Visual COBOL チュートリアル

COBOL と Java の相互運用

1 目的

ビジネスの基幹システムで使用されている開発言語として COBOL は長く利用し続けられていますが、Web システムなど他システムでは Java をはじめ、様々な開発言語で記述されています。このことは、開発言語の優劣を表すものではなく、それぞれの開発言語の得意分野を有効に選択している表れです。例えば、Web システムをあげてみますと、Java には、様々なフレームワークが提供されており、すでに実装経験・知識を持つ開発者も多いことから、Java を選択することが最適となる場合が多いでしょう。これとは反対に、COBOL で記述された基幹システム、多くの計算処理が含まれているようなアプリケーションの機能拡張を考えたとき、Java で新規開発するよりも COBOL で記述するほうが計算精度を保ち、保守性を維持することができます。

これからのシステムは、以前のような単独で稼働する形態から、様々なシステム・アプリケーションとの連携が求められます。また、基幹システムが稼働を開始した後、別システムの開発・運用を行っていく中で、ある機能が基幹システムでも利用できたら、と感じることもあるかもしれません。基幹システムで利用するために、同じ機能を新たに COBOL で開発となると、保守性の劣化が懸念されますが、新規開発が不要、もしくは、容易に導入できるのであれば、どうでしょうか。

本チュートリアルでは、このようなシステム間連携を見据え、COBOL アプリケーションが Java 資産を利用する方法について、また、Java アプリケーションから COBOL 資産を利用する方法について紹介します。

2 前提

- 本チュートリアルで使用したマシン： Windows 11
- Adoptium OpenJDK21
- Visual COBOL 11.0 Patch Update 01 for Eclipse 製品をインストールし、COBOL 開発が行える環境

本チュートリアルでは、一部の手順において、下記リンク先のサンプルファイルを使用します。事前にダウンロードをお願いします。

サンプルプログラムのダウンロード

内容

3 COBOL から Java 資産を呼び出す

COBOL から Java 資産を呼び出す代表的な方法は以下になります。

- SYSTEM ルーチンを利用した Java プログラムを起動
- Java 資産をサービスとして運用し、COBOL から利用
- COBOL/Java 相互運用機能を利用
- COBOL アプリケーションの実行環境を Java 仮想マシンに移して Java 資産とともに Java として利用

3.1 SYSTEM ルーチンを利用した Java プログラムを起動

最も簡単な Java プログラムの起動方法は、SYSTEM ルーチンを利用した Java プロセスの起動です。例えば、以下のプログラムでは、java のバージョン情報を表示します。

```
working-storage section.
01 cmd pic x(50) value "java -version".
procedure division.
call "system" using cmd.
```

補足）

java に限らず、任意のプログラムを実行することができます。

しかし、別プロセスとしての起動となることに加え、起動する Java プログラムへ情報を渡そうとした場合、実行時引数、もしくは、ファイルやデータベースの利用などが必要となることから、単純なプログラム起動以外には適しません。

以降に紹介する方法では、このような問題を解決する方法を紹介します。

3.2 Java 資産をサービスとして運用し、COBOL から利用

一般的なシステム間連携機能として Web API、サービスがあげられますが、このような機能を提供することで、COBOL を含めた様々な開発言語、アプリケーションとの連携が可能になります。本節では、JSON データを送受信する REST API を利用する方法について紹介します。なお、Java 資産をサービスとして運用する方法については、別途インターネット文献などを参照ください。

本節では、サービスプロバイダが提供するサービスの定義を利用して COBOL クライアントプログラムを生成し、そのクライアントを用いたサービス連携を行います。

補足）

本チュートリアルでは、Java 資産のサービス運用を前提に紹介していますが、サービスの開発言語は Java に限定されず、C# などの .NET 言語、Node.js なども利用できます。

3.2.1 前提条件

このチュートリアルは、REST API でアクセス可能なサービスの開発、稼働については対象外です。また、以降で説明する手順では、以下のオンラインで公開されているテストサービスの 1 つを利用します。

https://jsonplaceholder.typicode.com/

パス： posts/{postId}/comments

HTTP メソッド： GET

例）https://jsonplaceholder.typicode.com/posts/1/comments

多くの開発言語では、公開 API に対するクライアントプログラムは、API が提供するサービスを定義したファイルから生成できます。これをスキーマ駆動開発と呼びますが、Visual COBOL 製品を利用することで COBOL でもスキーマ駆動開発を利用できます。本チュートリアルで使用するサービスに対応するサービス定義ファイルは、サービス提供サイトからは提供されていないため、上記サービスに対応する定義をサンプルファイル内に get_post.xml として用意しています。このファイルは、OpenAPI 仕様に沿った yaml 形式で記述されています。OpenAPI については、以下の公式サイトなどを参照ください。

https://www.openapis.org/

3.2.2 サービス定義ファイルからのクライアントプログラムの生成

1) スタートメニューより、[Rocket Visual COBOL] > [Visual COBOL Command Prompt (64-bit)] を選択します。
2) プロンプト上で、サンプルファイルを解凍したフォルダ配下の apiservice フォルダに移動します。
3) プロンプト上で、以下のコマンドを実行して、クライアントプログラムを生成します。

   imtkmake -genclientjson clientjson=get_post.yaml

```
C:¥COBOLJavaInteroperability¥apiservice>imtkmake -genclientjson clientjson=get_post.yaml
Rocket Software Interface Mapping Toolkit v11.0.00261
(C) Copyright 1984-2025 Rocket Software, Inc. or its affiliates. All Rights Reserved
C:¥COBOLJavaInteroperability¥apiservice>
```

生成される COBOL ファイルとコピーブックは以下になります。

ファイル名	説明
get_post-app.cbl	サービスとの接続確認を行う対話形式のコンソールアプリケーション
get_post-proxy.cbl	get_post-app.cbl から呼び出され、サービスとの通信を行うプログラム
get_post-copy.cpy	上記 2 ファイルより参照されるコピーファイル

3.2.3 クライアントプログラムの動作確認

1) 前手順に引き続き、Visual COBOL コマンドプロンプト上で、以下のコマンドを実行し、プログラムのコンパイルを行います。

cobol get_post-app.cbl gnt;

cobol get_post-proxy.cbl gnt;

```
C:¥COBOLJavaInteroperability¥apiservice>cobol get_post-app.cbl gnt;
Rocket (R) COBOL
Version 11.0 (C) 1984-2025 Rocket Software, Inc. or its affiliates.
* チェック終了：エラーはありません - コード生成を開始します
* Generating get_post-app
* Data:       138512      Code:         5788      Literals:         2288
C:¥COBOLJavaInteroperability¥apiservice>cobol get_post-proxy.cbl gnt;
Rocket (R) COBOL
Version 11.0 (C) 1984-2025 Rocket Software, Inc. or its affiliates.
* チェック終了：エラーはありません - コード生成を開始します
* Generating get_post-proxy
* Data:         1296      Code:         4338      Literals:          368
C:¥COBOLJavaInteroperability¥apiservice>
```

2) プロンプト上で、以下のコマンドを実行します。

runw get_post-app.gnt

表示された画面上で、以下の入力を行ってください。

Service Address: 何も入力せず Enter キーを押す

Supplemental Query String: 何も入力せず Enter キーを押す

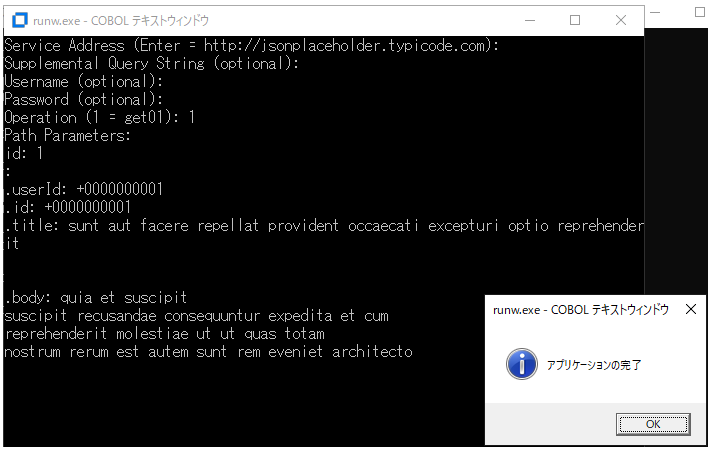
Username: 何も入力せず Enter キーを押す

Password: 何も入力せず Enter キーを押す

Operation: “1” を入力して Enter キーを押す

Path Parameters: id: “1” を入力して Enter キーを押す

サービスからの応答結果が表示されます。



[OK] をクリックして、アプリケーションを終了します。

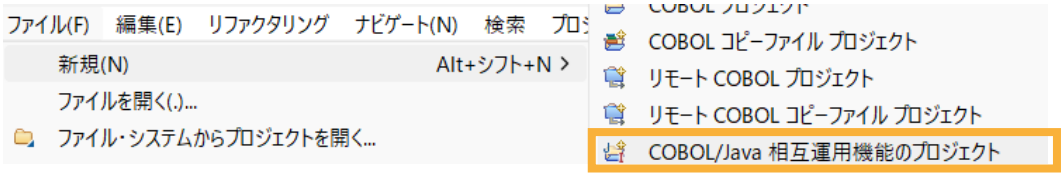
3.3 COBOL/Java 相互運用機能を利用

3.2 で紹介した方法は、Java 資産がサービスとして運用され、クライアント環境からアクセスできることが必要です。しかし、運用環境によっては、サービスやシステムが別環境上で稼働、環境間にファイアウォールが存在するなど、サービスの通信ポートへのアクセスがブロックされていることがあります。また、新たにサービスを立ち上げたくない、というケースも考えられます。このような課題をクリアしつつ、COBOL から Java 資産を呼び出すことができる方法が、本節で紹介する COBOL/Java 相互運用機能です。

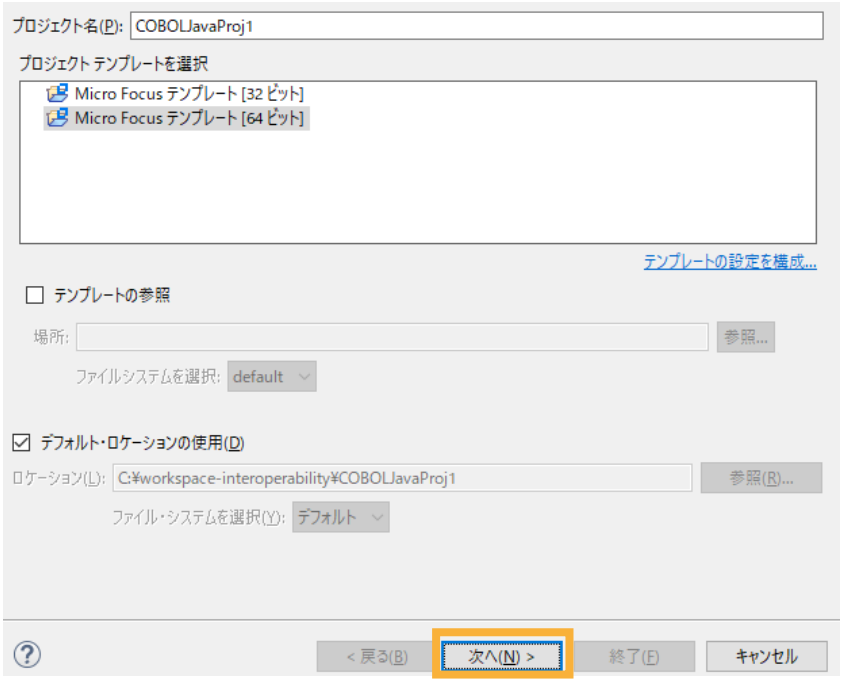
3.3.1 COBOL/Java 相互運用機能のプロジェクトの利用

本項で使用するプロジェクトは、COBOL 資産と Java 資産を同じプロジェクト配下で管理します。
しかし、Eclipse IDE 上でデバッグ実行ができる対象は COBOL 資産に限定されます。Java 資産については、別途 Java プロジェクトなどを作成したうえで、デバッグ作業を実施してください。

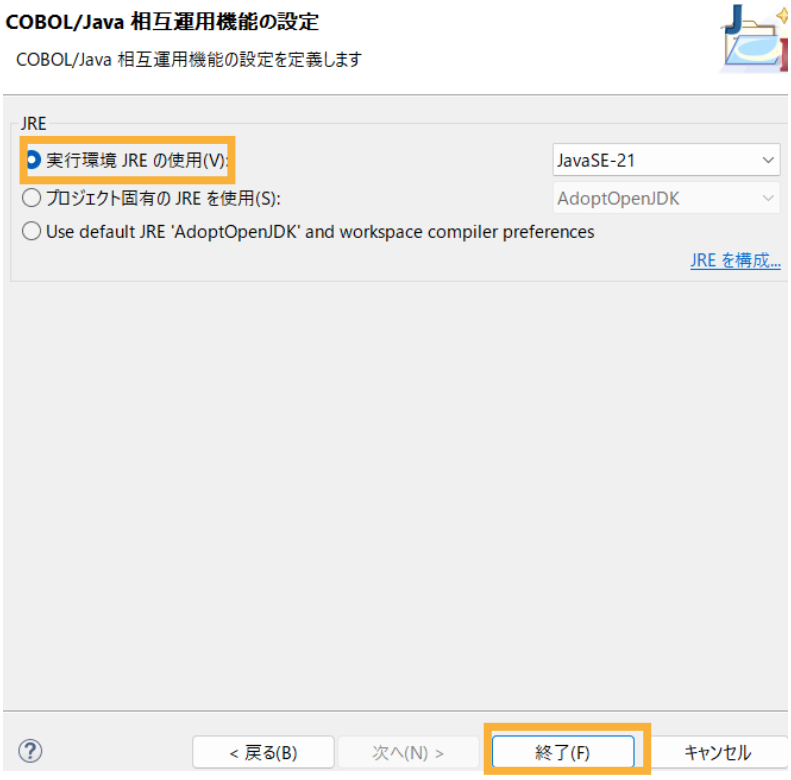
1) Windows スタートメニューより、[Rocket Visual COBOL] > [Visual COBOL for Eclipse] を選択して、Visual COBOL for Eclipse を起動します。
   ワークスペースは任意のフォルダでかまいません。以降の手順では、c:¥workspace-interoperability を使用します。
   起動後、ようこそ画面は閉じてください。
2) ワークスペースに対する文字コード設定を行います。
   5.1 の手順を実施してください。
3) [ファイル(F)] > [新規(N)] > [COBOL/Java 相互運用機能のプロジェクト] を選択します。



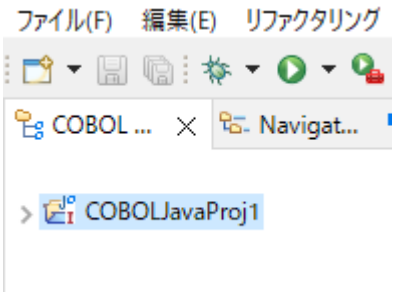
以下の入力を行い、[次へ(N)] をクリックします。
プロジェクト名： “COBOLJavaProj1”
プロジェクトテンプレート： “Rocket テンプレート(64 ビット)”



4) JRE に [実行環境 JRE の使用] を選択し、[終了(F)] をクリックします。



COBOLJavaProj1 プロジェクトが作成されます。



5)　プロジェクトに対する文字コードの設定を行います。

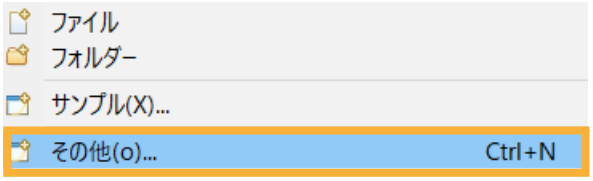
5.2 の手順を実施してください。

6)　COBOLJavaProj1 プロジェクトを選択し、マウスの右クリックによりコンテキストメニューを開き、[新規作成(N)] > [COBOL プログラム] を選択します。

そのまま、[終了(F)] をクリックします。

作成された Program1.cbl を、サンプルファイルを解凍したフォルダ内の COBOLtoJava フォルダ配下の Program1.cbl の内容で上書きしてください。

7) COBOLJavaProj1 プロジェクトを選択し、[ファイル(F)] > [新規(N)] > [その他(o)] を選択します。

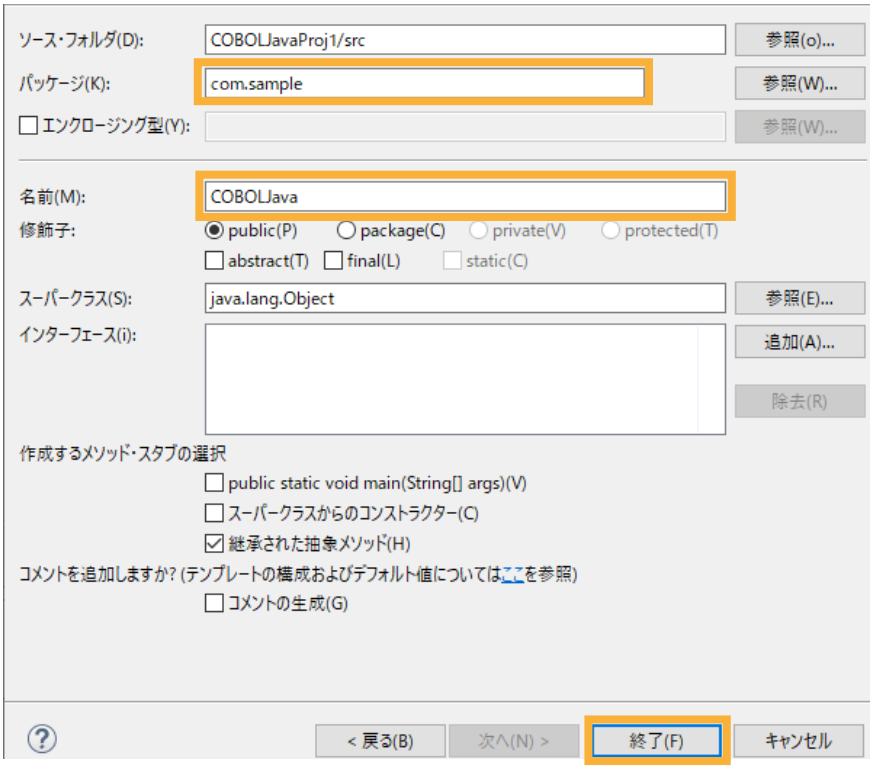


8) [Java] > [クラス] を選択して、[次へ(N)] をクリックします。

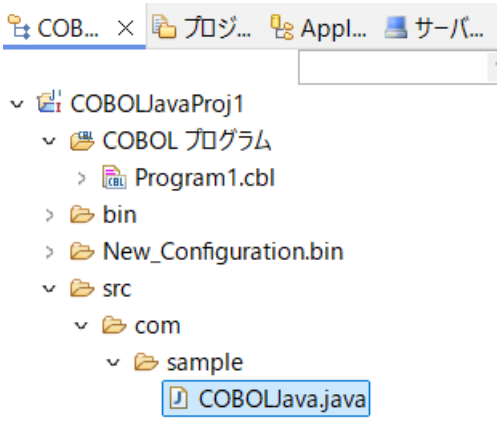
以下の入力を行ったうえで、[終了(F)] をクリックします。

パッケージ： “com.sample”

名前： “COBOLJava”

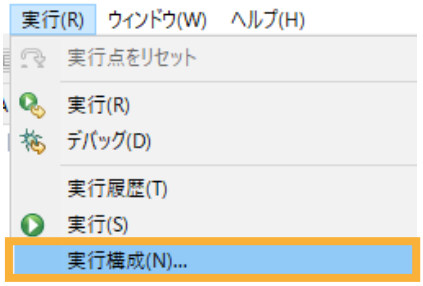


COBOLJavaProj1 プロジェクト配下に src¥com¥sample フォルダが作成され、COBOLJava.java が作成されます。

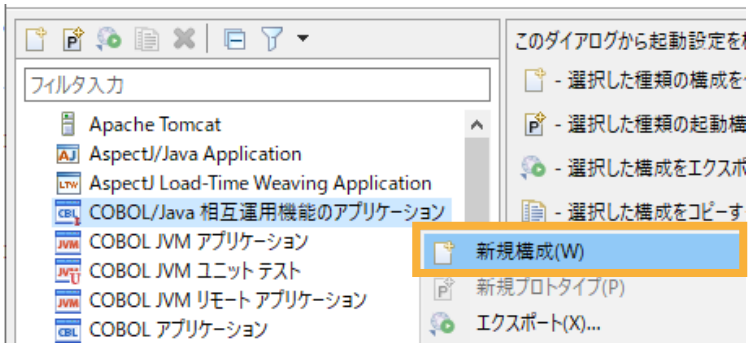


作成された COBOLJava.java を、エディター上でサンプルファイルを解凍したフォルダ内の COBOLtoJava フォルダ配下の COBOLJava.java の内容で上書き保存してください。

9) COBOLJavaProj1 プロジェクトを選択し、[実行(R)] > [実行構成(N)] を選択します。



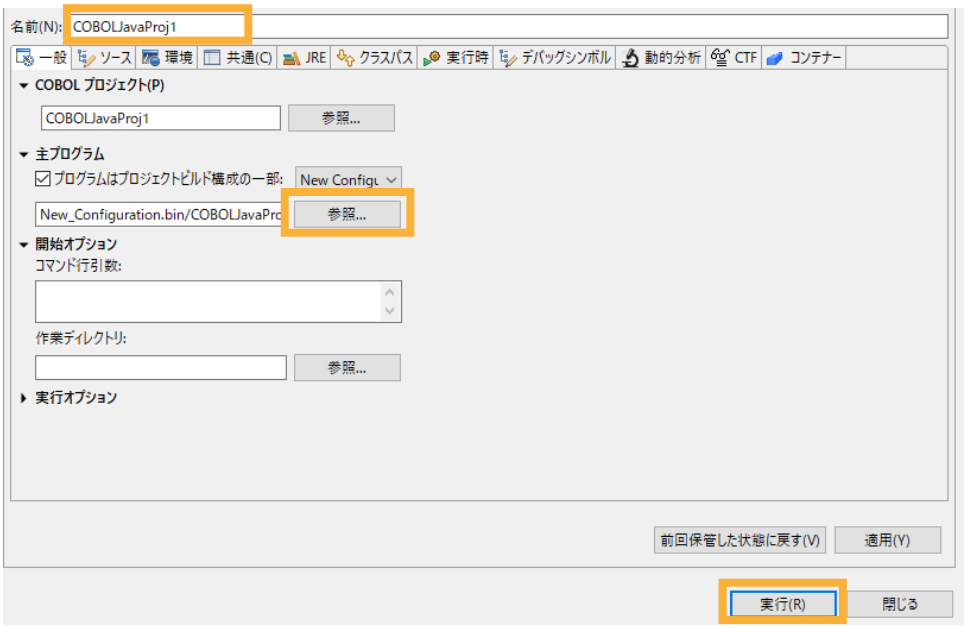
[COBOL/Java 相互運用機能のアプリケーション] を選択したうえで、マウスの右クリックによりコンテキストメニューを開き、[新規構成(W)] を選択します。



10) 以下の入力を行い、[実行(R)] をクリックします。

名前： COBOLJavaProj1

主プログラム： [参照] をクリックし、”COBOLJavaProj1.dll” を選択



COBOL から Java が呼び出され、COBOL からの情報が出力されます。続いて、Java から戻された値が COBOL から出力されます。

```
オリジナル配列順序>>>
赤
緑
青
橙
藍
Java でのソート結果>>>
橙
緑
藍
赤
青
Java でセットされた値の表示>>>
橙
黄
青
藍
紫
続行するには何かキーを押してください . . .
```

何かキーを押して、アプリケーションを終了します。

3.3.2 COBOL プロジェクトから外部の Java 資産の利用

前項では、COBOL 資産と Java 資産を同じプロジェクト内に配置しました。しかし、同時に開発を行わない限り、一般的には Java 資産は別なフォルダ、すなわち、プロジェクト外に保存されます。

本項では、COBOL, Java の２つのプロジェクトを使用した COBOL/Java 相互運用機能を利用する方法を紹介します。

注意）

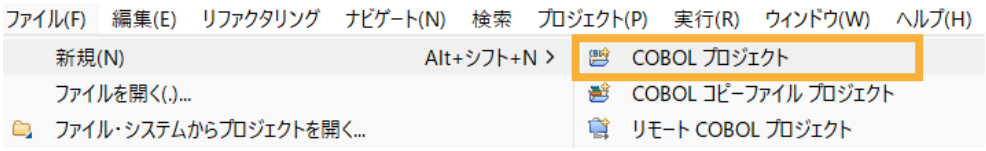
以降の手順で参照する Java クラスは 3.3.1 で作成したものです。こちらの手順の実施前に、3.3.1 を実施してください。

1) Visual COBOL for Eclipse の起動

Windows スタートメニューより、[Rocket Visual COBOL] > [Visual COBOL for Eclipse] を選択して、Visual COBOL for Eclipse を起動します。

ワークスペースは、さきほどの c:¥workspace-interoperability を使用します。

2) [ファイル(F)] > [新規(N)] > [COBOL プロジェクト] を選択します。

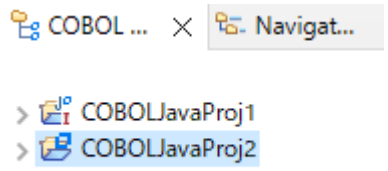


以下の入力を行い、[終了(F)] をクリックします。

プロジェクト名： “COBOLJavaProj2”

プロジェクトテンプレート： “Rocket テンプレート(64 ビット)”

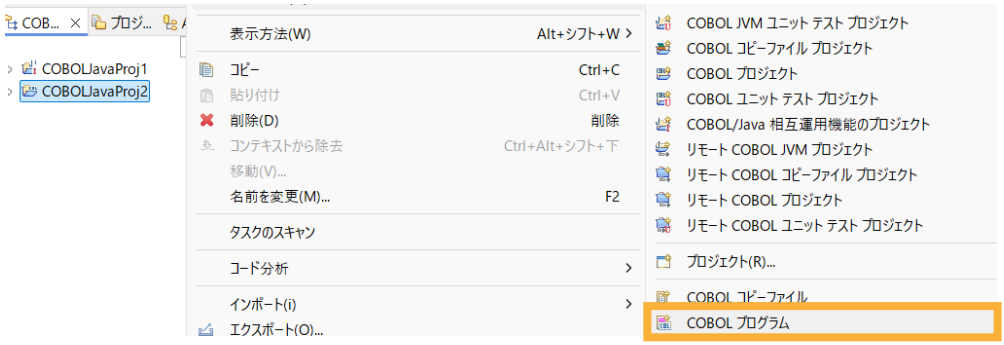
COBOLJavaProj2 プロジェクトが作成されます。



3) プロジェクトに対する文字コード設定を行います。

5.2 の手順を実施してください。

4) COBOLJavaProj2 プロジェクトを選択し、マウスの右クリックによりコンテキストメニューを開き、[新規作成(N)] > [COBOL プログラム] を選択します。

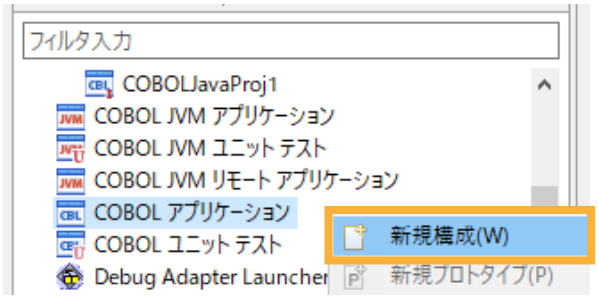


そのまま、[終了(F)] をクリックします。

作成された Program1.cbl を、エディター上でサンプルファイルを解凍したフォルダ内の COBOLtoJava フォルダ配下の Program1.cbl の内容で上書き保存してください。

5) [実行(R)] > [実行構成(N)] を選択します。

[COBOL アプリケーション] を選択したうえで、マウスの右クリックによりコンテキストメニューを開き、[新規構成(W)] を選択します。



6) 以下の入力を行い、[実行(R)] をクリックします。

名前： “COBOLJavaProj2”

[環境] タブを選択

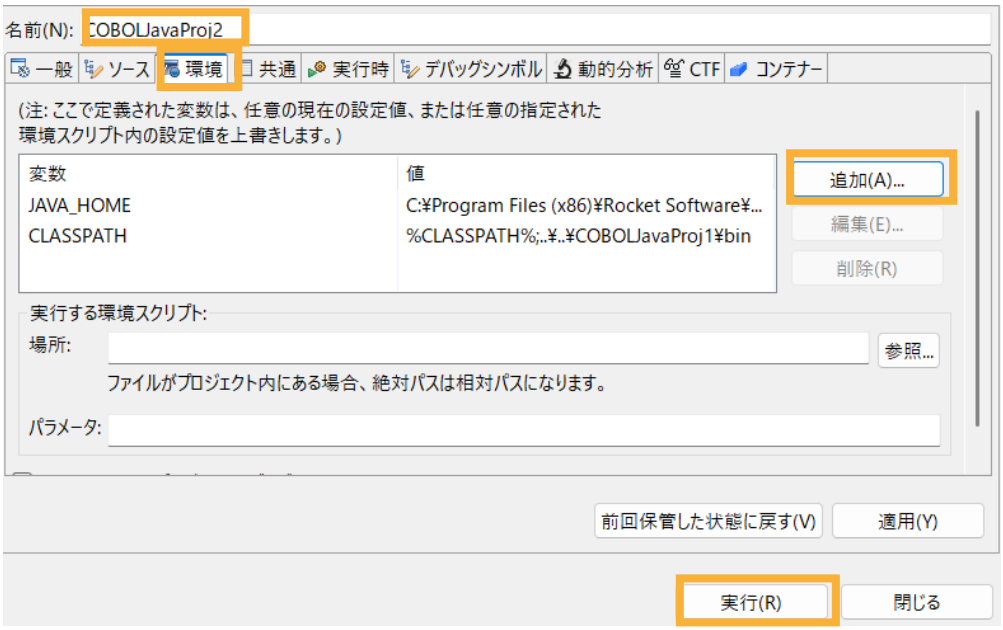
[追加(A)] をクリックし、以下の環境変数を追加します。

- JAVA_HOME

  “C:¥Program Files (x86)¥Rocket Software¥Visual COBOL¥AdoptOpenJDK”

- CLASSPATH

  “%CLASSPATH%;..¥..¥COBOLJavaProj1¥bin”



注意）
上記の JAVA_HOME 環境変数は、製品のデフォルトインストールした場合のパスです。
インストール先を変更した場合や、3.3.1 の手順で JRE 環境の設定を変更した場合は、設定した環境を指定してください。

3.3.1 と同じ結果が戻されます。

```
オリジナル配列順序>>>
赤
緑
青
橙
藍
Java でのソート結果>>>
橙
緑
藍
赤
青
Java でセットされた値の表示>>>
橙
黄
青
藍
紫
続行するには何かキーを押してください . . .
```

なにかキーを押して、アプリケーションを終了します。

3.4 COBOL アプリケーションの実行環境を Java 仮想マシンに移して Java 資産とともに Java として利用

この方法は、製品が提供する JVM COBOL 機能を利用します。別途チュートリアルが提供されていますので、以下のチュートリアルを参照ください。

https://www.amc.rocketsoftware.co.jp/manuals/VC100/Eclipse/index.html?t=GUID-D10DC512-FDEF-44CF-8A9B-32839729B493.html

Visual COBOL 11.0 for Eclipse のチュートリアルトップからは、以下のように進んでください。

[ここからはじめよう] > [Getting Started] > [JVM COBOL チュートリアル]

4 Java から COBOL 資産を呼び出す

Java から COBOL 資産を呼び出す代表的な方法は以下になります。

- Runtime.exec() を利用して COBOL プログラムを呼び出す
- 製品に付属する COBOL 専用のアプリケーションサーバーを利用して、COBOL 資産をサービスとして利用
- COBOL/Java 相互運用機能を利用
- COBOL アプリケーションの実行環境を Java 仮想マシンに移して Java 資産とともに Java として運用

4.1 Runtime.exec() を利用して COBOL プログラムを呼び出す

Java は、別なプロセスを起動するための API として、Runtime クラスの exec メソッドを提供しています。最も簡単な COBOL プログラムの起動方法は、このメソッドの利用です。例えば、以下のプログラムでは、MyCOBOLApp.exe モジュールを実行します。

```
public class SampleMain {
    public static void main(String[] args) {
        Runtime r = Runtime.getRuntime();
        try {
            r.exec("MyCOBOLApp.exe", args);
        } catch (java.io.IOException e) {
            e.printStackTrace();
        }
    }
}
```

補足）

この方法は、COBOL に限らず、任意のプログラムを実行することができます。なお、上記方法では、実行時に COBOL アプリケーションが実行できる環境設定が行われている必要があります。

しかし、起動する COBOL プログラムへ情報を渡そうとした場合、exec() メソッドの第二引数、上記サンプルでは args の使用、もしくは、ファイルやデータベースの利用などが必要となるため、単純なプログラム起動以外には適しません。

以降に紹介する方法では、このような問題を解決する方法を紹介します。

4.2 製品に付属する COBOL 専用のアプリケーションサーバーを利用して、COBOL 資産をサービスとして利用

Visual COBOL 製品には、COBOL 専用のアプリケーションサーバー機能が提供されており、このサーバー上で COBOL 資産を容易にサービスとして運用することができます。このサービスは、Eclipse IDE 上で、サービスの新規開発からテスト、デプロイまで作業を完結できます。

こちらの方法は、別途チュートリアルが提供されていますので、以下のチュートリアルを参照ください。

https://www.amc.rocketsoftware.co.jp/manuals/CMN/MFVC_1100_ECLWSVC01.pdf

Visual COBOL 11.0 for Eclipse のチュートリアルトップからは、以下のように進んでください。

[ここからはじめよう] > [Getting Started] > [ネイティブ COBOL チュートリアル] > [Interface Mapping Toolkit - RESTful Web サービスによる COBOL 資産の再利用]

4.3 COBOL/Java 相互運用機能を利用

COBOL/Java 相互運用機能を利用することで、COBOL 資産のサービス化を行うことなく、Java から COBOL 資産を呼び出すことができます。

4.3.1 COBOL/Java 相互運機能のプロジェクトを利用

1) Visual COBOL for Eclipse の起動

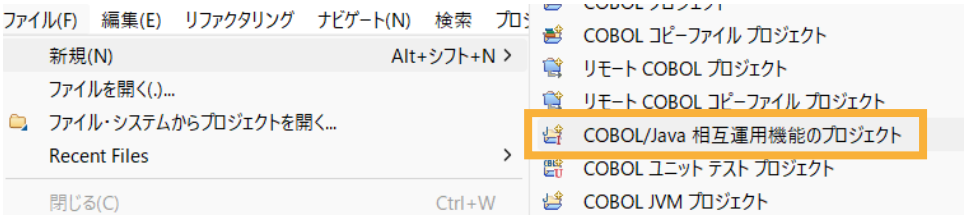
Windows スタートメニューより、[Rocket Visual COBOL] > [Visual COBOL for Eclipse] を選択して、Visual COBOL for Eclipse を起動します。

ワークスペースは任意のフォルダでかまいません。以降の手順では、c:¥workspace-interoperability を使用します。

2) ワークスペースに対する文字コードの設定を行います。

5.1 の手順を実施してください。

3) [ファイル(F)] > [新規(N)] > [COBOL/Java 相互運用機能のプロジェクト] を選択します。

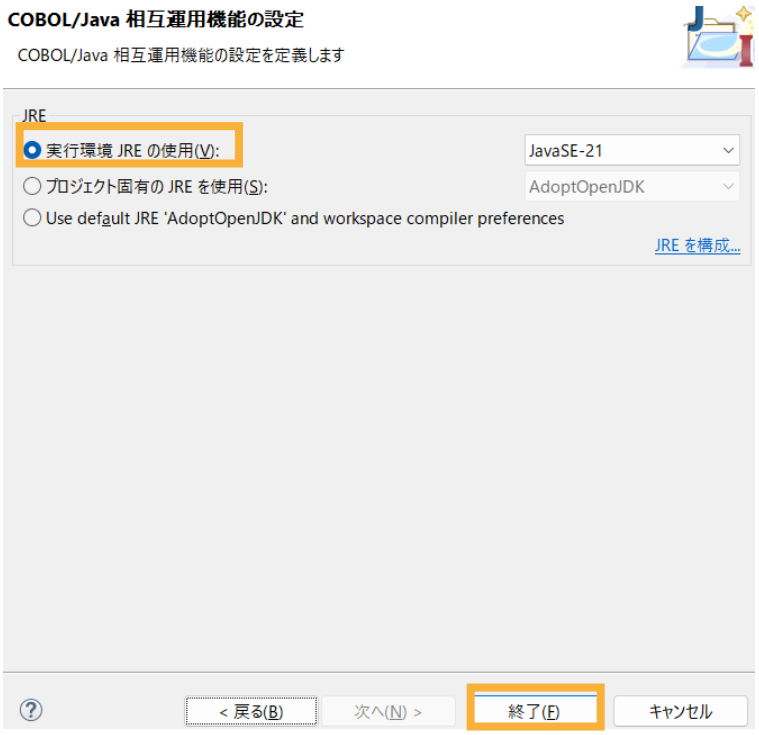


以下の入力を行い、[次へ(N)] をクリックします。

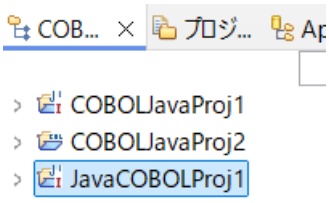
プロジェクト名： “JavaCOBOLProj1”

プロジェクトテンプレート： “Rocket テンプレート(64 ビット)”

4) JRE に [実行環境 JRE の使用] を選択し、[終了(F)] をクリックします。



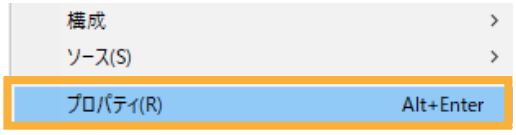
JavaCOBOLProj1 プロジェクトが作成されます。



5) プロジェクトに対する文字コード設定を行います。

5.2 の手順を実施してください。

6) JavaCOBOLProj1 プロジェクトを選択し、マウスの右クリックによりコンテキストメニューを開き、[プロパティ(R)] を選択します。



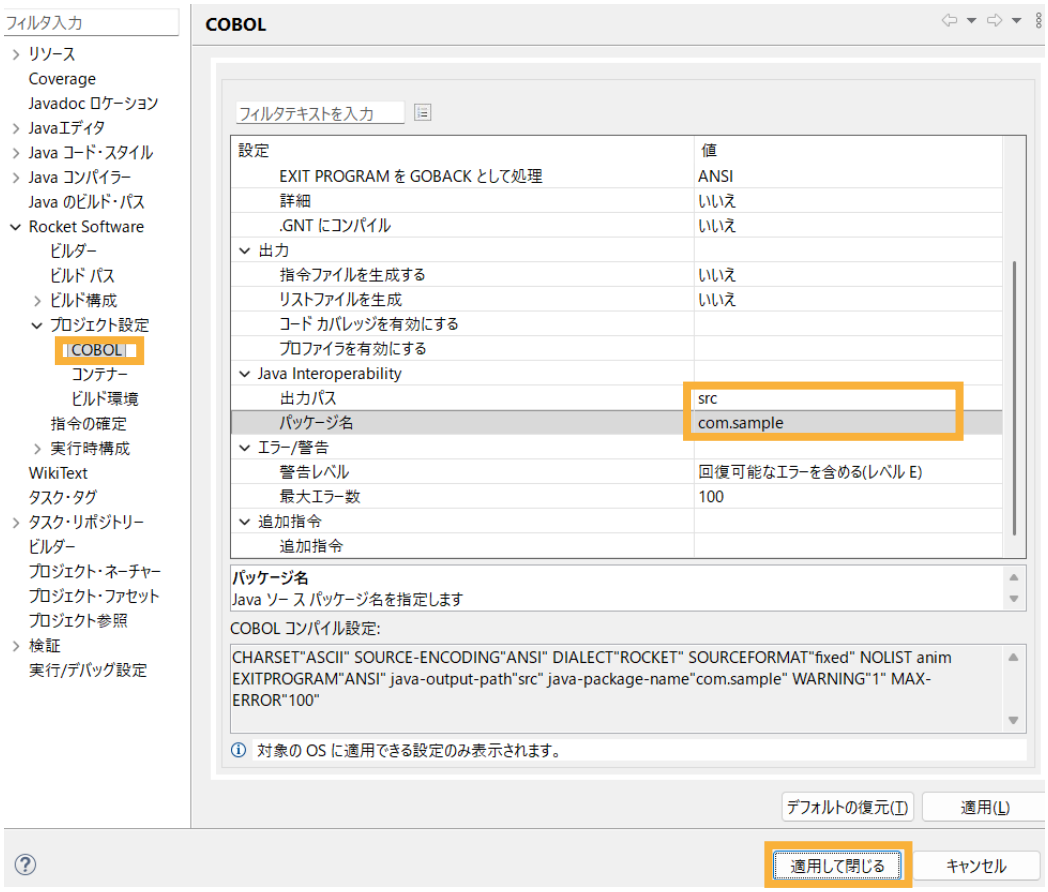
左側のツリーより [Rocket Software] > [プロジェクト設定] > [COBOL] を選択し、以下の選択を行ったうえで、[適用して閉じる] をクリックします。

出力パス： “src”

パッケージ名： “com.sample”

補足）

出力パス “src” が、Java プログラムのソースフォルダになります。このフォルダ配下に、COBOL プログラムにアクセスするためのラッパープログラムが生成されます。

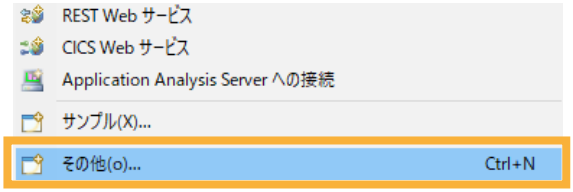


7) JavaCOBOLProj1 プロジェクトを選択し、マウスの右クリックによりコンテキストメニューを開き、[新規作成(N)] > [COBOL プログラム] を選択します。

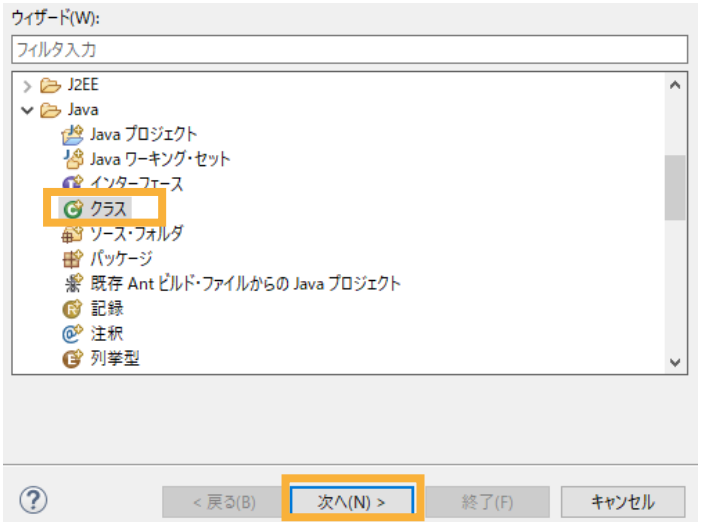
そのまま、[終了(F)] をクリックします。

作成された Program1.cbl を、サンプルファイルを解凍したフォルダ内の JavaToCOBOL フォルダ配下の Program1.cbl の内容で上書きしてください。

8) JavaCOBOLProj1 プロジェクトを選択の上、[ファイル(F)] > [新規(N)] > [その他(o)] を選択します。



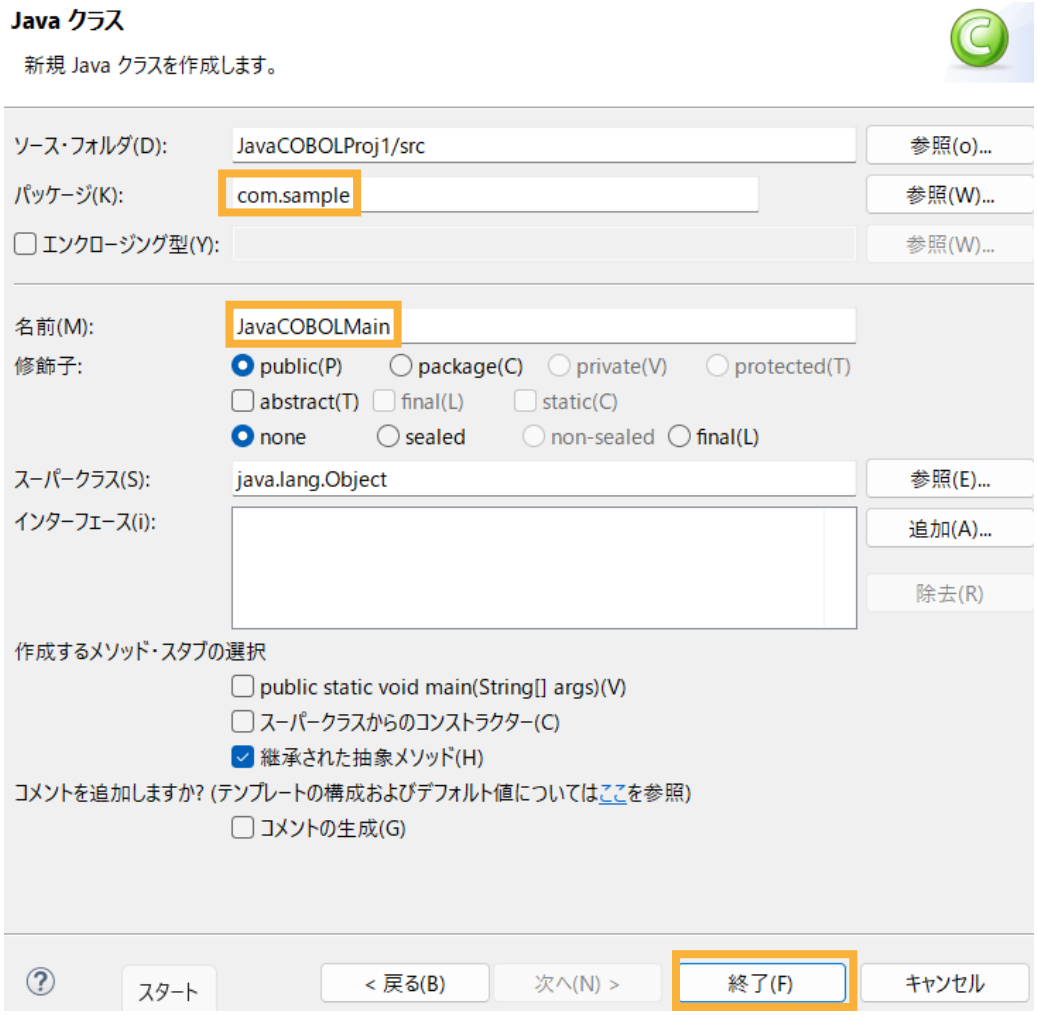
9) [Java] > [クラス] を選択して、[次へ(N)] をクリックします。



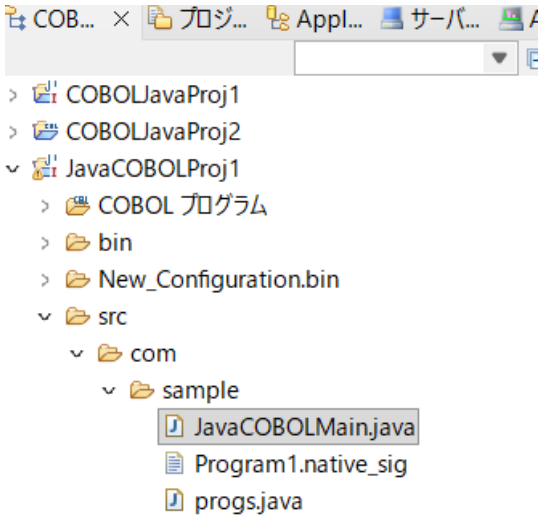
以下の入力を行ったうえで、[終了(F)] をクリックします。

パッケージ： “com.sample”

名前： “JavaCOBOLMain”

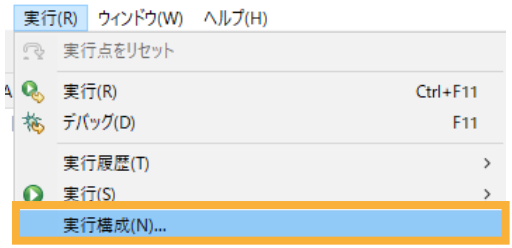


JavaCOBOLProj1 プロジェクト配下の　src¥com¥sample の下に JavaCOBOLMain.java が作成されます。



作成された JavaCOBOLMain.java を、エディター上でサンプルファイルを解凍したフォルダ内の JavatoCOBOL フォルダ配下の JavaCOBOLMain.java の内容で上書き保存してください。

10) JavaCOBOLProj1 プロジェクトを選択し、[実行(R)] > [実行構成(N)] をクリックします。

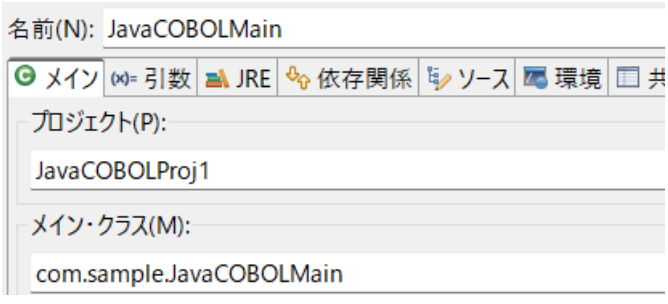


[Java アプリケーション] を選択したうえで、マウスの右クリックによりコンテキストメニューを開き、[新規構成(W)] を選択します。

11) 以下の入力を行います。

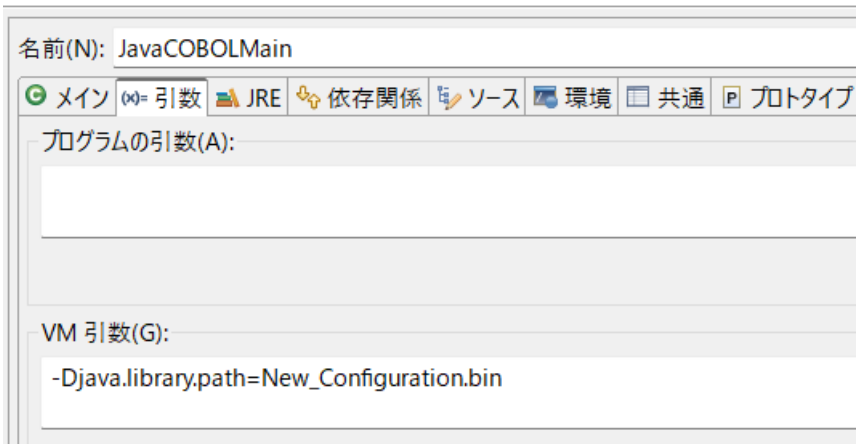
名前： “JavaCOBOLMain”

メイン・クラス： “com.sample.JavaCOBOLMain”



[引数] タブを選択

VM 引数： “-Djava.library.path=New_Configuration.bin”



12) [実行(R)] をクリックします。

以下の結果がコンソールビューに表示されます。

```
COBOL 0000000001 青
COBOL 0000000002 黄
COBOL 0000000003 赤
COBOL 0000000004 緑
Prime number from Java
2
3
5
7
11
13
17
19
23
29
```

このサンプルでは、Java から色名称の配列を COBOL に渡し、COBOL 側で出力しています。

COBOL からは素数のリストを Java に戻し、その結果を Java 側で出力しています。

4.3.2 COBOL と Java を別プロジェクトで利用

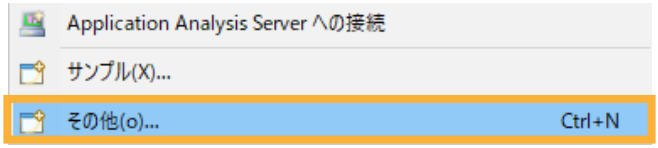
前項では、Java 資産と COBOL 資産を同じプロジェクト内に配置しました。しかし、同時開発を行わない限り、別々に管理されることが一般的です。本項では、COBOL, Java の 2 つのプロジェクトを使用した COBOL/Java 相互運用機能を利用する方法を紹介します。

1) Visual COBOL for Eclipse の起動

Windows スタートメニューより、[Rocket Visual COBOL] > [Visual COBOL for Eclipse] を選択して、Visual COBOL for Eclipse を起動します。

ワークスペースは、さきほどの c:¥workspace-interoperability を使用します。

2) [ファイル(F)] > [新規(N)] > [その他(o)] を選択します。



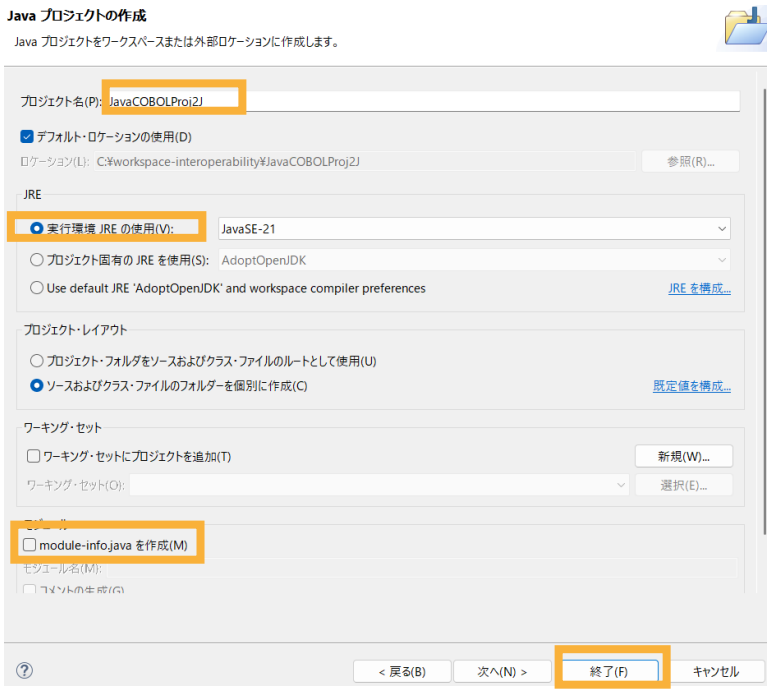
3) [Java] > [Java プロジェクト] を選択したうえで [次へ] をクリックします。

4) 以下の入力を行い、[終了(F)] をクリックします。

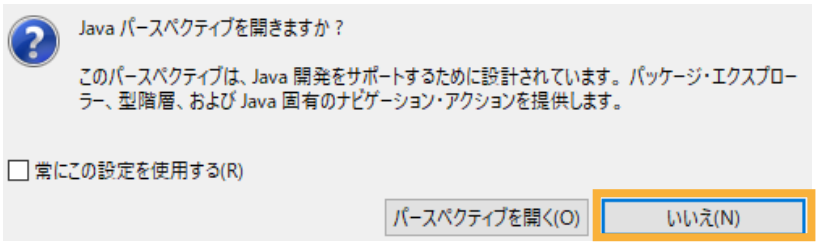
プロジェクト名： “JavaCOBOLProj2J”

[実行環境 JRE の使用] を選択

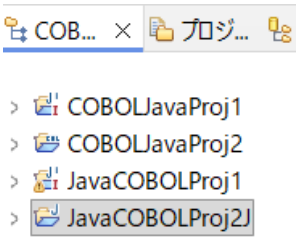
[module-info,java を作成] のチェックを外す



パースペクティブの切り替えダイアログでは、[いいえ(N)] をクリックします。

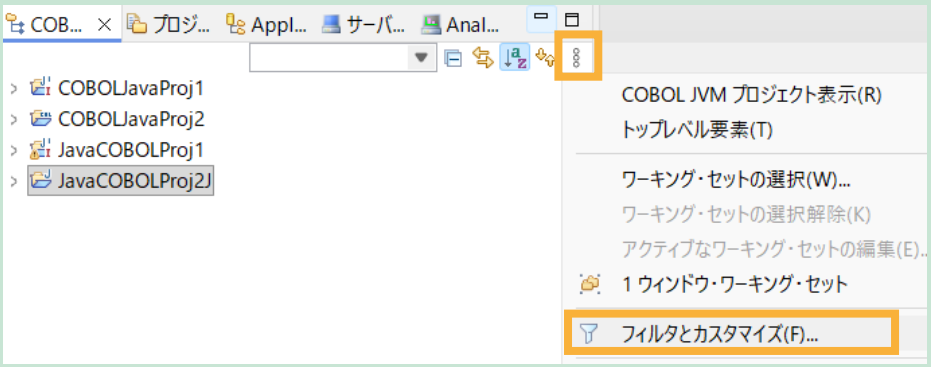


JavaCOBOLProj2J プロジェクトが作成されます。

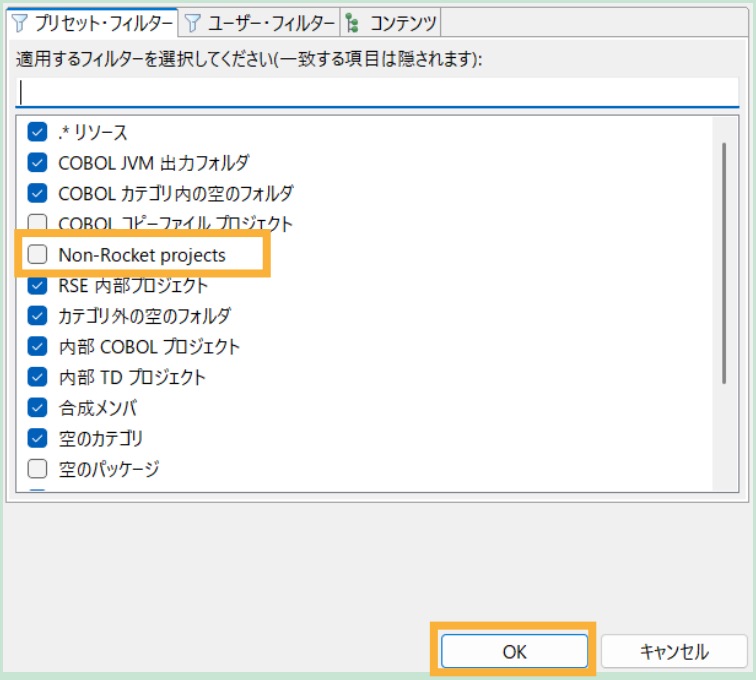


JavaCOBOLProj2J プロジェクトが表示されない場合）

エクスプローラービューの右上をクリックし、[フィルタとカスタマイズ(F)] をクリックします。

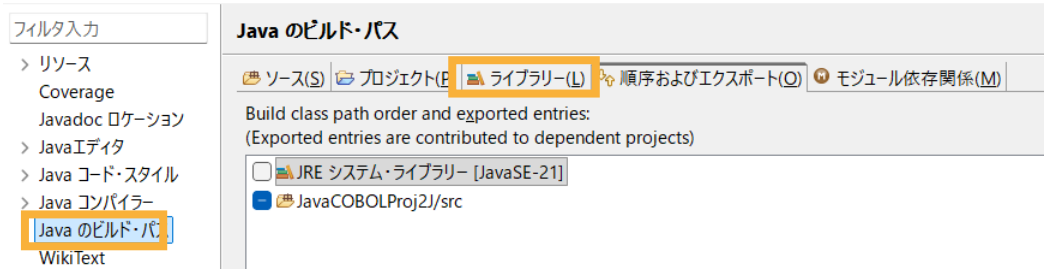


[Non-Rocket Projects] のチェックを外したうえで、[OK] をクリックします。

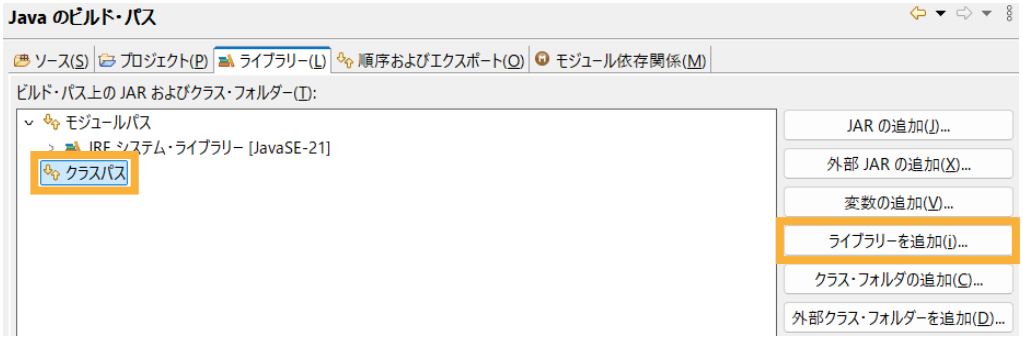


5) JavaCOBOLProj2J プロジェクトを選択したうえで、マウスの右クリックによりコンテキストメニューを開き、[プロパティ(R)] を選択します。

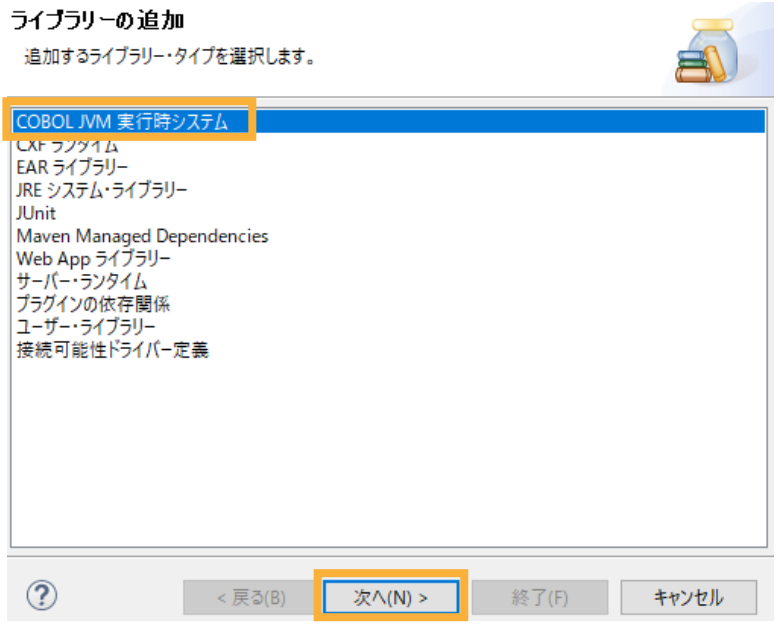
左側のツリーより [Java のビルド・パス] を選択し、[ライブラリー(L)] を選択します。



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

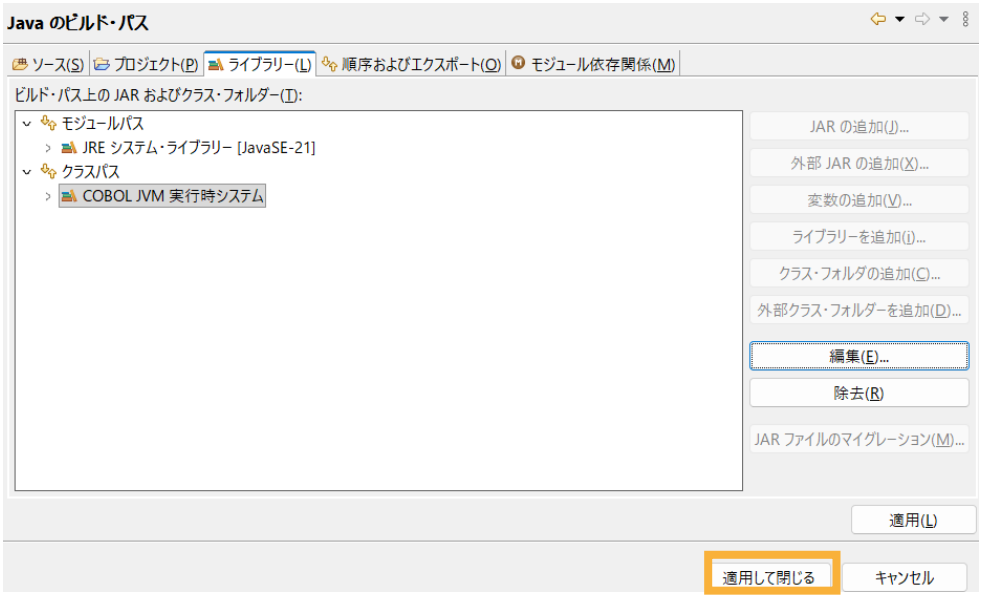


[COBOL JVM 実行時システム] を選択し、[次へ(N)] をクリックします。



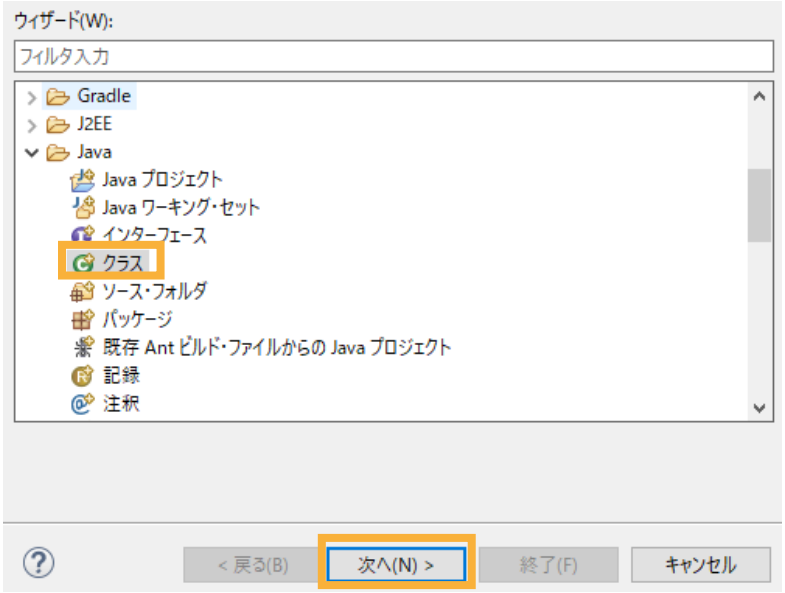
そのまま、[終了(F)] をクリックします。

“COBOL JVM 実行時システム” が追加されたことを確認したうえで、[適用して閉じる] をクリックします。



6) JavaCOBOLProj2J プロジェクトを選択したうえで、[ファイル(F)] > [新規(N)] > [その他(o)] を選択します。

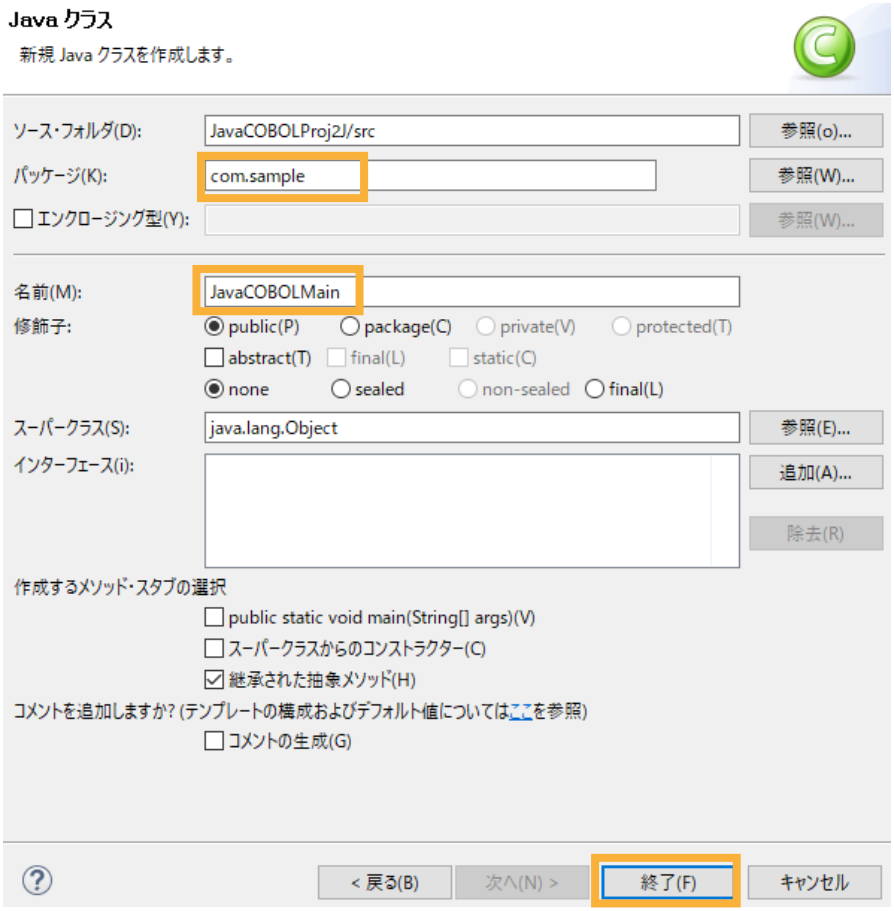
[Java] > [クラス] を選択し、[次へ(N)] をクリックします。



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.sample”

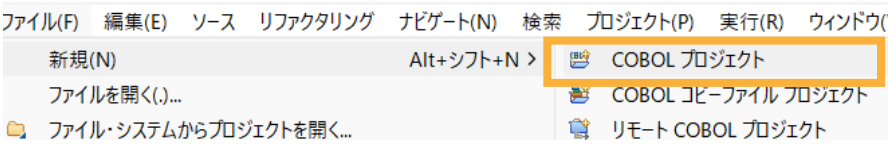
名前： “JavaCOBOLMain”



作成された JavaCOBOLMain.java を、サンプルファイルを解凍したフォルダ内の JavatoCOBOL フォルダ配下の JavaCOBOLMain.java の内容で上書きしてください。

この時点では、9 行目がエラーとなりますが無視してください。

7) [ファイル(F)] > [新規(N)] > [COBOL プロジェクト] を選択します。

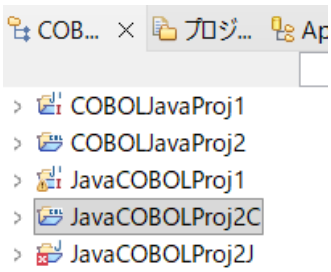


以下の入力を行い、[終了(F)] をクリックします。

プロジェクト名： “JavaCOBOLProj2C”

プロジェクトテンプレート： “Rocket テンプレート(64 ビット)”

JavaCOBOLProj2C プロジェクトが作成されます。



8) プロジェクトに対する文字コード設定を行います。

5.2 の手順を実施してください。

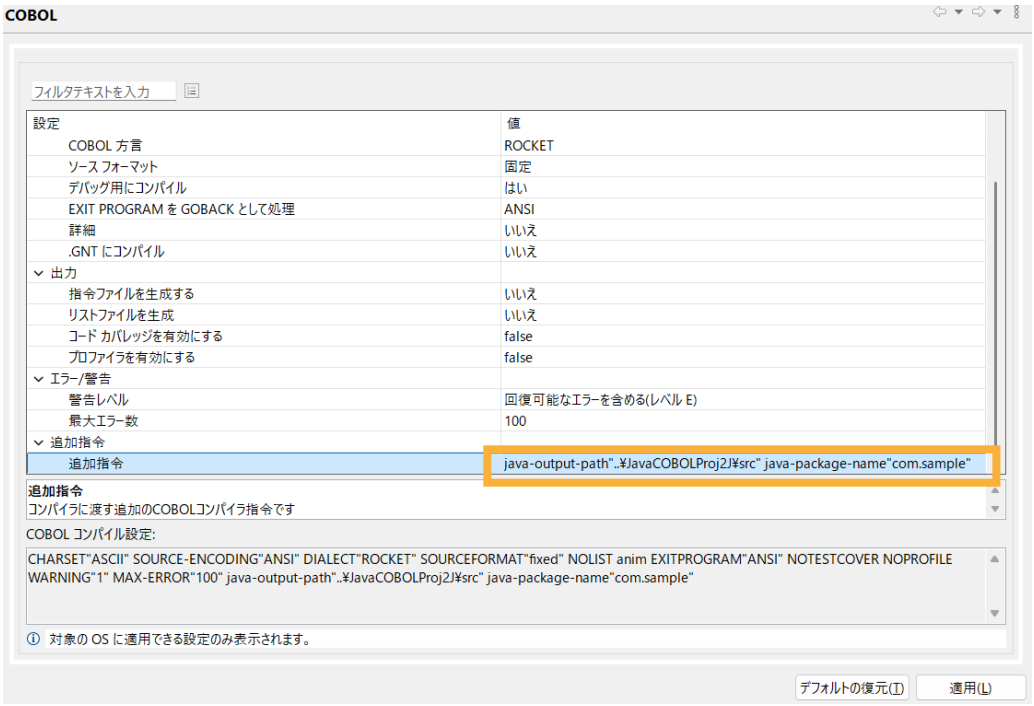
9) JavaCOBOLProj2C プロジェクトを選択し、マウスの右クリックによりコンテキストメニューを開き、[プロパティ(R)] を選択します。

以下の設定を行ったうえで、[適用(L)] をクリックします。

[Rocket Software] > [プロジェクト設定] > [COBOL]　を選択

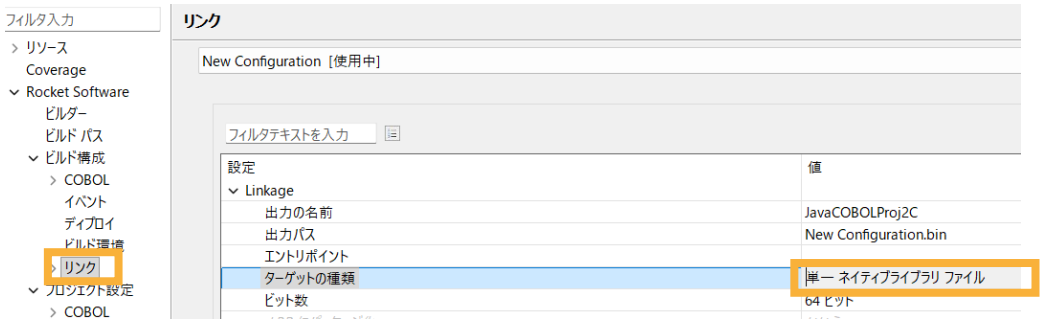
追加指令：半角スペースをデリミタとして、以下の２つを入力

- java-output-path"..¥JavaCOBOLProj2J¥src"
- java-package-name"com.sample"



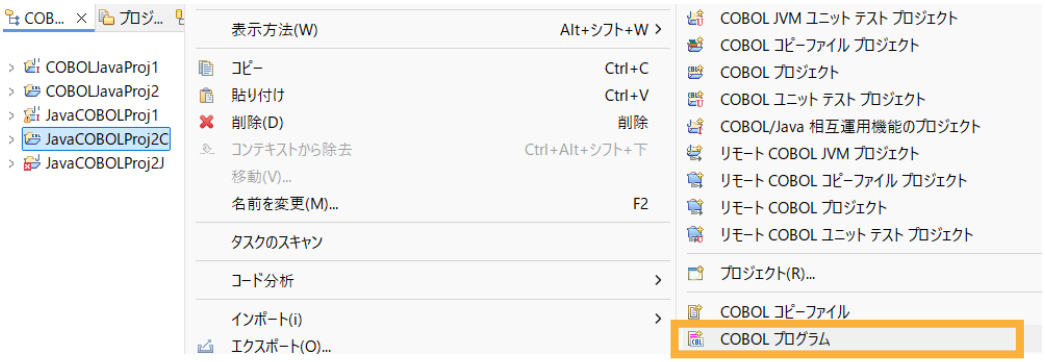
[Rocket Software] > [ビルド構成] > [リンク] を選択

ターゲットの種類： “単一ネイティブライブラリファイル”



10) [適用して閉じる] をクリックします。

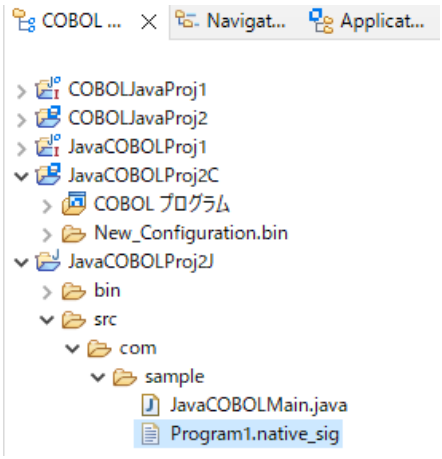
11) JavaCOBOLProj2C プロジェクトを選択し、マウスの右クリックによりコンテキストメニューを開き、[新規作成(N)] > [COBOL プログラム] を選択します。



そのまま、[終了(F)] をクリックします。

作成された Program1.cbl を、サンプルファイルを解凍したフォルダ内の JavatoCOBOL フォルダ配下の Program1.cbl の内容で上書きしてください。

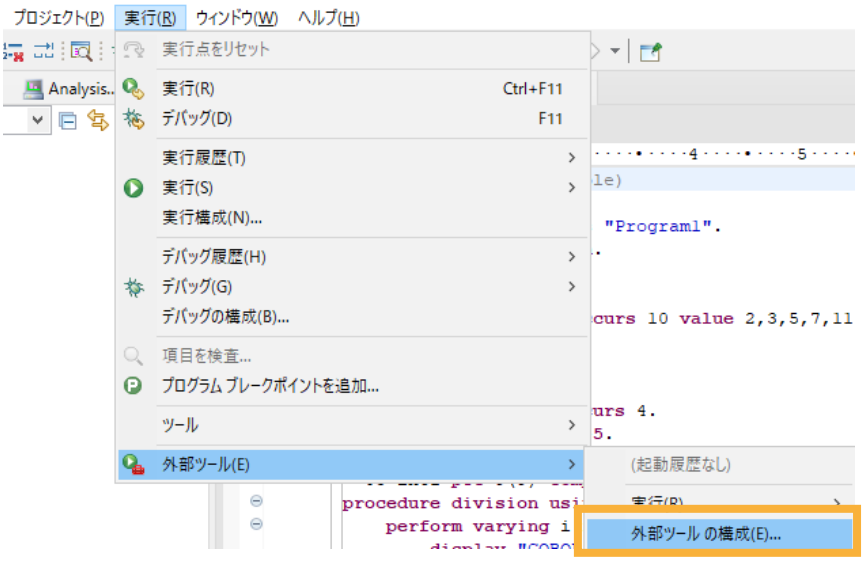
自動でビルドが行われ、前手順で指定した追加指令によって、JavaCOBOLProj2J プロジェクト配下の src¥com¥sample フォルダ配下に Program1.native_sig が作成されます。



補足）

表示されない場合は、src¥com¥sample フォルダを選択し、[ファイル(F)] > [更新(F)] を選択してください。

12) COBOL 呼出しに必要なラッパープログラムを生成するため、[実行(R)] > [外部ツール(E)] > [外部ツールの構成(E)] を選択します。



13) [プログラム]をダブルクリックしたうえで、以下の入力を行い、[実行(R)] をクリックします。

名前： “genjava-for-JavaCOBOLProj2C”

ロケーション：

“C:¥Program Files (x86)¥Rocket Software¥Visual COBOL¥bin64¥genjava.exe”

作業ディレクトリー：

“C:¥workspace-interoperability¥JavaCOBOLProj2J¥src”

引数：

“JavaCOBOLProj2C -p Program1 -k com.sample”

補足）

ロケーション

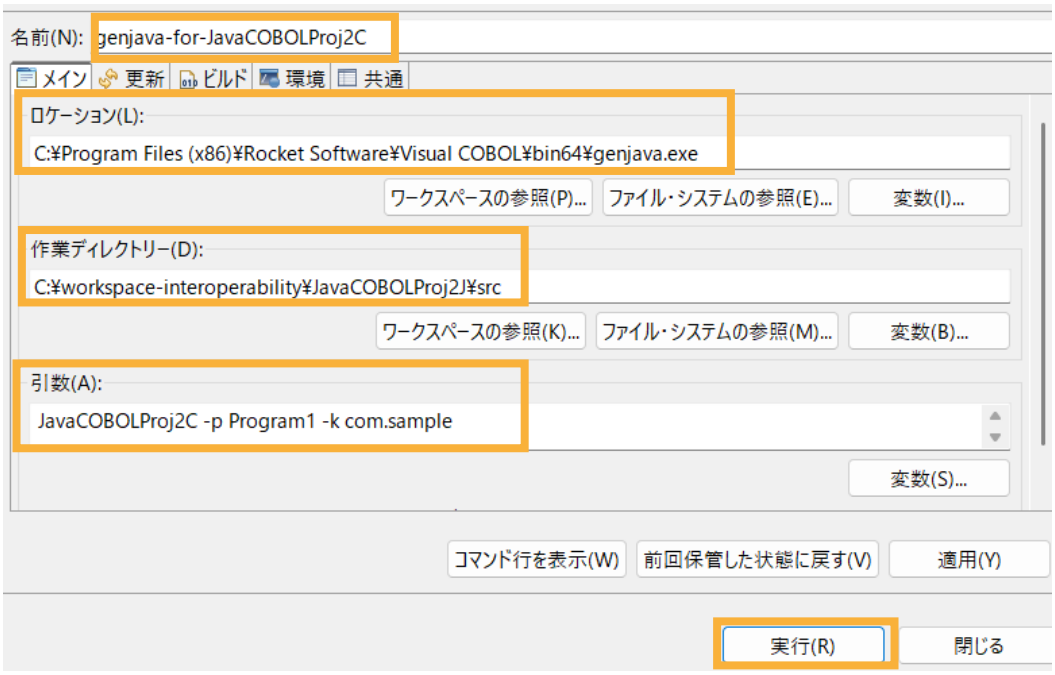
上記で指定している genjava.exe は、Visual COBOL 製品のインストール先がデフォルトの場合となります。異なるフォルダにインストールした場合は、<製品インストールフォルダ>¥bin64¥genjava.exe を指定してください。

作業ディレクトリー

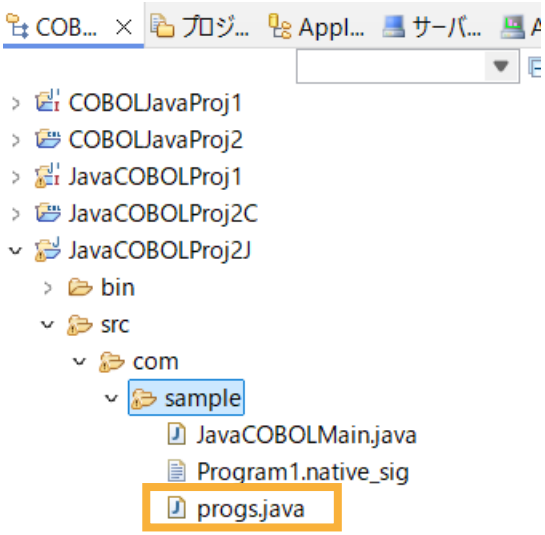
さきほど作成した JavaCOBOLProj2J プロジェクト配下の src フォルダまでの絶対パスを指定してください。

引数

最初に指定している JavaCOBOLProj2C は、さきほど作成した COBOL プロジェクト“JavaCOBOLProj2C” の成果物である JavaCOBOLProj2C.dll を指定しています。



JavaCOBOLProj2J プロジェクト配下の src¥com¥sample 配下を更新すると、progs.java が生成されます。



14) JavaCOBOLProj2J プロジェクトを選択したうえで、[実行(R)] > [実行構成(N)] を選択します。

[Java アプリケーション] をダブルクリックし、以下の入力を行います。

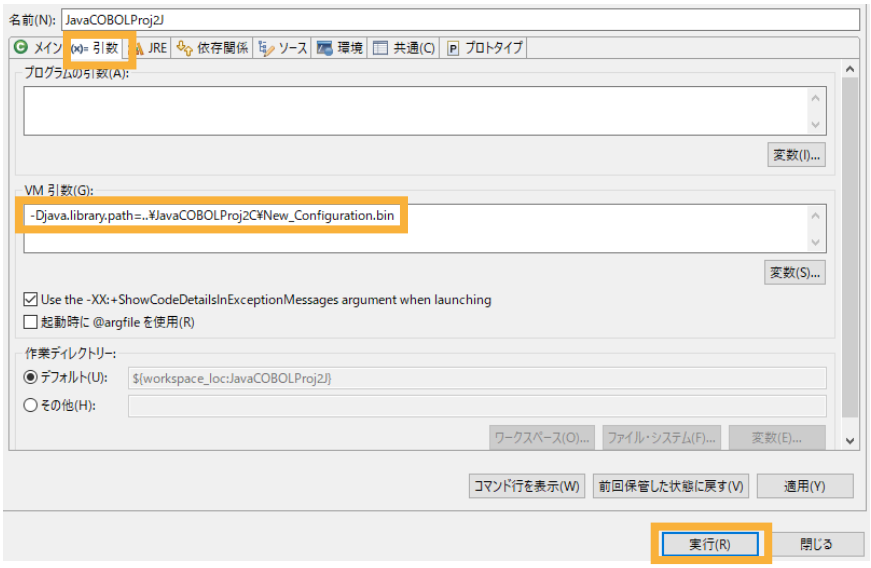
名前： “JavaCOBOLProj2J”

メイン・クラス： “com.sample.JavaCOBOLMain”

[引数] タブを選択

VM 引数：

“-Djava.library.path=..¥JavaCOBOLProj2C¥New_Configuration.bin”



15) [実行(R)] をクリックします。

4.3.1 と同様の結果がコンソールビューに表示されます。

```
COBOL 0000000001 青
COBOL 0000000002 黄
COBOL 0000000003 赤
COBOL 0000000004 緑
Prime number from Java
2
3
5
7
11
13
17
19
23
29
```

4.4 COBOL アプリケーションの実行環境を Java 仮想マシンに移して Java 資産とともに Java として運用

この運用方法は、製品が提供する JVM COBOL 機能を利用します。別途チュートリアルが提供されていますので、以下のチュートリアルを参照ください。

https://www.amc.rocketsoftware.co.jp/manuals/VC110/Eclipse/index.html?t=GUID-D10DC512-FDEF-44CF-8A9B-32839729B493.html

Visual COBOL 11.0 for Eclipse のチュートリアルトップからは、以下のように進んでください。

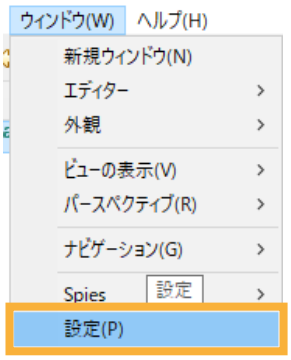
[ここからはじめよう] > [Getting Started] > [JVM COBOL チュートリアル]

5 Visual COBOL for Eclipse 上の文字コード設定について

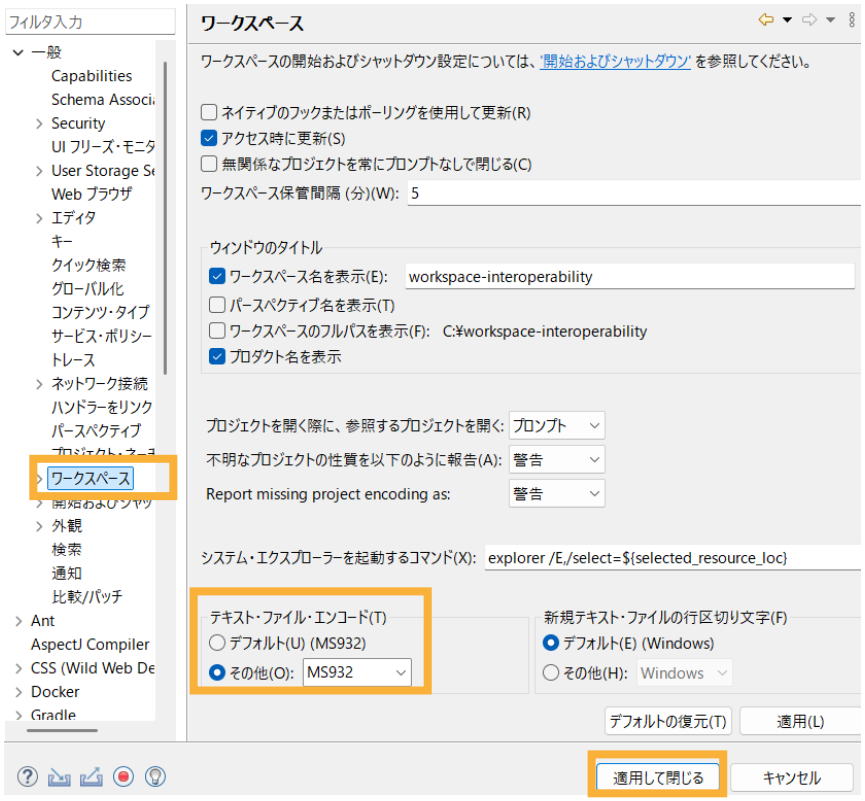
Eclipse IDE のバージョンによっては、文字コードのデフォルトに UTF-8 が採用されていますが、COBOL 資産は長年利用されていることから、多くは UTF-8 ではなく SJIS が採用されています。文字コード設定は、ワークスペース全体の設定と、プロジェクト毎の設定の 2 つがあります。これらの設定と、プログラムファイルの文字コードに不整合があると、文字化けの原因となります。本チュートリアルで使用するサンプルファイルは、文字コード SJIS を採用しているため、Eclipse IDE 上で SJIS 資産を正しく扱うための設定手順について紹介します。

5.1 ワークスペースに対する文字コード設定

1) Visual COBOL for Eclipse を起動したうえで、 [ウィンドウ(W)] > [設定(P)] をクリックします。



2) [一般] > [ワークスペース] を選択し、[テキスト・ファイル・エンコード] に “MS932” が選択されているかを確認してください。異なる値が設定されている場合は、”MS932” を選択したうえで、[OK] をクリックします。



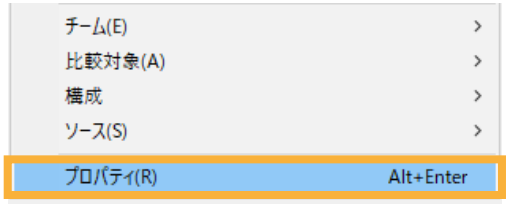
補足）

環境によっては、Windows-31J と表示されることがあります。その場合は、MS932 を Windows-31J に読み替えてください。

Preference Recorder のダイアログが表示された場合は、[Recorder enabled] のチェックを外し、[キャンセル] をクリックします。

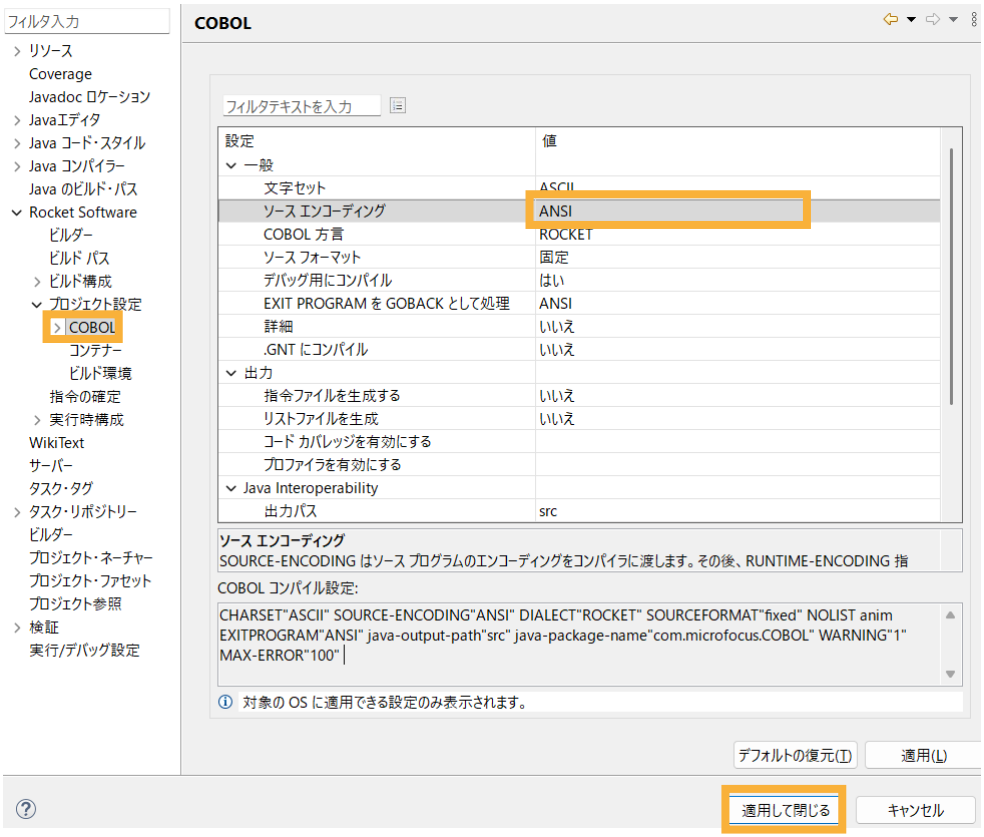
5.2 プロジェクトに対する設定

1) エクスプローラービュー上で、対象のプロジェクトを選択し、マウスの右クリックによりコンテキストメニューを開き、[プロパティ(R)] を選択します。



2) [Rocket Software] > [プロジェクト設定] > [COBOL] を選択し、以下の選択を行ったうえで、[適用して閉じる] をクリックします。

ソース エンコーディング： “ANSI”



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。