本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。