

# Micro Focus Visual COBOL チュートリアル

# COBOL 開発: Eclipse - ネイティブ COBOL 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

- 本チュートリアルで使用したマシン OS : Windows 10
- Micro Focus Visual COBOL 7.0 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、任意のフォルダに解凍しておいてください。 サンプルプログラムのダウンロード



# 内容

- 1. 目的
- 2. 前提
- 3. チュートリアル手順の概要
- 3.1. IDE からの実行
  - 3.1.1. Eclipse の起動
  - 3.1.2. チュートリアルプロジェクトのインポート
  - 3.1.3. MFUnit テストの作成
  - 3.1.4. MFUnit テストの実行
- 3.2. コマンドラインからの実行



# 3. チュートリアル手順の概要

# 3.1. IDE からの実行

# 3.1.1. Eclipse 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)] ボタンをクリックします。

O Eclipse ランチャー	×
ディレクトリーをワークスペースとして選択	
Eclipse は、ワークスペースディレクトリを使用して、環境設定と開発成果物を保存します。	
$\Box_{-}$ / $\Delta T_{-}^{R}$ - 7000. C/Ywarkspace tut pative mfunit	关照(D)
	1≫ XR ( <u>D</u> )
□ この避損をデコナルトレース使用し、会後この質問を書テしたい(1)	
▶ <u>最近のリークスペース(R)</u>	
起動(L)	キャンヤル



# 3.1.2. 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。
  - ファイル(F) 編集(E) リファクタリング ナビゲート(N) 検索 プロジェクト(P) 新規(N) Alt+シフト+N > 🕴 🚽 💡 ファイルを開く(.)... サーバー □ ファイル・システムからプロジェクトを開く... × [ 最近のファイル > 閉じる(C) Ctrl+W すべて閉じる(L) Ctrl+シフト+W 📄 保管(S) Ctrl+S 圖 別名保存(A)... 💼 すべて保管(E) Ctrl+シフト+S 前回保管した状態に戻す(T) 移動(V)... F2 ▶ 名前を変更(M)... F5 🜒 更新(F) 行区切り文字の変換(D) > 🕒 印刷(P)... Ctrl+P ≥」 インポート(I)... 🖆 エクスポート(O)... プロパティ(R) Alt+Enter
- [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) >] ボタンをクリックします。

Ľ
2
1
*



3) 「ルート・ディレクトリーの選択(T)」欄に、チュートリアルプロジェクトへのパスを指定します。 [参照(R)] ボタンを押してサンプ ルファイルを展開したフォルダ内の AirportDemoMFUnit フォルダを指定してください。

● インポート	_		×
プロジェクトのインボート 既存の Eclipse プロジェクトを検索するディレクトリーを選択します。			
<ul> <li>・ディレクトリーの選択(I):</li> <li>アーカイブ・ファイルの選択(A):</li> </ul>	~	<b>参照(<u>R</u></b> 参照( <u>R</u>	) ))
プロジェクト( <u>P</u> ):	選折	すべて選択( Rをすべて解 更新( <u>E</u> )	S) (除( <u>D</u> )

4) 「プロジェクトをワークスペースにコピー(C)」項目にチェックを入れた上で、[終了(F)] ボタンをクリックします。

シルート・ティレクトリーの選択(刀)	C#vc-einativetestframework#AirportD	✓ 参照(R)
)アーカイブ・ファイルの選択(A):		
7ロジェクト(P):		
AirportDemoTutNative	MFUnit(Cr#vc-eInativetestframework#Ai	すべて銀沢(S)
		<b>潮沢をすべて解除()</b>
		更新(E)
<ul> <li>ワロジェクトをワークスペースにこ</li> <li>Close newly imported proj</li> <li>ロークスペースに既に存在する。</li> </ul>	ビー(C) ects upon completion 7Dにナクトを届す(1)	
ワーキング・セット		
□ ワーキング・セットにプロジェクト	· 在追加(T)	新規(W)
		1766.
2-107-101-101		



以下のダイアログが表示された場合、そのまま [OK] ボタンをクリックします。 ● 
響き新し、愛愛保存
X



AirportDemoTutNativeMFUnit プロジェクトが作成されることを確認します。



#### 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。



 5) 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。

\$	更新(F) プロジェクトを閉じる(S) Close Unrelated Project	F5
	リモートシステムビューで表示 検証(V)	
Q	Coverage As	>
0	実行(R)	>
*	デバッグ(D)	>
	プロファイル(P)	>
	ローカル履歴から復元(Y)	
	チーム(E)	>
	比較対象(A)	>
	構成	>
	ソース(S)	>
	プロパティ(R)	Alt+Enter

ッリーメニューより、[Micro Focus] > [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] ボタンをクリックします。

New Configuration [使用中]	✓ 構成の管理.
7-11/97年ストを入力	
記定 → Linkage	* D
ニカパス エントリポイント	New Configuration.bin
ターゲットの種類	#AT INT/GNT 2#416
ビット家	64 ピット
LBRにパッケージ化	1117
サービスを COBOL アーカイブ (.car)	ファイルにパッケージ化 いいえ *
ターゲットの種類 (生成する出力ファイルの種類、および単一フ 定します(必須) デフォルト値:単一変行可能ファイル	アイルモ作成するか、ソースごとに1 僕のファイルモ作成するかを操
	デフォルトの保元(1) 適用(1)
	アイルタテキストを入力     設定     シ Linkage     ニュニッドッド     シーンドリポイント     ターグットの種類     ビット数     ムSR にパッケージ化     サービスを COBOL アーカイブ (.can)     ワープットの種類     作成する出力ファイルの種類、および単一フ     定します (必須)     デフォルト磁:単一集行可能ファイル

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" を選択していただく必要があります。



# 3.1.3. 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。

ファイ	ル(F)	編集(E)	リファクタリング	ナビゲート(N)	検索	プロジ	፤//	(P) 実行(R) ウィンドウ(W) ヘルプ(H)
	新規(	N)		А	lt+シフト	+N >	2	COBOL プロジェクト
	ファイノ	レを開く(.)					📸	COBOL コピーファイル プロジェクト
	ファイノ	レ・システム	からプロジェクトを開	罰く			۲ <b>Ω</b>	リモート COBOL プロジェクト
	最近0	のファイル				>	ġ	リモート COBOL コピーファイル プロジェクト
	閉じる	(C)			Ctrl	+W	Ê	COBOL ユニット テスト プロジェクト
	すべて	問じる(L)		Ct	rl+シフト	+W	曾	COBOL JVM プロジェクト

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)] ボタンをクリックします。

● COBOL ユニット テスト プロジェクトの新規作成	_		×
<b>COBOL ユニット テスト プロジェクト</b> ワークスペースに COBOL ユニット テスト プロジェクトを新規作成します。		7	
プロジェクト名(P): AirportDemoTutNativeMFUnitTest プロジェクト テンプレートを選択 ② Micro Focus テンプレート [32 ビット] ③ Micro Focus テンプレート [64 ビット]			
<b> テンプレートの参照</b> 場所: ファイルシステムを選択: default 〜	<del>7274-1</del>	<u>の設定を構</u> 参照	<u>E.</u>
☑ デフォルト・ロケーションの使用(型)			
ロケーション( <u>し</u> ): C:¥workspace_tut_native_mfunit¥AirportDemoTut ファイル・システムを選択( <u>Y</u> ): <mark>デフォルト</mark> 〜	tNative№	参照( <u>R</u> )	
?	了( <u>F</u> )	キャンセ	μ
\ <u>\</u>			
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。



AirportDemoTutNativeMFUnitTest プロジェクトが作成されていることを確認してください。



3) AirportDemoTutNativeMFUnitTest プロジェクトを選択した上で、Eclise 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。

アイル(F)	編集(E) リファクタリング	ナビゲート(N) 検索 ブ	ロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
新規 ファイ ファイ 最近	(N) ルを開く(.) ル・システムからプロジェクトを のファイル	Alt+シフト+N 闘く	<ul> <li>COBOL プロジェクト</li> <li>COBOL コピーファイル プロジェクト</li> <li>リモート COBOL プロジェクト</li> <li>リモート COBOL コピーファイル プロジェクト</li> </ul>
閉じる すべて	ō(C) 【閉じる(L)	Ctrl+W Ctrl+シフト+W	COBOL ユニット テスト プロジェクト     ごのBOL JVM プロジェクト     COBOL JVM プロジェクト     COBOL JVM フロジェクト     COBOL DVM フロット ニフト プロジェクト     COBOL DVM フロット     COBOL DVM     CO
保管 別名 すべて 前回	(S) 保存(A) C保管(E) 保管した状態に戻す(T)	Ctrl+S Ctrl+シフト+S	COBOL JVM ユ_9ト チスト フロシェクト     ジェクト COBOL JVM プロジェクト     プロジェクト(R)     COBOL プログラム     COBOL プログラム     COBOL プログラム
移動 名前 更新 行区	(V) を変更(M) (F) 切り文字の変換(D)	F2 F5	COBOL JE- ノアイル  COBOL JE- ノアイル  COBOL JE- ノアイル  COBOL JE- スタンドアロン ファイル  UE-ト スタンドアロン ファイル  COBOL JE-ト スタン
日 日 日 日 日 日 日 日 日 日 日 日 日 日 日 日 日 日 日	(P) モート(I) ポート(O)	Ctrl+P	Directory Server 接続     COBOL ユニットテスト     COBOL JVM クラス

4) 「プログラムのユニットテストを作成する」項目を選択し、[参照] ボタンをクリックします。

💿 COBOL ユニット テ	ストの新規作成		×
<b>COBOL ユニット ラ</b> ⊗ テストする有効なソ	<b>-スト</b> -スファイルを指定する必要があります	 C	Ů
含まれるプロジェクト: 新規ファイル名:	AirportDemoTutNativeMFUnitTest TestProgram1.cbl	 *	9照
<ul> <li>プログラムのユニット</li> <li>テスト対象のプログラ</li> </ul>	>テストを作成する	参	照



5) AirportDemoTutNativeMFUnitTest プロジェクト内の「aircode.cbl」を選択した上で、[OK] ボタンをクリックします。

🥯 ソースファイル			$\times$
ソースファイルを選択します			
<ul> <li>AirportDemoTutNativeMFUnit</li> <li>Settings</li> <li>New_Configuration.bin</li> <li>aircode.cbl</li> <li>main.cbl</li> </ul>			
0	K I	++>>t	2.11

6) テスト対象のプログラムに、さきほど選択した aircode.cbl が表示されていることを確認して、[終了(F)] ボタンをクリックしま す。

💿 COBOL ユニット テ	ストの新規作成					$\times$
<b>COBOL ユニットラ</b>	テ <b>スト</b> きる COBOL ユニットテスト ファイルを	新規作成します。			в	\$
						U
含まれるプロジェクト:	AirportDemoTutNativeMFUnitTo	est			参照	a
新規ファイル名:	TestProgram1.cbl					
◎ プログラムのユニッ	トテストを作成する					
テスト対象のプログラ	ム: AirportDemoTutNativeMFUr	it/aircode.cbl			参照	
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	_ツト ナストを1F成 9 つ					
Micro Foc	us テンプレート					
				<u>テンプレ-</u>	トの設定を構成	戎
🗌 テンプレートの参	▶照					
場所;					参照	
ファイルシス	テムを選択: default ~					
?		< 戻る(B)	次へ(N) >	終了(F)	キャンセル	,



単体テストプログラムが作成されたことを確認します。



## 7) 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。

0	a TestProgram1.cbl 🕱								
	TestProgram1.cbl >								
	01 i pic 99.								
	Θ	procedure division. goback returning 0							
		entry MFU-TC-PREFIX & TEST-TESTAIRCODE.							
		<pre>call "AIRCODE" using by value lnk-function by value lnk-airport1 by value lnk-airport2 by value lnk-prefix-text by reference lnk-rec by reference lnk-distance-result by reference lnk-matched-codes-array</pre>							
		*> Verify the outputs here							
		goback returning MFU-PASS-RETURN-CODE .							
	Θ	<pre>\$region TestCase Configuration</pre>							
		<pre>entry MFU-TC-SETUP-PREFIX &amp; TEST-TESTAIRCODE. perform InitializeLinkageData    *&gt; Add any other test setup code here    goback returning 0 .</pre>							



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。前処理の代表例としては、ファイルを あらかじめオープンしておく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。

8) 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内の 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-msg 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。

### 3.1.4. 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。

0	実行(R)	>	CBL	1 COBOL アプリケーション	Alt+シフト+X,N
*	デパッグ(D)	>	œU	2 COBOL ユニット テスト	Alt+シフト+X,F
	プロファイル(P)	>		実行 の構成(N)	
	検証(V)		1	[mfurun.exe	Finished: SUC

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、[New Configuration] を選択します。



C 🛛 🗶 🗐 🗞	7 -	このダイアログから	起動設定を構成します:
フィルタ入力		📑 - 選択した種	重類の構成を作成するには、「新規構成
COBOL アプリケーション		▲ 📝 - 選択した種	重類の起動構成プロトタイプを作成する
COBOL ユニット テスト	Ľ	新規構成(W)	クスポートするには、「エクスポー
● Eclipse アプリケーション	P	新規プロトタイプ(P)	ピーするには、「複製」ボタンを排
📔 Eclipse データ・ツール	<b>\$</b>	エクスポート(X)	除するには、「削除」ボタンを担

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





4) 以下の情報を入力し、[OK] ボタンをクリックします。

変数: "dd\_airports"

值: "..¥..¥AirportDemoTutNativeMFUnit¥airports.dat"

◎ 変奏 環境:	<ul> <li>● <sup>変数を追加</sup> ×</li> <li>環境変数を追加または変更します</li> </ul>							
変数: <u>値</u> :	dd_airports ¥¥AirportDemoTutNativeMFUnit¥airports.dat							
?		OK ŧ	ャンセル					

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)] ボタンをクリックします。
 ● 素行構成 - □ ×

構成の作成、管理、および実行						
📑 🖻 💫 🗎 🗶 🗖	名前(N): AirportDemoTutNativeMFUnitTest					
フィルタ入力	□	動的分析 ○ CTF     ○     ○     ○     □				
Apache Tomcat AspectJ Load-Time Weavi	(注:ここで定義された変数は、任意の現在の設定値、または任意の指定された 環境スクリプト内の設定値を上書きします。)					
ASPECT/Java Application COBOL JVM アプリケーション	変数値	追加( <u>A</u> )				
COBOL JVM ユニット テスト	dd_airports¥¥AirportDemoTutNativeMFUr	nit¥ai 编集(E)				
COBOL JVM リモート アブリク  COBOL アプリケーション						
✓ @ COBOL ユニット テスト						
■ 新規構成						
Eclipse データ・ツール						
Gradle Project						
Grunt	実行する環境スクリプト:					
Guip HTTP プレビュー	場所:	参照				
🗄 J2EE プレビュー	ファイルがプロジェクト内にある場合、絶対パスは相対パスになります。					
」 Java アプリケーション	パラメータ:					
Ju JUnit						
🚏 JUnit プラグイン・テスト	□ 関連付けられたプロジェクトのビルド環境から値を継承					
m2 Maven Build						
30 項目のうち 28 項目がフィルターに-	前回保管した状態	態に戻す(⊻) 適用(¥)				
?		実行( <u>R</u> ) 閉じる				



6) [Micro Focus Unit Testing] ビューが自動的に表示され、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。

AT 1 1 1 1 4 4 1 1 1 1	C. C. Sarra . Winner and the same re-		+ 1 mm ee
117: 2/2	0 I5-1 0	単 失敗 0	
W HI AmortDemoTutNativ	MR.initTest (16 ms)		サスト結果
v de TestProgramT.cbl			
MFUT_TESTAIRC	(0 mil) 3000		

 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た上で、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。

なお、本例では、3.1.3 で作成した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。

Te	stProgram1.cbl 🕨
	• • * A · 1 · B · • • • • 2 · • • • • • 3 · • • • • • • • • • • 5 · • • • • • 6 · • • • • • 7 · • • • • 8 · • • • 9 · • •
	entry MFU-TC-PREFIX & "testDistance"
	set get-distance to true
	move "HND" to lnk-airport1
	move "LHR" to lnk-airport2
	call "AIRCODE" using
	by value lnk-function
	by value lnk-airport1
	by value lnk-airport2
	by value lnk-prefix-text
	by reference lnk-rec
	by reference lnk-distance-result
	by reference lnk-matched-codes-array
	move 9592 to wk-distance-km
	display wk-distance-km " / " distance-km
	if distance-km = wk-distance-km
	then
	goback returning MFU-PASS-RETURN-CODE
	else
	<pre>string "expected " wk-distance-km ", but " distance-km into err-msg end-string</pre>
	call MFU-ASSERT-FAIL-Z using err-msg
	end-if.

- 読 記: 図: 枠 ▼ ● ▼ ▲ ▼ ② ② ② ダ ▼ ↓ ▼ ▼ ▼ ◆ サーバー エクス... □ E ▼ □ ⑤ ♪ ② ◆ 実行(R) → 実行の構成(N).... お気に入りの編成(V)....

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。

) コンソール 🖹 問題 壇 タスク 🔲 プロパティー	Table Results 👸 Filter Definitions	😨 Micro Focus Unit Testing 🙁 📴 コードカパレッジ	
ļ行: 2/2	<b>¤</b> Iラ-: 0	☑ 失敗: 1	
AirportDemoTutNativeMFUnitTest (1 ms)			テスト結果



# 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。

ここでは、3.1.1 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、Micro Focus Visual COBOL 配下の Visual COBOL コマンドプロンプト をクリックします。



3.1.1 にて選択したビットと同様のプロンプトを使用してください。

2) 作業フォルダを作成し、作成したフォルダに移動します。

C:¥>mkdir VCCommandTutorial
C:¥>cd VCCommandTutorial
C:¥VCCommandTutorial>

- 3) 3.1 で利用したワークスペースフォルダを 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:\u00e4workspace\_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¥airc ode.cbl;

(コマンド実行中の出力内容を省略)

C:¥VCCommandTutorial>cobol %ECLIPSE\_WORKSPACE%¥AirportDemoTutNativeMFUnitTest¥



TestProgram1.cbl sourceformat(variable); (コマンド実行中の出力内容を省略) C:¥VCCommandTutorial>cbllink -D aircode.obj (コマンド実行中の出力内容を省略) C:¥VCCommandTutorial>cbllink -D TestProgram1.obj (コマンド実行中の出力内容を省略)

#### 以下のファイルが作成されていることを確認します。

C:¥VCComman	C:¥VCCommandTutorial>dir								
ドライブ C のボリ	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	レ番号は 68	308-353	9 です						
C:¥VCComma	IndTutoria	Iのディレ	クトリ						
2019/06/13 1	13:29	<dir></dir>							
2019/06/13 1	3:29	<dir></dir>							
2019/06/13 1	3:29		17,408	aircode.dl	l				
2019/06/13 1	13:26		10,121	aircode.ob	oj				
2019/06/13 1	3:29		15,360	TestProgra	am1.dll				
2019/06/13 1	3:26		7,372	TestProgra	am1.obj				
	4 個のファ・	イル		50,261	バイト				
	2 個のディ	レクトリ	42,538,	450,944 /	「イトの空き領	稢			

- 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%¥AirportDemoTutNativeMFUnit¥airports.dat				
C:¥VCCommandTutorial	>mfurun -report:junit -outdir:result TestProgram1.dll				
Micro Focus COBOL - mi	furun 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				



outdir オプションにより、出力結果を result フォルダに保存されていることを確認してください。



### WHAT'S NEXT

• 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。

### 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。