Rocket software

Visual COBOL チュートリアル

Eclipse – ネイティブ COBOL プログラムの単体テスト

1 目的

本チュートリアルでは、ネイティブ COBOL プログラムに対するテスト作成、実行方法、および、テスト結果を表示させる方法の習得を目的としています。

MFUnit は、Visual COBOL に搭載された xUnit 系の単体テストフレームワークです。xUnit はオブジェクト指向型の単体テストフレームワーク SUnit に起源を持つ JUnit や RUnit 等の単体テストフレームワークの総称です。MFUnit は xUnit の設計アーキテクチャーや仕組みは取り入れつつも COBOL 開発者にとって扱いやすい手続き型の COBOL を対象とした単体テストフレームワークという設計思想の下、開発されました。

MFUnit は COBOL 開発作業に以下の利点を提供します。

- テストを繰返し実行させることができるため、修正作業時などのテスト工数の削減が見込める
- Jenkins などの継続的インテグレーション（Continuous Integration）ツールと連携によりテストの自動化が行え、DevOps サイクルの導入が足がかりを作れる

2 前提条件

- 本チュートリアルで使用したマシン OS ： Windows 11
- Visual COBOL 11.0 Patch Update 01 for Eclipse がインストール済みであること

本資料は、ネイティブ COBOL に対する単体テストフレームワークの利用方法を記載したチュートリアルです。JVM COBOL の単体テスト実現方法については、別チュートリアルを参照ください。

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。

サンプルファイルのダウンロード

内容

3　チュートリアルについて

MFUnit は、単一処理の結果を判定する基本的なテストプログラムからデータ駆動型、自己完結型といった複数のテストタイプを用意しており、順に説明していきます。特定のテストタイプのみを実施したい場合においても、「3.1.1 前準備」は先に実施してください。

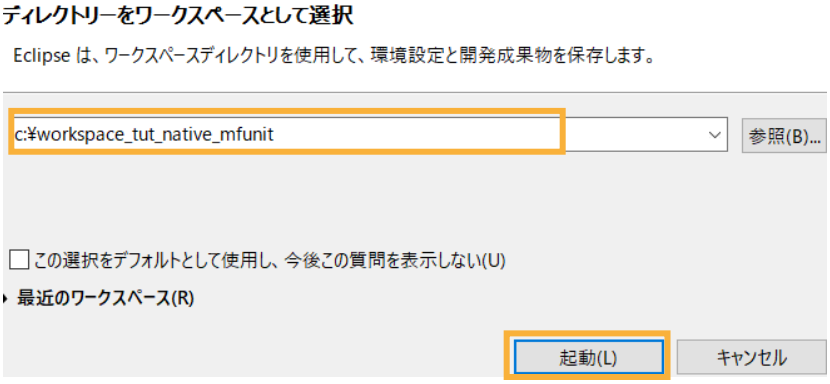
3.1　IDE からの実行

3.1.1 前準備

3.1.1.1 Eclipse IDE の起動

1) スタートメニューより、Visual COBOL for Eclipse を起動します。

2) ワークスペースを指定し、[起動(L)] をクリックします。

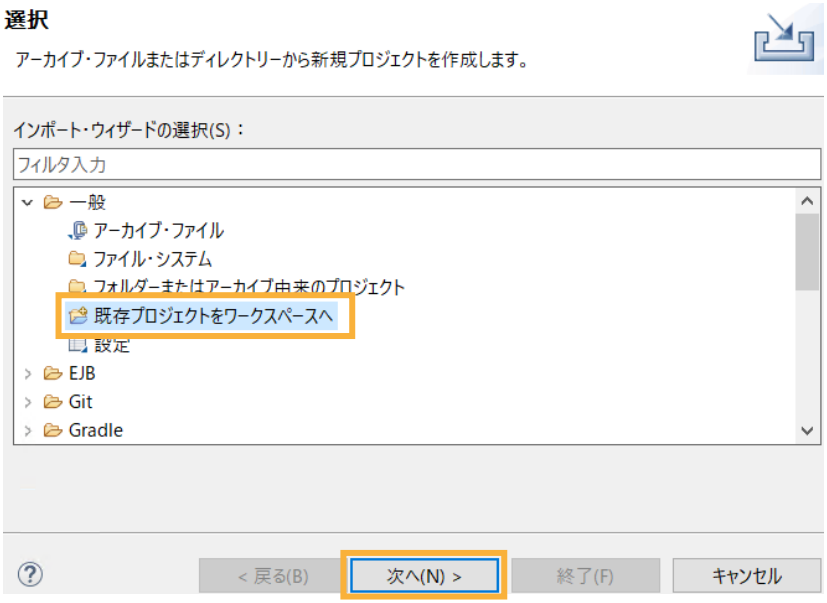


起動後、ようこそ画面は閉じてください。

3.1.1.2 チュートリアルプロジェクトのインポート

1) Eclipse IDE メニューより、[ファイル(F)] > [インポート(I)] を選択してください。

2) [一般] > [既存プロジェクトをワークスペースへ] を選択し、[次へ(N) >] をクリックします。

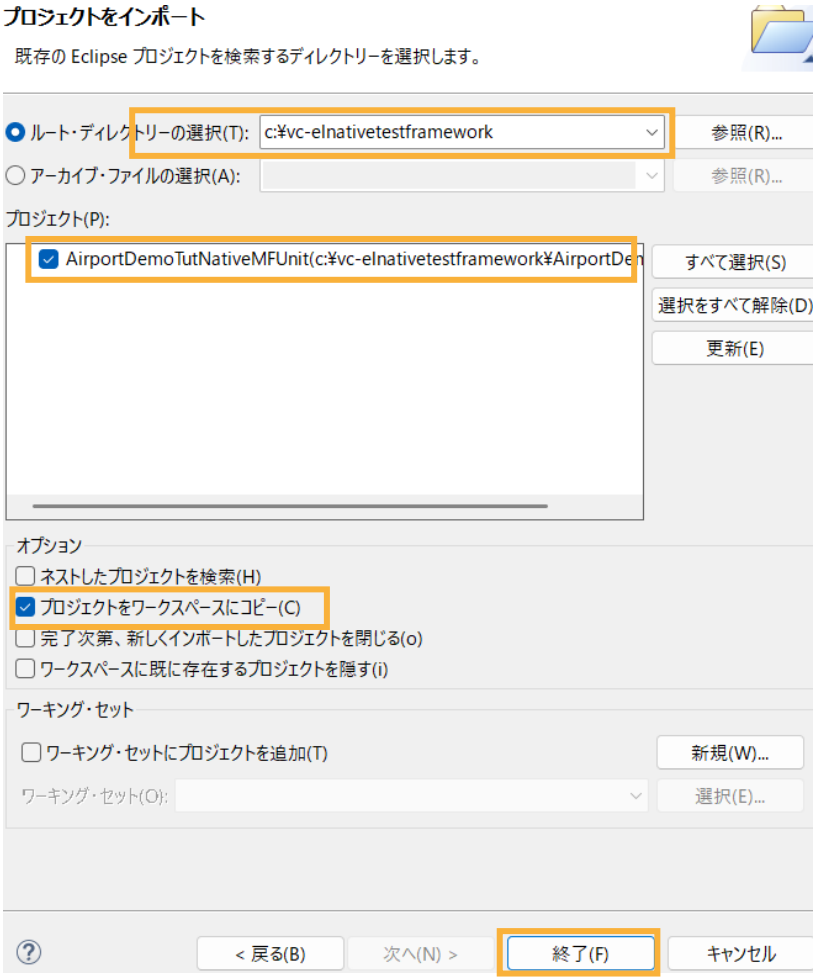


3) 以下の入力を行い、[終了(F)] をクリックします。

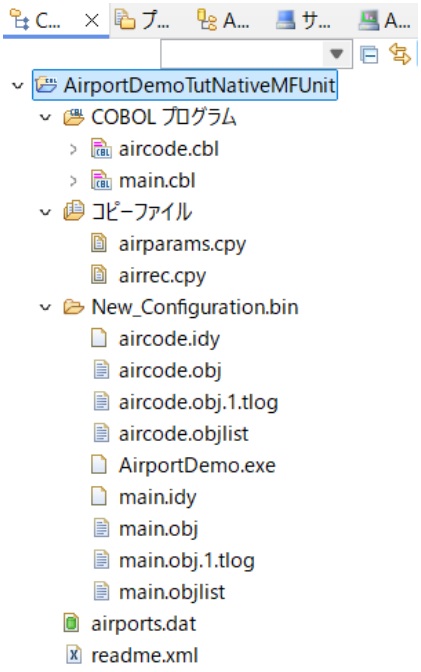
ルートディレクトリの選択：

[参照(R)] をクリックし、サンプルファイルを展開したフォルダー内の AirportDemoMFUnit フォルダーを指定

プロジェクトをワークスペースにコピー： チェックをつける



AirportDemoTutNativeMFUnit プロジェクトが作成されます。

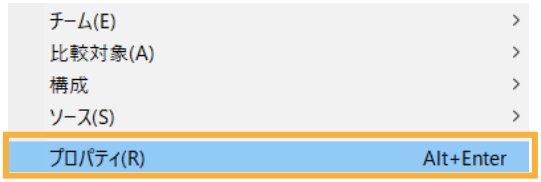


補足）

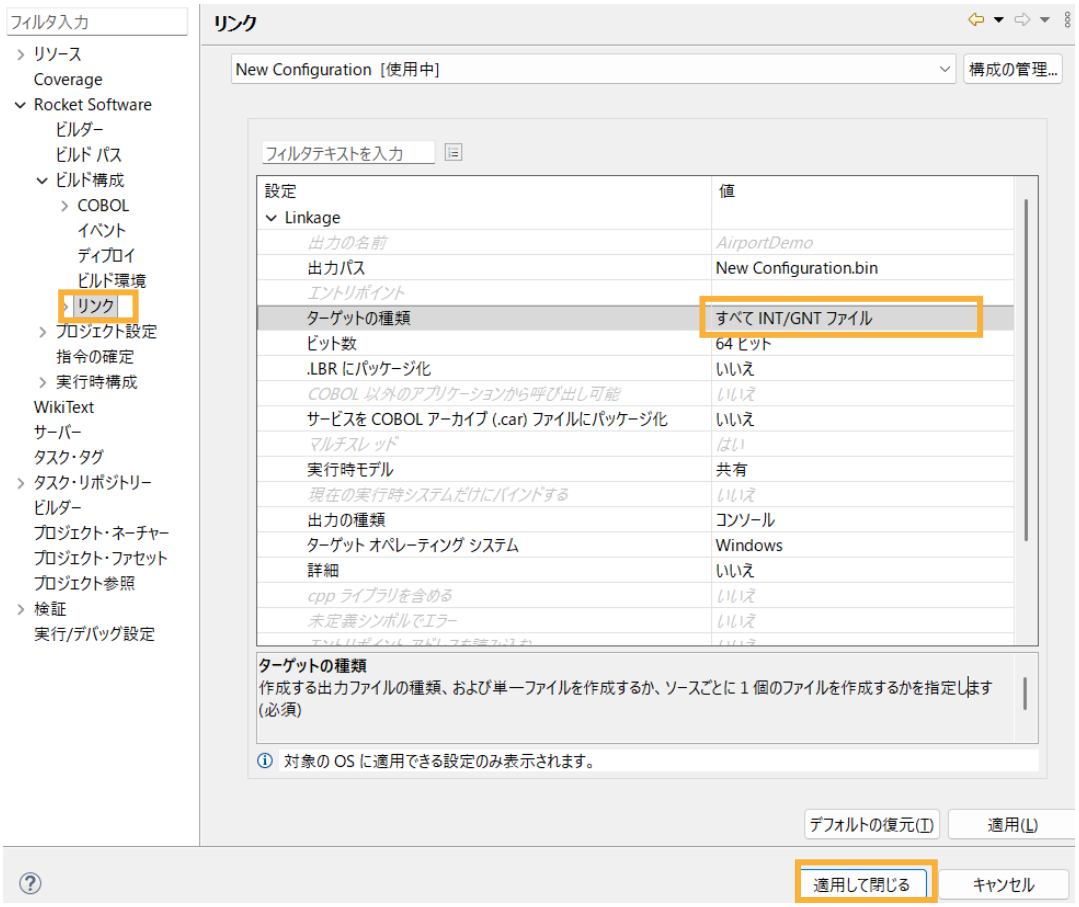
COBOL 開発を行うためには、COBOL パースペクティブという画面レイアウトを使用します。異なるパースペクティブを開いている場合、Eclipse IDE メニューの [ウィンドウ(W)] > [パースペクティブ(R)] > [パースペクティブを開く(O)] > [その他(o)] をクリックした上で、COBOL をクリックすることで、COBOL パースペクティブを開くことができます。

4) 単体テストを行なうために、出力形式を int 形式に変更します。以下の手順を実行してください。

AirportDemoTutNativeMFUnit プロジェクト名を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[プロパティ(R)] を選択します。



ツリーメニューより、[Rocket Software] > [ビルド構成] > [リンク] を選択し、「ターゲットの種類」を “すべて INT/GNT ファイル” に変更した上で、[適用して閉じる] をクリックします。



注意）

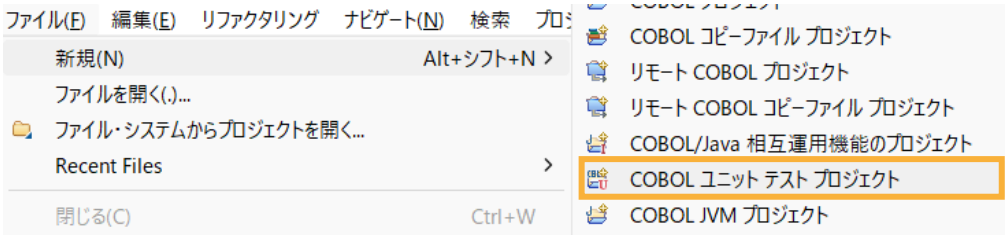
上記画面では、「ビット数」に “64 bit” を指定しています。“32 bit” 指定も可能ですが、その場合、以降の手順でも “32 bit” を選択してください。

3.1.2 基本的なテスト

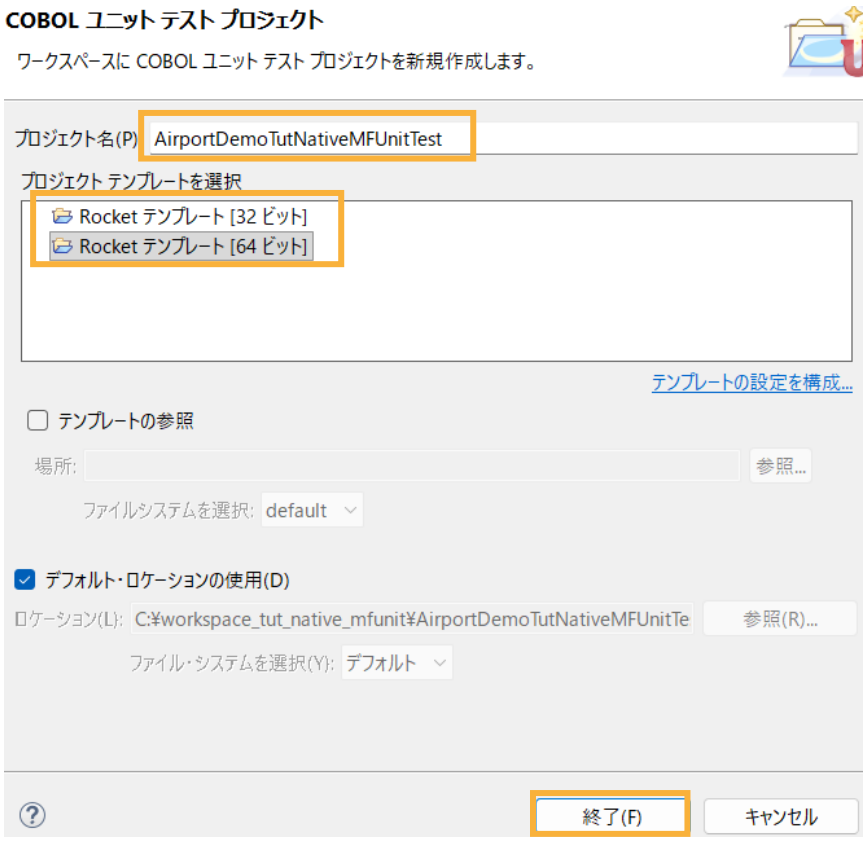
この方式では、１つのテストを１つのテストブロックで記述していきます。

3.1.2.1 MFUnit テストの作成

1) Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテストプロジェクト] を選択します。



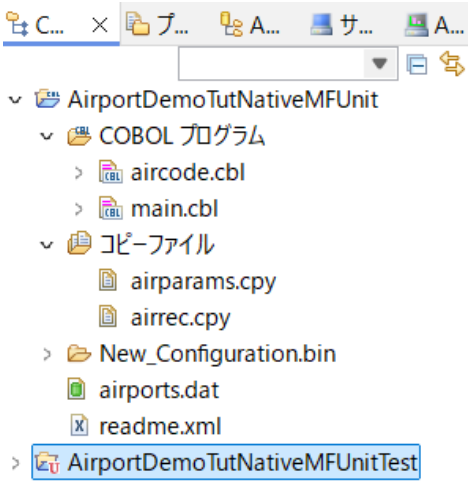
2) 「プロジェクト名」に “AirportDemoTutNativeMFUnitTest” を入力し、実行環境に合わせたプロジェクトテンプレートを選択した上で、[終了(F)] をクリックします。



注意）

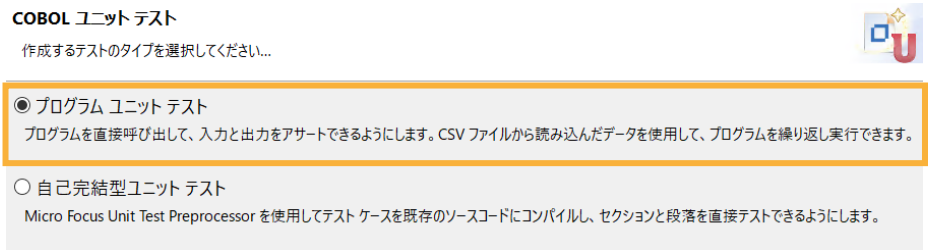
先行作業にて指定したビット数と同じテンプレートを選択してください。

プロジェクトが作成されます。



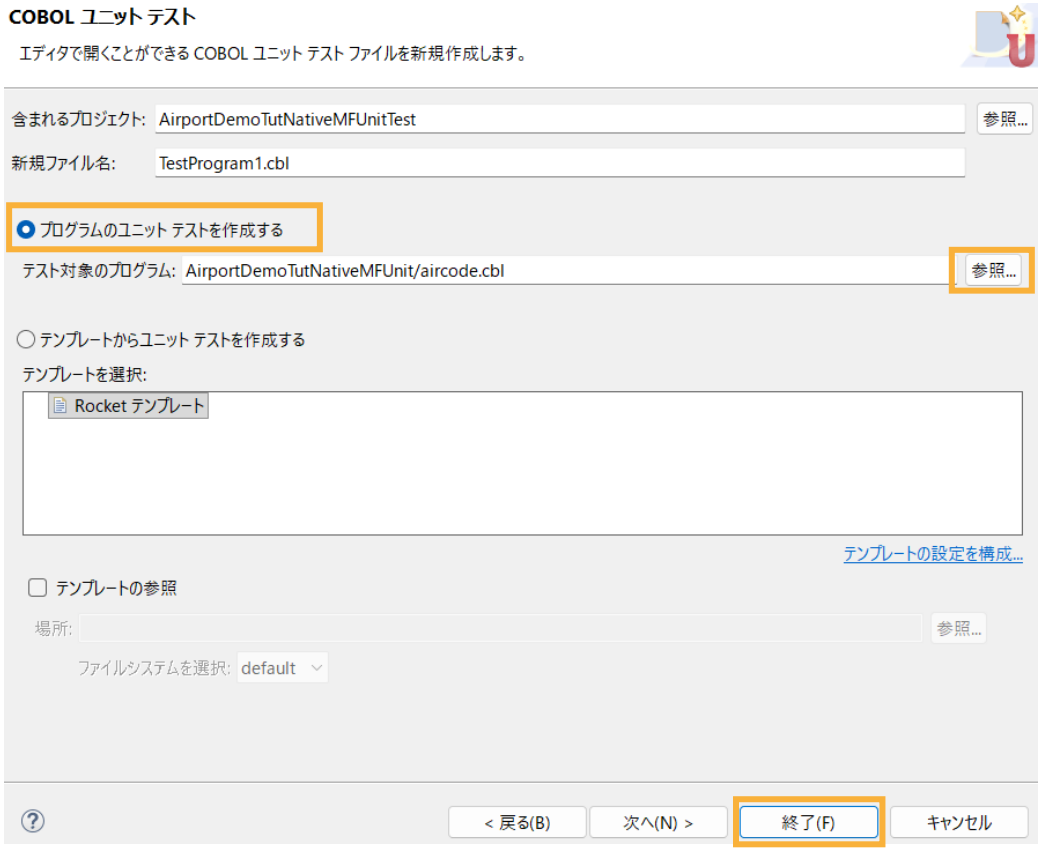
3) AirportDemoTutNativeMFUnitTest プロジェクトを選択した上で、Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテスト] を選択します。

4) [COBOL ユニット テストの新規作成] ウィンドウが表示されるので [プログラム ユニット テスト] 項目を選択し、[次へ(N)] をクリックします。

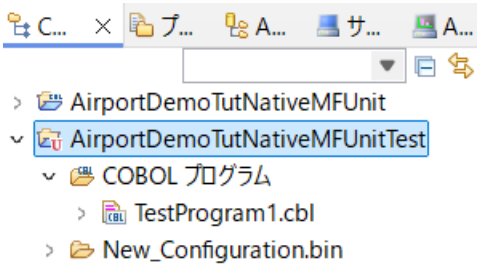


5) 以下の項目を入力し、[終了(F)] をクリックします。

- [プログラムのユニットテストを作成する] を選択
- [参照] をクリックして、AirportDemoTutNativeMFUnitTest プロジェクト内の「aircode.cbl」を選択

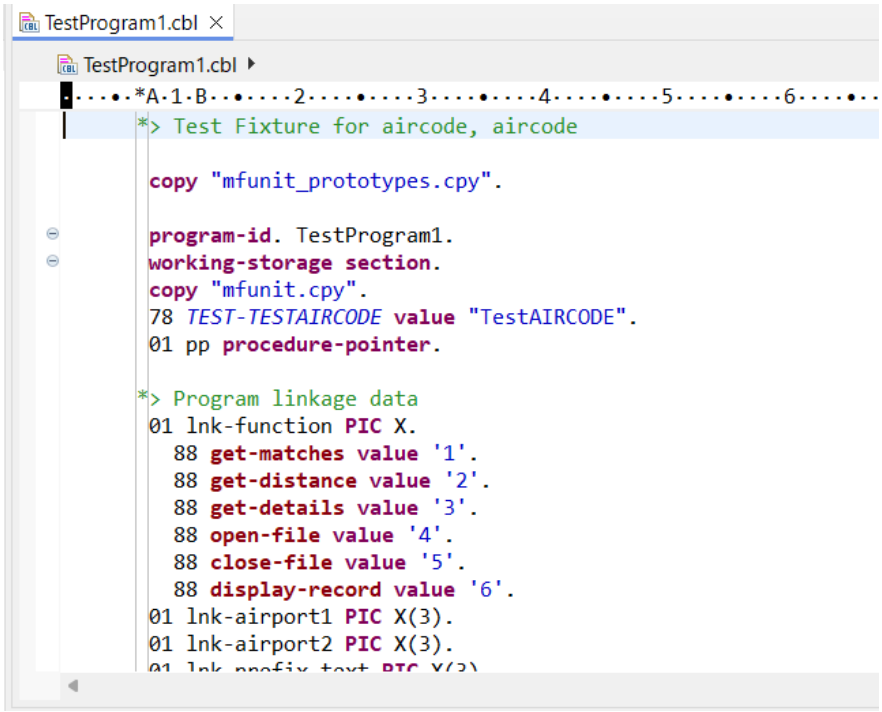


単体テストプログラムが作成されたことを確認します。



6) テストプログラムを確認します。

AirportDemoTutNativeMFUnitTest プロジェクト内の「TestProgram1.cbl」をダブルクリックして、コードを表示します。



以下のコードが記述されていることが分かります。

- entry MFU-TC-PREFIX & TEST-TESTAIRCODE
- entry MFU-TC-SETUP-PREFIX & TEST-TESTAIRCODE

MFUnit では、テストを下記のように決められた手順で実行しています。

テスト名 test1 を実行する場合、以下の順序で動作します。

① entry MFU-TC-SETUP-PREFIX & “test1”

② entry MFU-TC-PREFIX & “test1”

③ entry MFU-TC-TEARDOWN-PREFIX & “test1”

④ 次のテストを実行・・・

MFU-TC-SETUP-PREFIX で始まる entry にて、テストの前処理を定義できます。前処理の例として、ファイルの事前オープン処理などが考えられます。一方、MFU-TC-TEARDOWN-PREFIX で始まる entry では、テスト

実行後の処理を定義できます。前処理でオープンしたファイルをクローズするような処理が該当します。前処理、後処理ともに省略可能です。

自動生成されるテストプログラムはテンプレートであり、実際には、上記ルールに従い、テストを記述する必要があります。

7) 新規のテストケース（羽田・ロンドンヒースロー空間間の距離（km）のテスト）を追加した上で、実行を行ないます。

サンプルファイルを展開したフォルダー内の Basic¥TestProgram1.cbl の内容で、TestProgram1.cbl を上書きしてください。 これは、テストケース “testDistance” を途中まで作成しています。前述した MFU-TC-SETUP-PREFIX, MFU-TC-PREFIX, MFU-TC-TEARDOWN-PREFIX の 3 entry が追加されていますが、肝心な結果判定を記述していません。

結果判定処理を実装するため、82 行目に、以下のコードを追加して保存します。

```
            if distance-km = wk-distance-km
            then
                goback returning MFU-PASS-RETURN-CODE
            else
                string "expected " wk-distance-km ", but " distance-km z"" into err-ms
g end-string
                call MFU-ASSERT-FAIL-Z using err-msg
            end-if.
```

期待値である wk-distance-km (9591) と一致しているかを判定した上で、成功・失敗を戻します。

参考)

テスト失敗時の記述方法として、成功時同様に、戻り値で返す方法もあります。その場合は、以下のような例になります。

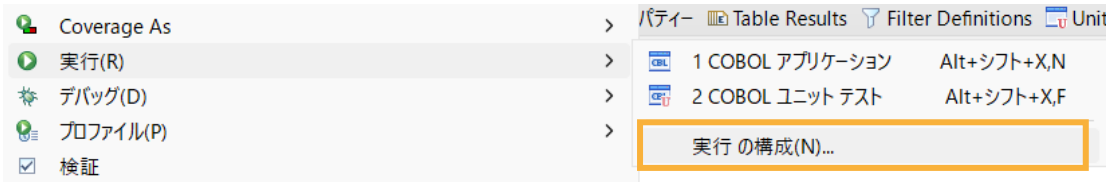
```
            if distance-km = wk-distance-km
            then
                goback returning MFU-PASS-RETURN-CODE
            else
                string "expected " wk-distance-km ", but " distance-km into err-msg end-
string
                display err-msg
                goback returning MFU-FAIL-RETURN-CODE
            end-if.
```

戻り値 MFU-FAIL-RETURN-CODE を利用する場合、テスト結果を確認するためにエラー情報を、display などで出力する必要があります。

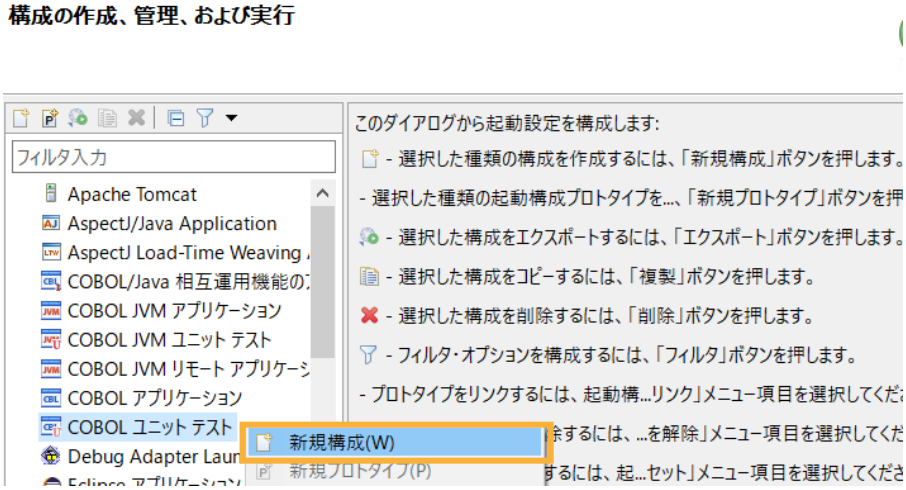
3.1.2.2 MFUnit テストの実行

本テスト対象のプログラムは、環境変数で設定された空港情報が保存されたデータファイルを参照するため、手順内で設定を行います。

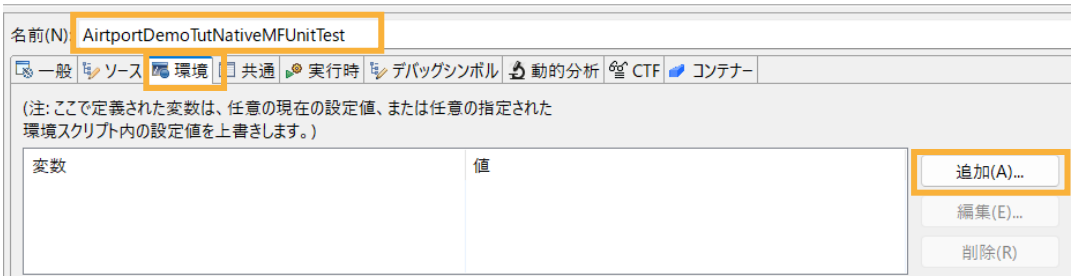
1) 「TestProgram1.cbl」を選択した状態で、マウスの右クリックでコンテクストメニューを表示し、[実行(R)] > [実行の構成(N)] をクリックします。



2) 画面左側より「COBOL ユニットテスト」を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[新規構成(W)] を選択します。



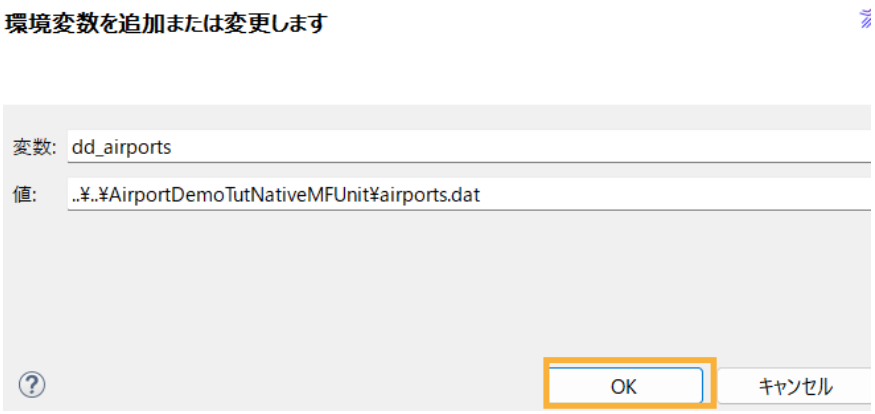
3) 「名前」に “AirtportDemoTutNativeMFUnitTest” を入力し、「環境」タブを選択した後、[追加(A)] をクリックします。



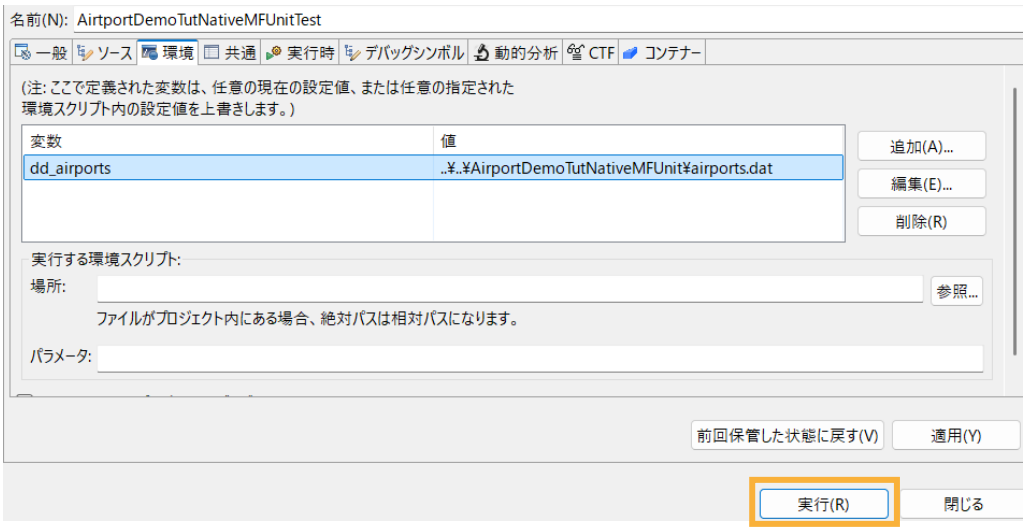
4) 以下の情報を入力し、[OK] をクリックします。

変数： “dd_airports”

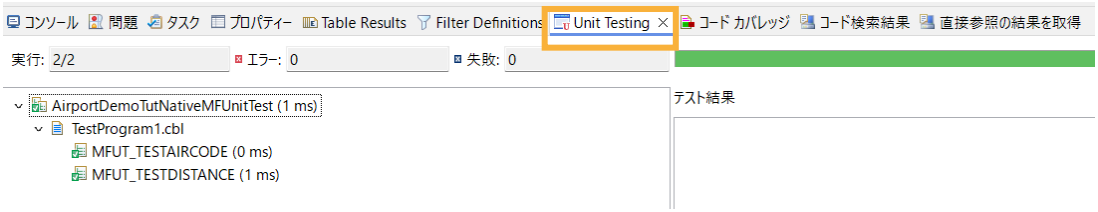
値： “..¥..¥AirportDemoTutNativeMFUnit¥airports.dat”



5) ささほど追加した dd_airports 環境変数が表示されていることを確認して、[実行(R)] をクリックします。

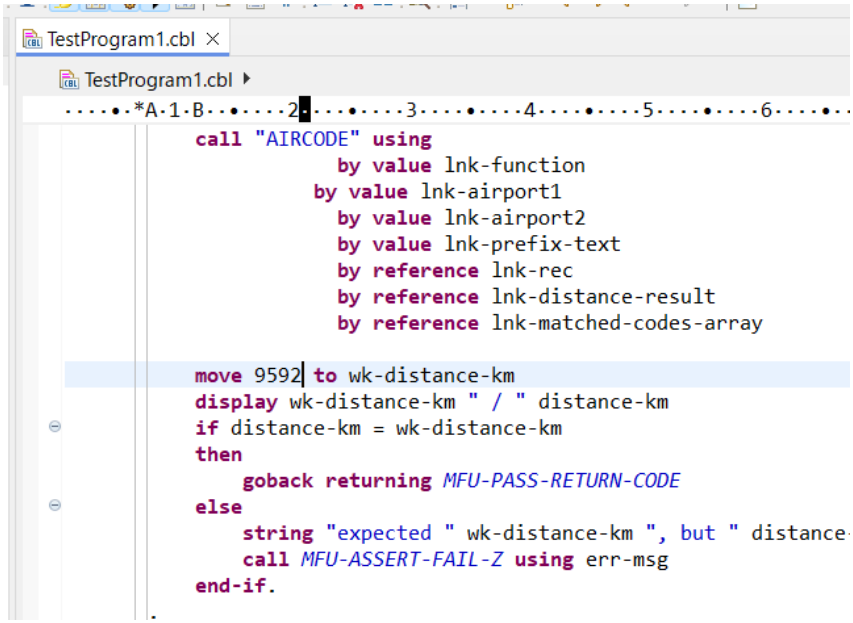


6) [Unit Testing] ビューをクリックすると、2 つのテストケースが緑色で表示されています。緑色は、テストが正常に終了したことを表しています。

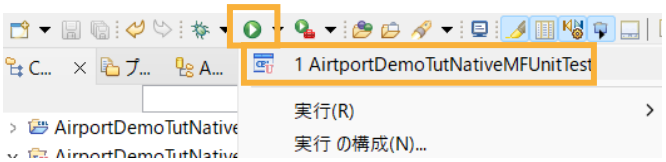


7) エラーケースを確認します。

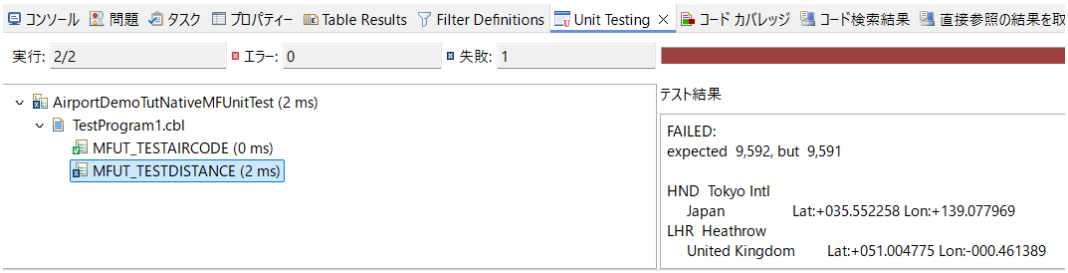
「TestProgram1.cbl」をエラーとなるように修正します。本例では、テストプログラム内に記載されていた期待値 9591 を 9592 に修正しています。



ツールバーより、[実行] アイコンの矢印をクリックし、「AirportDemoTutNativeMFUnitTest」をクリックします。



MFUT_TESTDISTANCE のテストで、エラーが発生したことが一覧から判断できます。

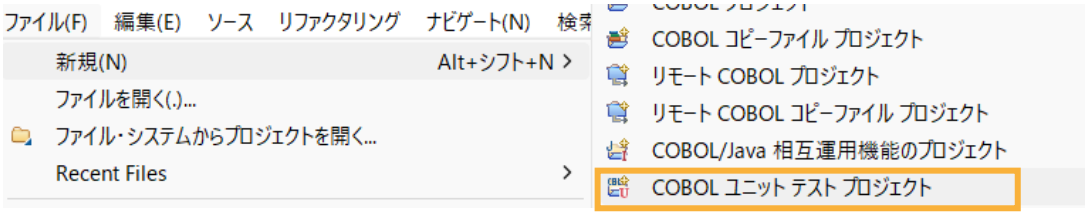


3.1.3 データ駆動型テスト

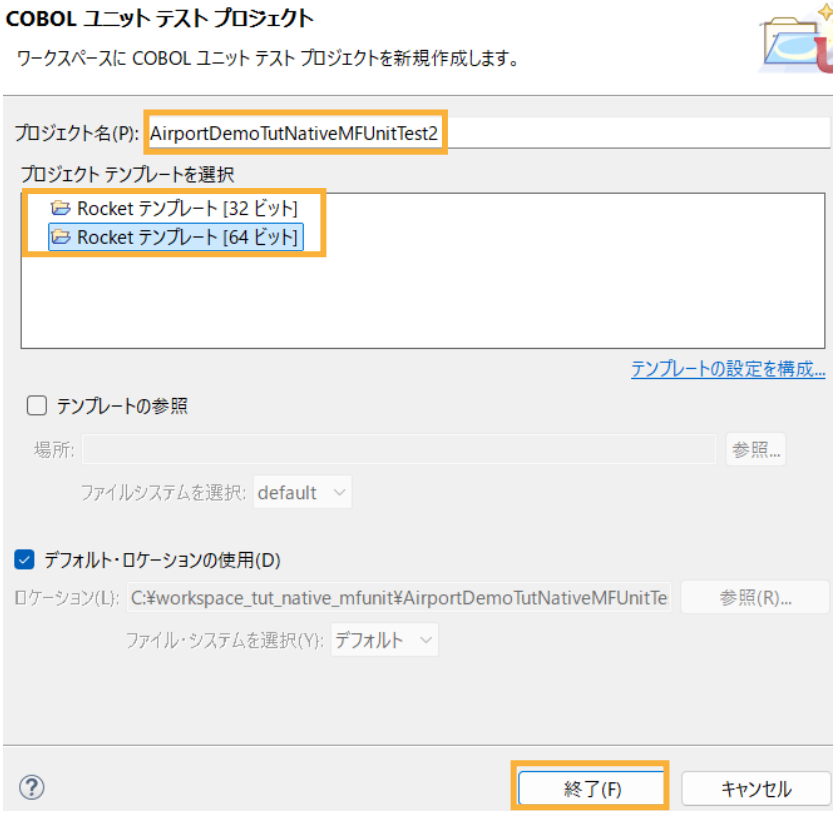
この方式では、複数のテストデータをデータファイルに設定しておくことで、データによって結果が異なるテストを効率よく行うことができます。

3.1.3.1 MFUnit テストの作成

1) Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテストプロジェクト] を選択します。



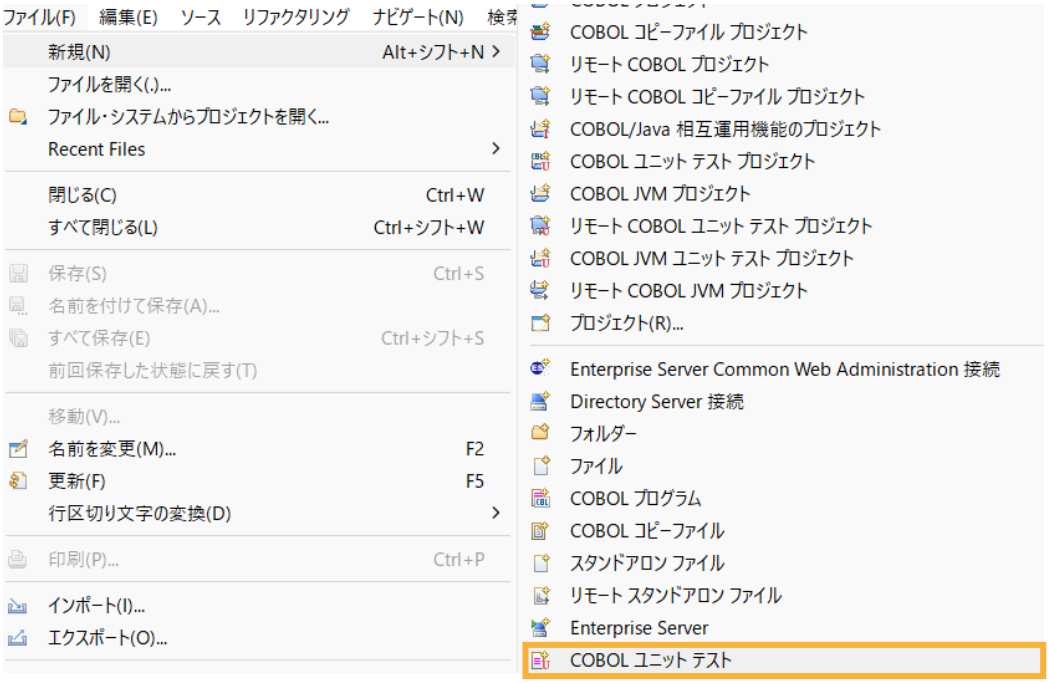
2) 「プロジェクト名」に “AirportDemoTutNativeMFUnitTest2” を入力し、実行環境に合わせたプロジェクトテンプレートを選択した上で、[終了(F)] をクリックします。



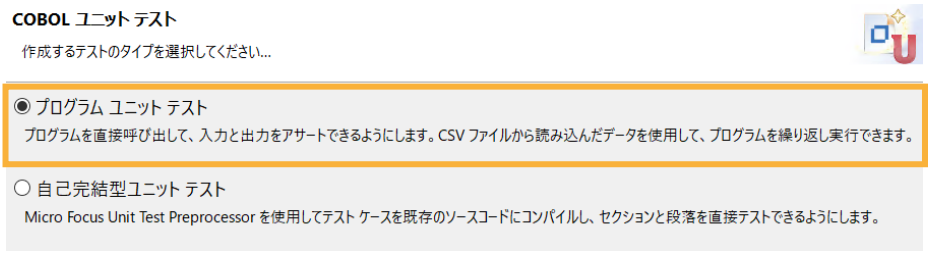
注意）

先行作業にて指定したビット数と同じテンプレートを選択してください。

3) AirportDemoTutNativeMFUnitTest2 プロジェクトを選択した上で、Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテスト] を選択します。

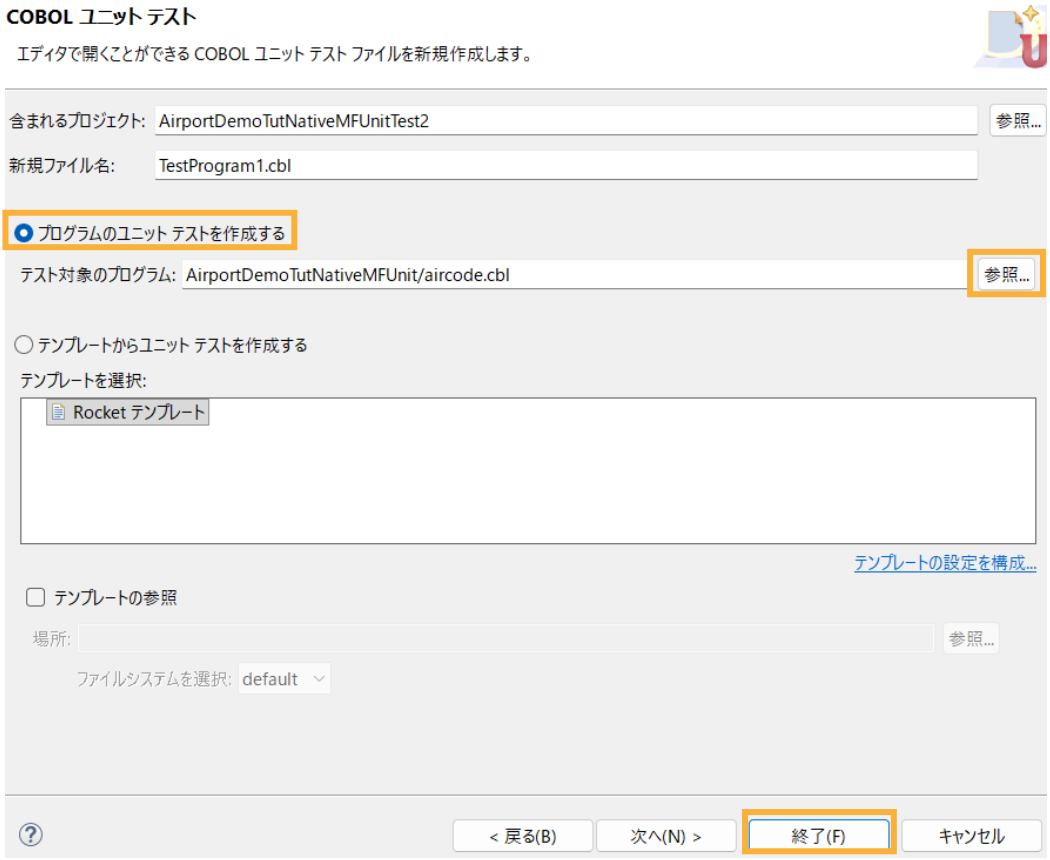


4) [COBOL ユニット テストの新規作成] ウィンドウが表示されるので [プログラム ユニット テスト] 項目を選択し、[次へ(N)] をクリックします。



5) 以下の項目を入力し、[終了(F)] をクリックします。

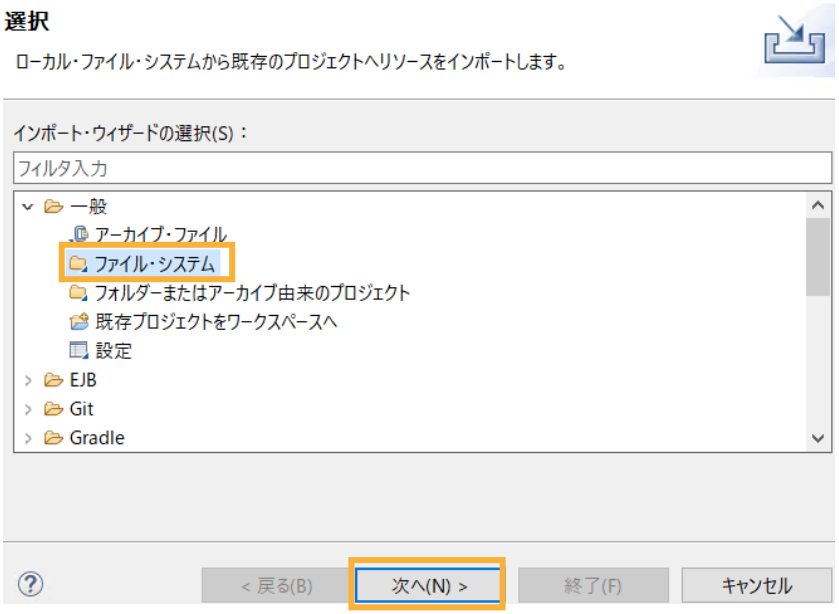
- [プログラムのユニットテストを作成する] を選択
- [参照] をクリックして、AirportDemoTutNativeMFUnitTest プロジェクト内の「aircode.cbl」を選択



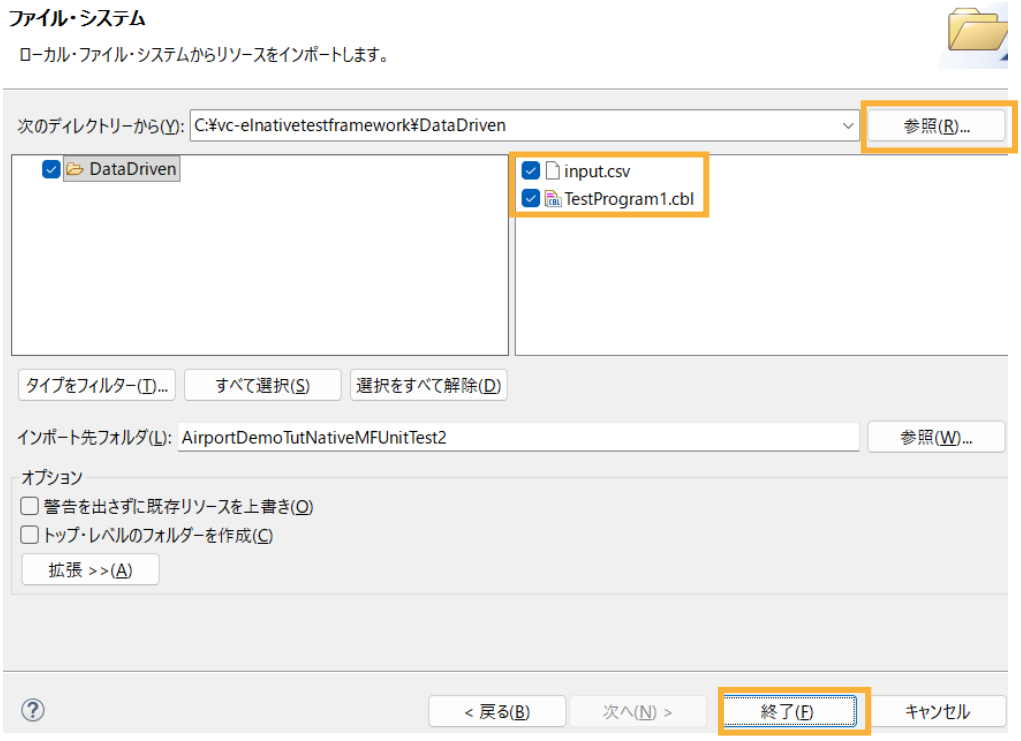
6) AirportDemoTutNativeMFUnitTest2 プロジェクト配下の TestProgram1.cbl を削除してください。

7) AirportDemoTutNativeMFUnitTest2 プロジェクトを選択したうえで、Eclipse IDE メニューから [ファイル(F)] > [インポート(I)] をクリックします。

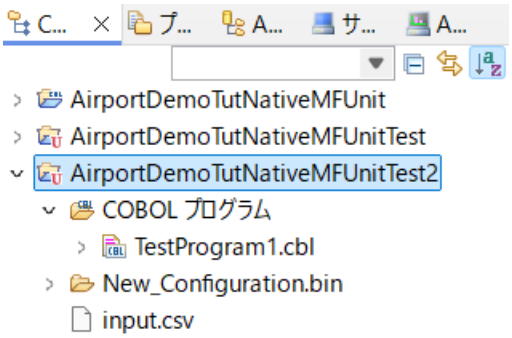
8) [一般] > [ファイル・システム] を選択し、[次へ(N)] をクリックします。



9) [参照(R)] をクリックした後、サンプルファイルを展開したフォルダー内の DataDriven を選択したうえで、「input.csv」,「TestProgram1.cbl」を選択、[終了(F)] をクリックします。



プロジェクトは以下のようになります。



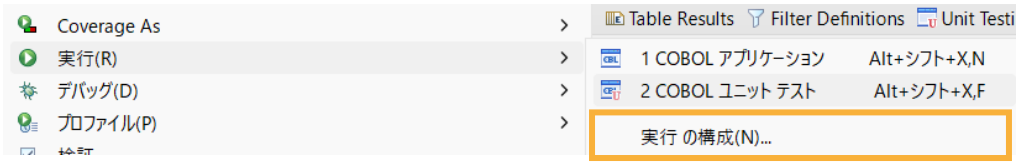
10) TestProgram1.cbl を開き、プログラムを確認します。

行数	説明
10 ～ 12	01 mfu-dd-airport1 is MFU-DD-VALUE external. 01 mfu-dd-airport2 is MFU-DD-VALUE external. 01 mfu-dd-distance-km is MFU-DD-VALUE external. テストデータファイル “input.csv” から取得する各テストデータの値が格納されます。
53	entry MFU-TC-PREFIX & TEST-TESTAIRCODE. 基本的な単体テストプログラムと同様の entry 句の構成ですが、テストデータ 1 行ごとに呼び出されるようになります。 また、基本的な単体テストプログラムでは、テスト失敗時には MFU-ASSERT-FAIL-Z を使用してエラー情報を戻していましたが、データ駆動型テストではこちらは使用できません。 このため、70 行目では goback での戻り値に MFU-FAIL-RETURN-CODE を設定しています。
77	entry MFU-TC-METADATA-SETUP-PREFIX & TEST-TESTAIRCODE. テストデータファイルを読み込みます。

3.1.3.2 MFUnit テストの実行

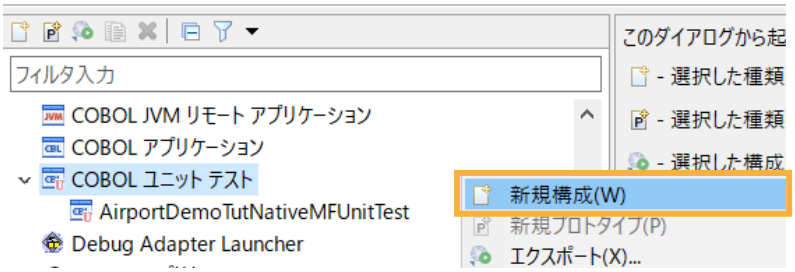
本テスト対象のプログラムは、環境変数で設定された空港情報が保存されたデータファイルを参照するため、手順内で設定を行います。

1) AirportDemoTutNativeMFUnitTest2 プロジェクト配下の「TestProgram1.cbl」を選択した状態で、マウスの右クリックでコンテクストメニューを表示し、[実行(R)] > [実行の構成(N)] をクリックします。

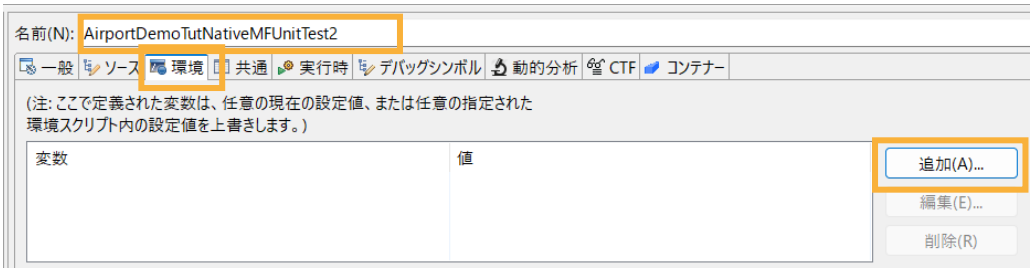


2) 画面左側より「COBOL ユニットテスト」を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[新規構成(W)] を選択します。

構成の作成、管理、および実行



3) [名前] に “AirtportDemoTutNativeMFUnitTest2” を入力し、 [環境] タブを選択した後、[追加(A)] をクリックします。

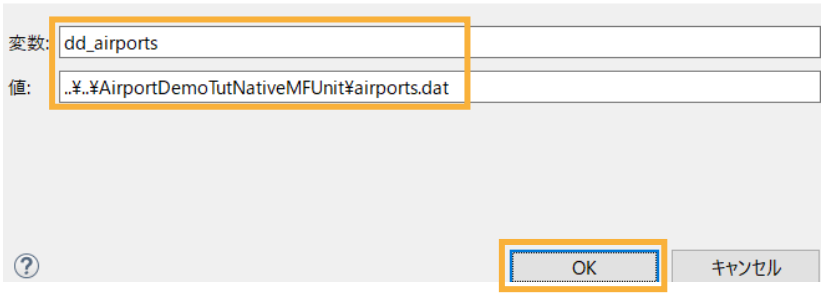


4) 以下の情報を入力し、[OK]をクリックします。

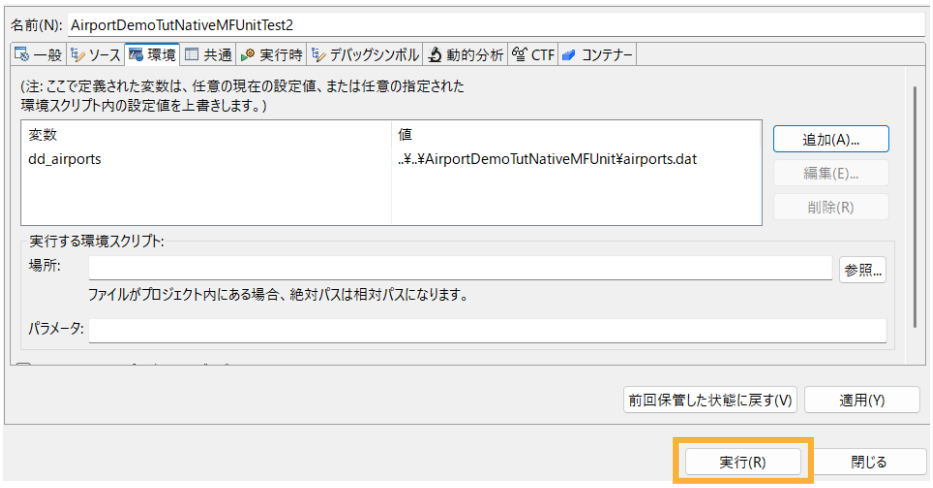
変数： “dd_airports”

値： “..¥..¥AirportDemoTutNativeMFUnit¥airports.dat”

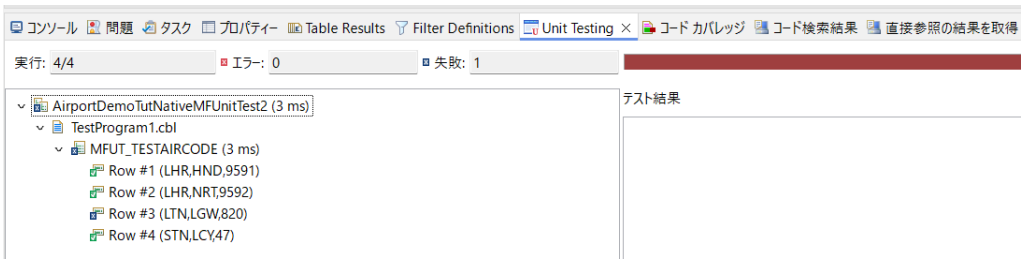
環境変数を追加または変更します



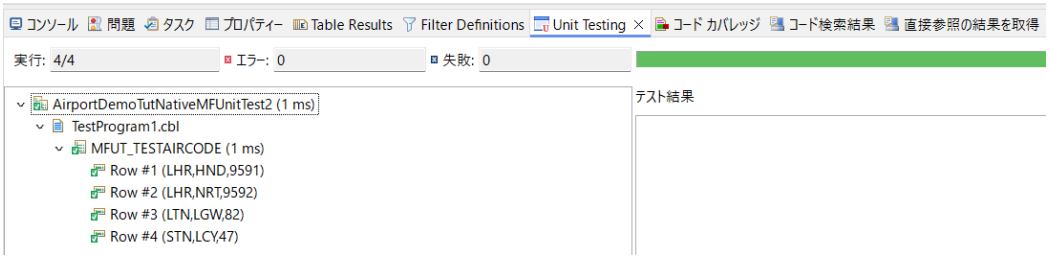
5) さきほど追加した dd_airports 環境変数が表示されていることを確認して、[実行(R)] をクリックします。



実行後に [Unit Testing] ビューを表示すると、以下のように 1 つテストが失敗しています。



テストデータファイルの 4 行目 LTN, LGW の距離がエラーとなっています。これは、テストデータファイルの 820 が誤りであり、正しい値は 82 です。テストデータファイルの 820 を 82 に修正したうえで、テストを再実行すると、以下のように全て成功します。

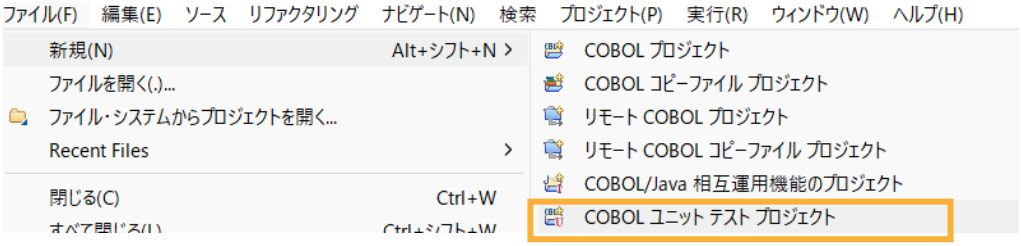


3.1.4 自己完結型テスト

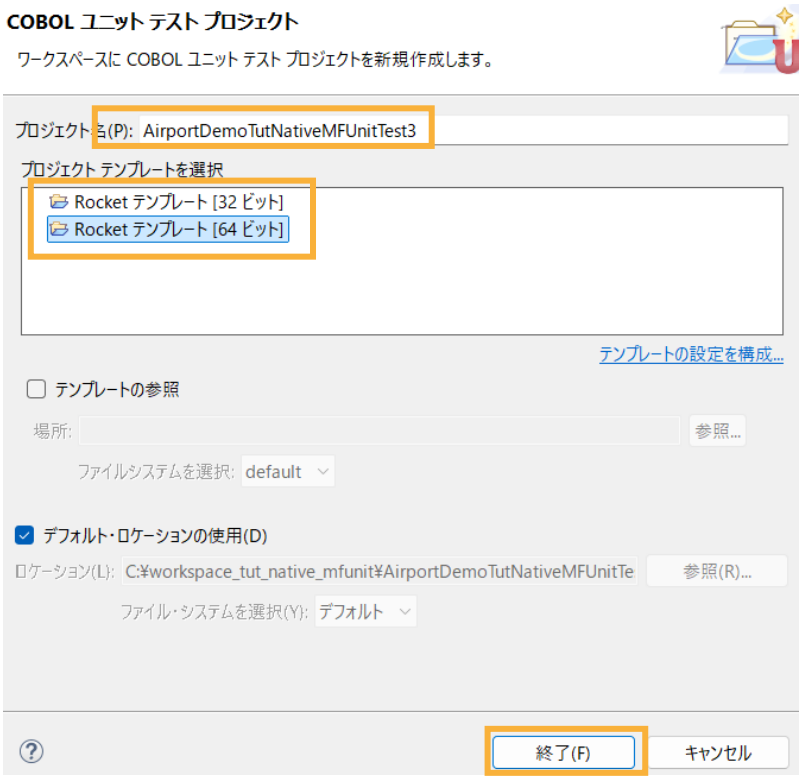
この方式では、テスト対象となるプログラム内にテストコードをコンパイル時に埋め込むことで、より粒度の小さなテストを行えます。

3.1.4.1 MFUnit テストの作成

1) Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテストプロジェクト] を選択します。



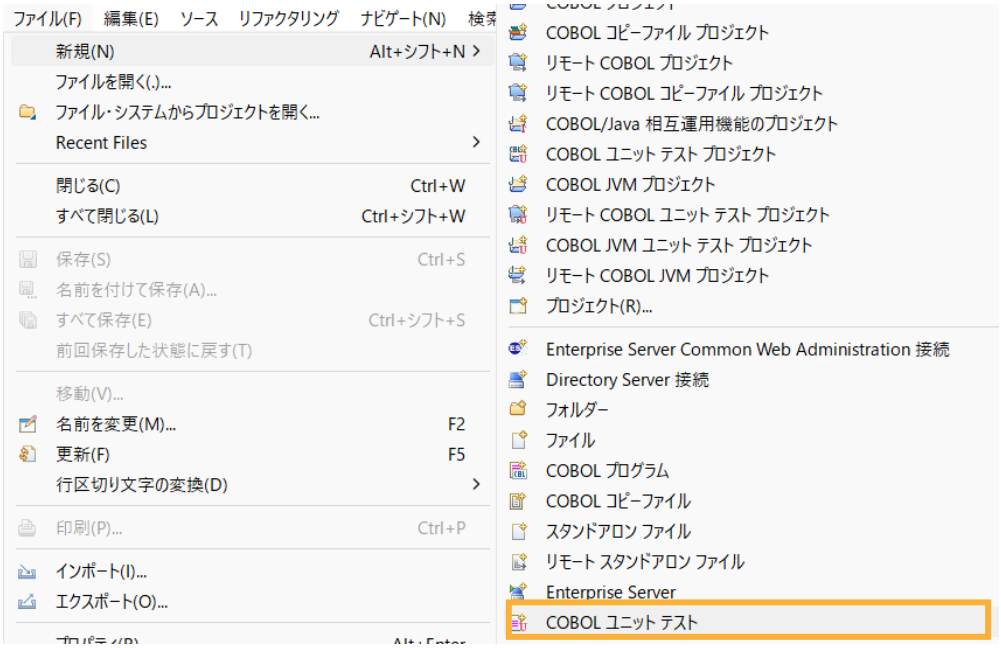
2) 「プロジェクト名」に “AirportDemoTutNativeMFUnitTest3” を入力し、実行環境に合わせたプロジェクトテンプレートを選択した上で、[終了(F)] をクリックします。



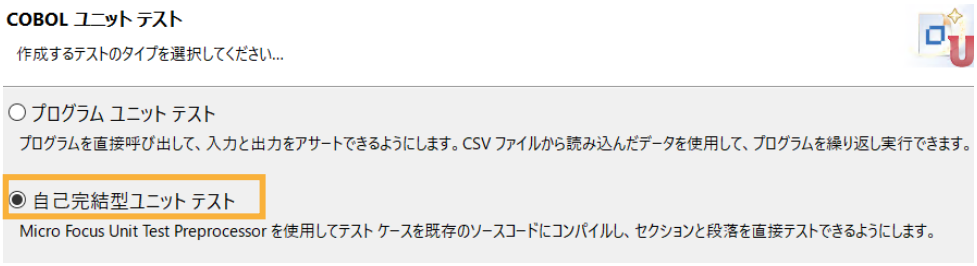
注意）

先行作業にて指定したビット数と同じテンプレートを選択してください。

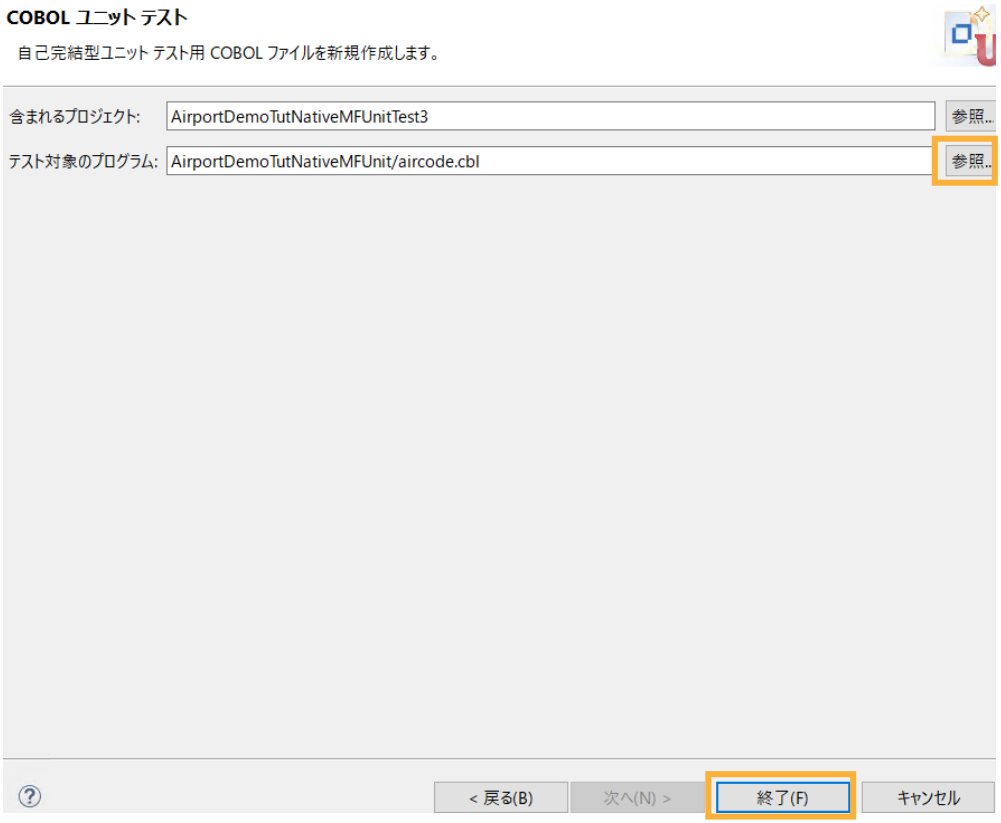
3) AirportDemoTutNativeMFUnitTest3 プロジェクトを選択した上で、Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL ユニットテスト] を選択します。



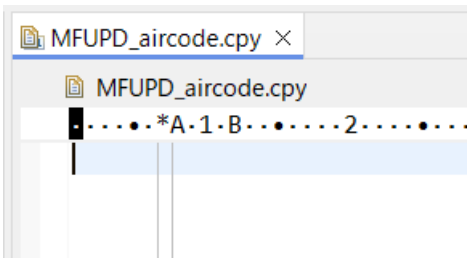
4) [自己完結型ユニットテスト] を選択して[次へ(N)]をクリックします。



5) テスト対象のプログラム欄の右にある [参照...]をクリックしたうえで、AirportDemoTutNativeMFUnit プロジェクト配下の aircode.cbl を選択し、[終了(F)] をクリックします。



以下のような空のコピーブックファイルが作成されます。



自己完結型のテストでは、このコピーブックがテスト対象プログラム（本手順では aircode.cbl）内にコンパイル時に埋め込まれます。このため、テストコードに必要なデータ項目は LINKAGE SECTION で定義された項目ではなく、WORKING-STORAGE SECTION 内で定義された実体のある項目で記述されている必要があります。

サンプルファイルを展開したフォルダー内の Self¥MFUPD_aircode.cpy の内容で、MFUPD_aircode.cpy を上書きしてください。

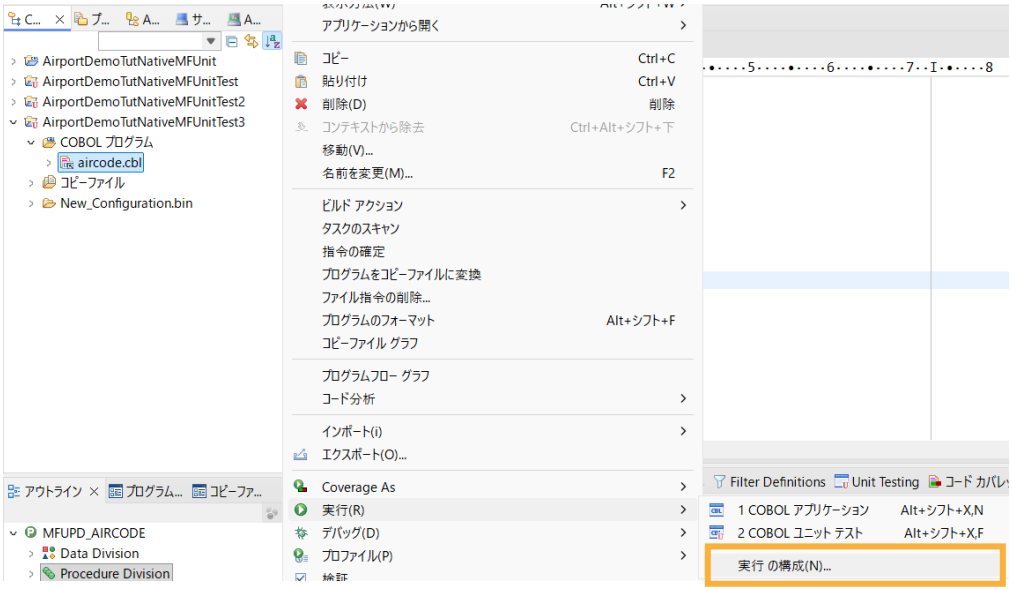
```
entry “MFUT_TESTAIRCODE”.
    perform open-airfile
    if fstat-1 not= 0
        goback returning 1
    end-if
        goback
.
entry “MFUT_TESTAIRCODE2”.
    perform close-airfile
    goback
.
```

この例では、open-airfile と close-airfile section のテストを行っています。open-airfile のテストコードを確認すると、fstat-1 項目の値を直接参照して処理結果を評価していることが確認できます。一方、close-airfile は特に判定材料がないため、常にテストは成功します。

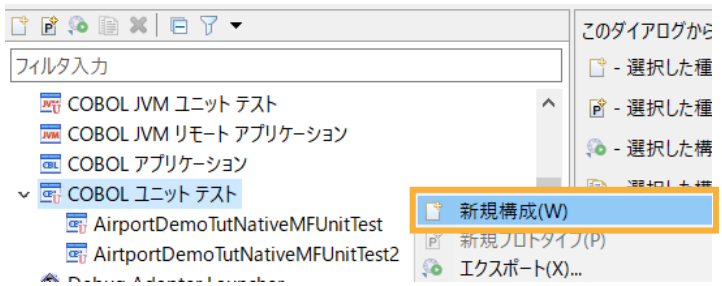
3.1.4.2 MFUnit テストの実行

本テスト対象のプログラムは、環境変数で設定された空港情報が保存されたデータファイルを参照するため、手順内で設定を行います。

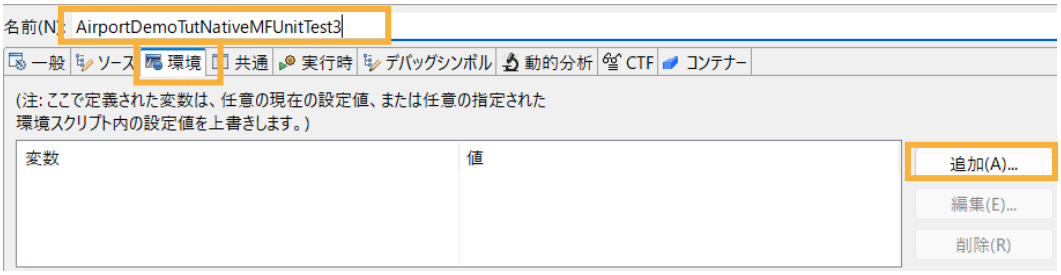
1) AirportDemoTutNativeMFUnitTest3 プロジェクト配下の「aircode.cbl」を選択した状態で、マウスの右クリックでコンテクストメニューを表示し、[実行(R)] > [実行の構成(N)] をクリックします。



2) 画面左側より「COBOL ユニットテスト」を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[新規構成(W)] を選択します。



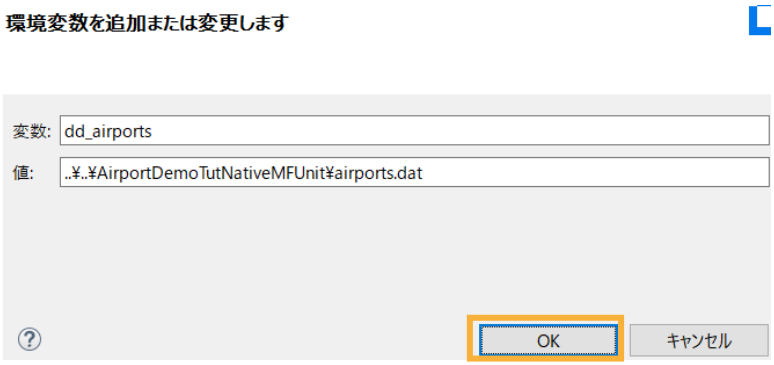
3) [名前] に “AirtportDemoTutNativeMFUnitTest3” を入力し、 [環境] タブを選択した後、[追加(A)] をクリックします。



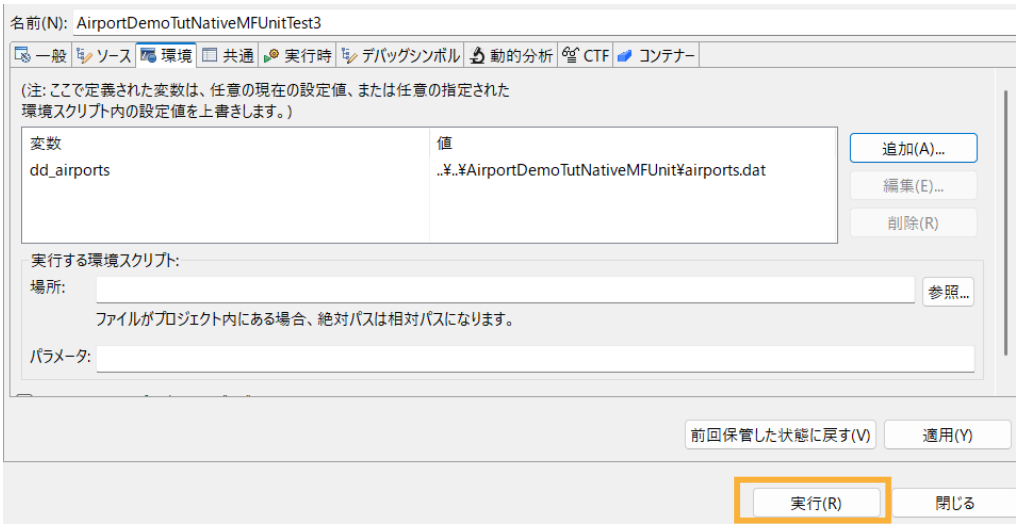
4) 以下の情報を入力し、[OK]をクリックします。

変数： “dd_airports”

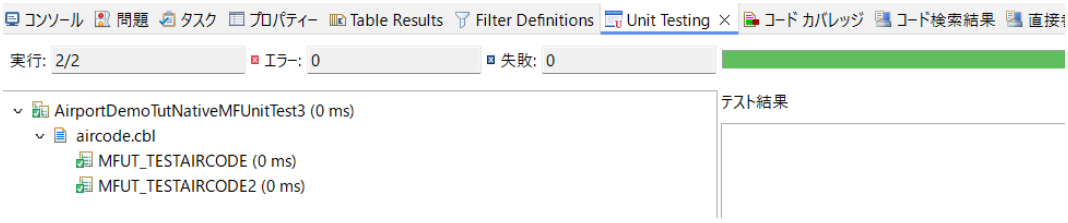
値： “..¥..¥AirportDemoTutNativeMFUnit¥airports.dat”



5) さきほど追加した dd_airports 環境変数が表示されていることを確認して、 [実行(R)] をクリックします。



実行後に［Unit Testing］ビューを表示すると、以下のように２つのテストがともに成功していることが確認できます。



3.2 コマンドラインからの実行

MFUnit によるテストは、Eclipse 上の画面からではなく、コマンドライン上からも行なうことができます。従来のスタイルでのテスト作業の効率化を図ることができ、Jenkins などの CI ツールと連携する事で、テストの自動実行を行なえるため、品質担保や作業工数の削減が見込めます。

どのテスト方式でも実行方法は同じであるため、ここでは 3.1.2 で作成したテストプログラムをコマンドラインから実行する方法について学びます。

1) スタートメニューより [Rocket Visual COBOL] > [Visual COBOL Command Prompt (64-bit)] をクリックします。

注意）
3.1.2 で指定したビット数と同じテンプレートを選択してください。

2) 作業フォルダーを作成し、作成したフォルダーに移動します。

```
C:¥>mkdir ¥VCCommandTutorial && cd ¥VCCommandTutorial
C:¥VCCommandTutorial>
```

3) 3.1 で利用したワークスペースフォルダーを ECLIPSE_WORKSPACE に指定した上で、下記コマンドを実行します。

- set ECLIPSE_WORKSPACE=c:¥workspace_tut_native_mfunit
- cobol %ECLIPSE_WORKSPACE%¥AirportDemoTutNativeMFUnit¥aircode.cbl;
- cobol %ECLIPSE_WORKSPACE%¥AirportDemoTutNativeMFUnitTest¥TestProgram1.cbl sourceformat(variable);
- cbllink -D aircode.obj
- cbllink -D TestProgram1.obj

```
C:¥VCCommandTutorial>set ECLIPSE_WORKSPACE=c:¥workspace_tut_native_mfunit
C:¥VCCommandTutorial>cobol %ECLIPSE_WORKSPACE%¥AirportDemoTutNativeMFUnit¥aircode.cbl;
（実行中のログを省略）
C:¥VCCommandTutorial>cobol %ECLIPSE_WORKSPACE%¥AirportDemoTutNativeMFUnitTest¥TestProgram1.cbl sourceformat(variable);
（実行中のログを省略）
C:¥VCCommandTutorial>cbllink -D aircode.obj
（実行中のログを省略）
C:¥VCCommandTutorial>cbllink -D TestProgram1.obj
（実行中のログを省略）
C:¥VCCommandTutorial>
```

以下のファイルが作成されていることを確認します。

```
C:¥VCCommandTutorial>dir
 ドライブ C のボリューム ラベルがありません。
 ボリューム シリアル番号は 3837-A4A9 です
 C:¥VCCommandTutorial のディレクトリ
2025/11/19  15:34    <DIR>          .
2025/11/19  15:34            17,920 aircode.dll
2025/11/19  15:34            10,126 aircode.obj
2025/11/19  15:34            16,384 TestProgram1.dll
2025/11/19  15:34             8,164 TestProgram1.obj
```

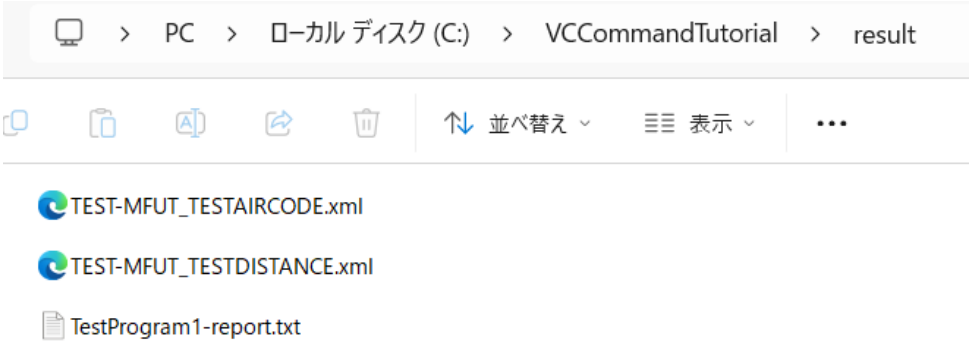
```
               4 個のファイル                 52,594 バイト
               1 個のディレクトリ  84,190,031,872 バイトの空き領域
C:¥VCCommandTutorial>
```

4) プロンプト上で下記コマンドを実行し、MFUnit を実行します。

- set dd_airports=%ECLIPSE_WORKSPACE%¥AirportDemoTutNativeMFUnit¥airports.dat
- mfurun -report:junit -outdir:result TestProgram1.dll

```
C:¥VCCommandTutorial>set dd_airports=%ECLIPSE_WORKSPACE%¥AirportDemoTutNativeMF
Unit¥airports.dat
C:¥VCCommandTutorial>mfurun -report:junit -outdir:result TestProgram1.dll
Rocket (R) COBOL - mfurun Utility
Unit Testing Framework for Windows/Native/64
Fixture : TestProgram1
Test Run Summary
 Overall Result            Passed
 Tests run                  2
 Tests passed               2
 Tests failed               0
 Total execution time       0
C:¥VCCommandTutorial>
```

outdir オプションにより、出力結果が result フォルダーに保存されます。



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。