
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 Server 2019
- Adoptium OpenJDK17
- Visual COBOL 9.0J 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

- 1 目的
- 2 前提
- 3 COBOL から Java 資産を呼び出す
 - 3.1 SYSTEM ルーチンを利用した Java プログラムを起動
 - 3.2 Java 資産をサービスとして運用し、COBOL から利用
 - 3.2.1 前提条件
 - 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
 - 3.2.3 クライアントプログラムの動作確認
 - 3.3 COBOL/Java 相互運用機能を利用
 - 3.3.1 COBOL/Java 相互運用機能のプロジェクトの利用
 - 3.3.2 COBOL プロジェクトから外部の Java 資産の利用
 - 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
- 4 Java から COBOL 資産を呼び出す
 - 4.1 Runtime.exec() を利用して COBOL プログラムを呼び出す
 - 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
 - 4.3 COBOL/Java 相互運用機能を利用
 - 4.3.1 COBOL/Java 相互運用機能のプロジェクトを利用
 - 4.3.2 COBOL と Java を別プロジェクトで利用
 - 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
- 5 Visual COBOL for Eclipse 上の文字コード設定について
 - 5.1 ワークスペースに対する文字コード設定
 - 5.2 プロジェクトに対する設定

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

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。

- 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

3.2 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。

3.3 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 など利用できます。

3.3.1 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/>

3.3.2 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、[Micro Focus Visual COBOL] > [Visual COBOL コマンドプロンプト (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
Micro Focus Interface Mapping Toolkit v9.0.00167
(C) Copyright 1984-2023 Micro Focus or one of its affiliates.
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

3.3.3 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;  
Micro Focus COBOL  
Version 9.0 (C) Copyright 1984-2023 Micro Focus or one of 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:      138352      Code:      5716      Literals:      2288  
C:\¥COBOLJavaInteroperability¥apiservice>cobol get_post-proxy.cbl gnt;  
Micro Focus COBOL  
Version 9.0 (C) Copyright 1984-2023 Micro Focus or one of 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:      3
```

- 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

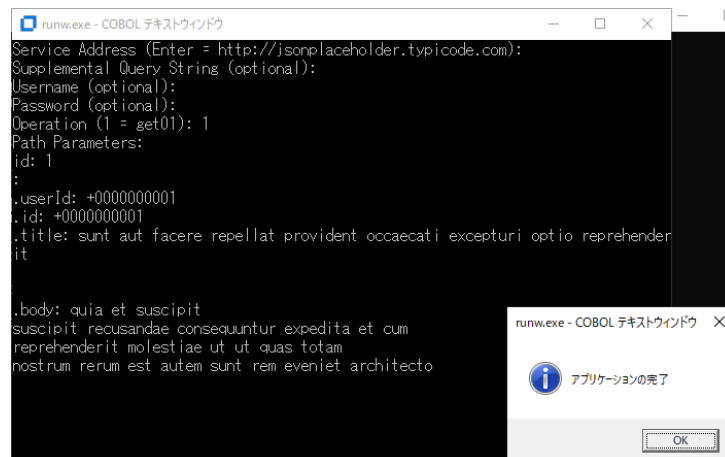
Username: 何も入力せず Enter キーを押す

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。

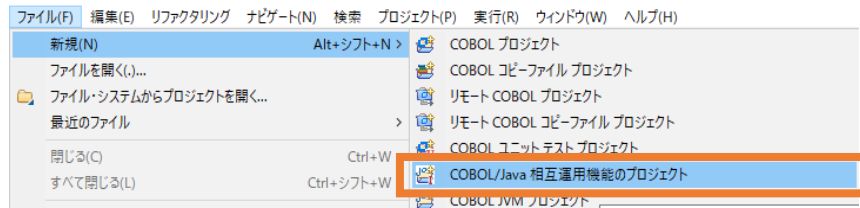
3.4 COBOL/Java 相互運用機能を利用

3.3 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。

3.4.1 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。

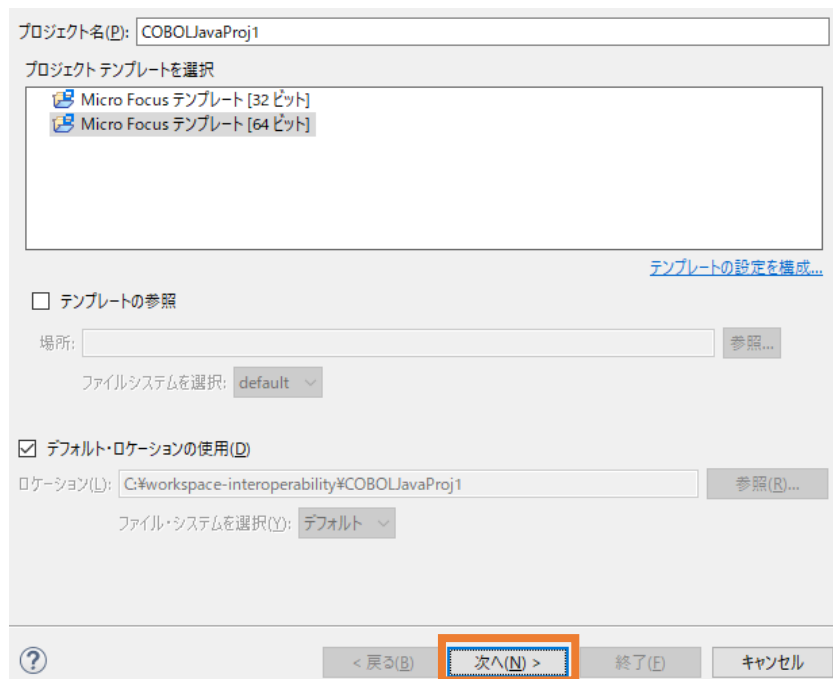
- 1) Windows スタートメニューより、[Micro Focus Visual COBOL] > [Visual COBOL for Eclipse] を選択して、Visual COBOL for Eclipse を起動します。
ワークスペースは任意のフォルダでかまいません。以降の手順では、c:\¥workspace-interopability を使用します。
- 2) ワークスペースに対する文字コード設定を行います。
5.1 の手順を実施してください。
- 3) [ファイル(F)] > [新規(N)] > [COBOL/Java 相互運用機能のプロジェクト] を選択します。



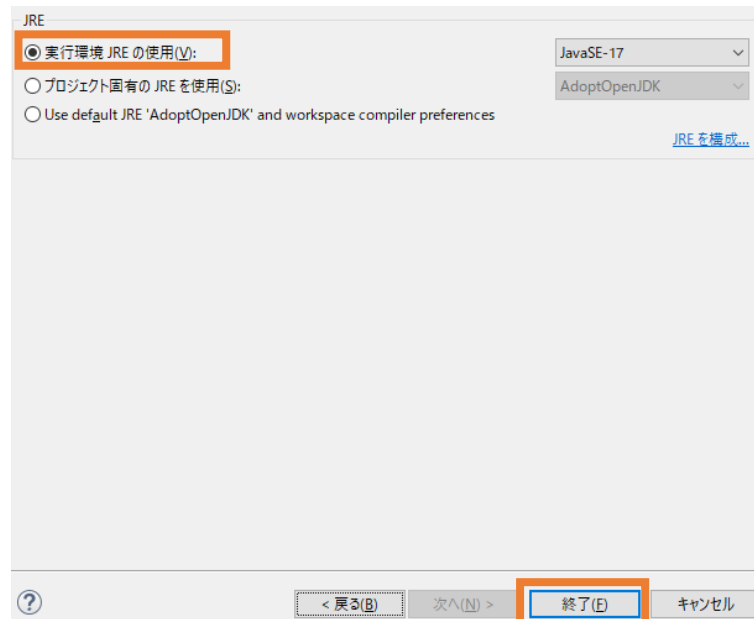
以下の入力を行い、[次へ(N)] をクリックします。

プロジェクト名 : “COBOLJavaProj1”

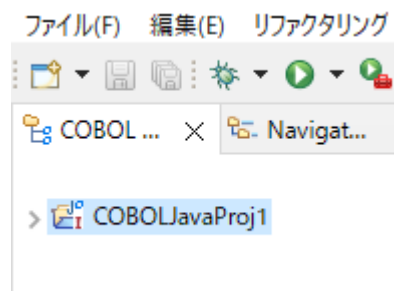
プロジェクトテンプレート : “Micro Focus テンプレート(64 ビット)”



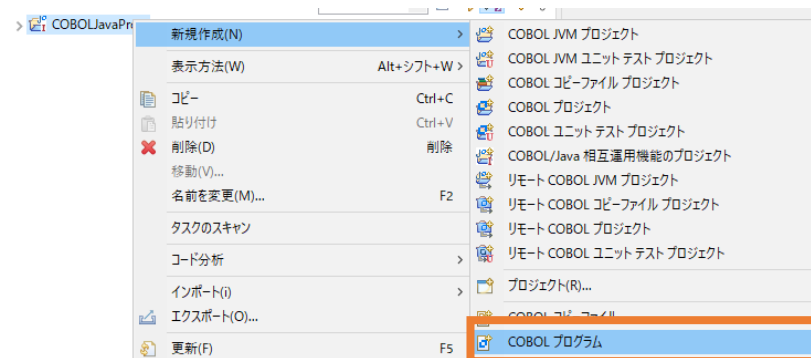
4) JRE に [実行環境 JRE の使用] を選択し、[終了(F)] をクリックします。



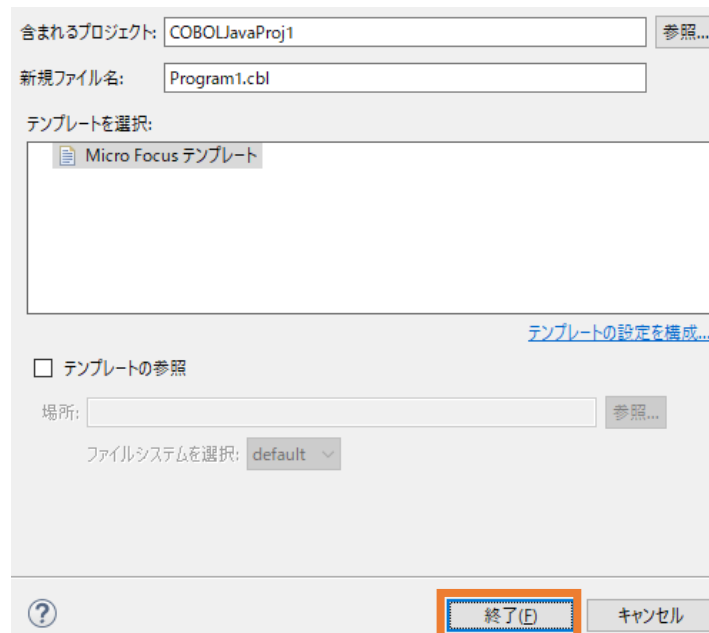
COBOLJavaProj1 プロジェクトが作成されます。



- 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。

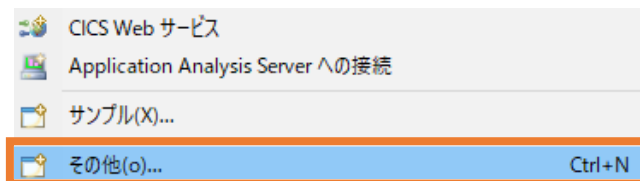


そのまま、[終了(F)] をクリックします。

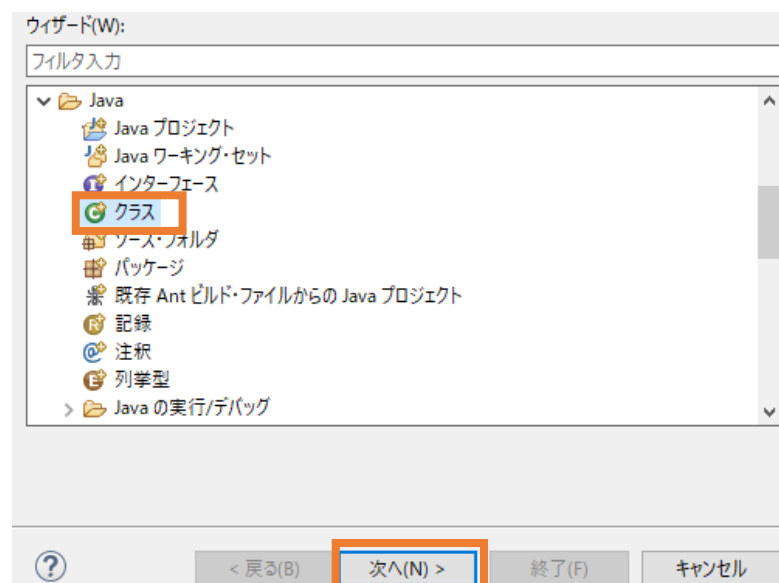


作成された Program1.cbl を、サンプルファイルを解凍したフォルダ内の COBOLtoJava フォルダ配下の Program1.cbl の内容で上書きしてください。

- 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”

ソース・フォルダ(D): COBOLJavaProj1/src 参照(o)...

パッケージ(K): com.sample 参照(W)...

エンクロージング型(Y): 参照(W)...

名前(M): COBOLJava

修飾子:
 public(P) package(C) private(V) protected(T)
 abstract(T) final(L) static(C)

スーパークラス(S): java.lang.Object 参照(E)...

インターフェース(i): 追加(A)...

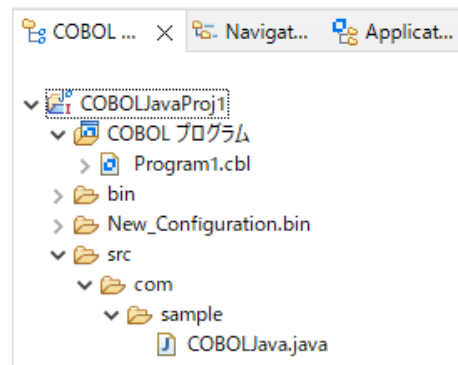
除去(R)

作成するメソッド・スタブの選択
 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(C)
 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)

? < 戻る(B) 次へ(N) > 終了(F) キャンセル

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。

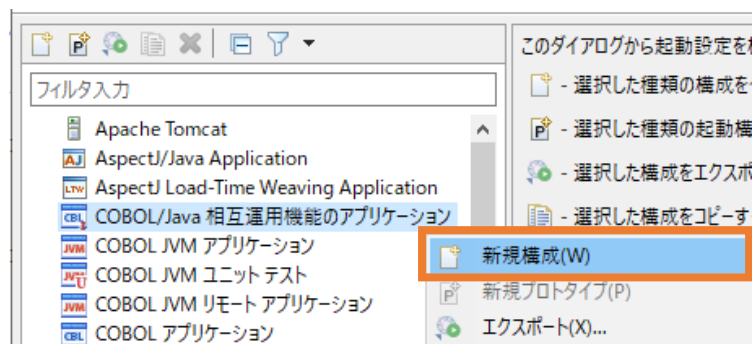


作成された COBOLJava.java を、エディター上でサンプルファイルを解凍したフォルダ内の COBOLtoJava フォルダ配下の COBOLJava.java の内容で上書き保存してください。

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。



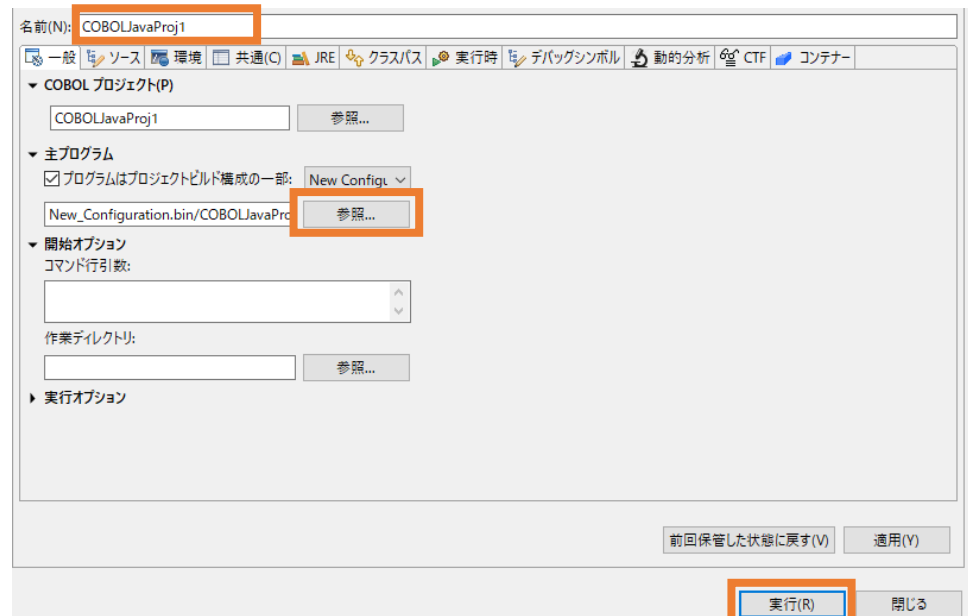
[COBOL/Java 相互運用機能のアプリケーション] を選択したうえで、マウスの右クリックによりコンテキストメニューを開き、[新規構成(W)] を選択します。



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。

3.4.2 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。

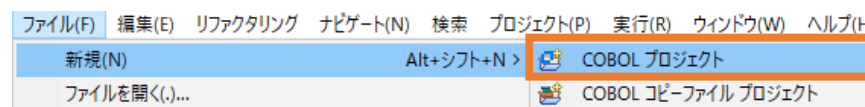
注意)
以降の手順で参照する Java クラスは 3.4.1 で作成したものです。こちらの手順の実施前に、3.4.1 を実施してください。

1) Visual COBOL for Eclipse の起動

Windows スタートメニューより、[Micro Focus 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) [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 : “Micro Focus テンプレート(64 ビット)”

プロジェクト名(P)

プロジェクトテンプレートを選択

- Micro Focus テンプレート [32 ビット]
- Micro Focus テンプレート [64 ビット]**

[テンプレートの設定を構成...](#)

テンプレートの参照

場所:

ファイルシステムを選択: default ▾

デフォルト・ロケーションの使用(D)

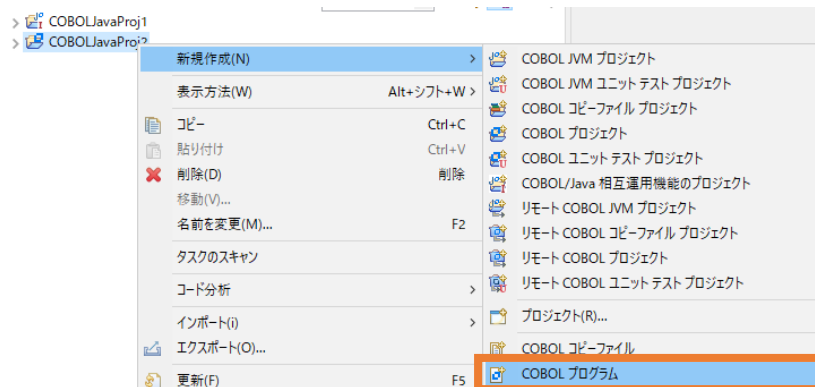
ロケーション(L):

ファイル・システムを選択(Y): デフォルト ▾

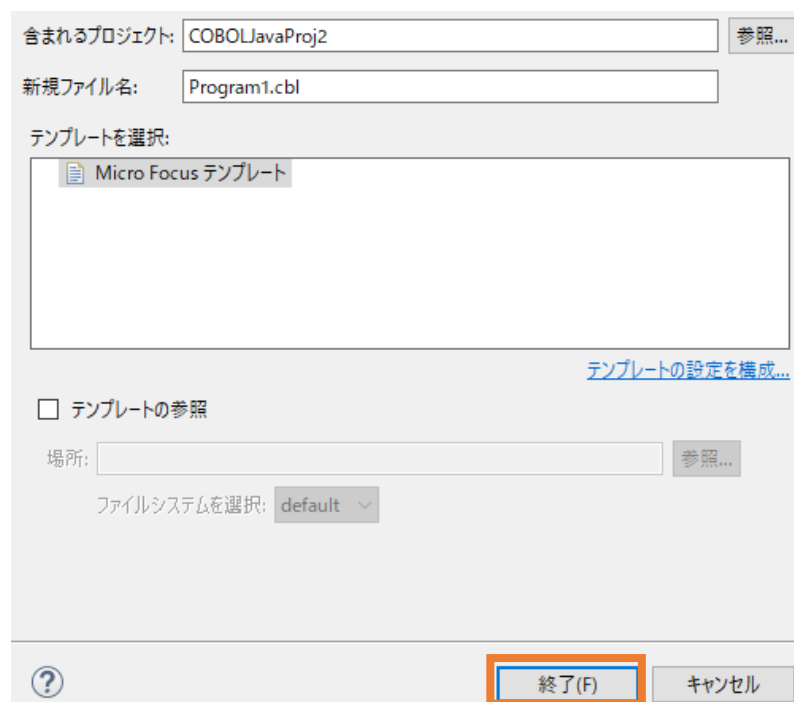
COBOLJavaProj2 プロジェクトが作成されます。



- 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。

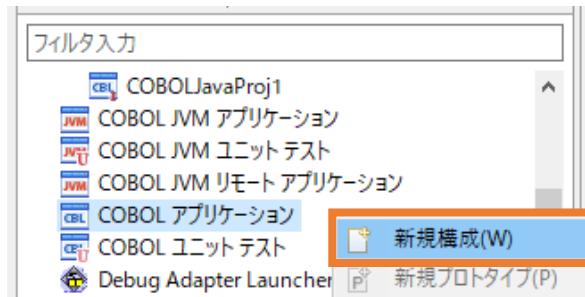


作成された Program1.cbl を、エディター上でサンプルファイルを解凍したフォルダ内の COBOLtoJava フォルダ配下の Program1.cbl の内容で上書き保存してください。

- 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。



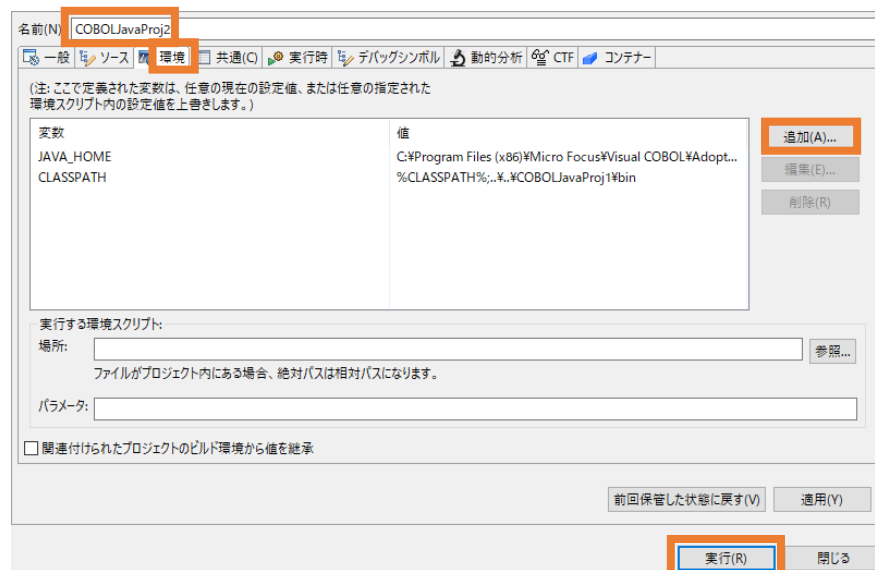
6) 以下の入力を行い、[実行(R)] をクリックします。

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)¥Micro Focus¥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、3.4.1 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。

3.4.1 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。

3.5 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.microfocus.co.jp/manuals/VC90/Eclipse/index.html?t=GUID-D10DC512-FDEF-44CF-8A9B-32839729B493.html>

Micro Focus Visual COBOL 9.0 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.microfocus.co.jp/manuals/CMN/MFVC_900_ECLWSVC01.pdf

Micro Focus Visual COBOL 9.0 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 相互運用機能のプロジェクトを利用

- 1) Visual COBOL for Eclipse の起動

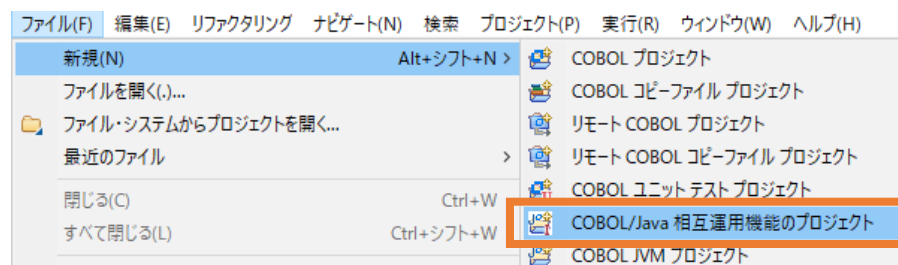
Windows スタートメニューより、[Micro Focus Visual COBOL] > [Visual COBOL for Eclipse] を選択して、Visual COBOL for Eclipse を起動します。

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-javacollaborate 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。

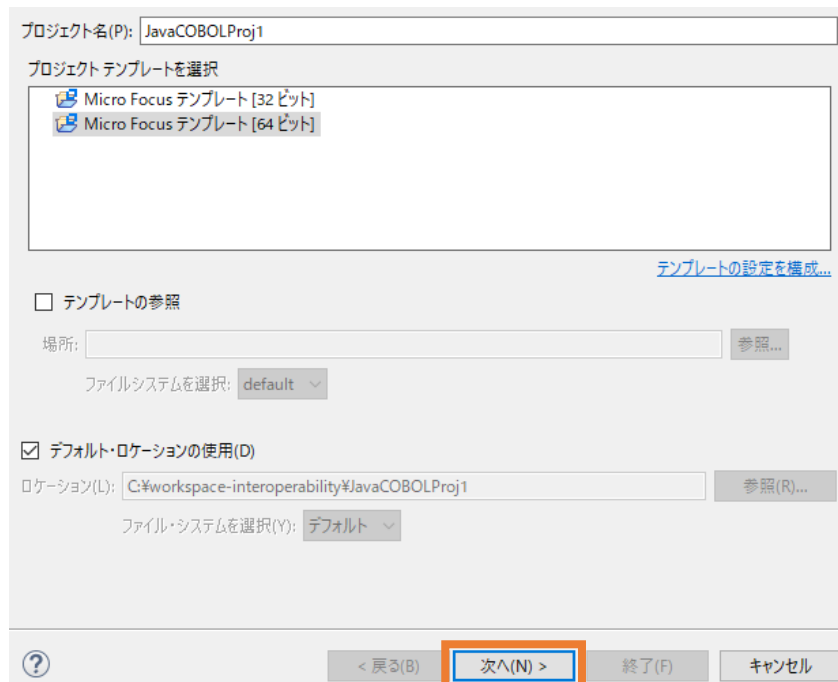
- 3) [ファイル(F)] > [新規(N)] > [COBOL/Java 相互運用機能のプロジェクト] を選択します。



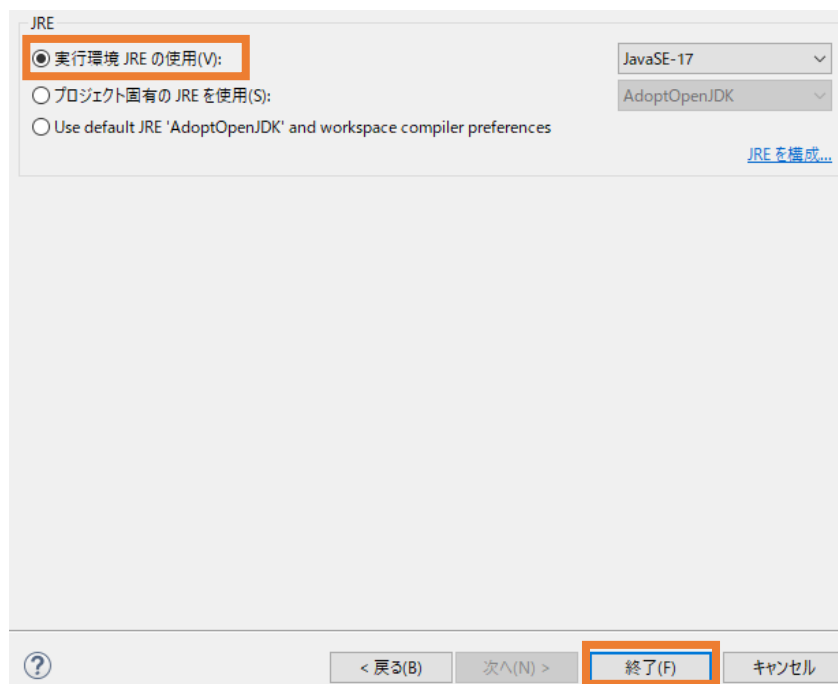
以下の入力を行い、[次へ(N)] をクリックします。

プロジェクト名: "JavaCOBOLProj1"

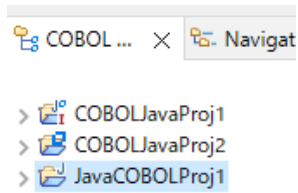
プロジェクトテンプレート: "Micro Focus テンプレート(64 ビット)"



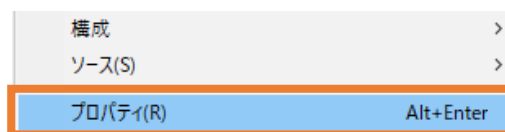
- 4) JRE に [実行環境 JRE の使用] を選択し、[終了(F)] をクリックします。



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。



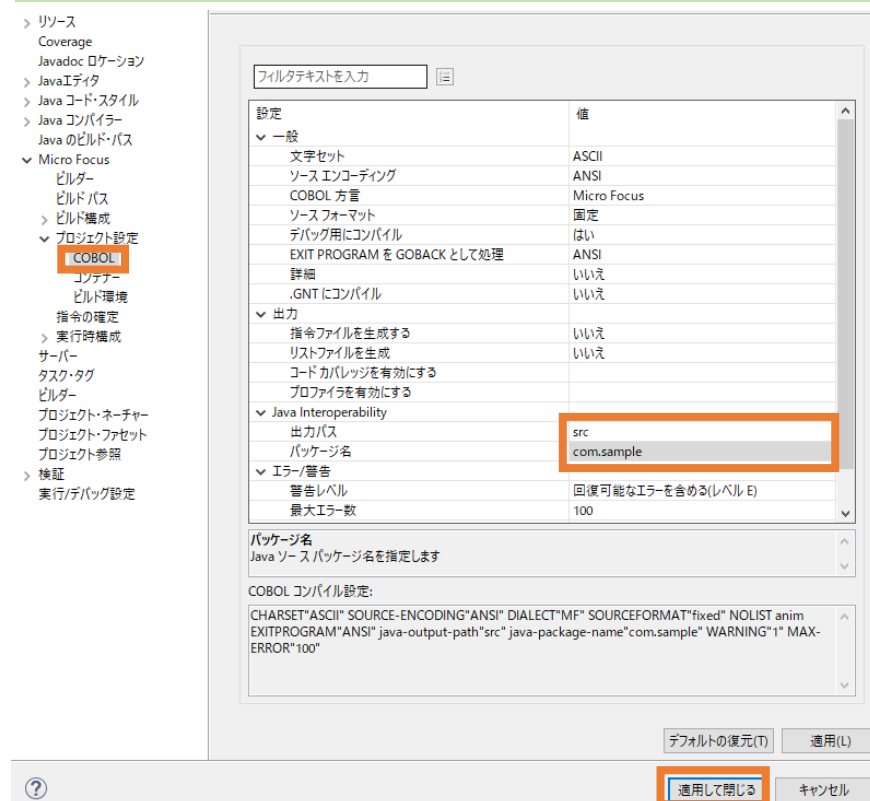
左側のツリーより [Micro Focus] > [プロジェクト設定] > [COBOL] を選択し、以下の選択を行ったうえで、[適用して閉じる] をクリックします。

出力パス: "src"

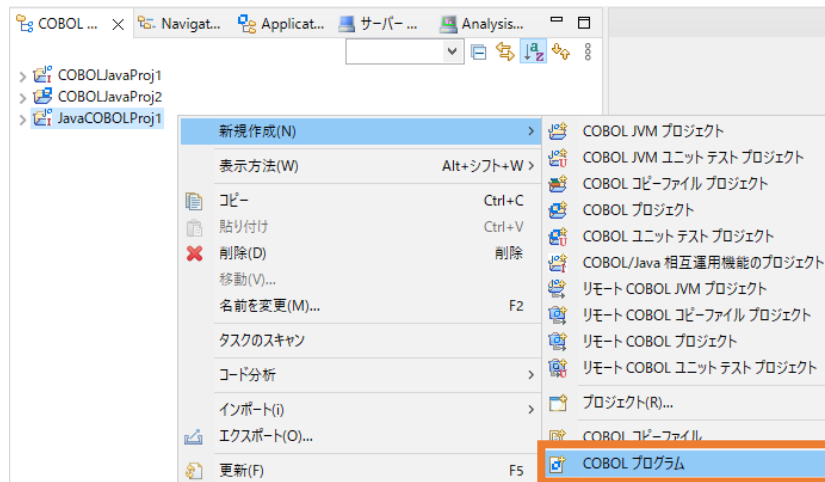
パッケージ名: "com.sample"

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。

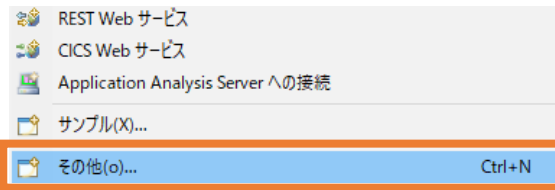
COBOL プログラム

エディタで開くことができる COBOL プログラムを新規作成します。

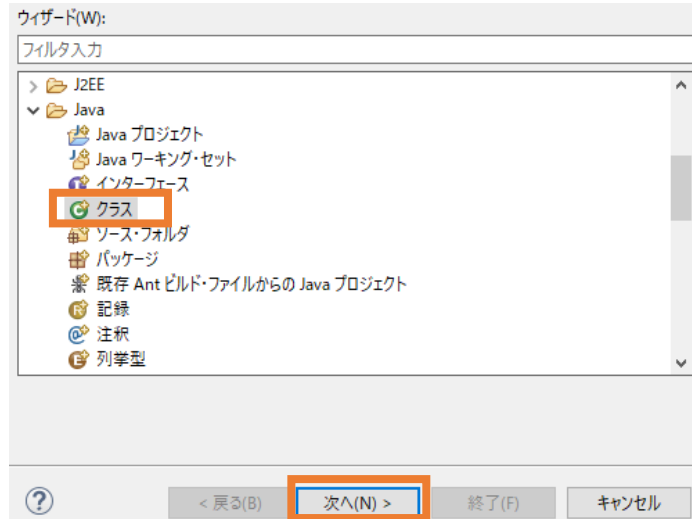


作成された Program1.cbl を、サンプルファイルを解凍したフォルダ内の JavaToCOBOL フォルダ配下の Program1.cbl の内容で上書きしてください。

- 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。



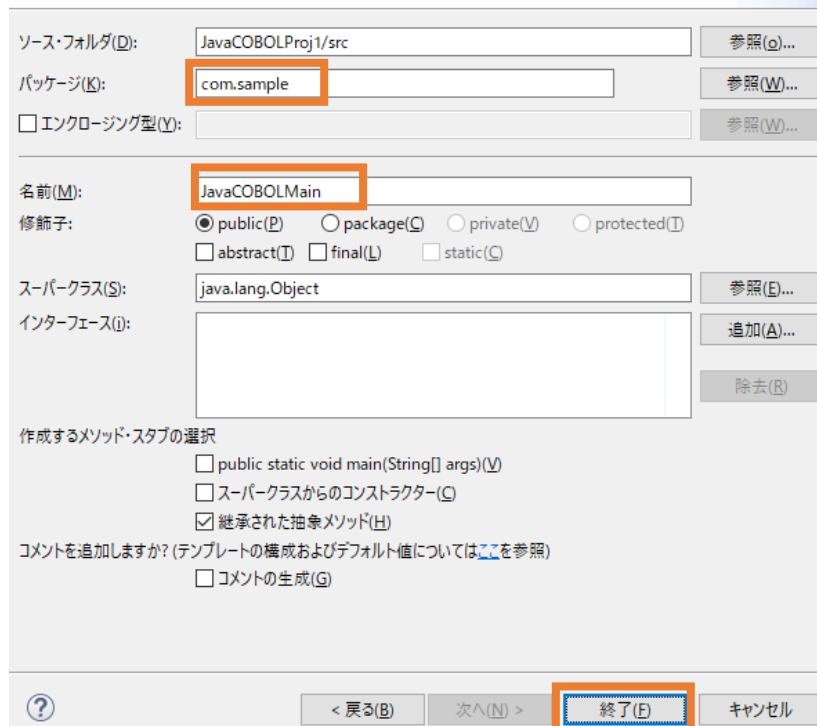
以下の入力を行ったうえで、[終了(F)] をクリックします。

パッケージ : "com.sample"

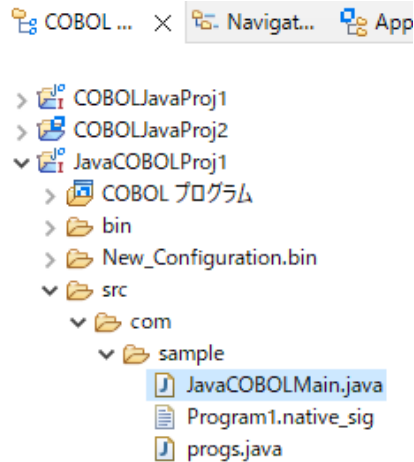
名前 : "JavaCOBOLMain"

Java クラス

新規 Java クラスを作成します。

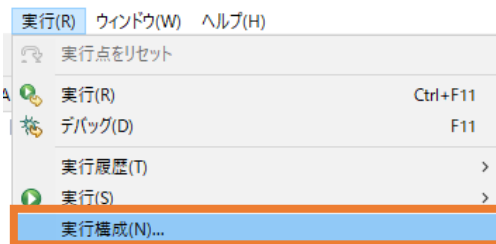


JavaCOBOLProj1 プロジェクト配下の src¥com¥sample の下に
JavaCOBOLMain.java が作成されます。



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。

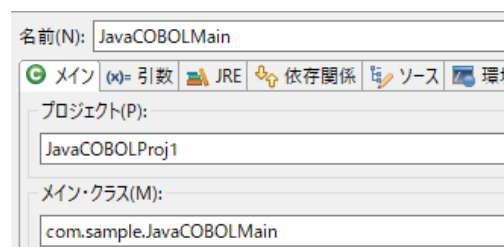


[Java アプリケーション] を選択したうえで、マウスの右クリックによりコンテキストメニューを開き、[新規構成(W)] を選択します。

- 11) 以下の入力を行い、[実行(R)] をクリックします。

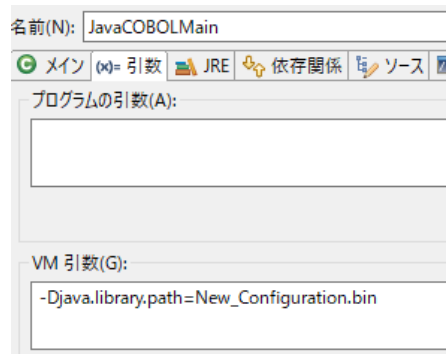
名前: "JavaCOBOLMain"

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"



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
27
```

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

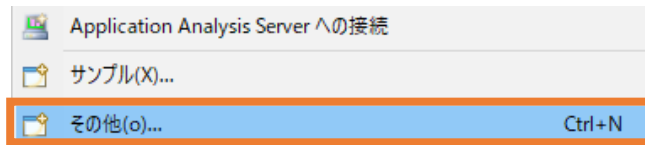
Windows スタートメニューより、[Micro Focus Visual COBOL] > [Visual COBOL for Eclipse] を選択して、Visual COBOL for Eclipse を起動します。

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-javacollaborate を使用します。

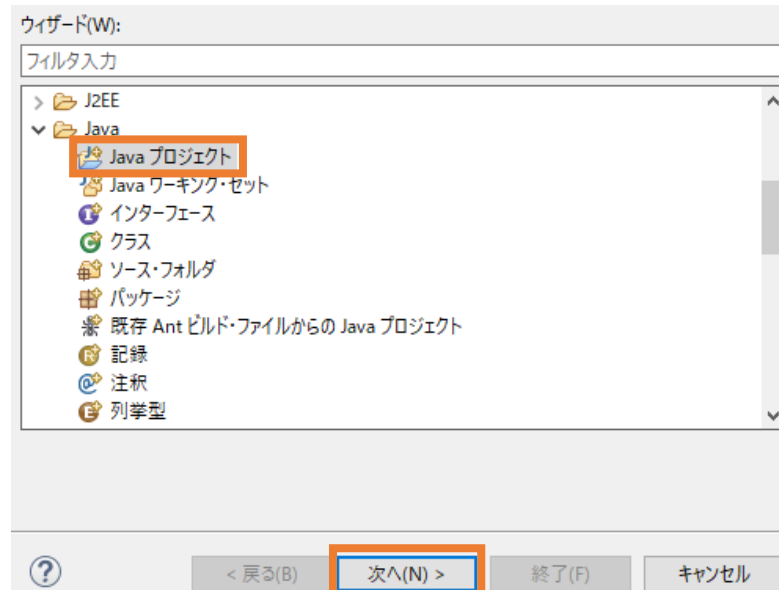
2) ワークスペースに対する文字コード設定を行います。

5.1 の手順を実施してください。

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



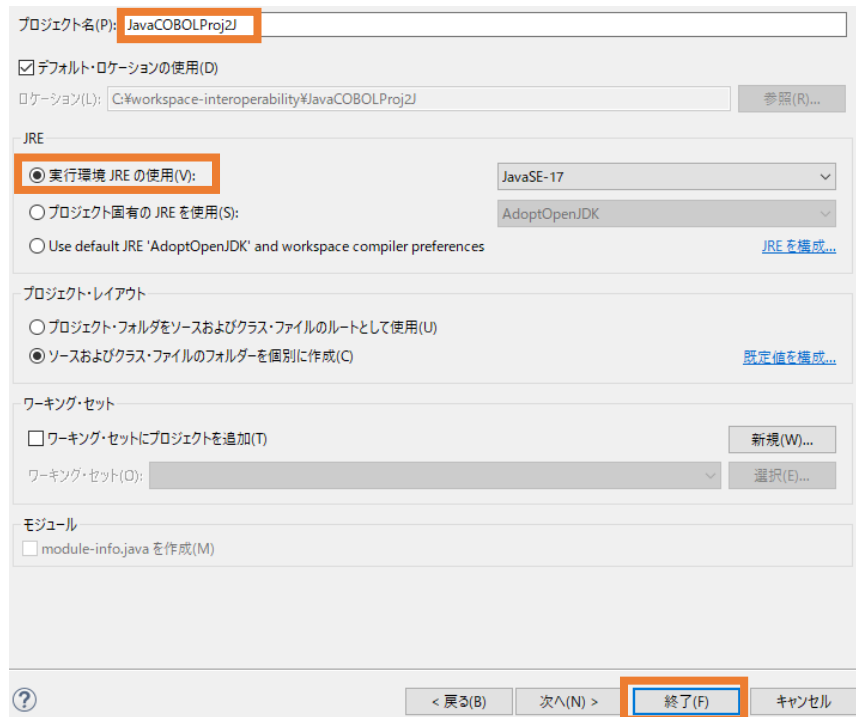
- 4) [Java] > [Java プロジェクト] を選択したうえで [次へ] をクリックします。



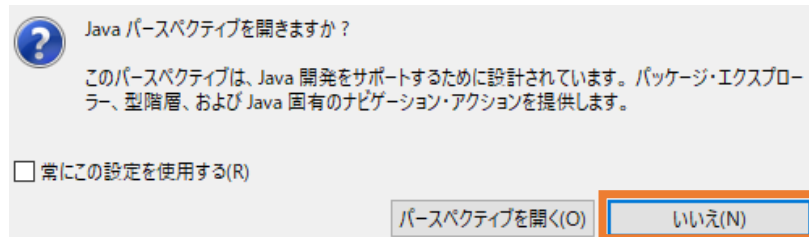
- 5) 以下の入力を行い、[終了(F)] をクリックします。

プロジェクト名 : "JavaCOBOLProj2J"

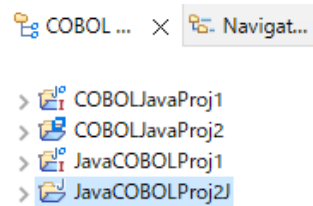
[実行環境 JRE の使用] を選択



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。

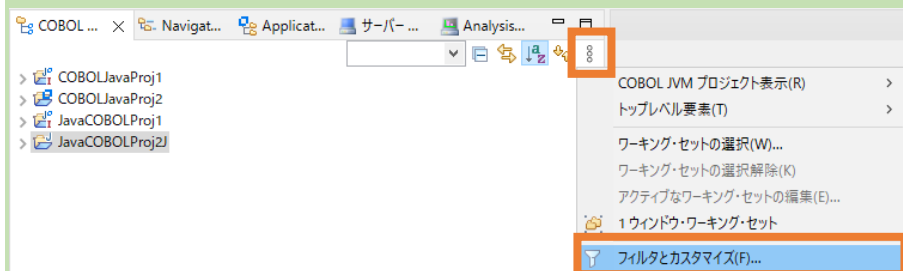


JavaCOBOLProj2J プロジェクトが作成されます。

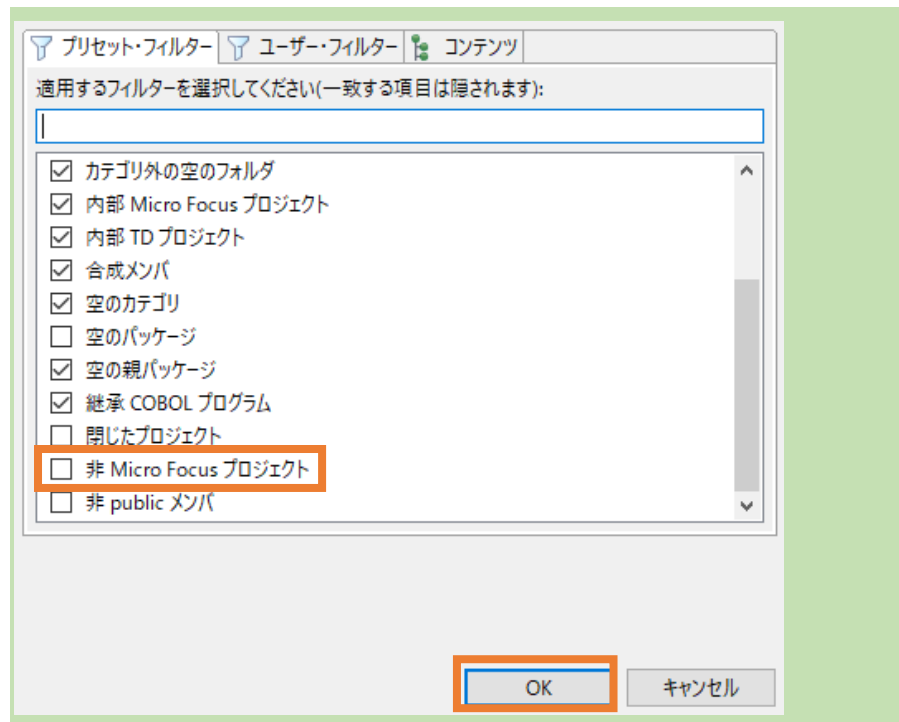


JavaCOBOLProj2J プロジェクトが表示されない場合)

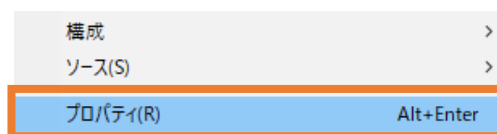
エクスプローラービューの右上をクリックし、[フィルタとカスタマイズ(F)] をクリックします。



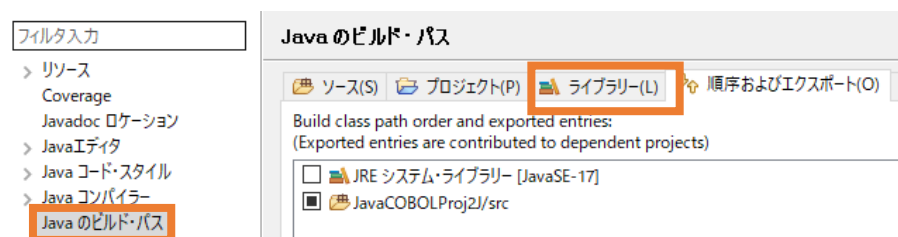
[非 Micro Focus プロジェクト] のチェックを外したうえで、[OK] をクリックします。



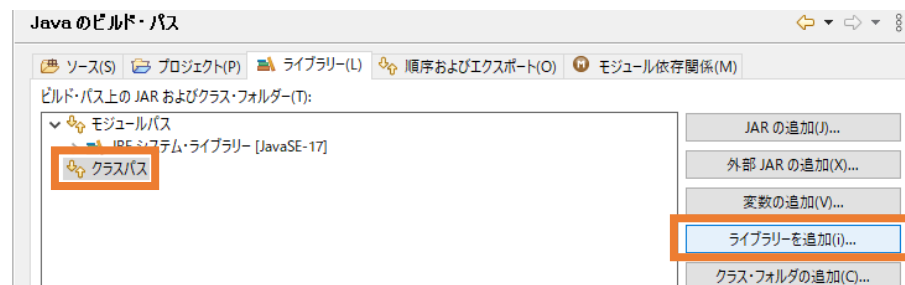
- 6) JavaCOBOLProj2J プロジェクトを選択したうえで、マウスの右クリックによりコンテキストメニューを開き、[プロパティ(R)] を選択します。



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。



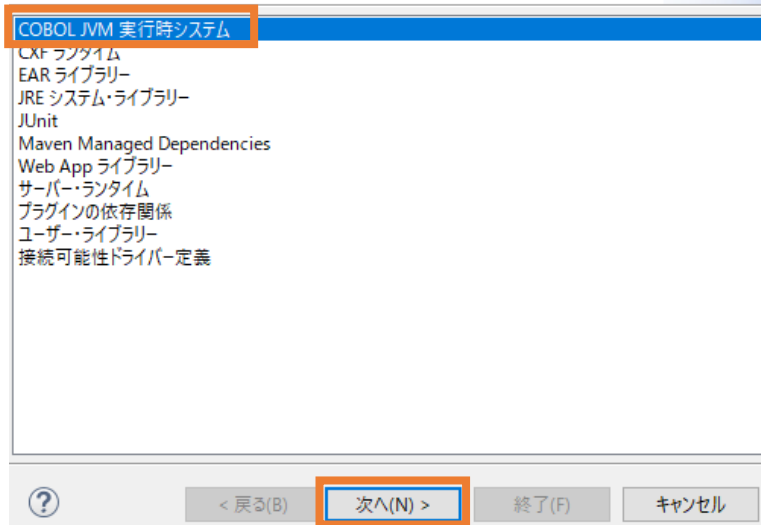
[クラスパス] を選択したうえで、[ライブラリーを追加(i)] をクリックします。



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

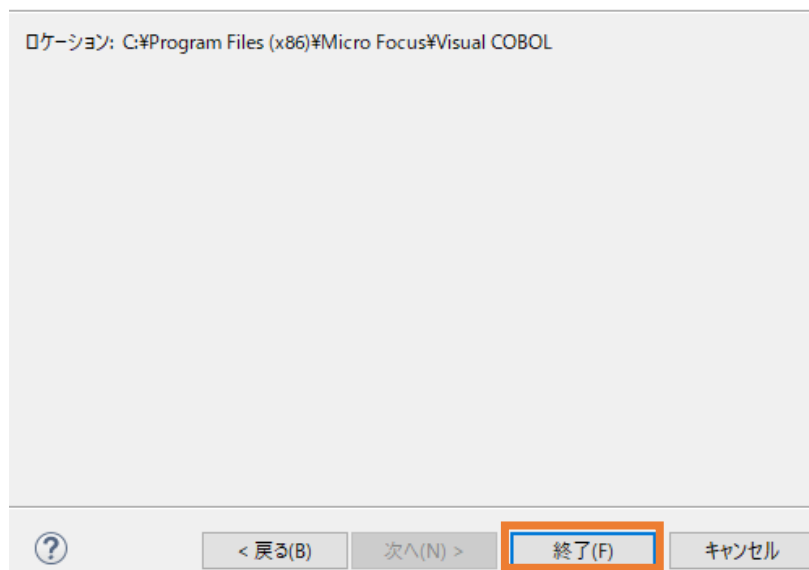
ライブラリーの追加

追加するライブラリー・タイプを選択します。

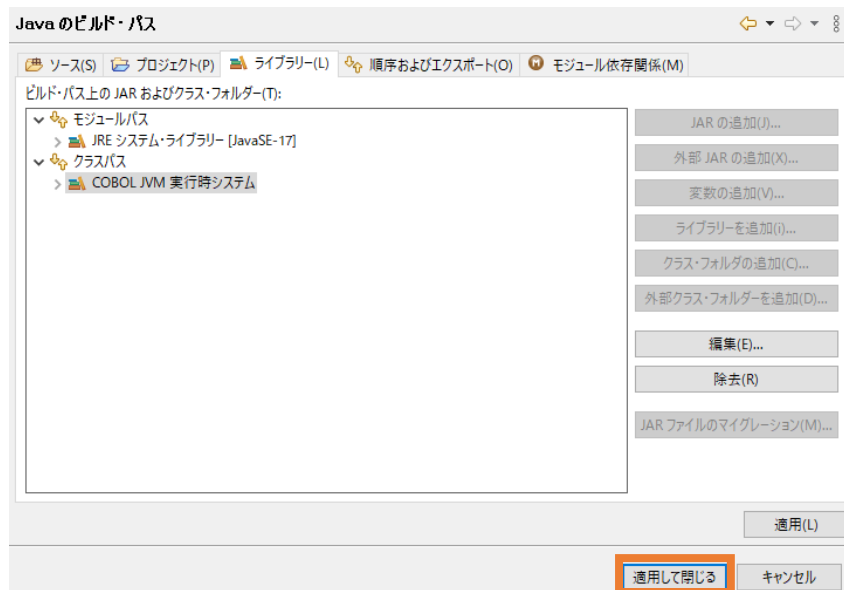


そのまま、[終了(F)] をクリックします。

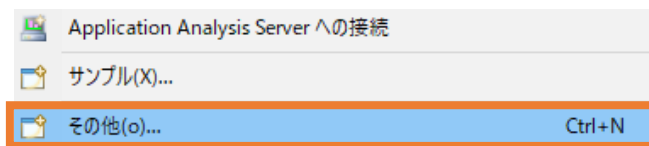
COBOL JVM 実行時システム



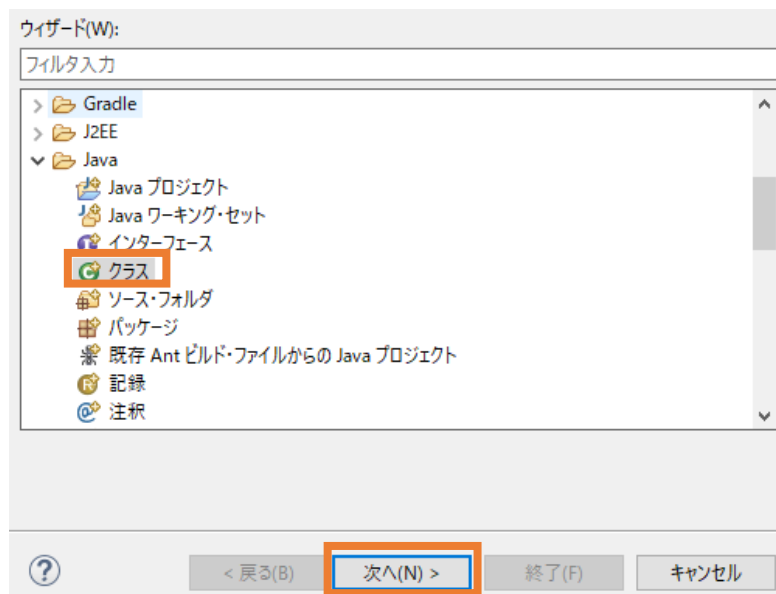
“COBOL JVM 実行時システム” が追加されたことを確認したうえで、[適用して閉じる] をクリックします。



- 7) JavaCOBOLProj2] 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"

Java クラス

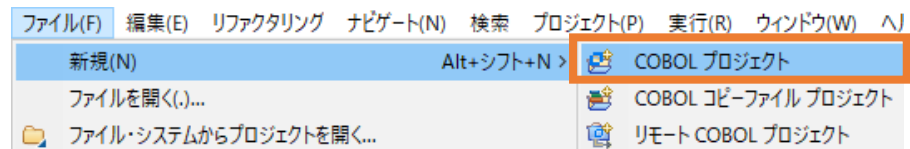
新規 Java クラスを作成します。



ソース・フォルダ(D):	JavaCOBOLProj2J/src	参照(o)...
パッケージ(K):	com.sample	参照(W)...
<input type="checkbox"/> エンクロージング型(Y):		参照(W)...
名前(M):	JavaCOBOLMain	
修飾子:	<input checked="" type="radio"/> public(P) <input type="radio"/> package(C) <input type="radio"/> private(V) <input type="radio"/> protected(T) <input type="checkbox"/> abstract(T) <input type="checkbox"/> final(L) <input type="checkbox"/> static(C) <input checked="" type="radio"/> none <input type="radio"/> sealed <input type="radio"/> non-sealed <input type="radio"/> final(L)	
スーパークラス(S):	java.lang.Object	参照(E)...
インターフェース(I):		追加(A)...
		除去(R)
作成するメソッド・スタブの選択	<input type="checkbox"/> public static void main(String[] args)(V) <input type="checkbox"/> スーパークラスからのコンストラクター(C) <input checked="" type="checkbox"/> 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)	<input type="checkbox"/> コメントの生成(G)	
? < 戻る(B) 次へ(N) > 終了(F) キャンセル		

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。

- 8) [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: "Micro Focus テンプレート(64 ビット)"

COBOL プロジェクト

ワークスペースまたは外部の場所にCOBOL プロジェクトを作成します。

プロジェクト名(P): JavaCOBOLProj2C

プロジェクト テンプレートを選択

- Micro Focus テンプレート [32 ビット]
- Micro Focus テンプレート [64 ビット]

[テンプレートの設定を構成...](#)

テンプレートの参照

場所:

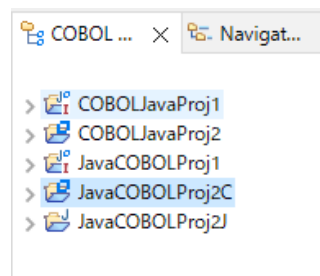
ファイルシステムを選択: default ▾

デフォルト・ロケーションの使用(D)

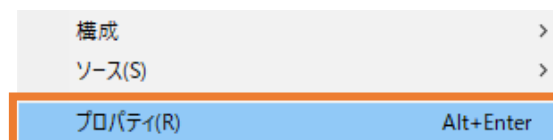
ロケーション(L): C:\workspace-interopability\JavaCOBOLProj2C

ファイル・システムを選択(Y): デフォルト ▾

JavaCOBOLProj2C プロジェクトが作成されます。



- 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。
- 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。

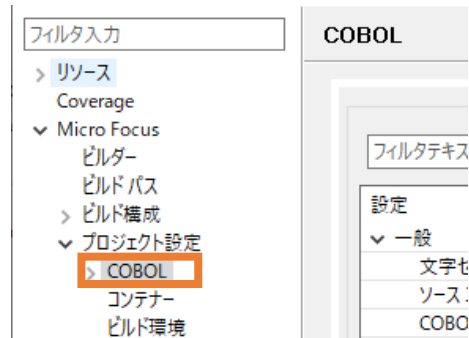


以下の設定を行ったうえで、[適用して閉じる] をクリックします。

[Micro Focus] > [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

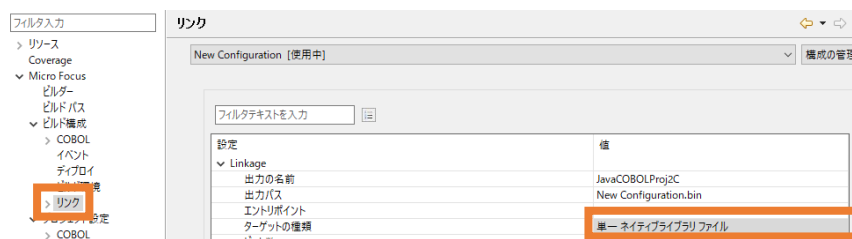
- java-output-path"..¥JavaCOBOLProj2¥src"
- java-package-name"com.sample"



設定	値
▼ 一般	
文字セット	ASCII
ソース エンコーディング	ANSI
COBOL 方言	Micro Focus
ソース フォーマット	固定
デバッグ用にコンパイル	はい
EXIT PROGRAM を GOBACK として処理	ANSI
詳細	いいえ
.GNT にコンパイル	いいえ
▼ 出力	
指令ファイルを生成する	いいえ
リストファイルを生成	いいえ
コードカバレッジを有効にする	false
プロファイラを有効にする	false
▼ エラー/警告	
警告レベル	回復可能なエラーを含める(レベル E)
最大エラー数	100
▼ 追加指令	
追加指令	java-output-path"..¥JavaCOBOLProj2¥src" java-package-name"com.sample"

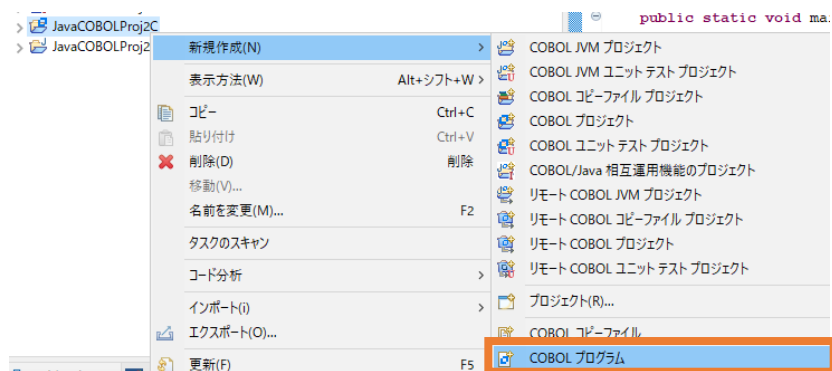
[Micro Focus] > [ビルド構成] > [リンク] を選択

ターゲットの種類：“単一ネイティブライブラリファイル”



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。

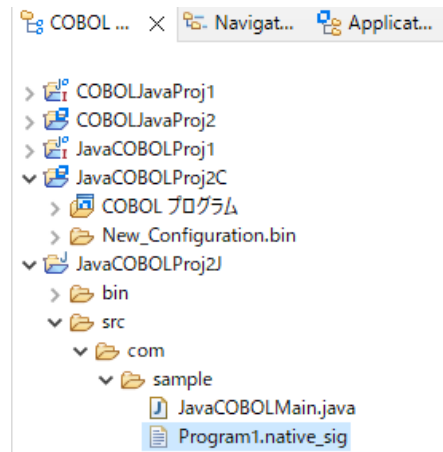
COBOL プログラム

エディタで開くことができる COBOL プログラムを新規作成します。



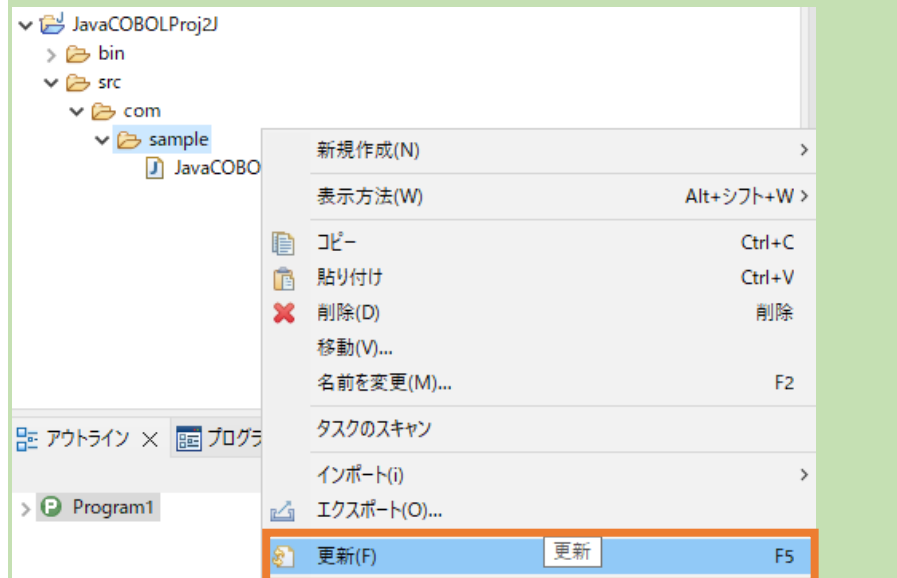
作成された Program1.cbl を、サンプルファイルを解凍したフォルダ内の JavatoCOBOL フォルダ配下の Program1.cbl の内容で上書きしてください。

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。

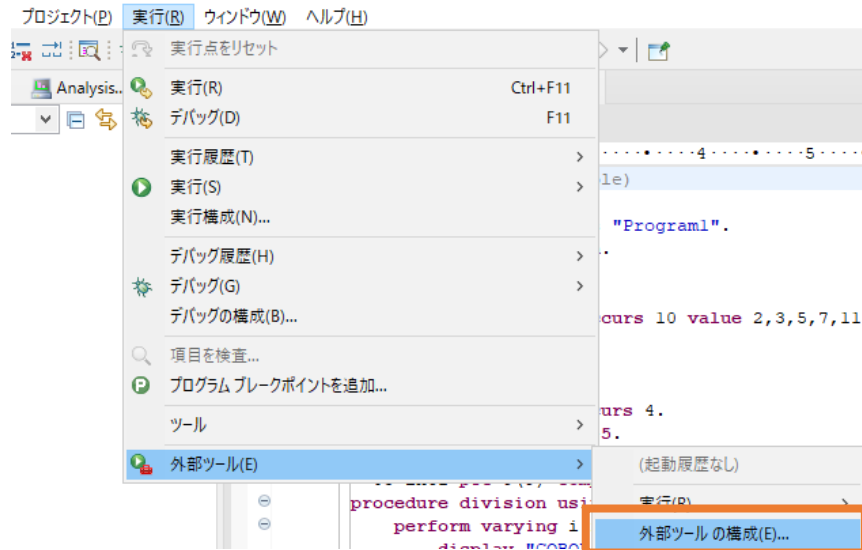


補足)

表示されない場合は、src¥com¥sample フォルダを選択し、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、[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)¥Micro Focus¥Visual COBOL¥bin64¥genjava.exe"

作業ディレクトリー：

"C:¥workspace-interopability¥JavaCOBOLProj2J¥src"

引数：

"JavaCOBOLProj2C -p Program1 -k com.sample"

補足)

ロケーション

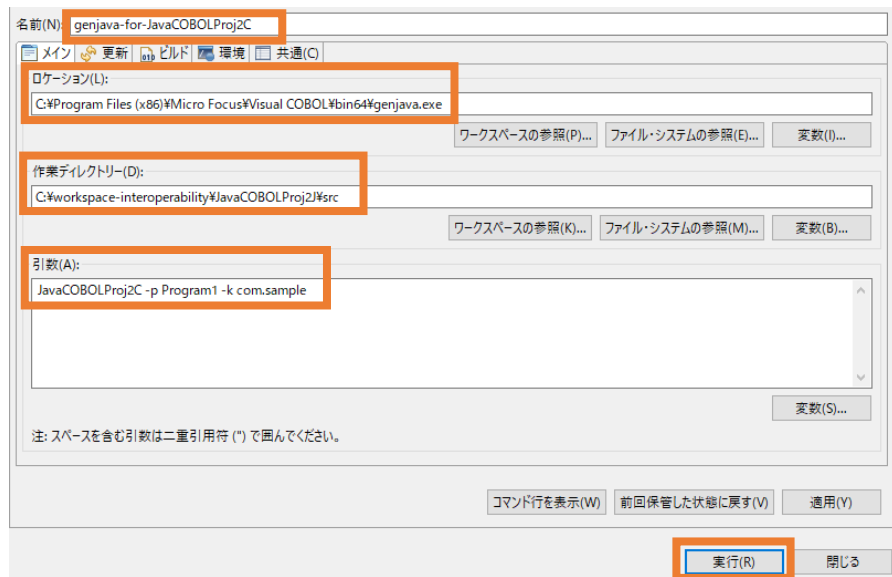
上記で指定している genjava.exe は、Visual COBOL 製品のインストール先がデフォルトの場合となります。異なるフォルダにインストールした場合は、<製品インストールフォルダ>¥bin64¥genjava.exe を指定してください。

作業ディレクトリー

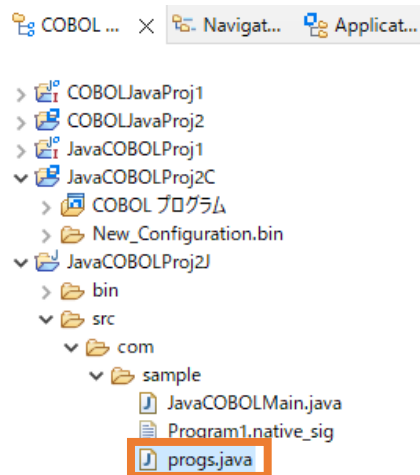
さきほど作成した JavaCOBOLProj2J プロジェクト配下の src フォルダまでの絶対パスを指定してください。

引数

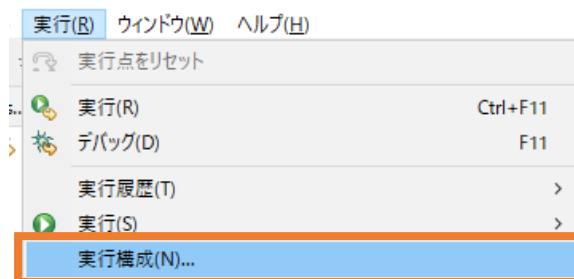
最初に指定している JavaCOBOLProj2C は、さきほど作成した COBOL プロジェクト "JavaCOBOLProj2C" の成果物である JavaCOBOLProj2C.dll を指定しています。



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。

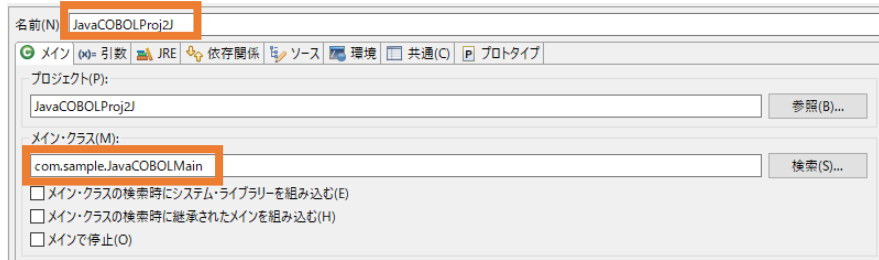


[Java アプリケーション] をダブルクリックします。

以下の入力を行い、[実行(R)] をクリックします。

名前: "JavaCOBOLProj2J"

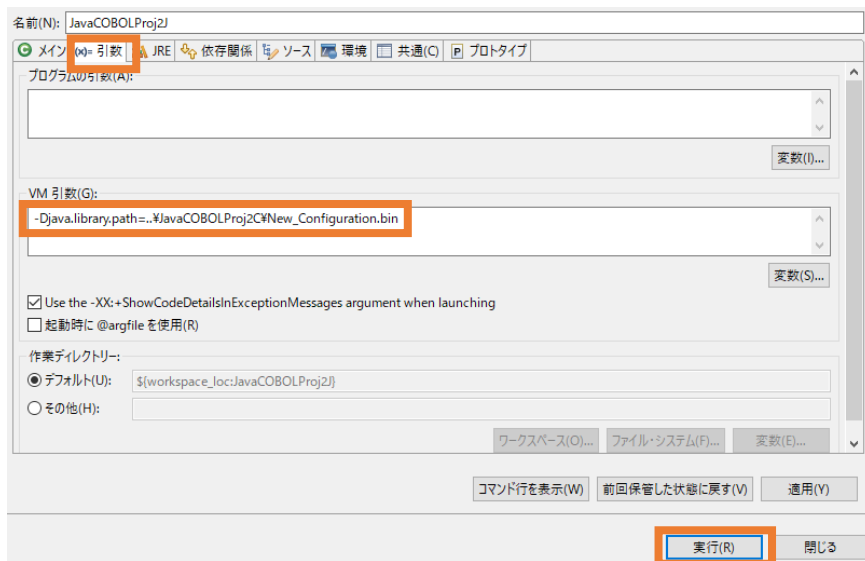
メイン・クラス: "com.sample.JavaCOBOLMain"



[引数] タブを選択

VM 引数 :

“-Djava.library.path=..¥JavaCOBOLProj2¥New_Configuration.bin”



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
27

```

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.microfocus.co.jp/manuals/VC90/Eclipse/index.html?t=GUID-D10DC512-FDEF-44CF-8A9B-32839729B493.html>

Micro Focus Visual COBOL 9.0 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。

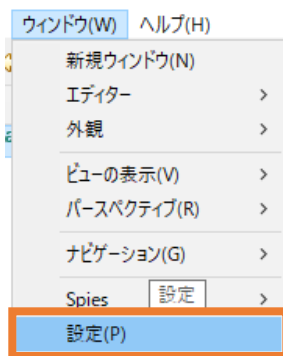
[ここからはじめよう] > [Getting Started] > [JVM COBOL チュートリアル]

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

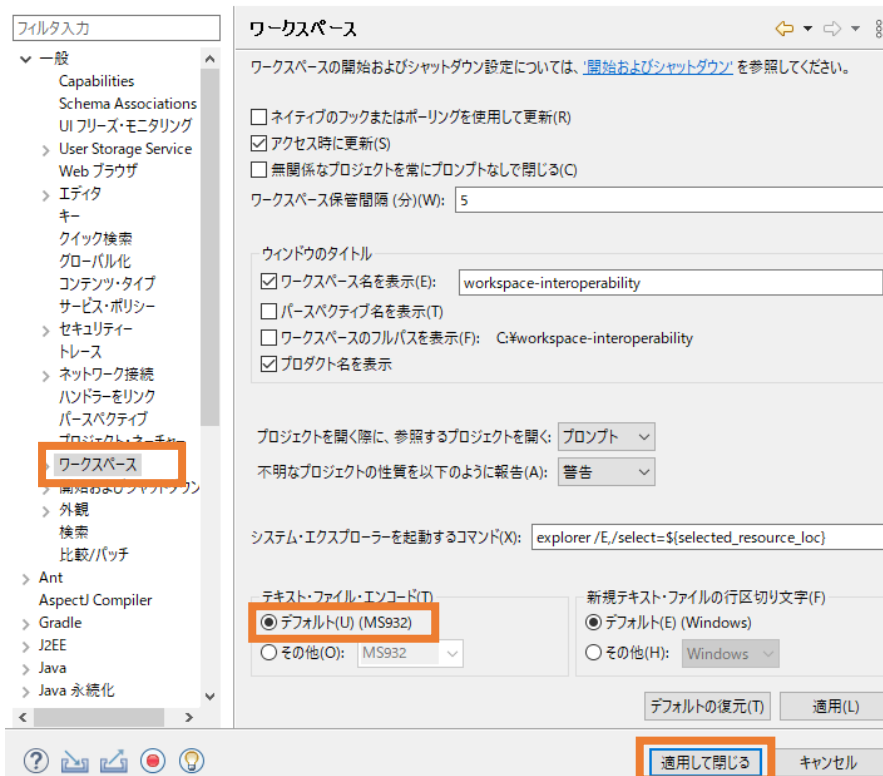
最新の Eclipse IDE 環境における文字コードのデフォルトは UTF-8 ですが、COBOL 資産は長年利用されていることから、多くは UTF-8 ではなく SJIS が採用されています。文字コード設定は、ワークスペース全体の設定と、プロジェクト毎の設定の 2 つがあります。これらの設定と、プログラムファイルの文字コードに不整合があると、文字化けの原因となります。本チュートリアルで使用するサンプルファイルは、文字コード SJIS を採用しているため、Eclipse IDE 上で SJIS 資産を正しく扱うための設定手順について紹介します。

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。



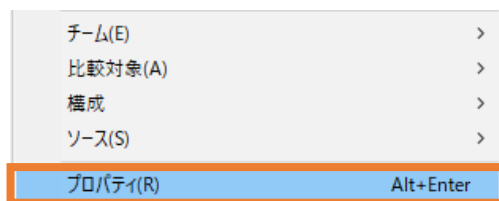
- 2) [一般] > [ワークスペース] を選択し、[テキスト・ファイル・エンコード] に “デフォルト(MS932)” を選択したうえで、[OK] をクリックします。



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) [Micro Focus] > [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"

設定	値
▼ 一般	
文字セット	ANSI
ソースエンコーディング	ANSI
COBOL 方言	Micro Focus
ソースフォーマット	固定
デバッグ用にコンパイル	はい
EXIT PROGRAM を GOBACK として処理	ANSI
詳細	いいえ
.GNT にコンパイル	いいえ
▼ 出力	
指令ファイルを生成する	いいえ
リストファイルを生成	いいえ
コードカバレッジを有効にする	
プロファイラを有効にする	
▼ Java Interoperability	
出力パス	src
パッケージ名	com.microfocus.COBOL
▼ エラー/警告	
警告レベル	回復可能なエラーを含める(レベル E)
最大エラー数	100

ソースエンコーディング
SOURCE-ENCODING はソース プログラムのエンコーディングをコンパイラに渡します。その後、RUNTIME-ENCODING 指令が指定されていない限り、実行時のエンコーディングの決定に使用されます。ソース ファイルに UTF-8、UTF-16、UTF-32、ASCII、ANSI、EBCDIC、UTF-16LE、UTF-16BE、UTF-32LE、UTF-32BE、UTF-7、UTF-7B、UTF-7E、UTF-7F、UTF-7G、UTF-7H、UTF-7I、UTF-7J、UTF-7K、UTF-7L、UTF-7M、UTF-7N、UTF-7O、UTF-7P、UTF-7Q、UTF-7R、UTF-7S、UTF-7T、UTF-7U、UTF-7V、UTF-7W、UTF-7X、UTF-7Y、UTF-7Z、UTF-7AA、UTF-7AB、UTF-7AC、UTF-7AD、UTF-7AE、UTF-7AF、UTF-7AG、UTF-7AH、UTF-7AI、UTF-7AJ、UTF-7AK、UTF-7AL、UTF-7AM、UTF-7AN、UTF-7AO、UTF-7AP、UTF-7AQ、UTF-7AR、UTF-7AS、UTF-7AT、UTF-7AU、UTF-7AV、UTF-7AW、UTF-7AX、UTF-7AY、UTF-7AZ、UTF-7BA、UTF-7BB、UTF-7BC、UTF-7BD、UTF-7BE、UTF-7BF、UTF-7BG、UTF-7BH、UTF-7BI、UTF-7BJ、UTF-7BK、UTF-7BL、UTF-7BM、UTF-7BN、UTF-7BO、UTF-7BP、UTF-7BQ、UTF-7BR、UTF-7BS、UTF-7BT、UTF-7BU、UTF-7BV、UTF-7BW、UTF-7BX、UTF-7BY、UTF-7BZ、UTF-7CA、UTF-7CB、UTF-7CC、UTF-7CD、UTF-7CE、UTF-7CF、UTF-7CG、UTF-7CH、UTF-7CI、UTF-7CJ、UTF-7CK、UTF-7CL、UTF-7CM、UTF-7CN、UTF-7CO、UTF-7CP、UTF-7CQ、UTF-7CR、UTF-7CS、UTF-7CT、UTF-7CU、UTF-7CV、UTF-7CW、UTF-7CX、UTF-7CY、UTF-7CZ、UTF-7DA、UTF-7DB、UTF-7DC、UTF-7DD、UTF-7DE、UTF-7DF、UTF-7DG、UTF-7DH、UTF-7DI、UTF-7DJ、UTF-7DK、UTF-7DL、UTF-7DM、UTF-7DN、UTF-7DO、UTF-7DP、UTF-7DQ、UTF-7DR、UTF-7DS、UTF-7DT、UTF-7DU、UTF-7DV、UTF-7DW、UTF-7DX、UTF-7DY、UTF-7DZ、UTF-7EA、UTF-7EB、UTF-7EC、UTF-7ED、UTF-7EE、UTF-7EF、UTF-7EG、UTF-7EH、UTF-7EI、UTF-7EJ、UTF-7EK、UTF-7EL、UTF-7EM、UTF-7EN、UTF-7EO、UTF-7EP、UTF-7EQ、UTF-7ER、UTF-7ES、UTF-7ET、UTF-7EU、UTF-7EV、UTF-7EW、UTF-7EX、UTF-7EY、UTF-7EZ、UTF-7FA、UTF-7FB、UTF-7FC、UTF-7FD、UTF-7FE、UTF-7FF、UTF-7FG、UTF-7FH、UTF-7FI、UTF-7FJ、UTF-7FK、UTF-7FL、UTF-7FM、UTF-7FN、UTF-7FO、UTF-7FP、UTF-7FQ、UTF-7FR、UTF-7FS、UTF-7FT、UTF-7FU、UTF-7FV、UTF-7FW、UTF-7FX、UTF-7FY、UTF-7FZ、UTF-7GA、UTF-7GB、UTF-7GC、UTF-7GD、UTF-7GE、UTF-7GF、UTF-7GG、UTF-7GH、UTF-7GI、UTF-7GJ、UTF-7GK、UTF-7GL、UTF-7GM、UTF-7GN、UTF-7GO、UTF-7GP、UTF-7GQ、UTF-7GR、UTF-7GS、UTF-7GT、UTF-7GU、UTF-7GV、UTF-7GW、UTF-7GX、UTF-7GY、UTF-7GZ、UTF-7HA、UTF-7HB、UTF-7HC、UTF-7HD、UTF-7HE、UTF-7HF、UTF-7HG、UTF-7HH、UTF-7HI、UTF-7HJ、UTF-7HK、UTF-7HL、UTF-7HM、UTF-7HN、UTF-7HO、UTF-7HP、UTF-7HQ、UTF-7HR、UTF-7HS、UTF-7HT、UTF-7HU、UTF-7HV、UTF-7HW、UTF-7HX、UTF-7HY、UTF-7HZ、UTF-7IA、UTF-7IB、UTF-7IC、UTF-7ID、UTF-7IE、UTF-7IF、UTF-7IG、UTF-7IH、UTF-7II、UTF-7IJ、UTF-7IK、UTF-7IL、UTF-7IM、UTF-7IN、UTF-7IO、UTF-7IP、UTF-7IQ、UTF-7IR、UTF-7IS、UTF-7IT、UTF-7IU、UTF-7IV、UTF-7IW、UTF-7IX、UTF-7IY、UTF-7IZ、UTF-7JA、UTF-7JB、UTF-7JC、UTF-7JD、UTF-7JE、UTF-7JF、UTF-7JG、UTF-7JH、UTF-7JI、UTF-7JJ、UTF-7JK、UTF-7JL、UTF-7JM、UTF-7JN、UTF-7JO、UTF-7JP、UTF-7JQ、UTF-7JR、UTF-7JS、UTF-7JT、UTF-7JU、UTF-7JV、UTF-7JW、UTF-7JX、UTF-7JY、UTF-7JZ、UTF-7KA、UTF-7KB、UTF-7KC、UTF-7KD、UTF-7KE、UTF-7KF、UTF-7KG、UTF-7KH、UTF-7KI、UTF-7KJ、UTF-7KK、UTF-7KL、UTF-7KM、UTF-7KN、UTF-7KO、UTF-7KP、UTF-7KQ、UTF-7KR、UTF-7KS、UTF-7KT、UTF-7KU、UTF-7KV、UTF-7KW、UTF-7KX、UTF-7KY、UTF-7KZ、UTF-7LA、UTF-7LB、UTF-7LC、UTF-7LD、UTF-7LE、UTF-7LF、UTF-7LG、UTF-7LH、UTF-7LI、UTF-7LJ、UTF-7LK、UTF-7LL、UTF-7LM、UTF-7LN、UTF-7LO、UTF-7LP、UTF-7LQ、UTF-7LR、UTF-7LS、UTF-7LT、UTF-7LU、UTF-7LV、UTF-7LW、UTF-7LX、UTF-7LY、UTF-7LZ、UTF-7MA、UTF-7MB、UTF-7MC、UTF-7MD、UTF-7ME、UTF-7MF、UTF-7MG、UTF-7MH、UTF-7MI、UTF-7MJ、UTF-7MK、UTF-7ML、UTF-7MM、UTF-7MN、UTF-7MO、UTF-7MP、UTF-7MQ、UTF-7MR、UTF-7MS、UTF-7MT、UTF-7MU、UTF-7MV、UTF-7MW、UTF-7MX、UTF-7MY、UTF-7MZ、UTF-7NA、UTF-7NB、UTF-7NC、UTF-7ND、UTF-7NE、UTF-7NF、UTF-7NG、UTF-7NH、UTF-7NI、UTF-7NJ、UTF-7NK、UTF-7NL、UTF-7NM、UTF-7NN、UTF-7NO、UTF-7NP、UTF-7NQ、UTF-7NR、UTF-7NS、UTF-7NT、UTF-7NU、UTF-7NV、UTF-7NW、UTF-7NX、UTF-7NY、UTF-7NZ、UTF-7OA、UTF-7OB、UTF-7OC、UTF-7OD、UTF-7OE、UTF-7OF、UTF-7OG、UTF-7OH、UTF-7OI、UTF-7OJ、UTF-7OK、UTF-7OL、UTF-7OM、UTF-7ON、UTF-7OO、UTF-7OP、UTF-7OQ、UTF-7OR、UTF-7OS、UTF-7OT、UTF-7OU、UTF-7OV、UTF-7OW、UTF-7OX、UTF-7OY、UTF-7OZ、UTF-7PA、UTF-7PB、UTF-7PC、UTF-7PD、UTF-7PE、UTF-7PF、UTF-7PG、UTF-7PH、UTF-7PI、UTF-7PJ、UTF-7PK、UTF-7PL、UTF-7PM、UTF-7PN、UTF-7PO、UTF-7PP、UTF-7PQ、UTF-7PR、UTF-7PS、UTF-7PT、UTF-7PU、UTF-7PV、UTF-7PW、UTF-7PX、UTF-7PY、UTF-7PZ、UTF-7QA、UTF-7QB、UTF-7QC、UTF-7QD、UTF-7QE、UTF-7QF、UTF-7QG、UTF-7QH、UTF-7QI、UTF-7QJ、UTF-7QK、UTF-7QL、UTF-7QM、UTF-7QN、UTF-7QO、UTF-7QP、UTF-7QQ、UTF-7QR、UTF-7QS、UTF-7QT、UTF-7QU、UTF-7QV、UTF-7QW、UTF-7QX、UTF-7QY、UTF-7QZ、UTF-7RA、UTF-7RB、UTF-7RC、UTF-7RD、UTF-7RE、UTF-7RF、UTF-7RG、UTF-7RH、UTF-7RI、UTF-7RJ、UTF-7RK、UTF-7RL、UTF-7RM、UTF-7RN、UTF-7RO、UTF-7RP、UTF-7RQ、UTF-7RR、UTF-7RS、UTF-7RT、UTF-7RU、UTF-7RV、UTF-7RW、UTF-7RX、UTF-7RY、UTF-7RZ、UTF-7SA、UTF-7SB、UTF-7SC、UTF-7SD、UTF-7SE、UTF-7SF、UTF-7SG、UTF-7SH、UTF-7SI、UTF-7SJ、UTF-7SK、UTF-7SL、UTF-7SM、UTF-7SN、UTF-7SO、UTF-7SP、UTF-7SQ、UTF-7SR、UTF-7SS、UTF-7ST、UTF-7SU、UTF-7SV、UTF-7SW、UTF-7SX、UTF-7SY、UTF-7SZ、UTF-7TA、UTF-7TB、UTF-7TC、UTF-7TD、UTF-7TE、UTF-7TF、UTF-7TG、UTF-7TH、UTF-7TI、UTF-7TJ、UTF-7TK、UTF-7TL、UTF-7TM、UTF-7TN、UTF-7TO、UTF-7TP、UTF-7TQ、UTF-7TR、UTF-7TS、UTF-7TT、UTF-7TU、UTF-7TV、UTF-7TW、UTF-7TX、UTF-7TY、UTF-7TZ、UTF-7UA、UTF-7UB、UTF-7UC、UTF-7UD、UTF-7UE、UTF-7UF、UTF-7UG、UTF-7UH、UTF-7UI、UTF-7UJ、UTF-7UK、UTF-7UL、UTF-7UM、UTF-7UN、UTF-7UO、UTF-7UP、UTF-7UQ、UTF-7UR、UTF-7US、UTF-7UT、UTF-7UU、UTF-7UV、UTF-7UW、UTF-7UX、UTF-7UY、UTF-7UZ、UTF-7VA、UTF-7VB、UTF-7VC、UTF-7VD、UTF-7VE、UTF-7VF、UTF-7VG、UTF-7VH、UTF-7VI、UTF-7VJ、UTF-7VK、UTF-7VL、UTF-7VM、UTF-7VN、UTF-7VO、UTF-7VP、UTF-7VQ、UTF-7VR、UTF-7VS、UTF-7VT、UTF-7VU、UTF-7VV、UTF-7VW、UTF-7VX、UTF-7VY、UTF-7VZ、UTF-7WA、UTF-7WB、UTF-7WC、UTF-7WD、UTF-7WE、UTF-7WF、UTF-7WG、UTF-7WH、UTF-7WI、UTF-7WJ、UTF-7WK、UTF-7WL、UTF-7WM、UTF-7WN、UTF-7WO、UTF-7WP、UTF-7WQ、UTF-7WR、UTF-7WS、UTF-7WT、UTF-7WU、UTF-7WV、UTF-7WW、UTF-7WX、UTF-7WY、UTF-7WZ、UTF-7XA、UTF-7XB、UTF-7XC、UTF-7XD、UTF-7XE、UTF-7XF、UTF-7XG、UTF-7XH、UTF-7XI、UTF-7XJ、UTF-7XK、UTF-7XL、UTF-7XM、UTF-7XN、UTF-7XO、UTF-7XP、UTF-7XQ、UTF-7XR、UTF-7XS、UTF-7XT、UTF-7XU、UTF-7XV、UTF-7XW、UTF-7XX、UTF-7XY、UTF-7XZ、UTF-7YA、UTF-7YB、UTF-7YC、UTF-7YD、UTF-7YE、UTF-7YF、UTF-7YG、UTF-7YH、UTF-7YI、UTF-7YJ、UTF-7YK、UTF-7YL、UTF-7YM、UTF-7YN、UTF-7YO、UTF-7YP、UTF-7YQ、UTF-7YR、UTF-7YS、UTF-7YT、UTF-7YU、UTF-7YV、UTF-7YW、UTF-7YX、UTF-7YY、UTF-7YZ、UTF-7ZA、UTF-7ZB、UTF-7ZC、UTF-7ZD、UTF-7ZE、UTF-7ZF、UTF-7ZG、UTF-7ZH、UTF-7ZI、UTF-7ZJ、UTF-7ZK、UTF-7ZL、UTF-7ZM、UTF-7ZN、UTF-7ZO、UTF-7ZP、UTF-7ZQ、UTF-7ZR、UTF-7ZS、UTF-7ZT、UTF-7ZU、UTF-7ZV、UTF-7ZW、UTF-7ZX、UTF-7ZY、UTF-7ZZ

COBOL コンパイル設定:
CHARSET"ASCII" SOURCE-ENCODING"ANSI" DIALECT"MF" SOURCEFORMAT"fixed" NOLIST anim
EXITPROGRAM"ANSI" java-output-path"src" java-package-name"com.microfocus.COBOL"
WARNING"1" MAX-ERROR"100"

デフォルトの復元(T) 適用(L)
適用して閉じる キャンセル

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、著作権法に基づき、適切な扱いを行ってください。