Rocket software

Visual COBOL チュートリアル

Eclipse – JVM 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、JVM 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。

Visual COBOL 製品に付属している MFUnit は、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

MFUnit は JVM COBOL に対するテストを記述することもできますが、JVM COBOL は 他の Java 言語と同様、Java バイトコードを生成します。このため、Java プログラムのように JUnit 単体テストフレームワークを使用することで、より統一性のあるテスト運用が可能です。本チュートリアルでは、JUnit 単体テストフレームワークを利用したテスト実装方法を学びます。

2 前提

- 本チュートリアルで使用したマシン OS ： Windows 10
- Visual COBOL 10.0 for Eclipse がインストール済みであること

本資料は、JVM 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。ネイティブ 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。

サンプルプログラムのダウンロード

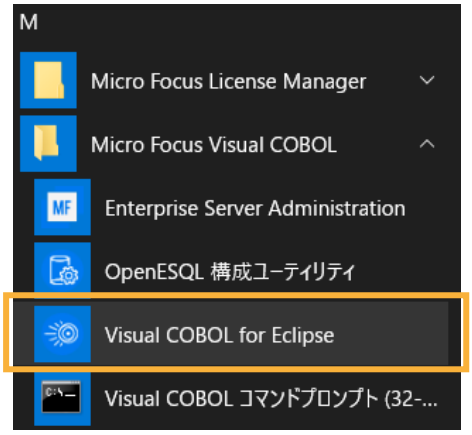
内容

3 チュートリアル手順

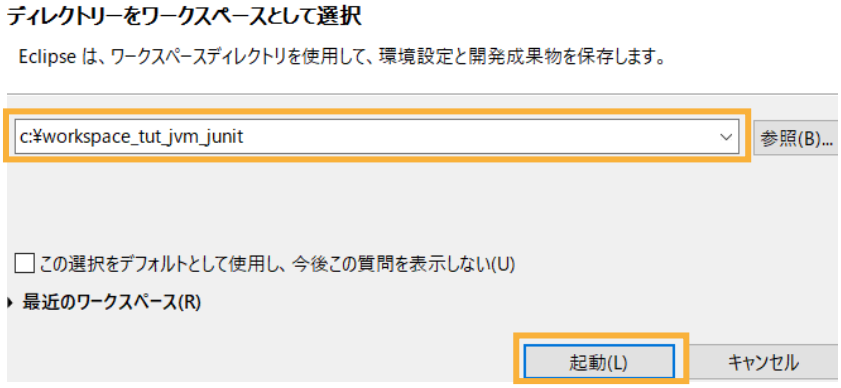
3.1 IDE からの実行

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。

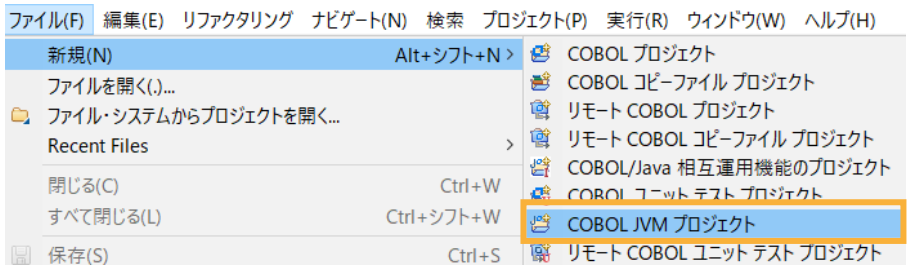


2) ワークスペースを指定し、[起動(L)] をクリックします。本書では、c:¥workspace_tut_jvm_junit を指定しています。



3.1.2 チュートリアルプロジェクトの作成

1) Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [COBOL JVM プロジェクト] を選択してください。



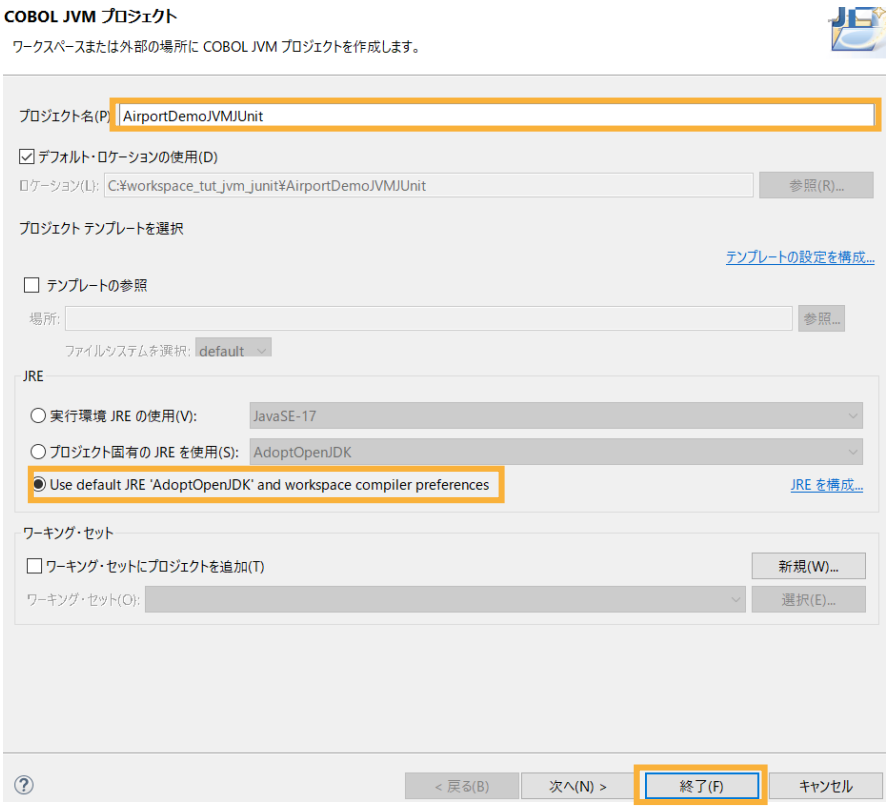
2) 以下の入力を行い、[終了(F)] をクリックします。

プロジェクト名： AirportDemoJVMJUnit

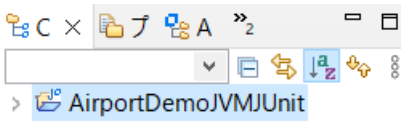
JRE： Use default JRE

補足）

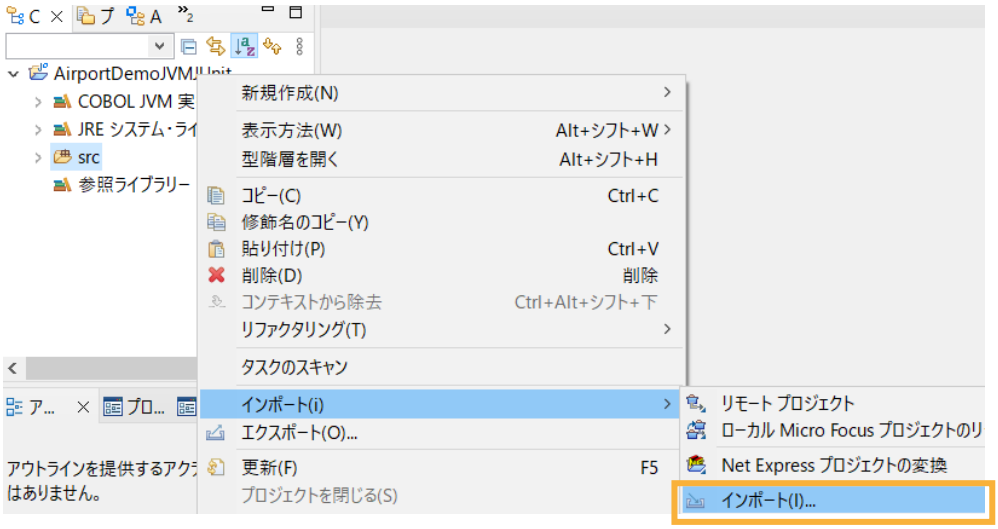
JRE については、各環境にあわせて異なる選択をして頂いても問題ありません。



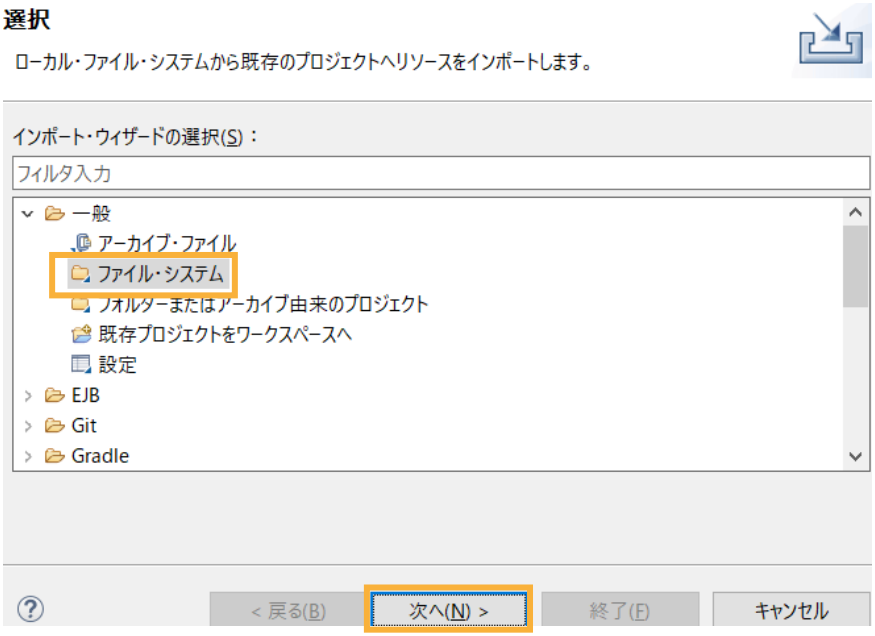
プロジェクトが作成されます。



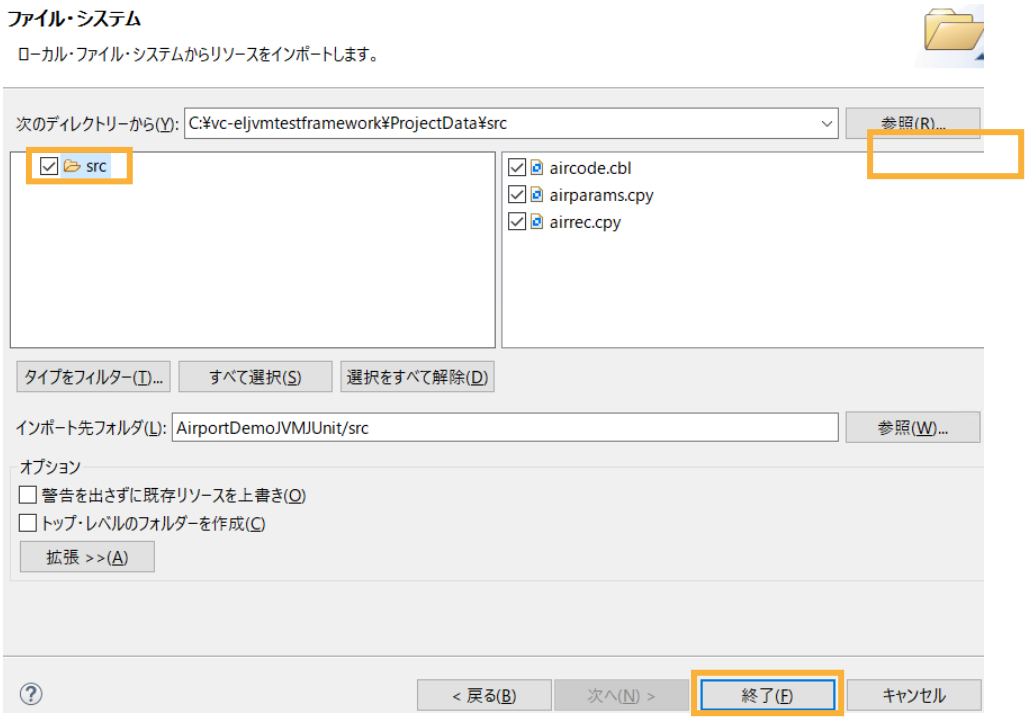
3) AirportDemoJVMJUnit プロジェクト配下の「src」フォルダーを選択した上で、マウスの右クリックにてコンテクストメニューを表示し、[インポート(I)] > [インポート(I)] を選択します。



4) [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。



5) [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 ProjectData¥src フォルダーを選択し、src フォルダーにチェックした上で、[終了(F)] をクリックします。

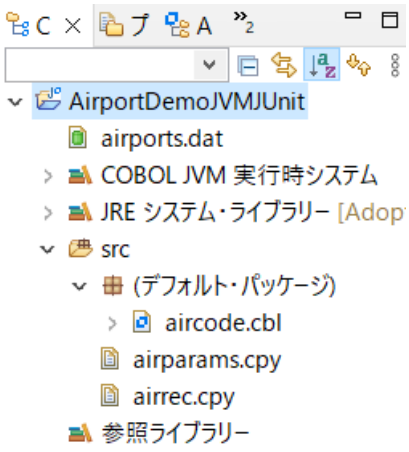


COBOL エクスプローラー画面より、インポートが完了したことを確認します。

6) AirportDemoJVMJUnit プロジェクト配下を選択した状態で、再度、上記インポート手順を以下の方法にて実施します。

- [参照(R)] ボタンクリック後のフォルダー選択にて、サンプルファイルを展開したフォルダー内の ProjectData を選択
- airports.dat をインポート

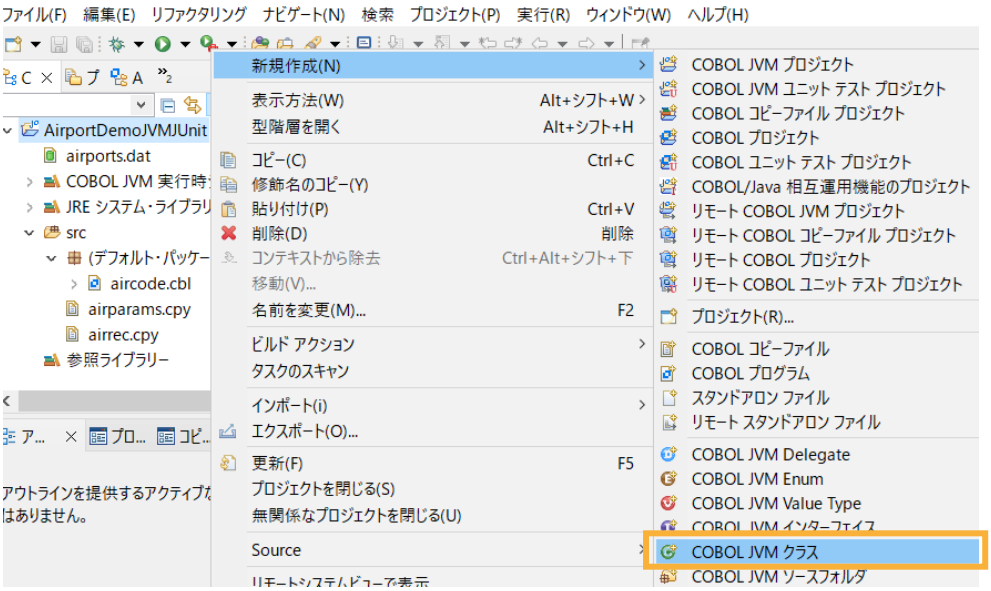
インポート後、以下のようになります。



3.1.3 JVM COBOL プログラムの作成

前手順でインポートした aircode.cbl はレガシーなネイティブ COBOL プログラムです。本手順では、aircode.cbl に一切の変更を加えることなく、JVM COBOL として機能するよう、Wrapper クラスを作成します。

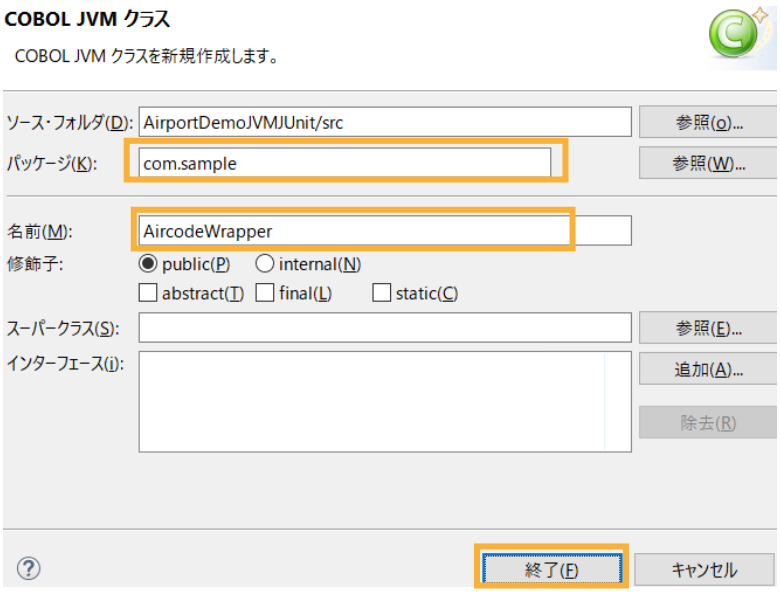
1) AirportDemoJVMJUnit プロジェクト名を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[新規作成(N)] > [COBOL JVM クラス] を選択します。



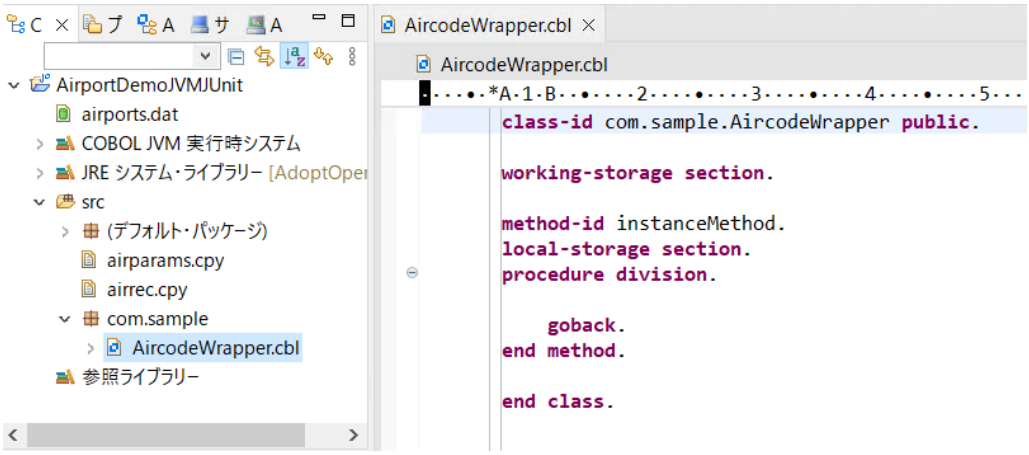
2) 以下の入力を行い、[終了(F)] をクリックします。

パッケージ： “com.sample”

名前： “AircodeWrapper”



AircodeWrapper プログラムが作成されたことを確認します。また、AircodeWrapper.cbl が自動的に開かれます。

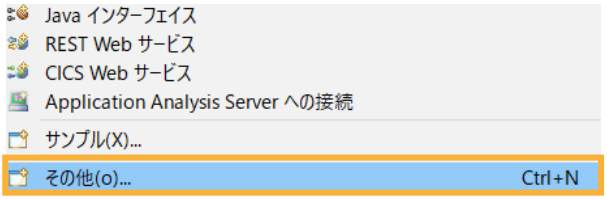


3) サンプルファイルを展開したフォルダー内の AircodeWrapper.cbl をメモ帳などで開き、その中身でエディター上の AircodeWrapper.cbl を上書きしたうえで、プログラムを [ファイル(F)] > [保存(S)] をクリックして上書き保存します。

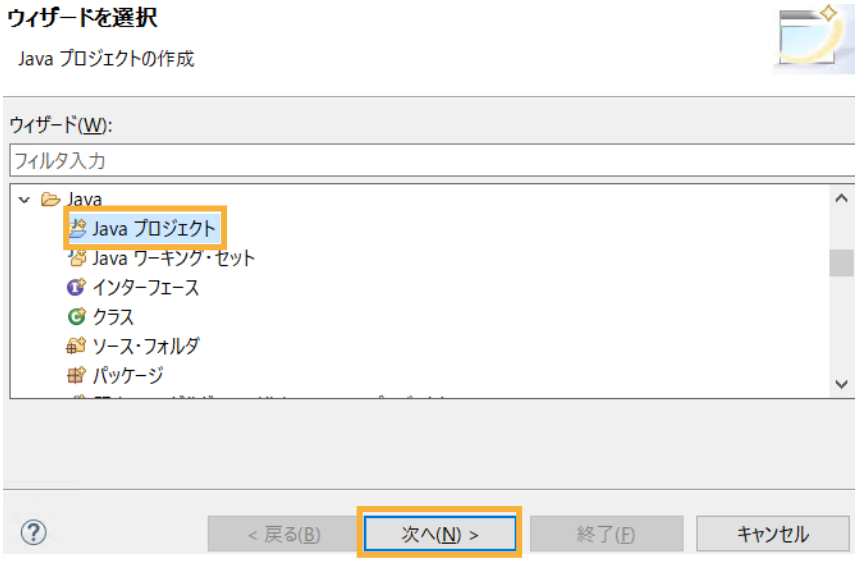
この Wrapper クラスは、com.sample.AircodeWrapper クラスとして、Java 言語から レガシーの COBOL を呼び出せるよう、各種メソッドを定義しています。

3.1.4 JUnit プロジェクトの作成

1) Eclipse IDE メニューより、[ファイル(F)] > [新規(N)] > [その他(o)] を選択します。



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



3) 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： AirportDemoJUnit

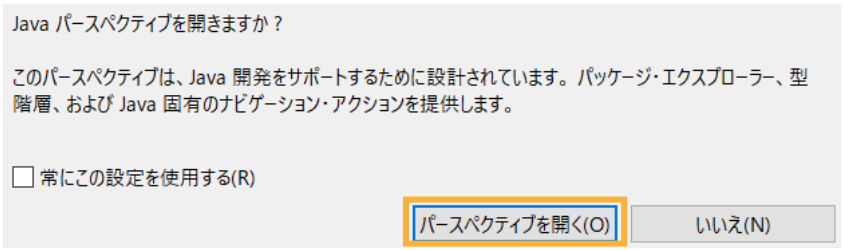
JRE： Use default JRE

module-info.java を作成： チェックを外す

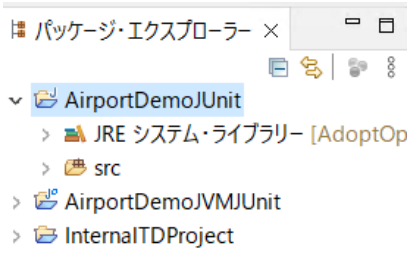
補足）

JRE については、各環境にあわせて異なる選択をして頂いても問題ありません。

以下のようなダイアログが表示された場合、[パースペクティブを開く(O)] をクリックします。

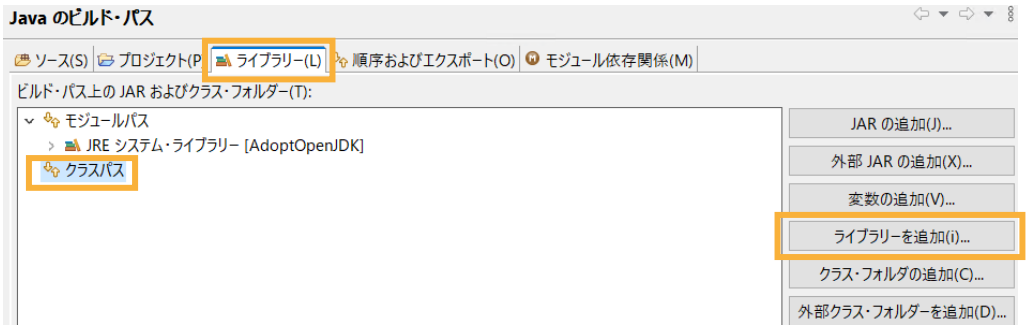


プロジェクトが作成されます。

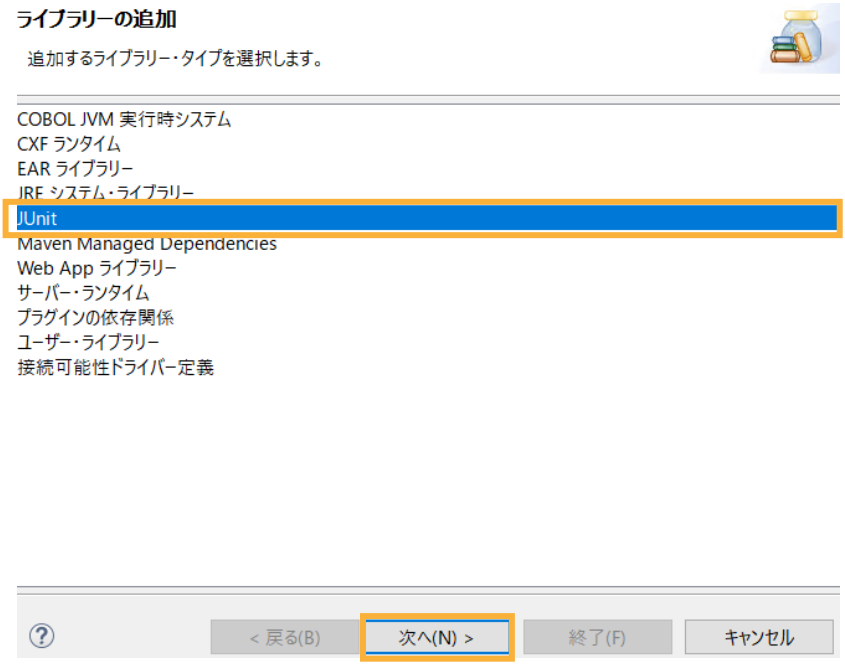


4) AirportDemoJUnit プロジェクト名を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[ビルド・パス(B)] > [ビルド・パスの構成(C)] を選択します。

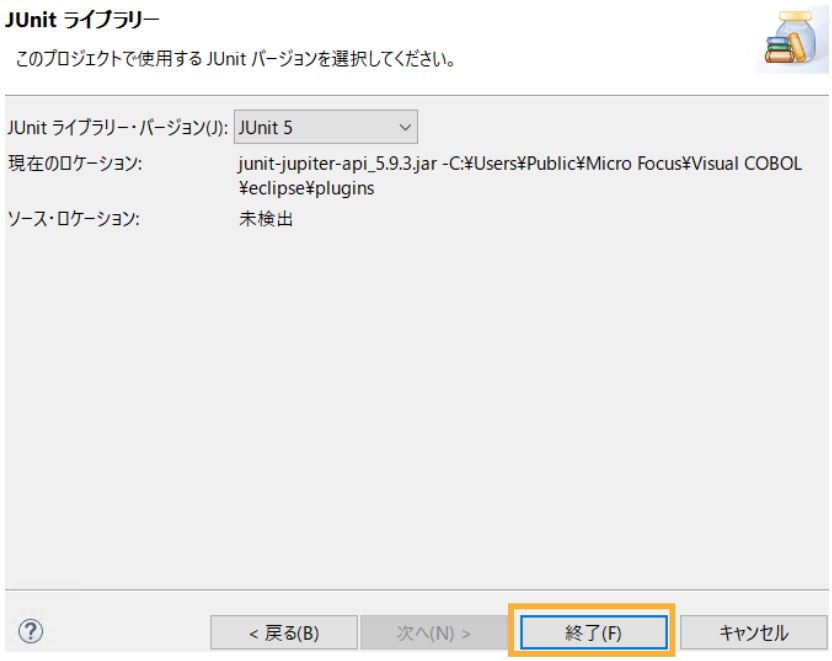
5) 「ライブラリー(L)」タブを選択し、[クラスパス] を選択したうえで [ライブラリーを追加(i)] をクリックします。



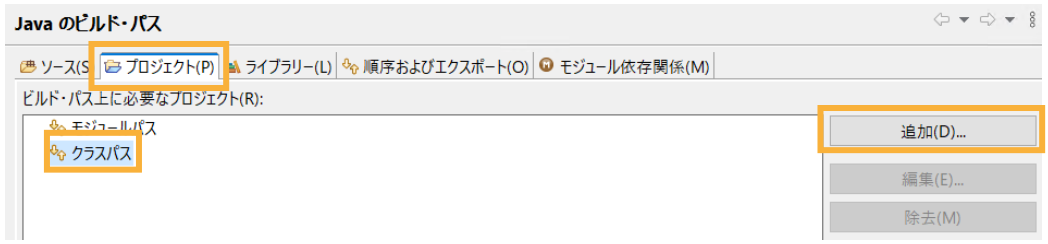
6) 「JUnit」を選択し、[次へ(N) >] をクリックします。



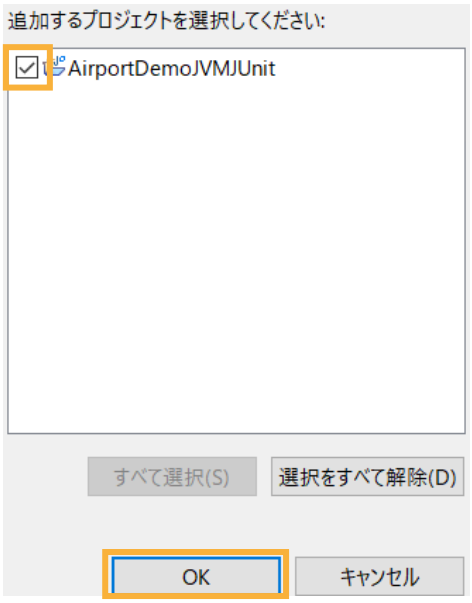
7) そのまま、[終了(F)] をクリックします。



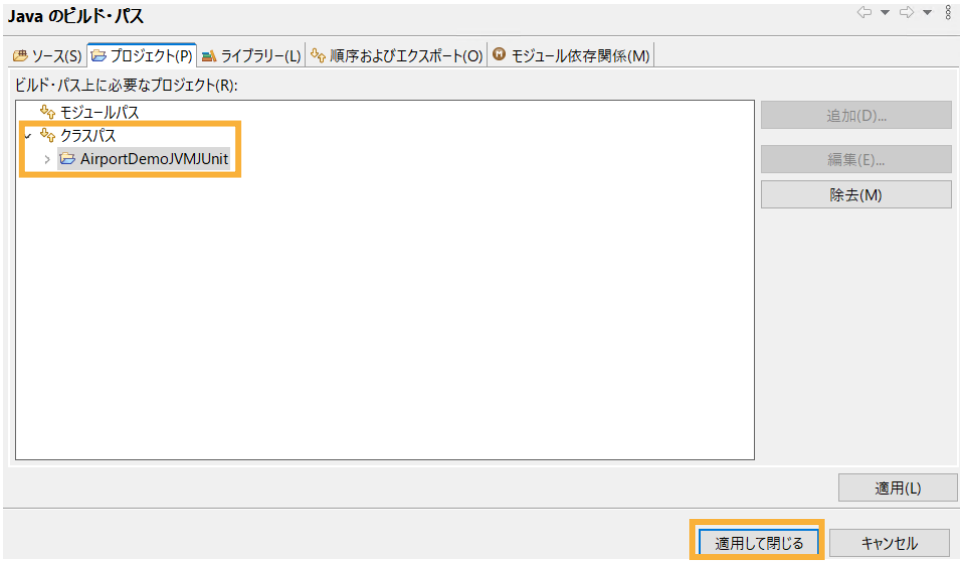
8) 「プロジェクト(P)」タブを選択し、[クラスパス] を選択したうえで [追加(D)] をクリックします。



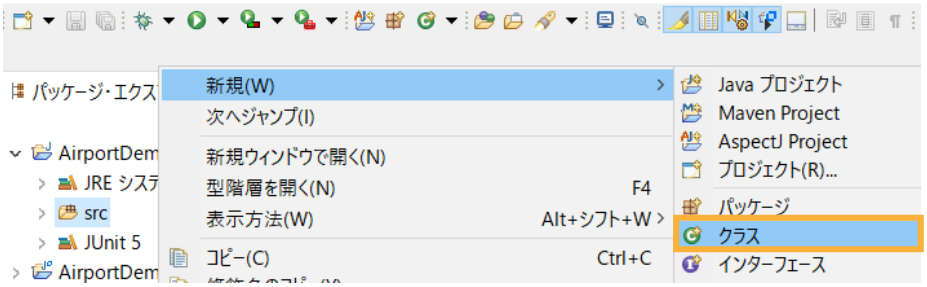
9) AirportDemoJVMJUnit プロジェクトにチェックを行い、[OK] をクリックします。



10) AirportDemoJVMJUnit プロジェクトが追加されたことを確認し、[適用して閉じる] をクリックします。



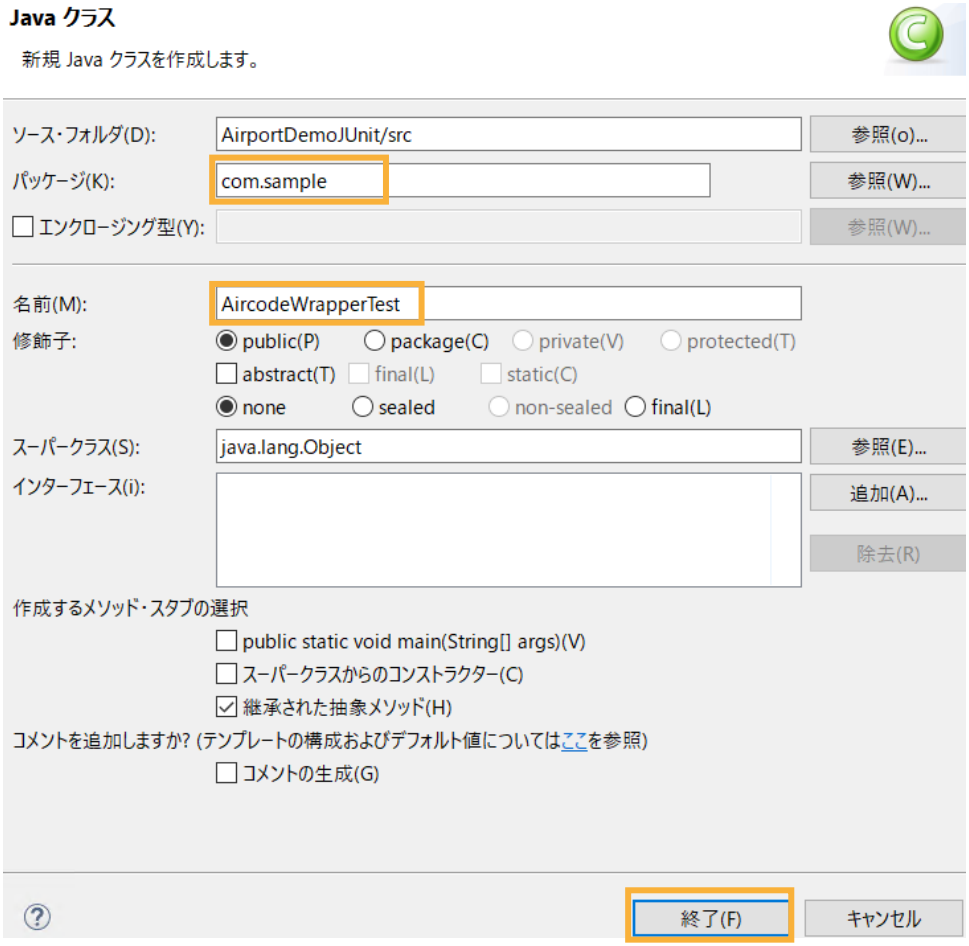
11) AirportDemoJUnit プロジェクト配下の src フォルダーを選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[新規(W)] > [クラス] を選択します。



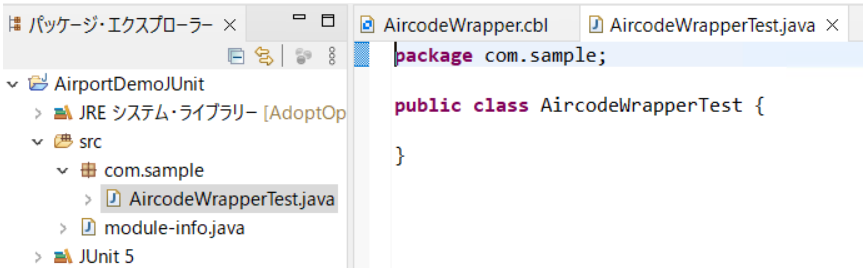
12) 以下の入力を行い、[終了(F)] をクリックします。

パッケージ： com.sample

名前： AircodeWrapperTest



AircodeWrapperTest.java が作成されます。



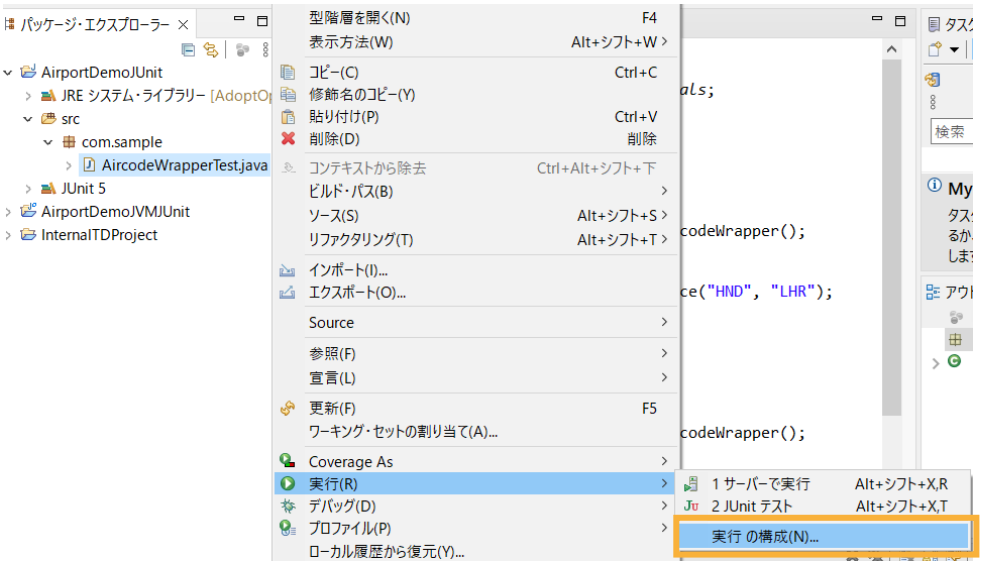
13) サンプルファイルを展開したフォルダー内の AircodeWrapperTest.java でコードを上書き保存します。

```java
package com.sample;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class AircodeWrapperTest {
    @Test
    public void testGetDistance1() {
        AircodeWrapper wrapper = new AircodeWrapper();
        wrapper.initialize();
        wrapper.openFile();
        int distance = wrapper.getDistance("HND", "LHR");
        assertEquals(9591, distance);
        wrapper.closeFile();
    }

    @Test
    public void testGetDistance2() {
        AircodeWrapper wrapper = new AircodeWrapper();
        wrapper.initialize();
        wrapper.openFile();
        int distance = wrapper.getDistance("NRT", "LHR");
```

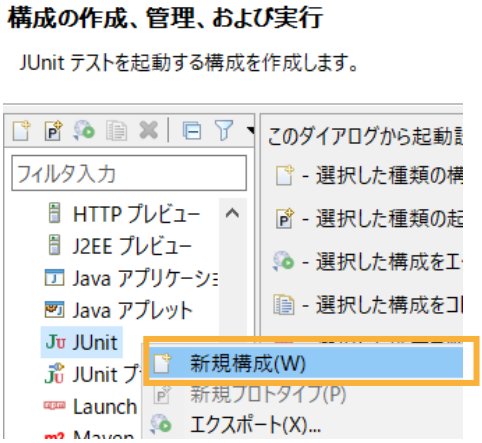
一般的な JUnit テストが記述されており、そのテストプログラム内で、さきほど作成した AircodeWrapper クラスが通常の Java クラスとして利用されていることが分かります。このように、JVM COBOL を利用することで、COBOL を意識させることなく、Java プログラム内で利用することができます。

3.1.5 JUnit テストプログラムの実行

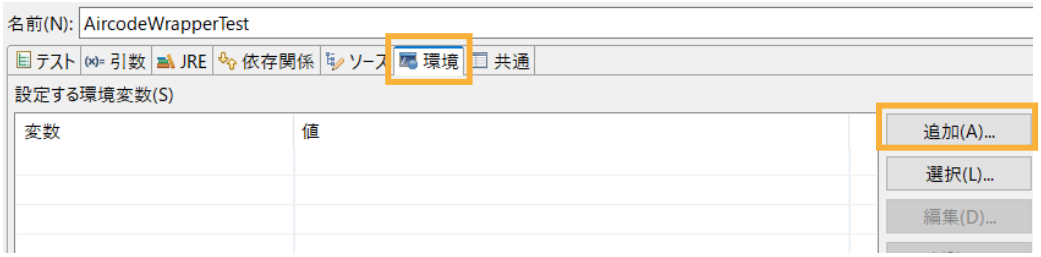
1) AirportDemoJUnit プロジェクトの配下の AircodeWrapperTest.java を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[実行(R)] > [実行の構成(N)] をクリックします。



2) 画面左側より「JUnit」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。



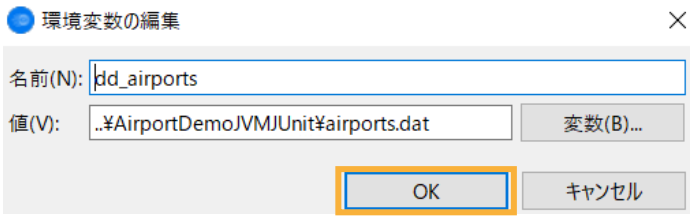
3) 「環境」タブを選択し、[追加(A)] をクリックします。



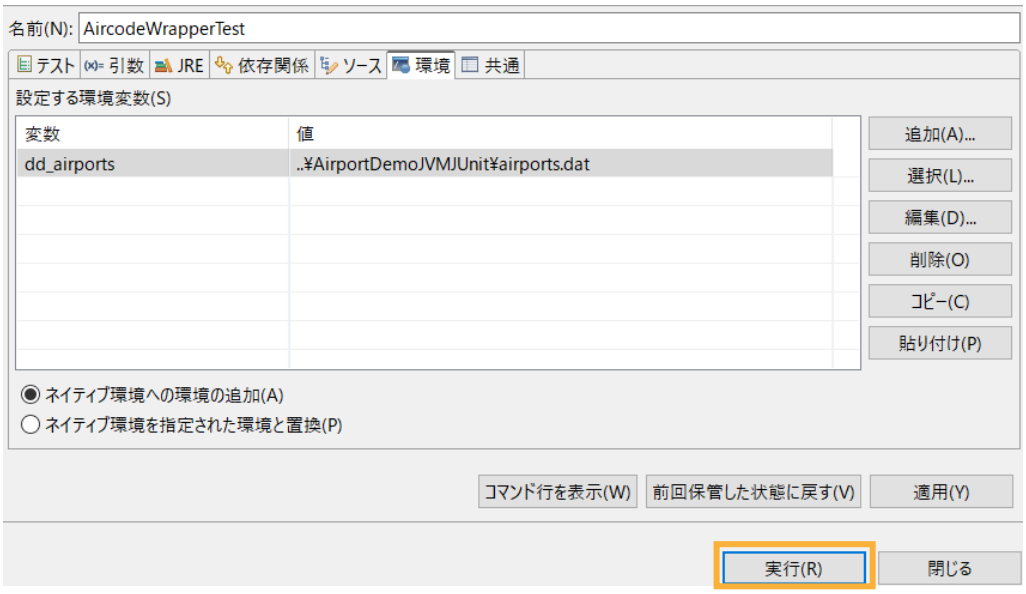
4) 以下の入力を行い、[OK] をクリックします。

名前： “dd_airports”

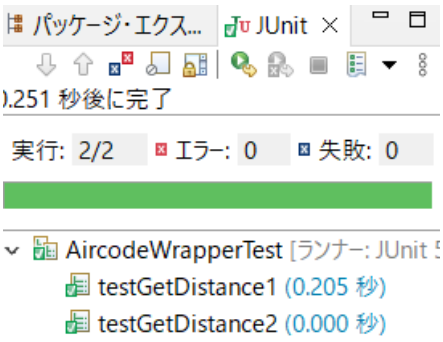
値： “..¥AirportDemoJVMJUnit¥airports.dat”



5) dd_airports 環境変数が追加されたことを確認して、[実行(R)] をクリックします。

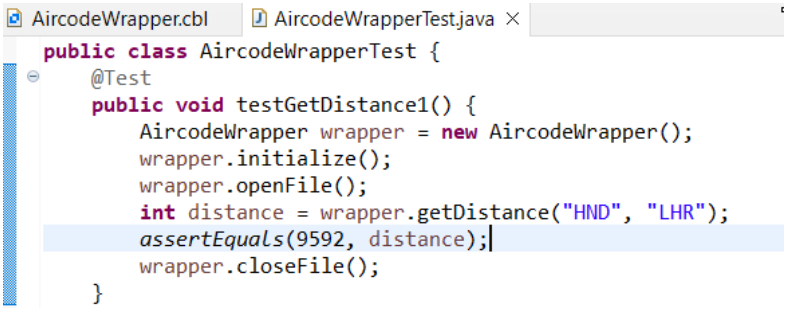


JUnit ビューが表示され、全てのテストが成功したことを示す緑色で表示されていることを確認します。

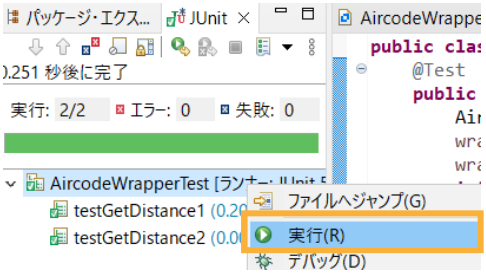


続いて、エラーケースを確認します。

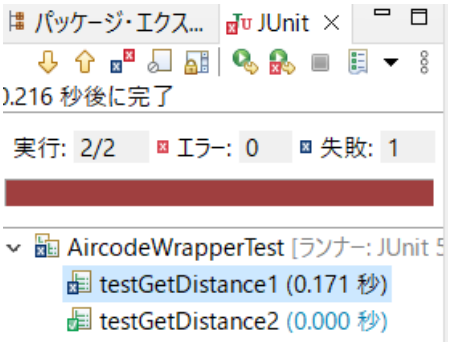
6) JUnit ビューより testGetDistance1 をダブルクリックし、9591 という数値を 9592 に修正した上で保存します。



7) Junit ビューの「com.sample.AircodeWrapperTest」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、[実行(R)] を選択します。



JUnit ビューが赤色となり、さきほど数値を修正した testGetDistance1 が失敗していることが分かります。



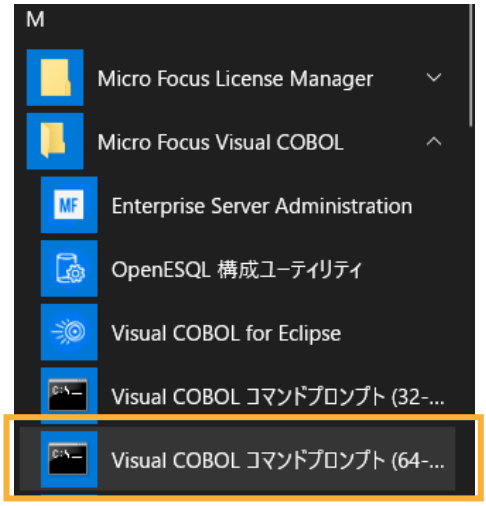
さきほど修正した 9592 を 9591 に再度修正し、ファイルを保存してください。

3.2 コマンドラインからの実行

JUnit テストプログラムはもちろん、JVM COBOL の開発・テストについても、コマンドラインから実施できます。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、IDE 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) Windows メニューより、[Micro Focus Visual COBOL] > [Visual COBOL コマンドプロンプト (64-bit ビット)] をクリックします。



2) サンプルファイルを解凍したフォルダー配下の command に移動します。以下では、C:¥vc-eljvmtestframework が解凍先となっています。

```
C:¥Users¥tarot¥Documents>cd ¥vc-eljvmtestframework¥command
C:¥vc-eljvmtestframework¥command>
```

3) env.ini をメモ帳などで開き、以下の変数をお客様の環境に合わせて修正し、保存します。

- VC_INSTALL_PATH
  Visual COBOL 製品のインストールフォルダー
- ECLIPSE_WORKSPACE
  IDE で指定したワークスペースフォルダー
- ECLIPSE_PLUGIN_PATH
  Visual COBOL 製品に同梱された Eclipse のインストールフォルダー

補足）
このファイルは、プログラムのビルドを行う build.bat、そして、テスト実行を行う run.bat から参照されます。それぞれのバッチファイルは、IDE 上と同じ環境を構築したうえで、ビルド、もしくはテスト実行を行います。

4) build.bat を実行します。

```
C:¥vc-eljvmtestframework¥command>build.bat
Micro Focus COBOL
Version 10.0 (C) Copyright 1984-2024 Micro Focus or one of its affiliates.
* チェック終了：エラーはありません
Micro Focus COBOL
Version 10.0 (C) Copyright 1984-2024 Micro Focus or one of its affiliates.
```

```
* チェック終了：エラーはありません
マニフェストが追加されました
aircode$_MF_LCTYPE_1.class を追加中です(入=823)(出=397)(51%収縮されました)
aircode.cbldat を追加中です(入=296)(出=64)(78%収縮されました)
aircode.class を追加中です(入=10061)(出=4466)(55%収縮されました)
com/を追加中です(入=0)(出=0)(0%格納されました)
com/sample/を追加中です(入=0)(出=0)(0%格納されました)
com/sample/AircodeWrapper.class を追加中です(入=4633)(出=1935)(58%収縮されました)
          1 個のファイルを移動しました。
C:¥vc-eljvmtestframework¥command>
```

5) run.bat を実行します。

```
C:¥vc-eljvmtestframework¥command>run.bat
JUnit version 4.13.2
.HND  Tokyo Intl
      Japan                  Lat:+035.552258 Lon:+139.077969
LHR  Heathrow
      United Kingdom          Lat:+051.004775 Lon:-000.461389
.NRT  Narita Intl
      Japan                  Lat:+035.764722 Lon:+140.038638
LHR  Heathrow
      United Kingdom          Lat:+051.004775 Lon:-000.461389
Time: 0.187
OK (2 tests)
C:¥vc-eljvmtestframework¥command>
```

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。