Micro Focus Visual COBOL for Eclipse

自習書

はじめに

Micro Focus Visual COBOL for Eclipse は、高品質、高機能なオープンソースの統合開発環境(IDE)として広く普及する Eclipse 上で COBOL アプリケーションプログラム開発を可能する COBOL 開発環境製品です。 COBOL プログラマが既存の COBOL 資産を Windows、UNIX/Linux といったオープン環境で活用するだけでなく、COBOL プログラミング経験のない Java プログラマが初めて COBOL アプリケーション開発を行う場合にも最適な製品です。

本書は、Micro Focus Visual COBOL for Eclipse(Windows)を学ぶための自習書です。 本書の読者は、プログラミングの基礎知識をもち、且つ Windows の基本操作を理解しているものとします。

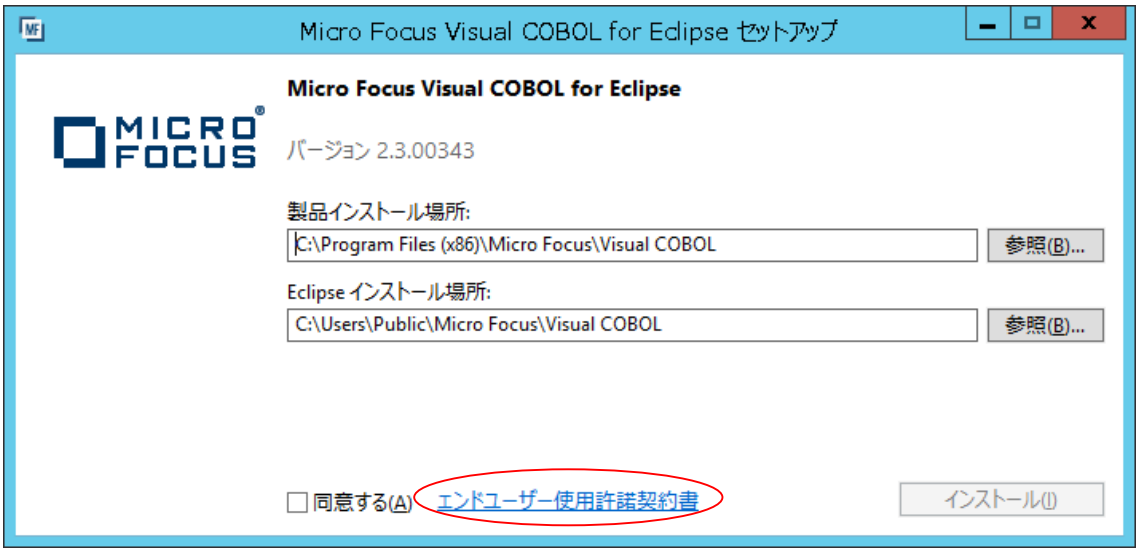
また、本書に掲載している画面イメージは Windows Server 2012 R2 でキャプチャしています。他の Windows OS では多少異なる場合がありますが、ご了承ください。

第1章　　自習環境の準備

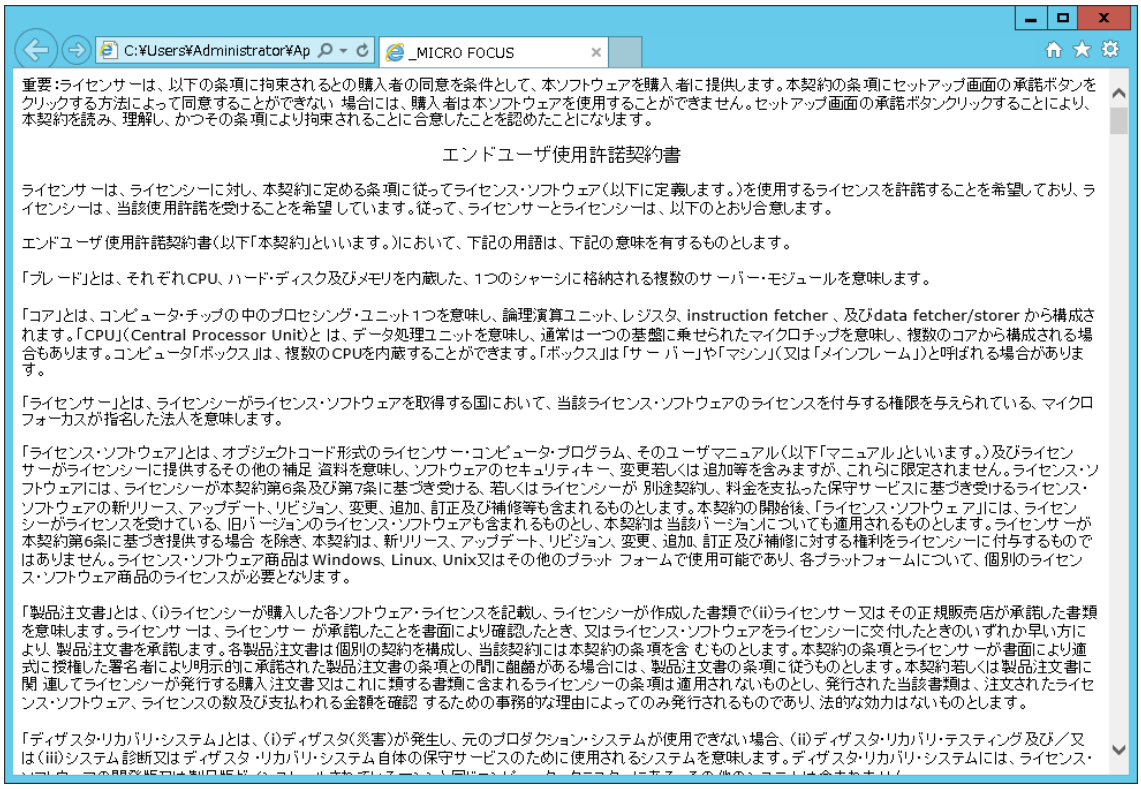
Micro Focus Visual COBOL for Eclipse は、COBOL プログラミングの IDE として Eclipse の IDE を利用します。本製品には、Eclipse 4.4.2 がバンドルされていますが、既に 32bit 版 Eclipse 4.2, 4.3, 4.4 をお使いの場合は、お使いの Eclipse にプラグイン形式で本製品をインストールすることも可能です。本章ではインストーラを使った方法を紹介いたします。

1　ダウンロードした **vce_23.exe** をダブルクリックします。

2　表示されるセットアップ画面で **エンドユーザ使用許諾契約書** をクリックします。

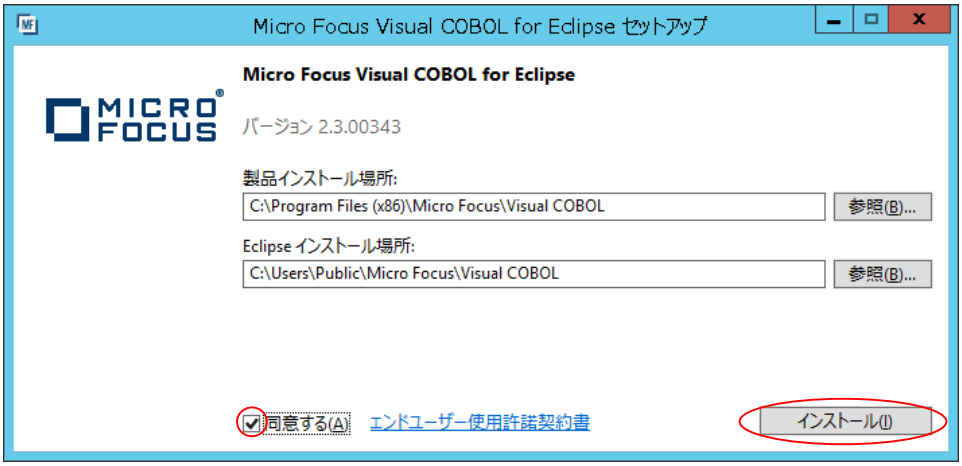


3　使用許諾契約書の内容を確認します。



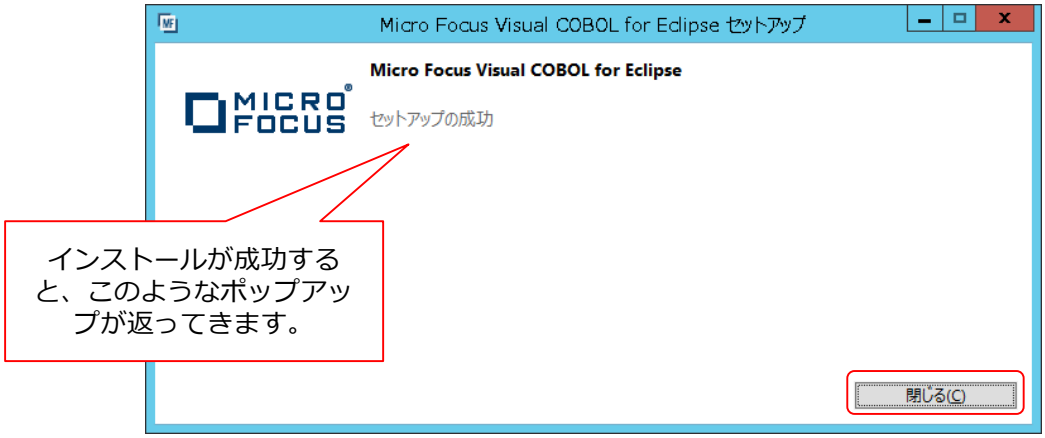
4　インストールを開始します。

問題がなければ、**同意します(A)** にチェックを入れ [**インストール(I)**] ボタンを押下してインストールを開始します。



5　セットアップを終了します。

[閉じる(C)] ボタンを押下します。

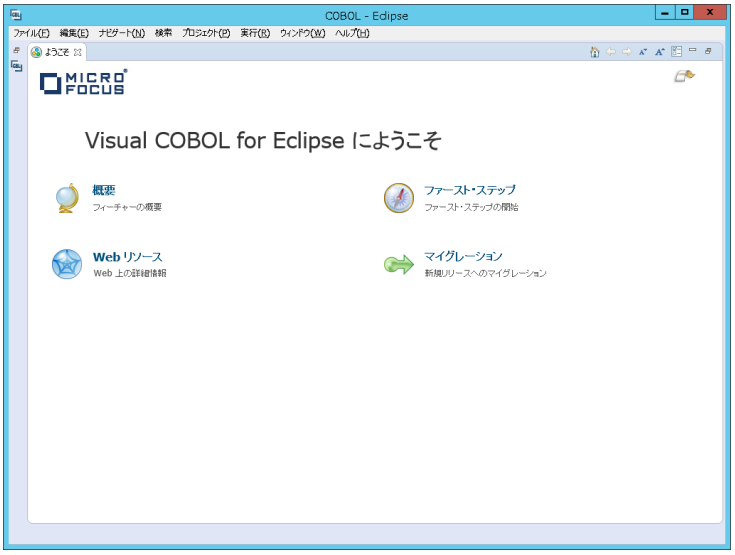


以上で、自習環境の準備は終了しました。 Windows のスタートメニューに「Micro Focus Visual COBOL」が登録されていることを確認してください。

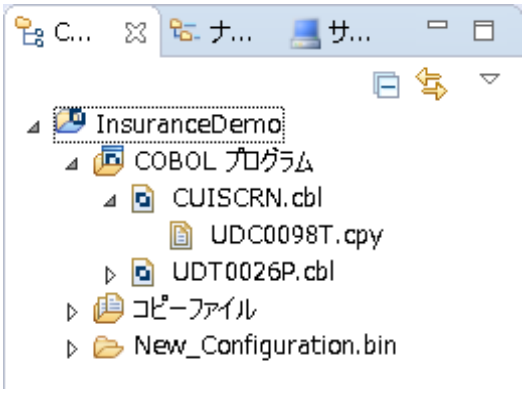
第2章　Eclipse IDE に慣れよう

Eclipse の IDE を初めて利用する COBOL プログラマのために、概要を簡単に説明します。 既に Eclipse に習熟されている方は、本章を読み飛ばしてください。

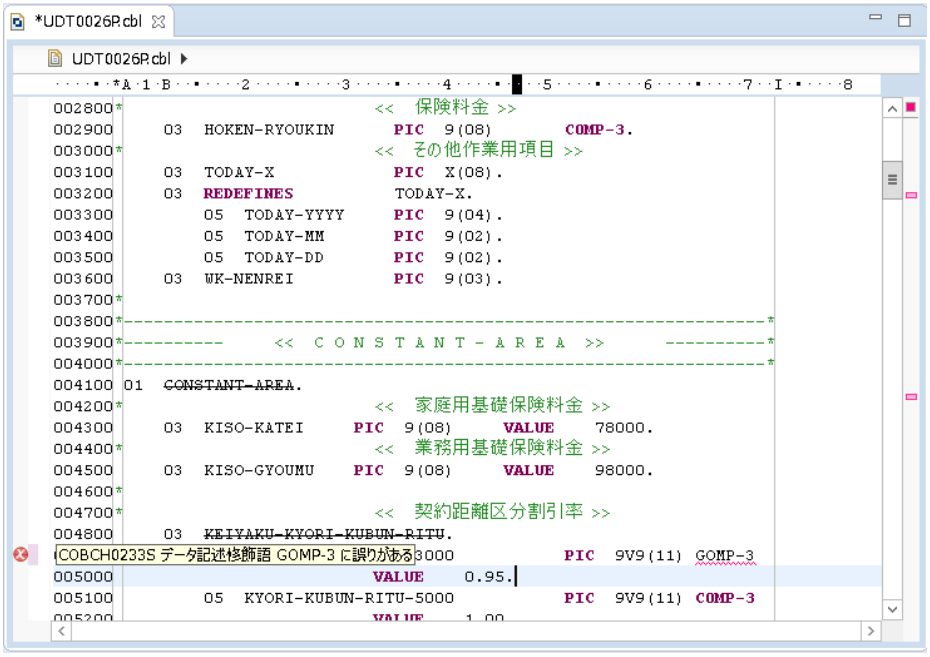
Eclipse の IDE を使う場合、ビュー、エディター、ツールバー、メニューバー、ステータスバー、パースペクティブから構成されるワークベンチというウィンドウ内で作業します。パースペクティブは、行いたい作業によって切り替えて利用するワークベンチのレイアウト（表示するビュー、メニュー、ツールバーの種類や場所）のことを指します。パースペクティブを切り替えることによって目的の作業に応じた構成要素を揃えることができます。各要素の配置はカスタマイズ可能です。



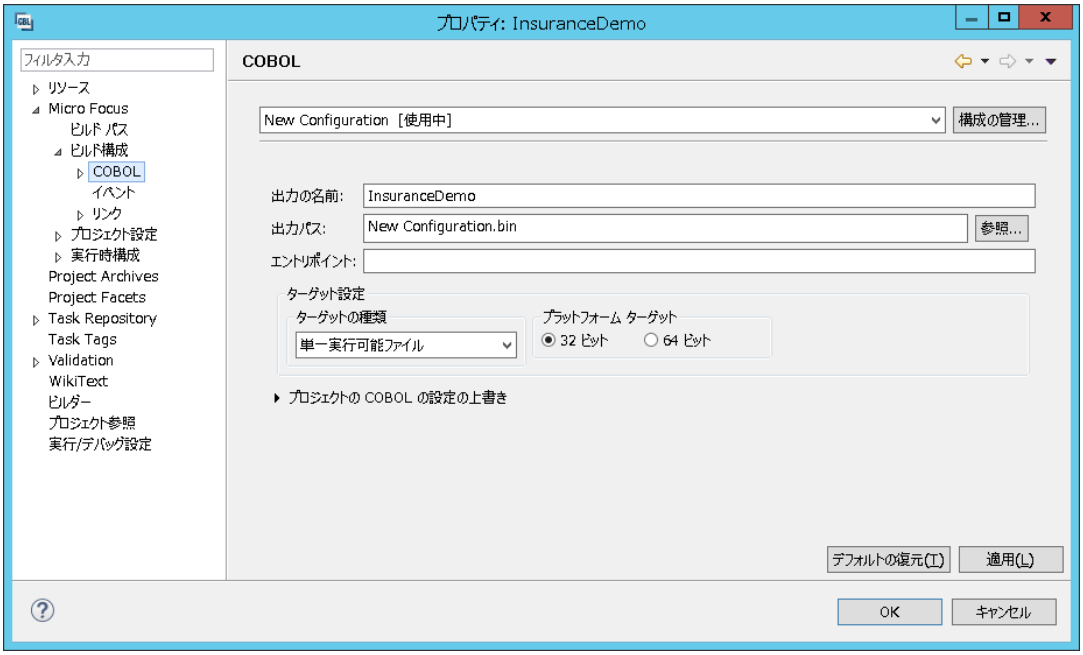
Eclipse のワークスペースとプロジェクトには、アプリケーションの作成に必要なビルドパス、データ接続、フォルダー、およびファイルを表す項目等が含まれています。 ワークスペースには複数のプロジェクトを含めることができ、プロジェクトには、通常、複数の項目が含まれます。COBOL エクスプローラには、ワークスペースに紐づけられたプロジェクト、それらのプロジェクト内の項目が階層状に表示されます。COBOL エクスプローラー上で目的の要素を検索し、編集するファイルを開く、プロジェクトに新規ファイルを追加する、プロジェクトおよび項目のプロパティを表示するなどの操作を実行できます。パースペクティブによっては異なる名称のビューが同等の用途のために用意されています。例えば Java パースペクティブであればパッケージエクスプローラという Java 開発時に必要な要素をフィルター表示するビューが紐つけられています。



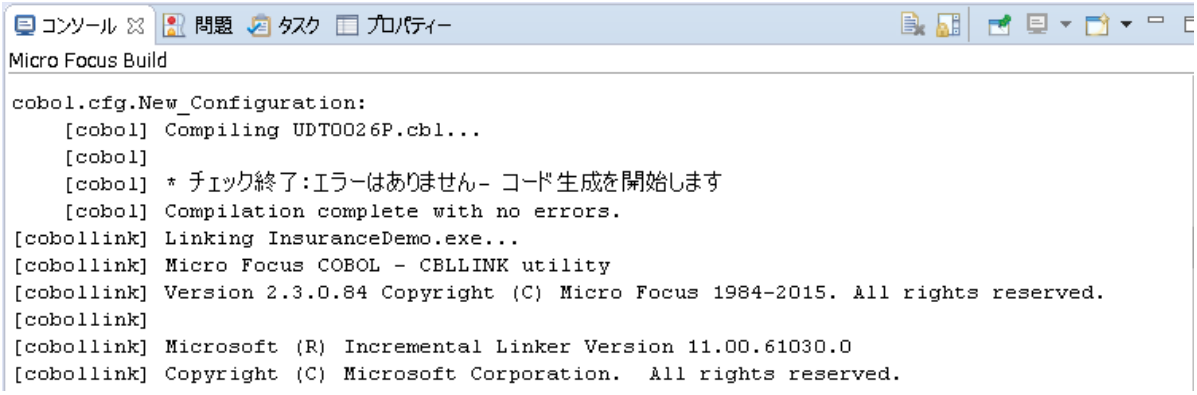
COBOL パースペクティブに紐つくエディターには、COBOL 予約語とデータ名や手続き名などの利用者語を色分け表示したり、COBOL スニペットなど COBOL 言語固有の機能拡張が含まれます。ソースコードを入力するとバックグラウンドチェックを実行して、赤の波線でエラー箇所を強調表示します。 そのエラー箇所にマウスポインタを移動すればエラー内容を確認したり、定義への移動、他の参照検索などの操作が可能です。



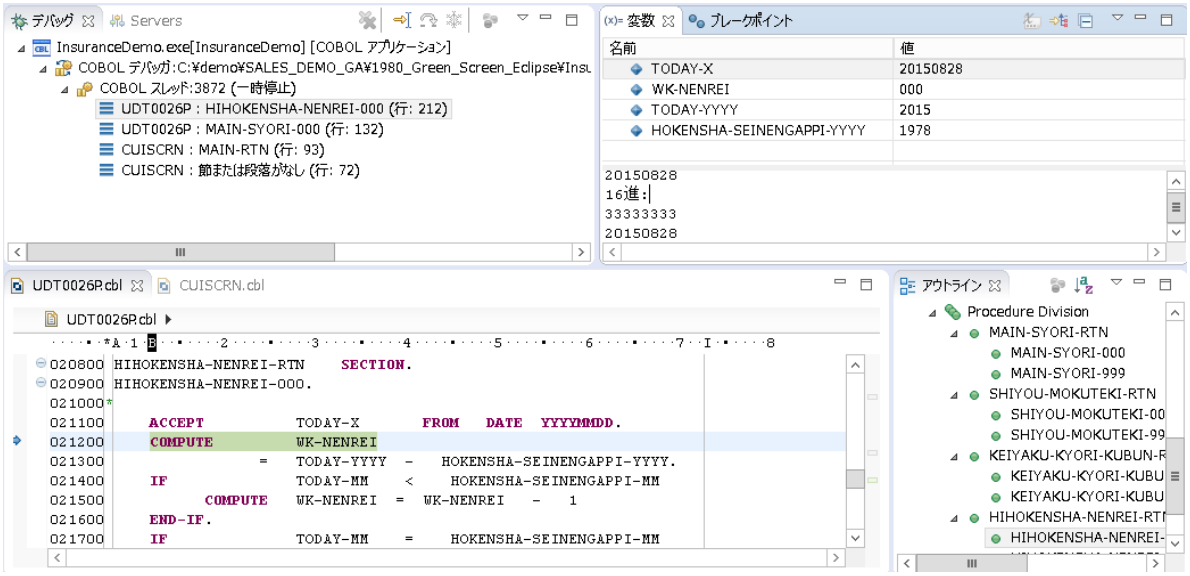
プロジェクトを右クリックし [**プロパティー(R)**] → [**Micro Focus COBOL**] と遷移し表示される各画面では COBOL アプリケーションのビルド方法等を構成します。



コンソールビューにはビルド時のメッセージやアプリケーションのコンソール出力等が表示されます。問題ビューには、不正な構文、キーワードのスペルミス、型の不一致などのコンパイルエラーが表示されます。



ビルドしたアプリケーションは、実行時の論理エラーやセマンティックエラーなどの問題を検出して修正するために、デバッグ機能を利用します。 Eclipse のデバッガーは、コードをステップ実行したり様々な条件を設定したブレークポイントで実行を中断させ、変数ビューを使用してローカル変数やその他の関連データを調べることができます。



デバッグが完了したアプリケーションは、アプリケーションサーバ等との連携機能を利用して自動配備するか、ファイルを手動でコピーして、本番環境に配置します。

なお、本番環境には COBOL Server が事前にインストールされている必要があります。

第3章　はじめての Visual COBOL

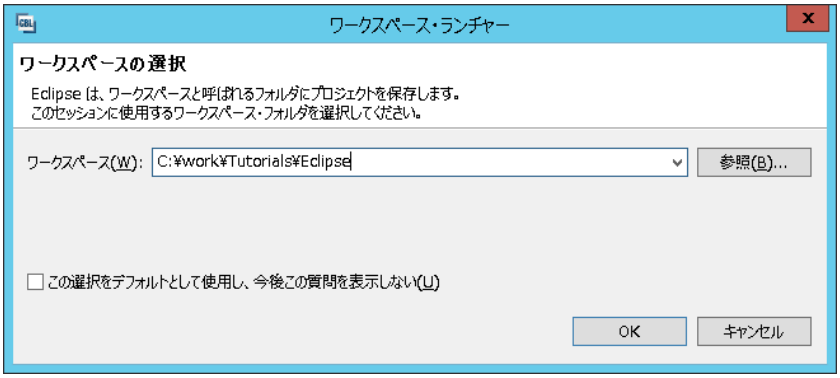
それでは、コマンドプロンプト画面に「Hello World」を表示する COBOL アプリケーションを Visual COBOL for Eclipse で作成します。

1　Visual COBOL for Eclipse を起動します。

Windows のスタートメニューから、**Visual COBOL for Eclipse** をクリックします。

2　ワークスペースの保存先を選択します。

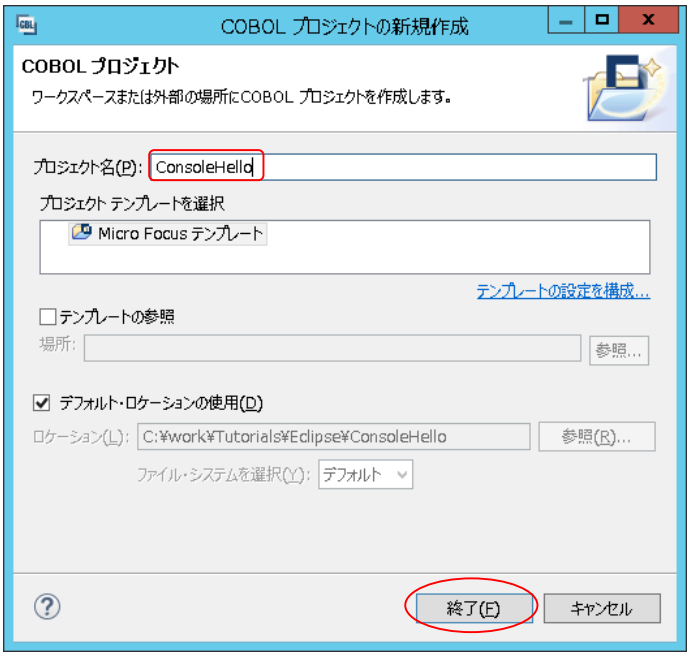
ワークスペースを保存する任意のフォルダを指定します。[**参照(B)**] ボタンを押下しエクスプローラ経由で選択することも可能です。



3　COBOL プロジェクトを作成します。

ファイル(F)メニューから **新規(N) → COBOL プロジェクト** を選択します。

プロジェクト名欄に **ConsoleHello** と入力し [**終了(F)**] ボタンを押下します。



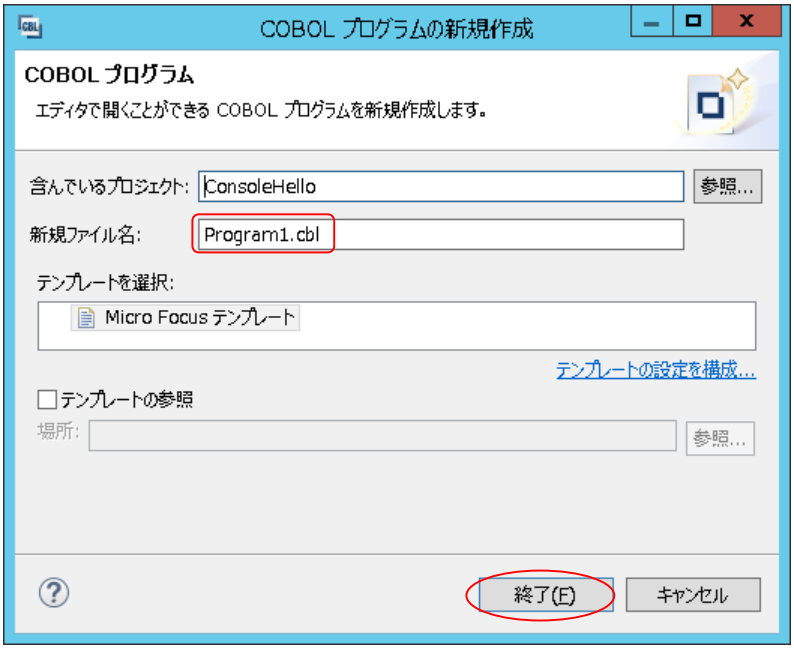
4　COBOL プログラムを追加します。

COBOL エクスプローラビューにて **プロジェクト** フォルダを右クリックし

新規作成(N) → COBOL プログラム

を選択します。

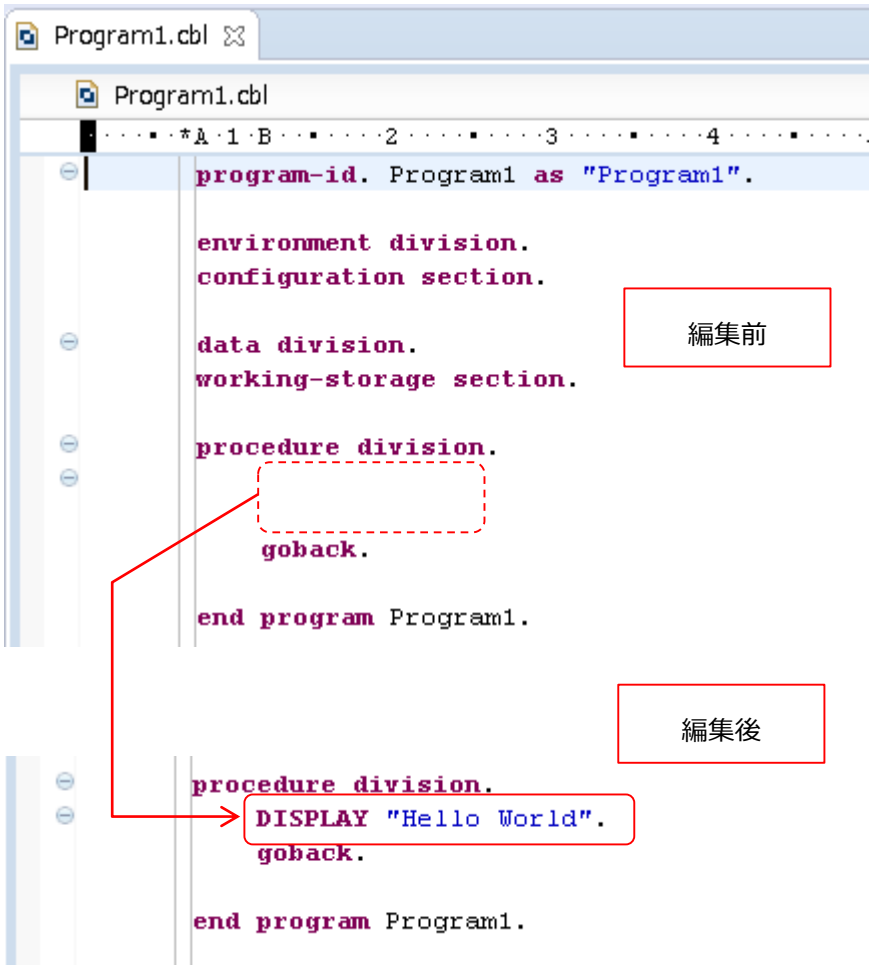
ファイル名はデフォルトの Program1.cbl を使用します。[**終了(F)**] ボタンを押下します。



5　エディターで COBOL ソースコードを入力します。

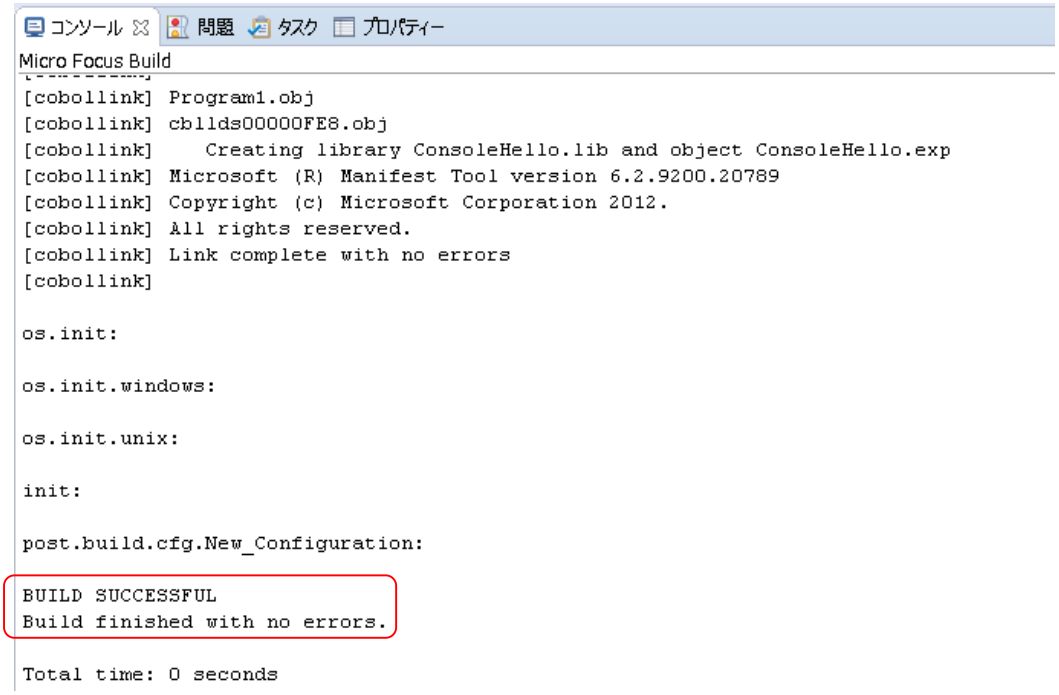
COBOL プロジェクトにて COBOL プログラムを新規に作成するとプログラムを構成する見出し部(identification division)、環境部(environment division)、データ部(data division)、手続き部(procedure division) や　program-id 段落が埋め込まれたかたちで生成されます。今回は「Hello World」を表示して終了するプログラムなので、手続き部３行目の goback 文の手前に以下のように display 文を加えます。

なお、COBOL 正書法ではエディタービュー左右にある線で区切られた領域を特別な領域として利用するので、通常のソースコードはこれを避けて入力します。



6　COBOL アプリケーションをビルドします。

終止符(ピリオド)を含めてスペルミスがなければ、エディタービュー上の Program1.cbl をアクティブにしたまた**ファイル(F)** メニューの**保管(S)** 或いは **Ctrl + S** キーで保存します。これにより自動的にコンパイラがキックされビルド処理が始まります。コンソールビューに正常にビルドできた旨のメッセージが出力されていることを確認します。



メモ：

Visual COBOL for Eclipse ではデフォルトでこのような自動ビルド機能が有効となっています。自動ビルド機能を無効にし、任意のタイミングでビルドしたい場合は

プロジェクトメニュー → **自動的にビルド**

についたチェックを外します。ビルドする際はプロジェクトメニュー配下の「すべてビルド」或いは「プロジェクトをビルド」を目的に応じて選択します。

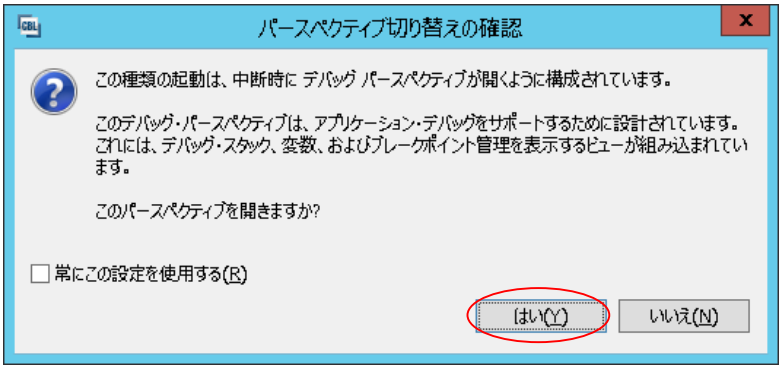
7　COBOL アプリケーションをデバッグ実行します。

COBOL エクスプローラ上の **Program1.cbl** を右クリックし

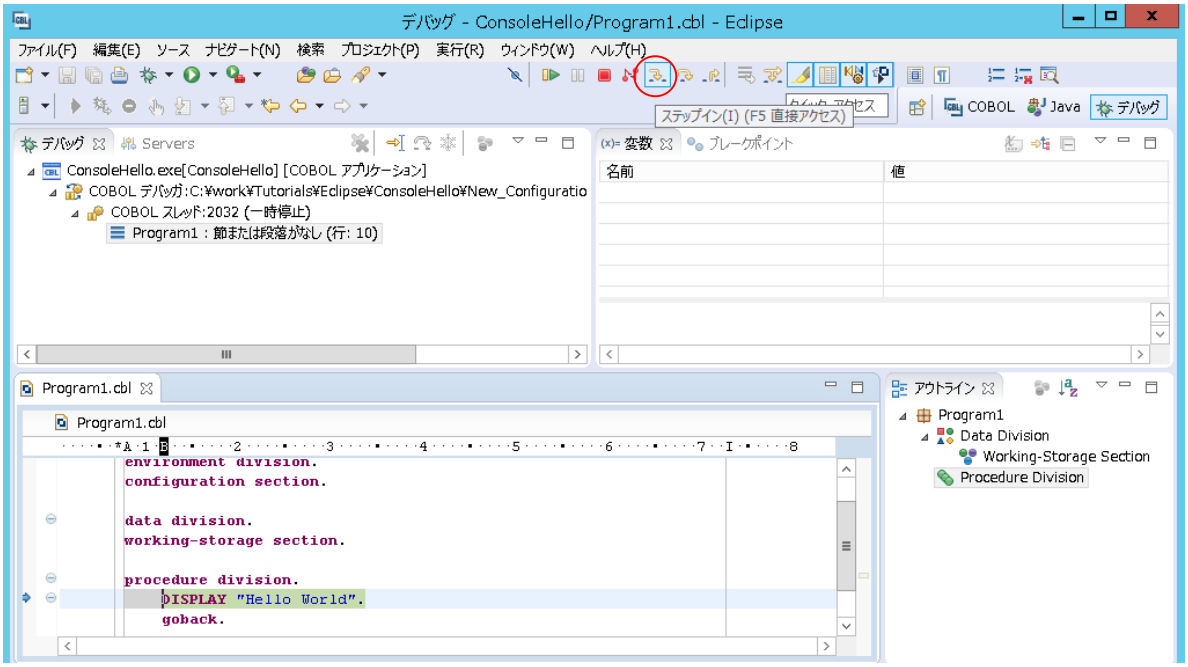
デバッグ(D) → COBOL アプリケーション

を選択します。

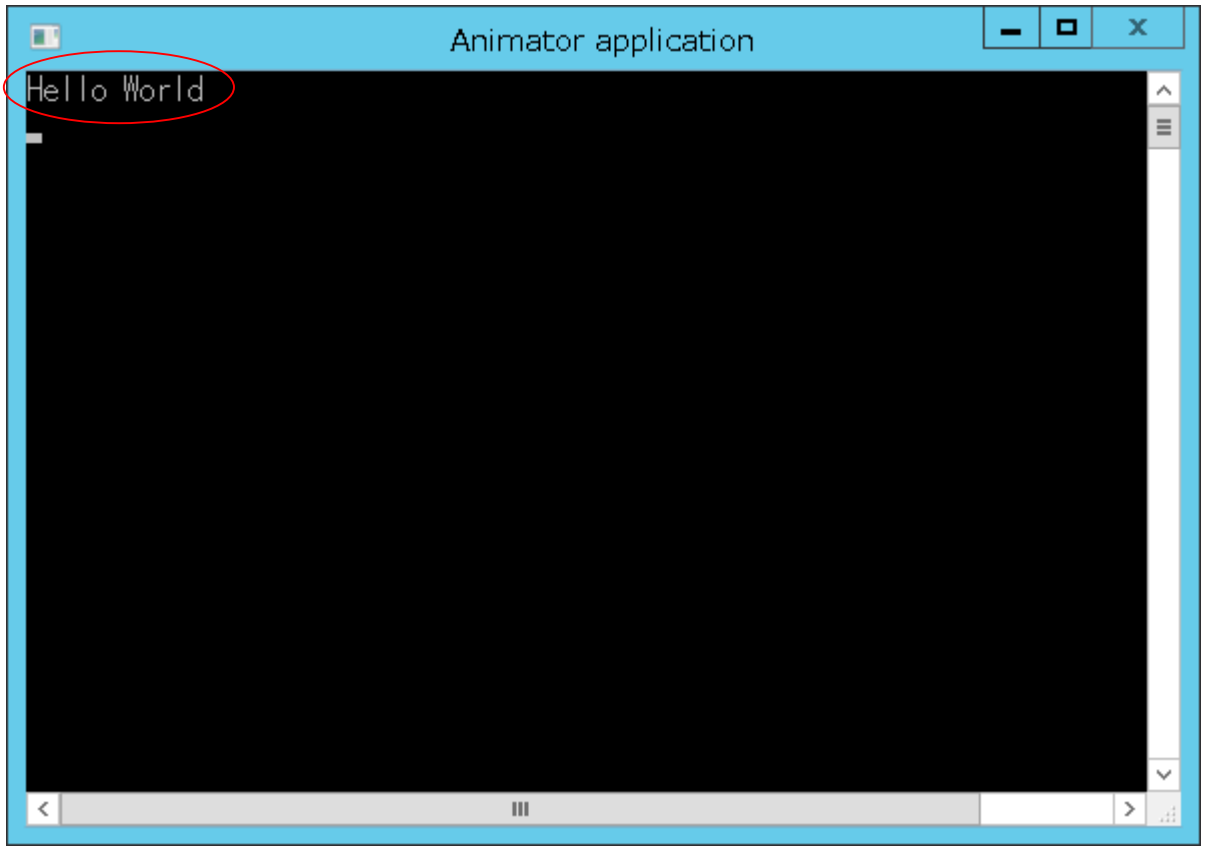
パースペクティブ切り替えに関する確認メッセージには［**はい(Y)**］を選択します。



デバッグパースペクティブに切り替わりましたら、DISPLAY 文を実行する手前でステップが一時停止していますので、［**ステップイン**］アイコンを一回クリックし DISPLAY 文を実行します。



Animator application 画面に「Hello World」が表示されたことを確認できましたら、[**ステップイン**] アイコンを再度クリックしデバッグを終了します。



第4章　COBOL JVM アプリケーションの作成

本章では、Visual COBOL for Eclipse の COBOL JVM 機能を利用して Java バイトコードにコンパイルされた COBOL アプリケーションを作成する方法を紹介します。ここでは、Java プログラムから「Hello World」を表示する COBOL プログラム呼び出すアプリケーションを作成します。

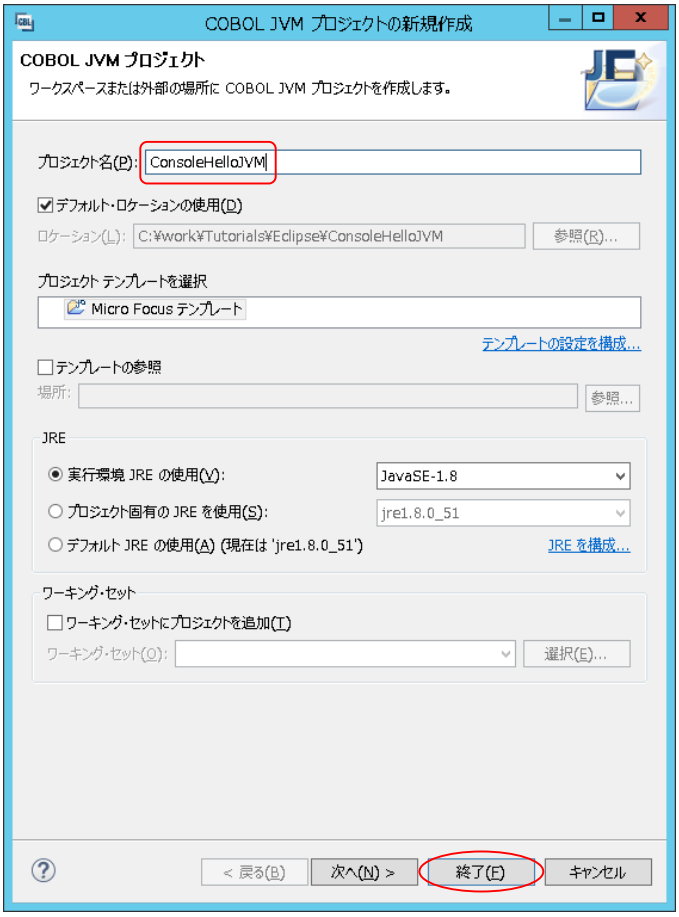
1　Visual COBOL for Eclipse を起動します。

Windows のスタートメニューから、**Visual COBOL for Eclipse** をクリックし、第３章で使用したワークスペースを選択します。第３章から続けて実施される場合は、このステップはスキップしワークスペースを継続して使用します。

2　COBOL JVM プロジェクトを作成します。

ファイル(F)メニューから **新規(N)** → **COBOL JVM プロジェクト** を選択します。

プロジェクト名欄に **ConsoleHelloJVM** と入力し［**終了(F)**］ボタンを押下します。

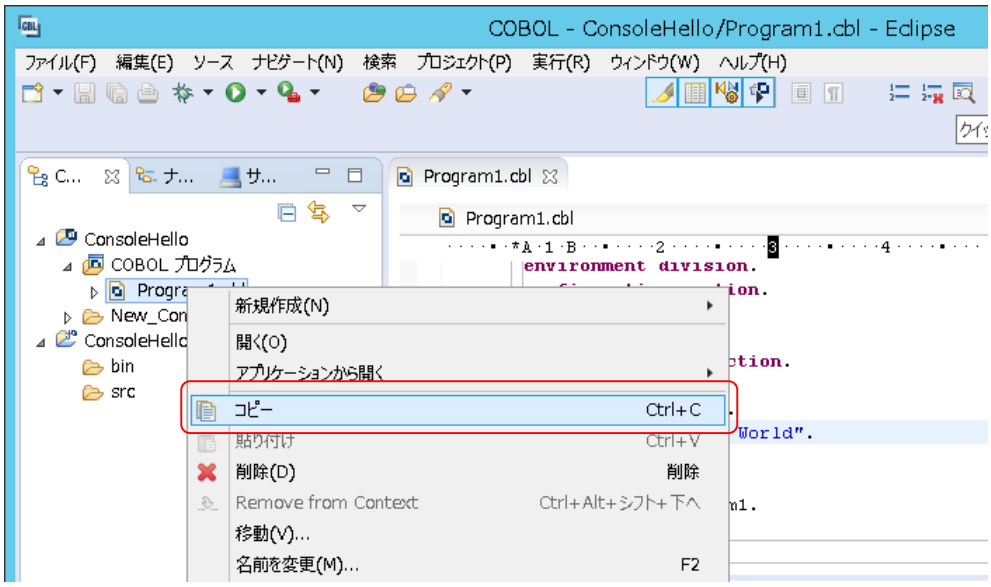


3　Native 用のプロジェクトで使用したプログラムをコピーします。

COBOL エクスプローラビューにて **ConsoleHello** プロジェクト配下の COBOL プログラムフォルダに前章で追加した Program1.cbl を右クリックし

コピー

を選択します。

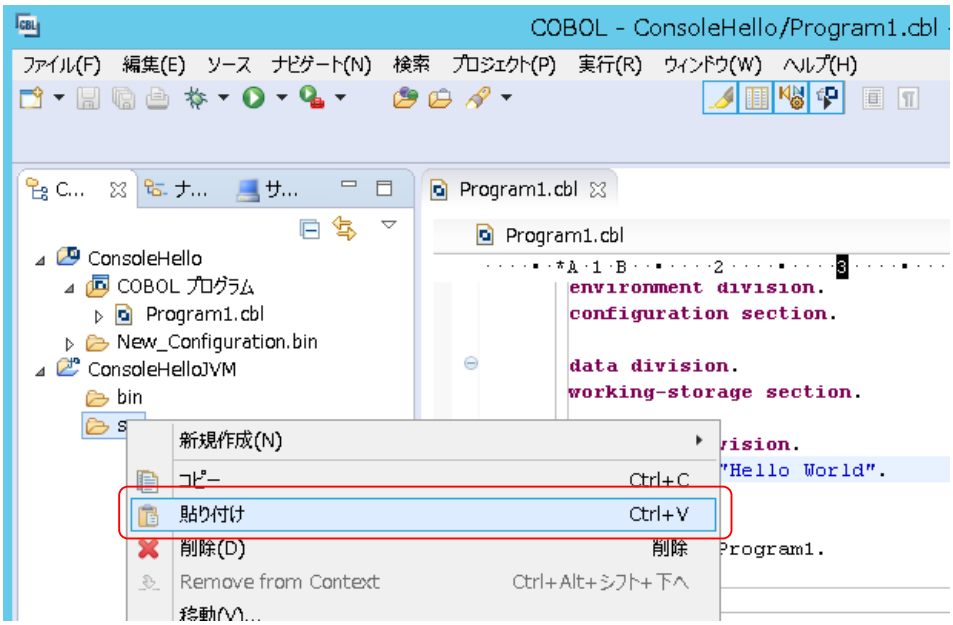


4　COBOL for JVM のプロジェクトにコピーしたプログラムを追加します。

COBOL for JVM のプロジェクト配下にある src フォルダを右クリックし

貼り付け

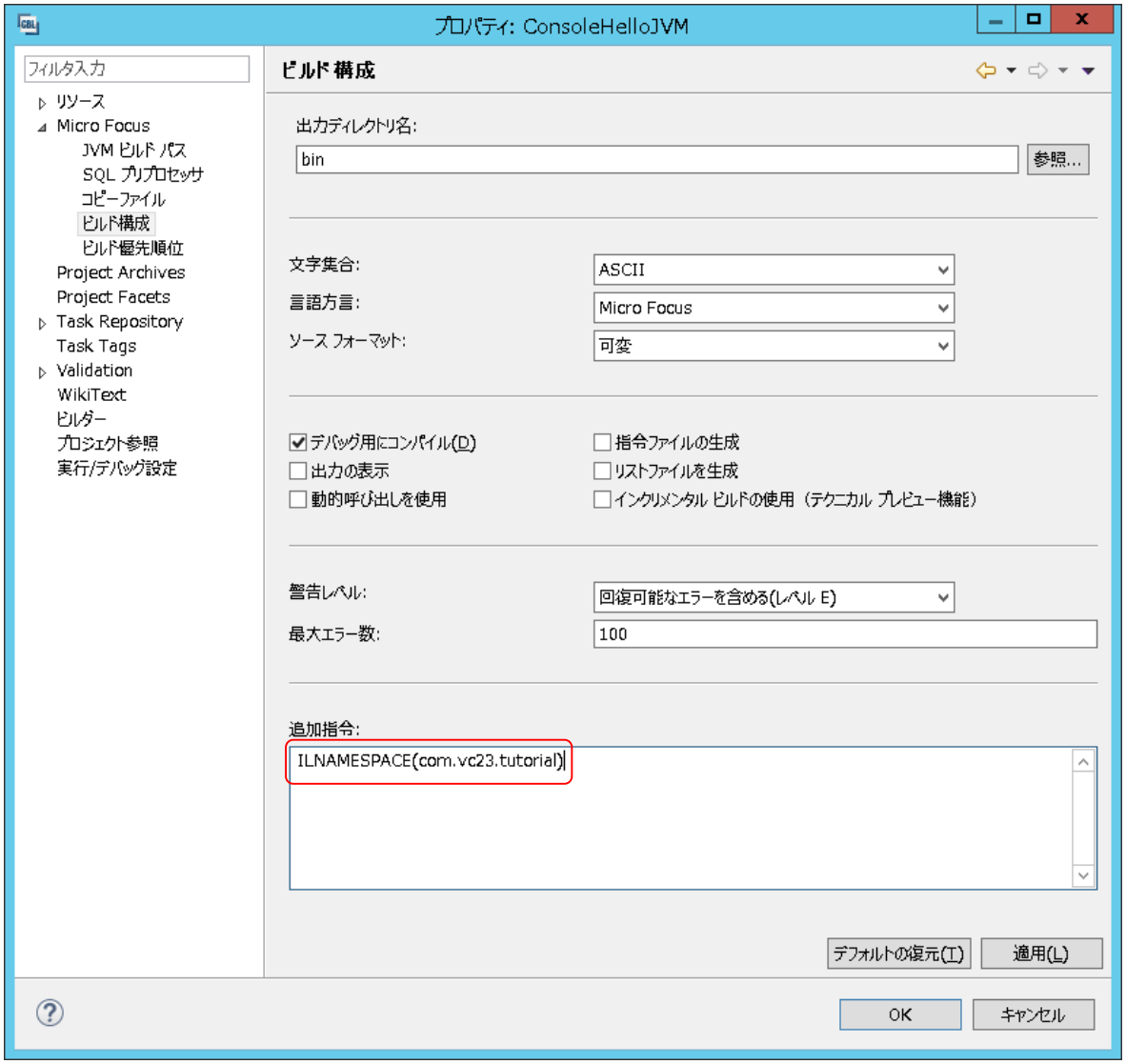
を選択します。



5　COBOL から生成された JVM クラスをパッケージ化します。

COBOL エクスプローラビューにて COBOL for JVM のプロジェクトを右クリックし、[**プロパティ(R)**] を選択します。

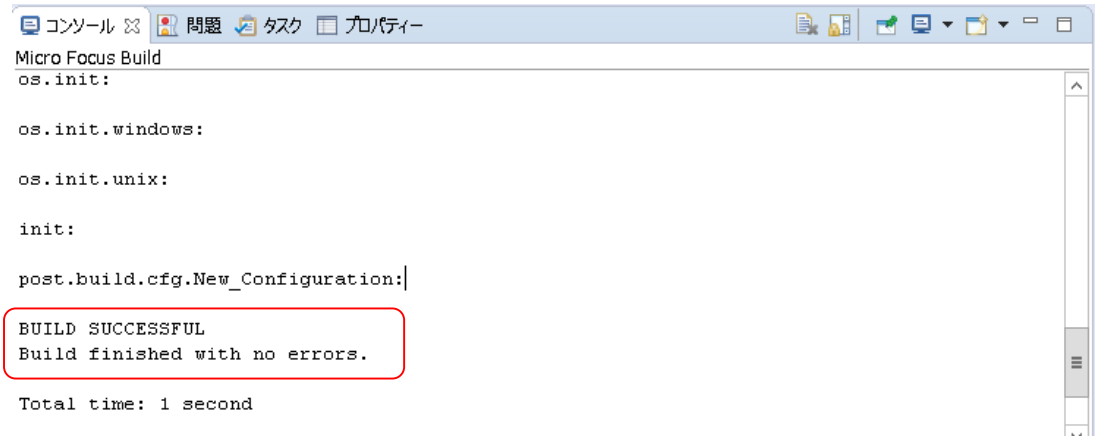
[**Micro Focus**] > [**ビルド構成**]
へとナビゲートし展開される画面中の [**追加指令**] 欄に
ILNAMESPACE(com.vc23.tutorial)
を記入し、[**OK**] ボタンを押下します。



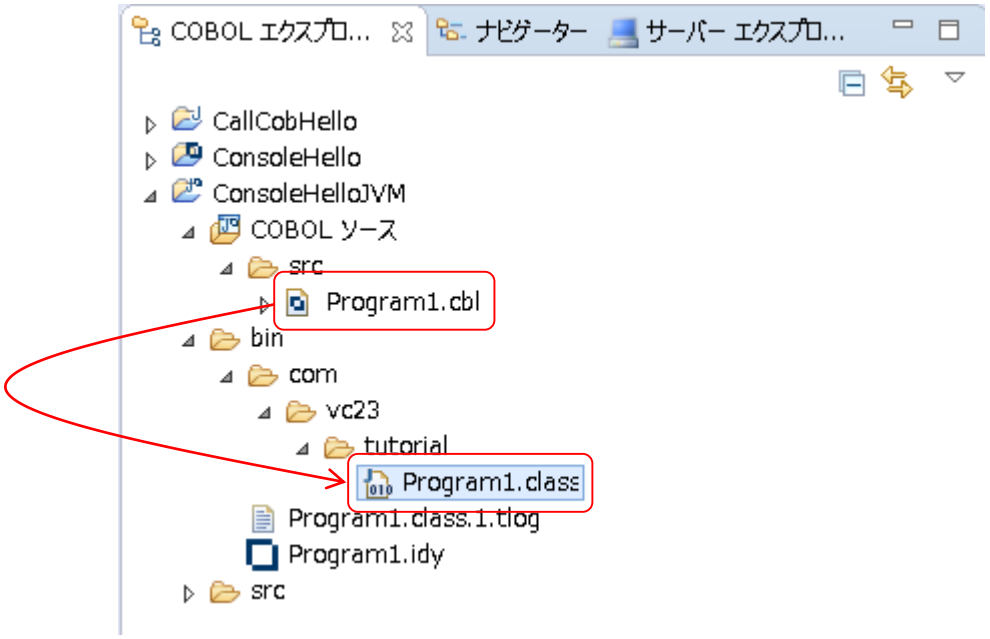
6　COBOL JVM Class ファイルをコンパイルします。

自動的にビルドが有効になっていれば、前ステップでプログラムを追加した時点でビルド処理が走ります。

コンソールビューで正常に処理できた旨のメッセージを確認できます。



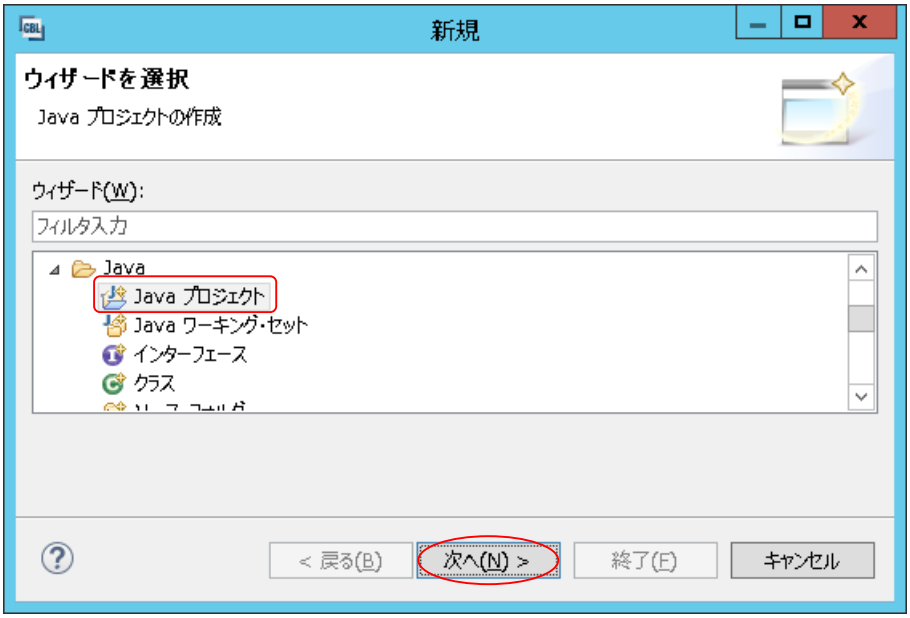
また、COBOL エクスプローラビューでは、追加したプログラムが .class の拡張子を持った JVM クラスとしてビルドされていることを確認できます。プログラム中にはパッケージ化する旨の記述は加えていませんが先のステップで指定したコンパイラ指令により、パッケージ化した JVM クラスとして生成されています



7　Java プロジェクトを作成します。

ファイル(F)メニューから **新規(N)** → **その他** を選択します。

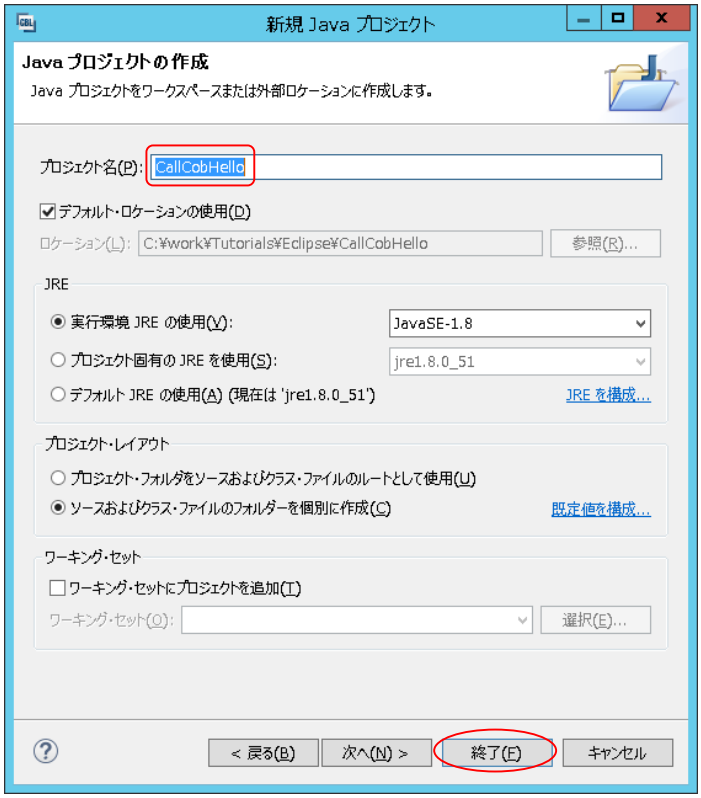
Java 配下の Java プロジェクトを選択し [**次へ(N)**] ボタンを押下します。



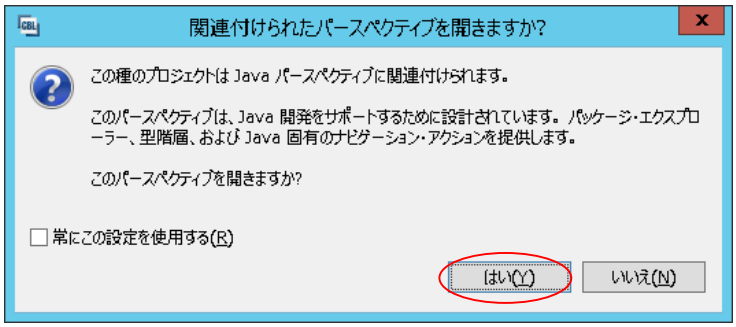
プロジェクト名欄に

CallCobHello

と入力し [**終了(F)**] ボタンを押下します。



Java パースペクティブに関連付けられる旨のメッセージには［**はい(Y)**］を選択します。



8　Java クラスファイルを追加します。

パッケージエクスプローラビューにて **CallCobHello** プロジェクト配下の src フォルダを右クリックし

新規(W) → クラス

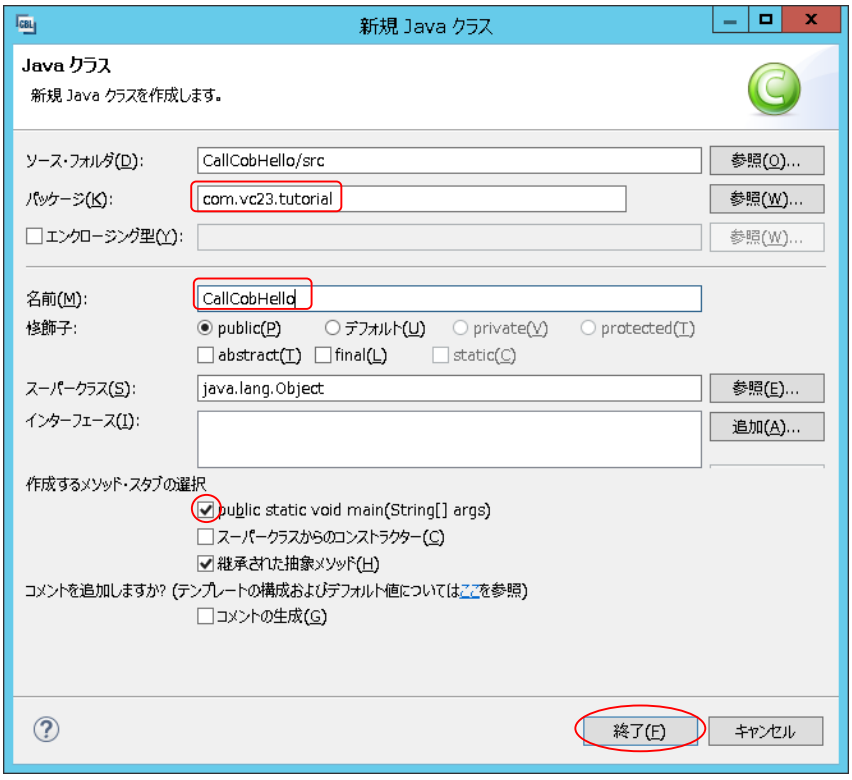
を選択します。

新規 Java クラスウィンドウでは

パッケージ欄　com.vc23.tutorial

名前欄　　CallCobHello

のように入力します。また、「public static void main(String[] args)」にチェックを入れ、［**終了(F)**］ボタンを押下します。

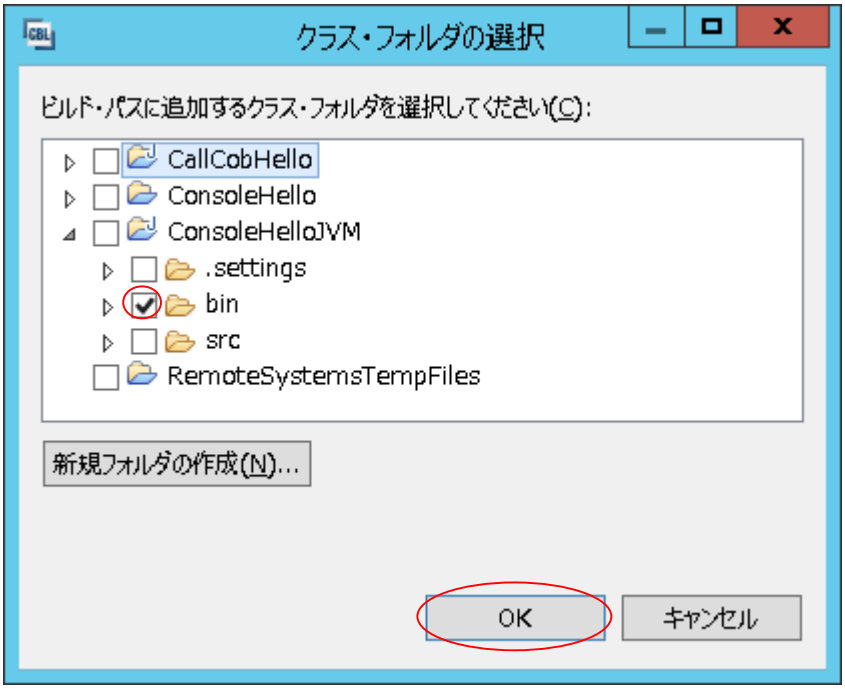


9　COBOL JVM プロジェクト及び Visual COBOL のランタイムをビルドパスに追加します。

パッケージエクスプローラビューにて **CallCobHello** プロジェクトを右クリックし［**プロパティ(R)**］を選択します。

［**Java のビルド・パス**］ページの［**ライブラリー**］タブを選択します。
［**クラスフォルダの追加(C)**］ボタンを押下します。

ConsoleHelloJVM プロジェクト配下の［**bin**］フォルダにチェックを入れ、［**OK**］ボタンを押下します。

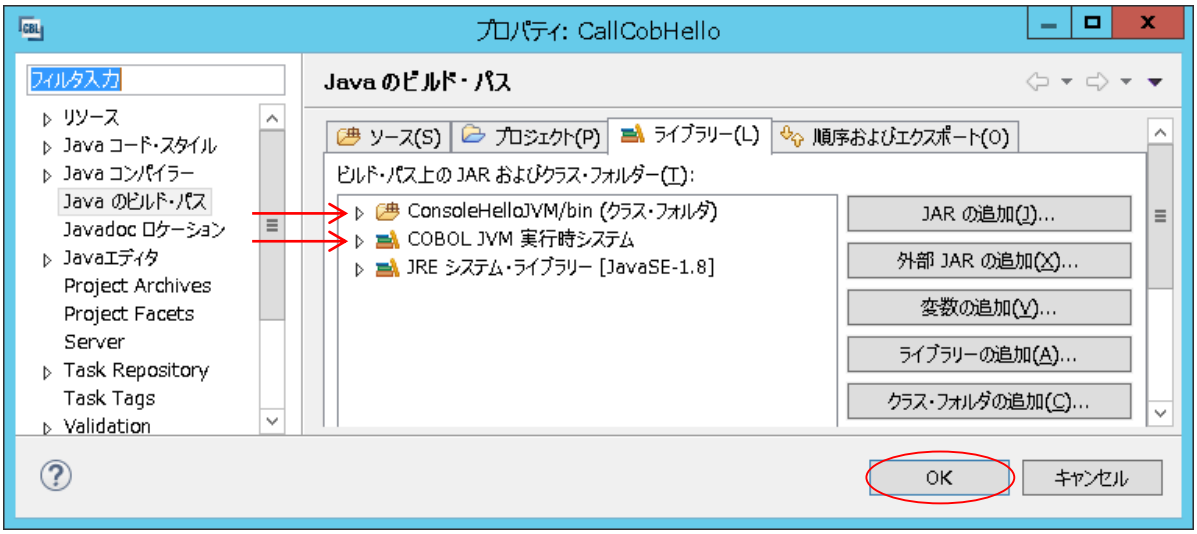


続いて、[**ライブラリーの追加(A)**] ボタンを押下します。

[**COBOL JVM 実行時システム**] を選択し [**次へ(N)**] ボタンを押下します。

[**終了(F)**] ボタンを押下します。

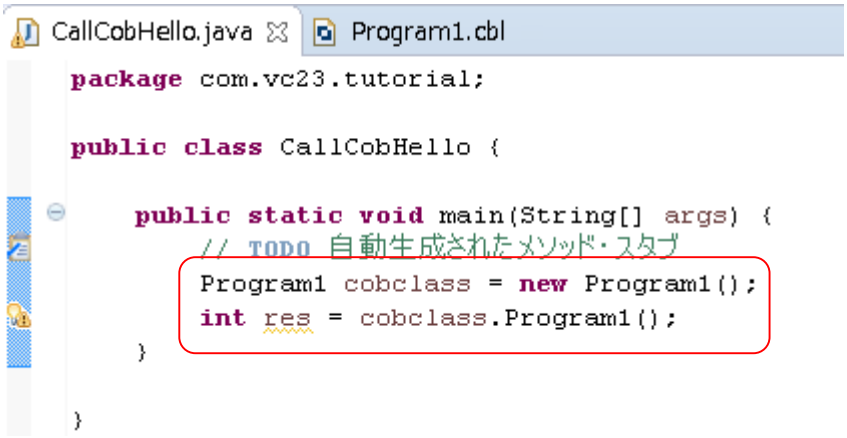
上で追加したライブラリーが画面に反映されていることを確認し、[**OK**] ボタンを押下しプロパティ画面を閉じます。



10 エディターで Java ソースコードを入力します。

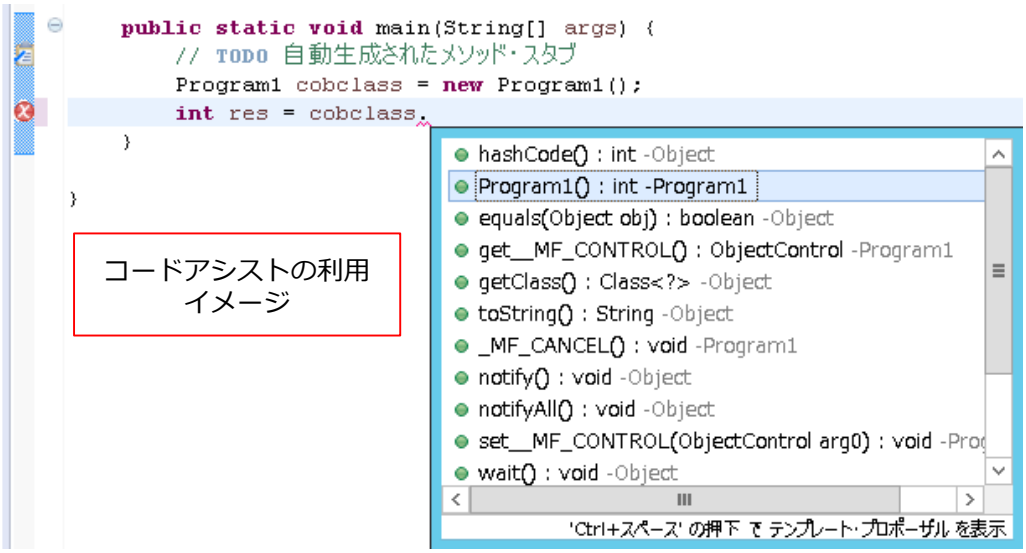
main メソッドに以下のコードを追記します。

```
Program1 cobclass = new Program1();
int res = cobclass.Program1();
```



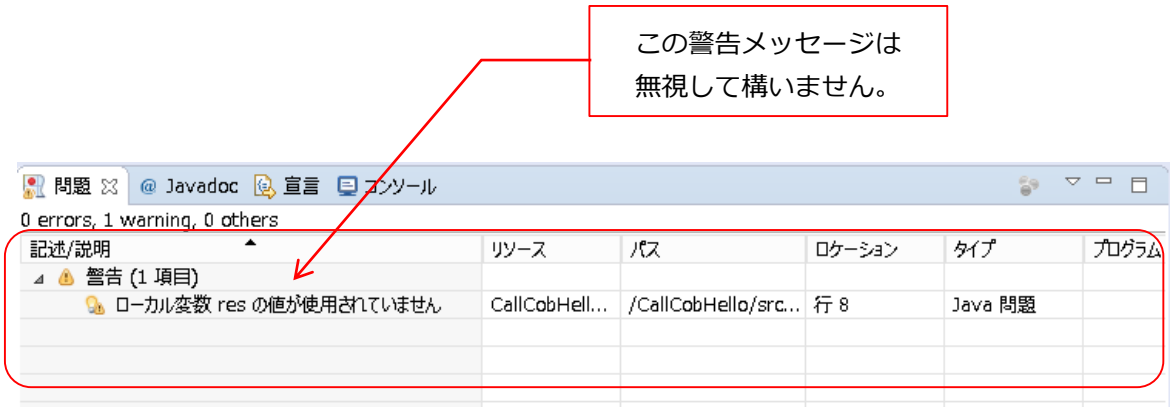
メモ：

従来の手続き型の COBOL プログラムを JVM クラスにコンパイルすると、プログラム名がクラス名となり、PROCEDURE DIVISION 以下で記載された命令は、プログラムと同名のインスタンスメソッドとして実装されます。生成された JVM クラスは Java から生成される JVM クラスと同様に扱えるため、Java 中で特別なロジックを記述することなく COBOL の呼び出し命令を記述できます。Eclipse 上の Java のエディタも COBOL から生成されたクラスのシンボルを Java と同様に認識できるため、Java エディタ上で COBOL プログラムに対してコードアシストの機能等を利用することができます。コードアシストは Ctrl + Space で起動できます。



11 COBOL JVM Class ファイルをコンパイルします。

スペルミスがなければ、エディタービュー上の CallCobHello.java をアクティブにしたまた**ファイル(F)** メニューの**保管(S)** 或いは **Ctrl + S** キーで保存します。問題ビューに何もエラーが出力されていないことを確認します。

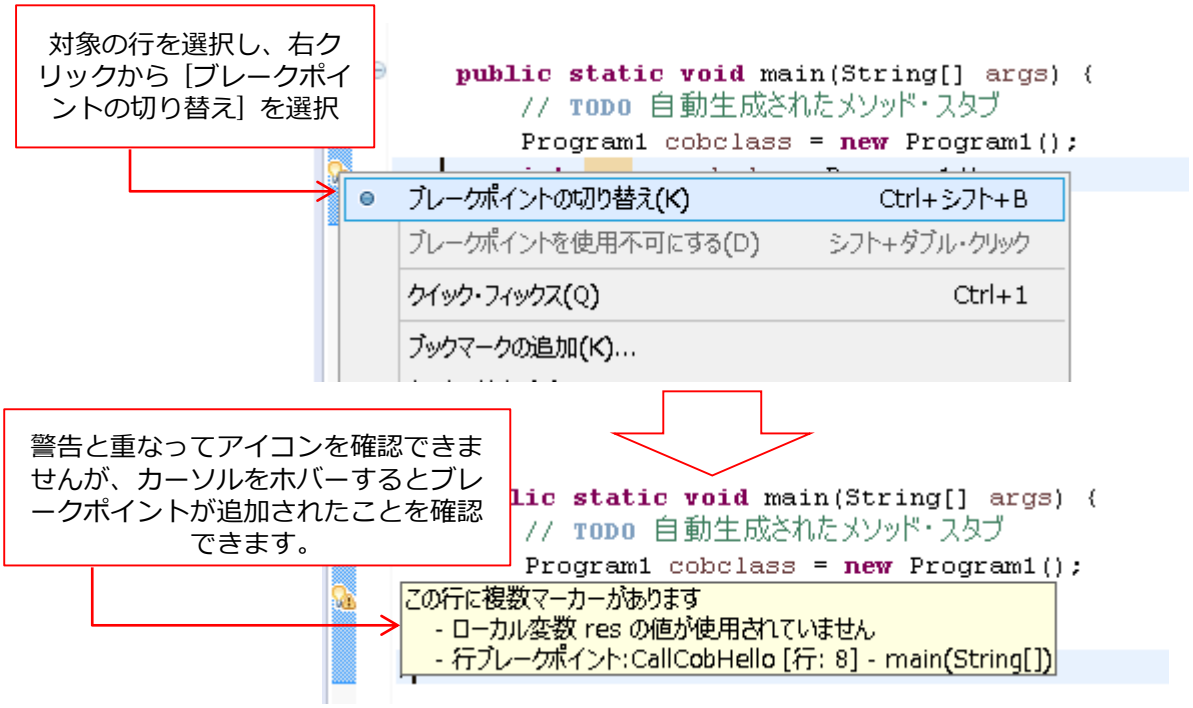


12 アプリケーションをデバッグ実行します。

COBOL プログラム中のメソッド呼び出す

```
int res = cobclass.Program1();
```

にブレークポイントを設定します。



CallCobHello > src > com.vc23.tutorial 配下の **CallCobHello.java** を右クリックし

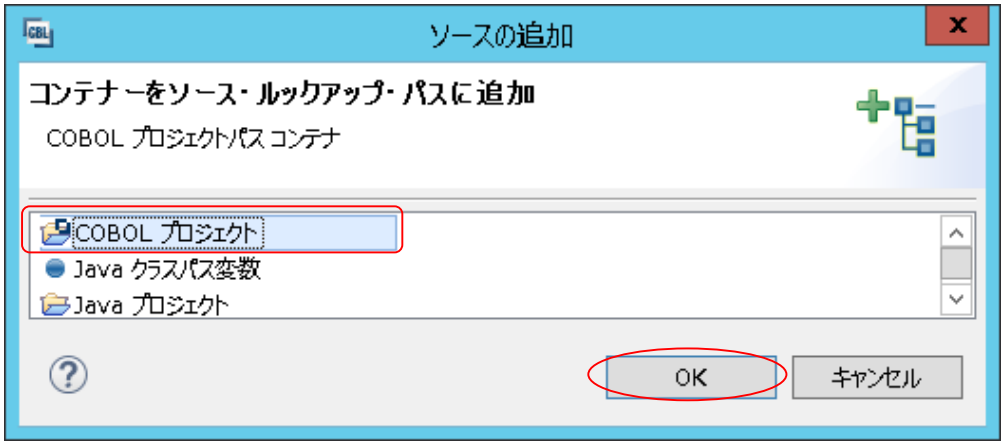
デバッグ(D) → デバッグの構成(B)

を選択します。

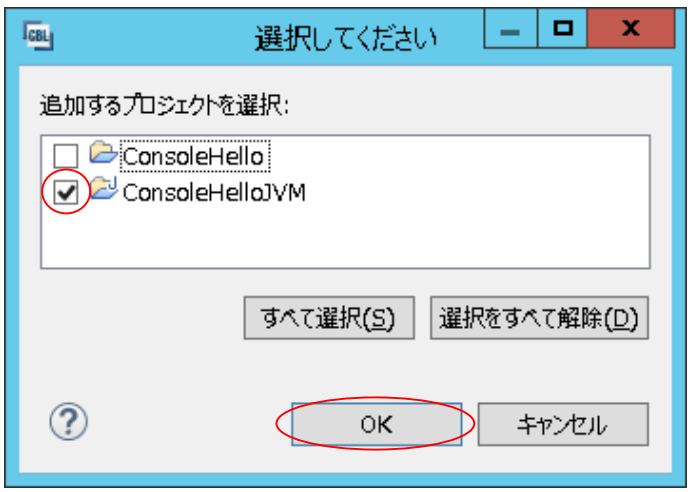
[**ソース**] タブをクリックします。

[**追加(A)**] ボタンを押下します。

[**COBOL プロジェクト**] を選択の上、[**OK**] ボタンを押下します。

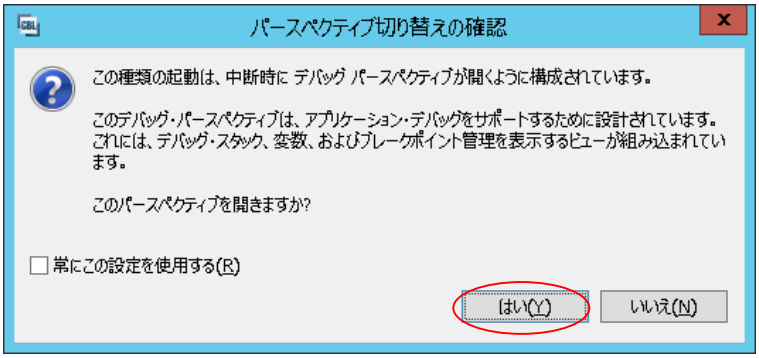


COBOL for JVM のプロジェクト [**ConsoleHelloJVM**] にチェックを入れ、[**OK**] ボタンを押下します。

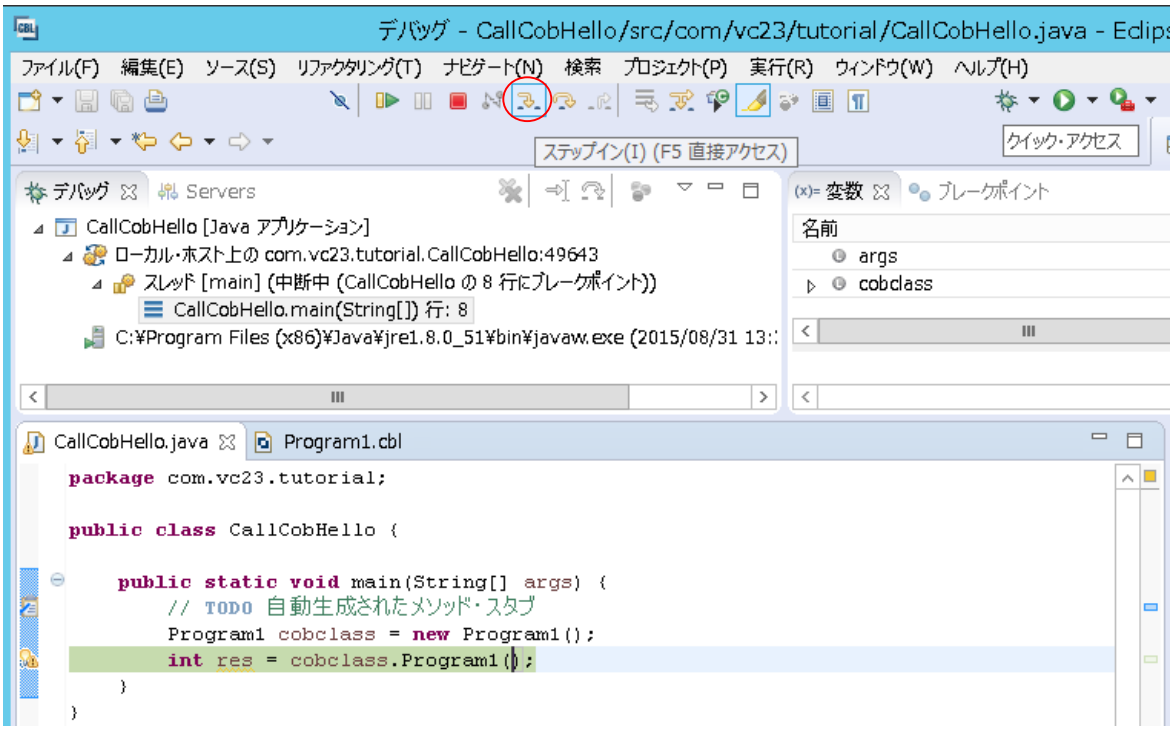


[**デバッグ(D)**] ボタンを押下します。

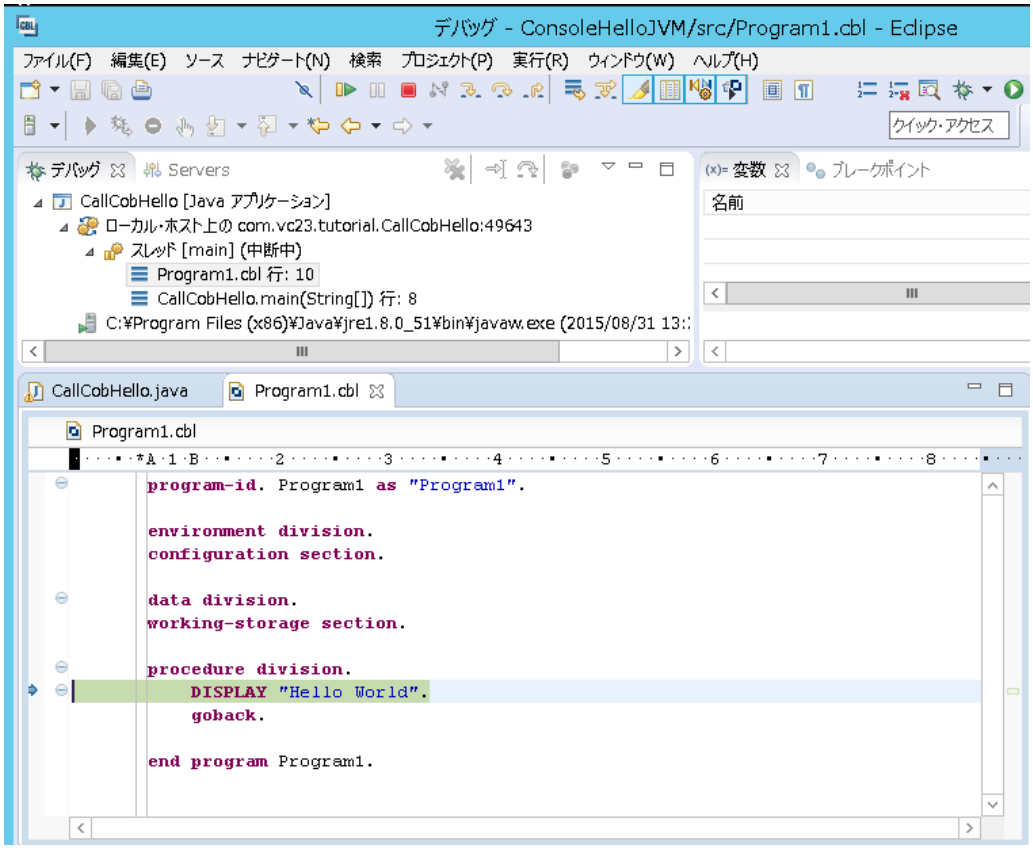
パースペクティブ切り替えの確認メッセージには [**はい(Y)**] を選択します。



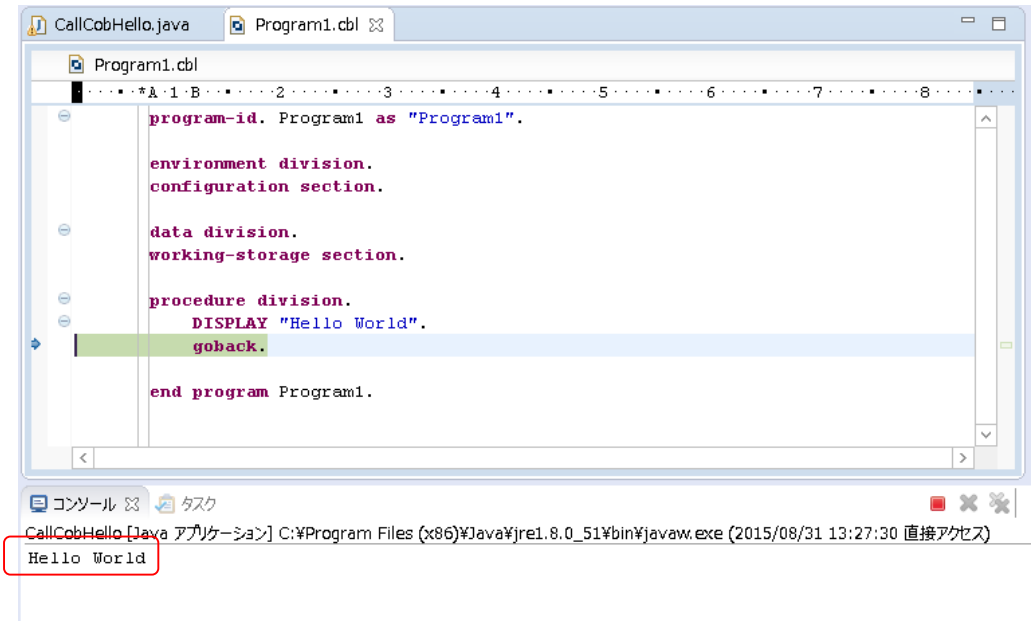
ステップインアイコンをクリックします。ステップ実行には、ステップイン、ステップオーバとメニューが用意されています。ステップインを選択した場合は Call 先のメソッドの中までステップ実行し、ステップオーバーを選択した場合は、メソッドの中にステップを進めずメソッドを実行します。ここでは、ステップインを選択し COBOL で書かれたメソッドの中までステップを進めます。



COBOL アプリケーションのソースの中にステップが進んでいることが確認できます。



再度、ステップインアイコンをクリックし、DISPLAY 文を実行するとコンソールに「Hello World」が表示されることが確認できます。



確認できましたら、アプリケーションが終了するまでステップを進めデバッグを終了させます。

第5章　Visual COBOL のファイル入出力

次に、エクセルやメモ帳で作成した CSV ファイルを読み込んで、固定長順編成ファイルを作成する COBOL アプリケーションを Visual COBOL for Eclipse で作成しましょう。

1　Visual COBOL for Eclipse を起動します。

Windows のスタートメニューから、**Visual COBOL for Eclipse** をクリックします。

ワークスペースは第３章で用意したものをそのままご使用いただいても構いません。

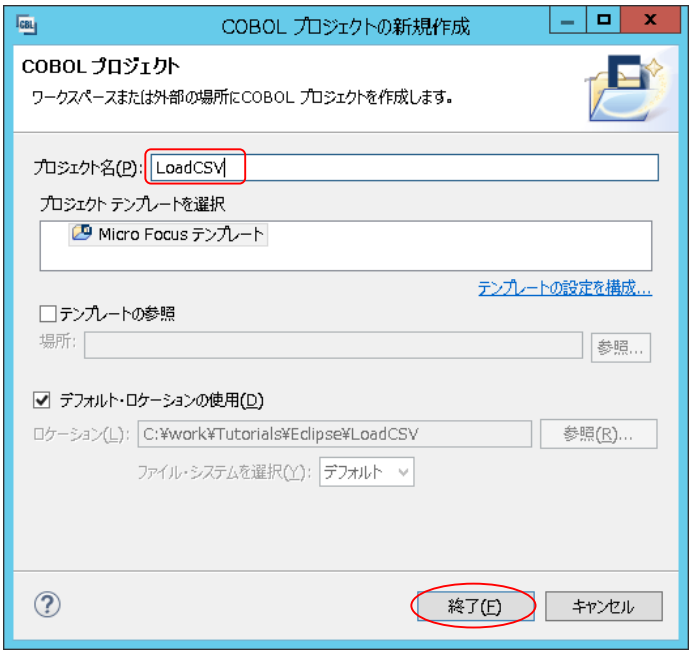
2　COBOL プロジェクトを作成します。

ファイル(F)メニューから **新規(N) → COBOL プロジェクト** を選択します。

メモ：

ファイル(F) メニュー → 新規(N) にて候補として表示されるプロジェクトはワークスペース中でアクティブなパースペクティブにより切り替わります。例えば、Visual COBOL が提供するパースペクティブがアクティブな場合、COBOL プロジェクトは候補として表示されます。一方、他の Java のパースペクティブ等がアクティブな場合は候補に含まれないため、パースペクティブを COBOL に切り替えてから作業します。

プロジェクト名欄に **LoadCSV** と入力し[**終了(F)**] ボタンを押下します。



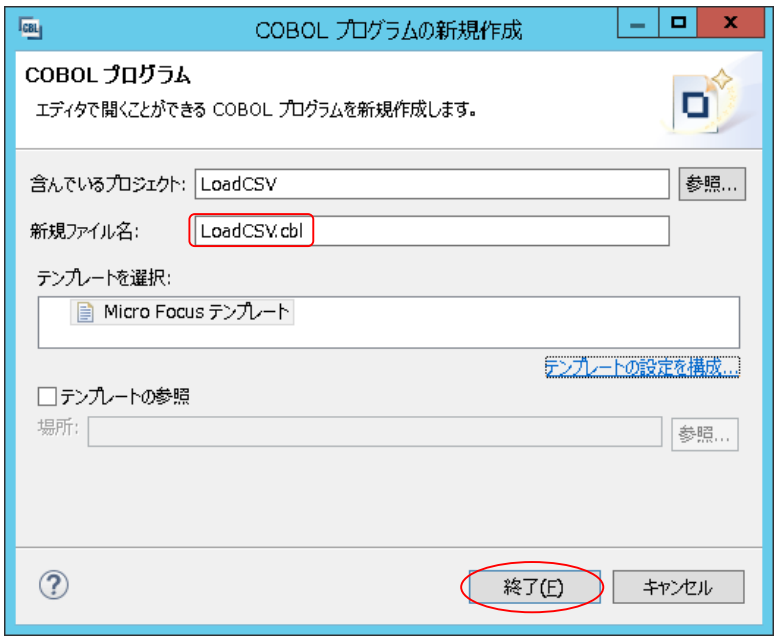
3　COBOL プログラムを追加します。

COBOL エクスプローラビューにて **LoadCSV プロジェクト** フォルダを右クリックし

新規(N) → COBOL プログラム

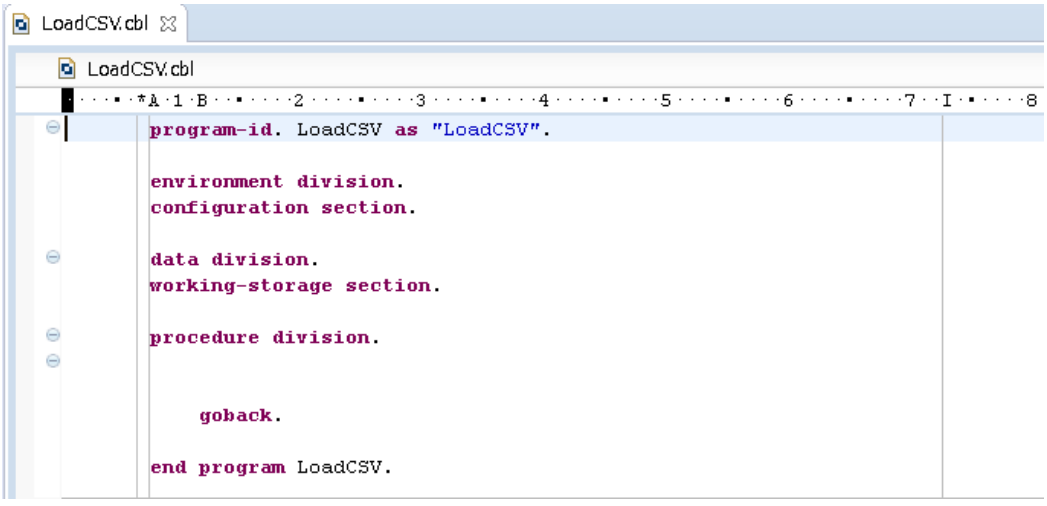
を選択します。

新規ファイル名欄には **LoadCSV.cbl** を指定します。[**終了(F)**] ボタンを押下します。



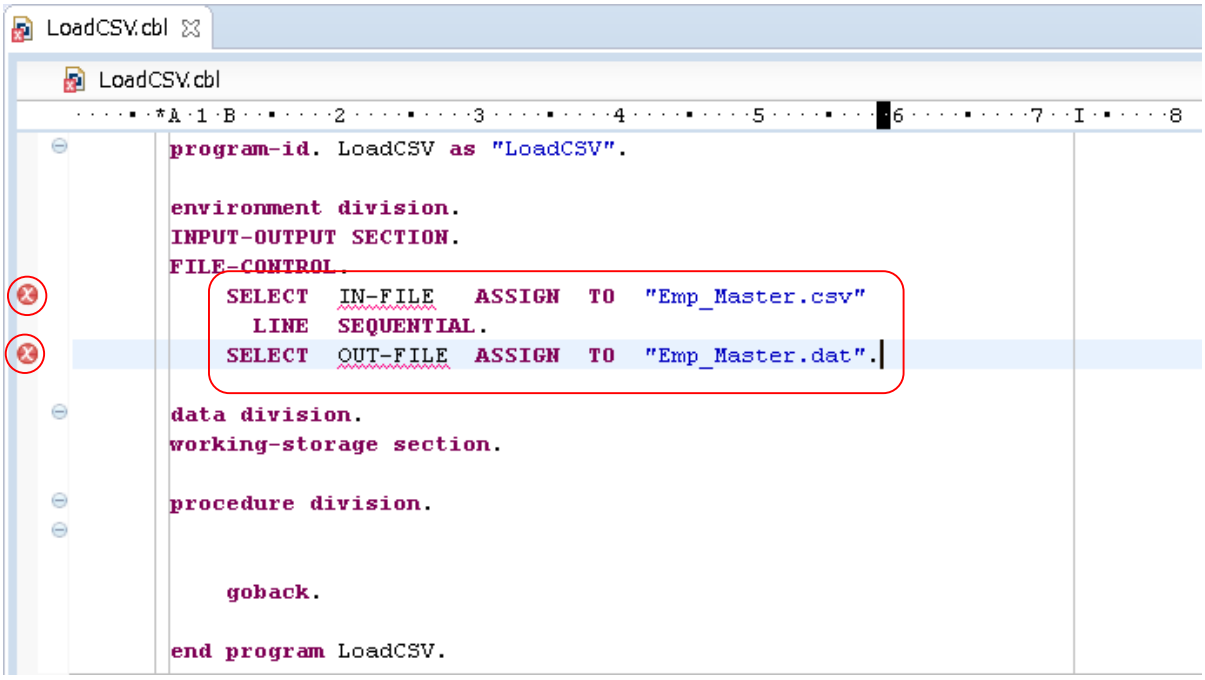
4　コードエディターで COBOL ソースコードを入力します。

COBOL ソースファイルを新規に作成した直後の段階ではコンソールアプリケーションのひな形が埋め込まれています。ここでは、環境部(environment division)、データ部(data division)、手続き部(procedure division) を書き換えます。



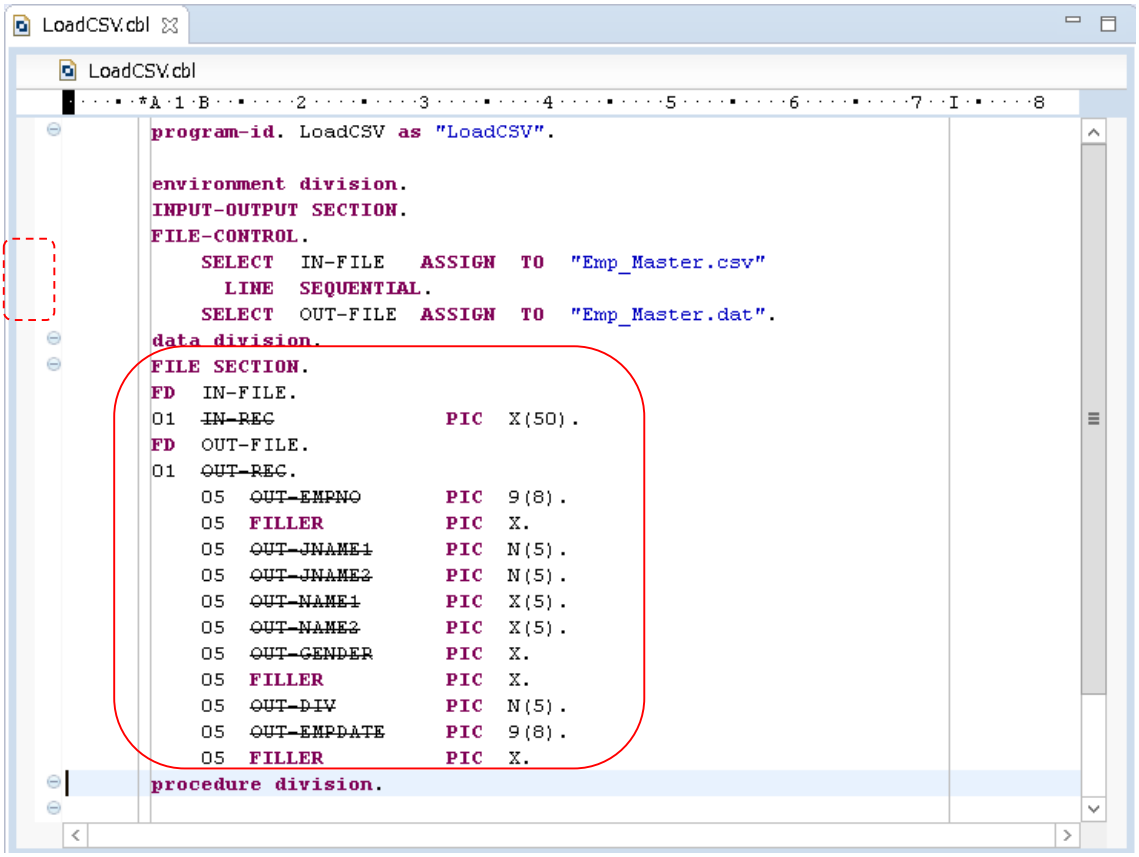
まず、環境部の構成節(configuration section) を削除し、以下の入出力節(input-output section) を追加します。 まだ、データ部のファイル定義が未入力なので in-file と out-file がエラーとなりますが、ここでは無視して構いません。

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT  IN-FILE   ASSIGN  TO  "Emp_Master.csv"
      LINE  SEQUENTIAL.
    SELECT  OUT-FILE  ASSIGN  TO  "Emp_Master.dat".
```



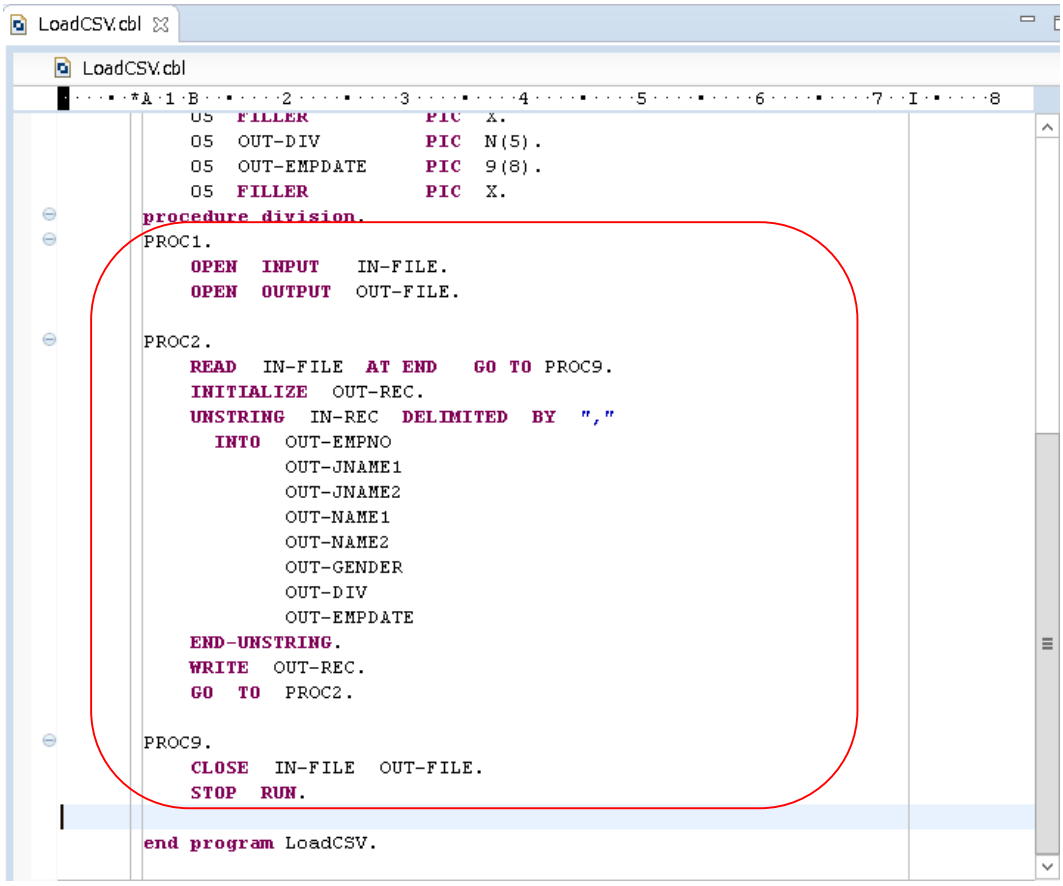
次に、データ部の作業場所節(working-storage section) を削除し、以下のファイル節(file section) を追加します。 なお、データ部のファイル定義を入力したので、環境部のエラーは無くなります。

```
FILE SECTION.
FD  IN-FILE.
01  IN-REC              PIC  X(50).
FD  OUT-FILE.
01  OUT-REC.
    05  OUT-EMPNO       PIC  9(8).
    05  FILLER          PIC  X.
    05  OUT-JNAME1      PIC  N(5).
    05  OUT-JNAME2      PIC  N(5).
    05  OUT-NAME1       PIC  X(5).
    05  OUT-NAME2       PIC  X(5).
    05  OUT-GENDER      PIC  X.
    05  FILLER          PIC  X.
    05  OUT-DIV         PIC  N(5).
    05  OUT-EMPDATE     PIC  9(8).
    05  FILLER          PIC  X.
```



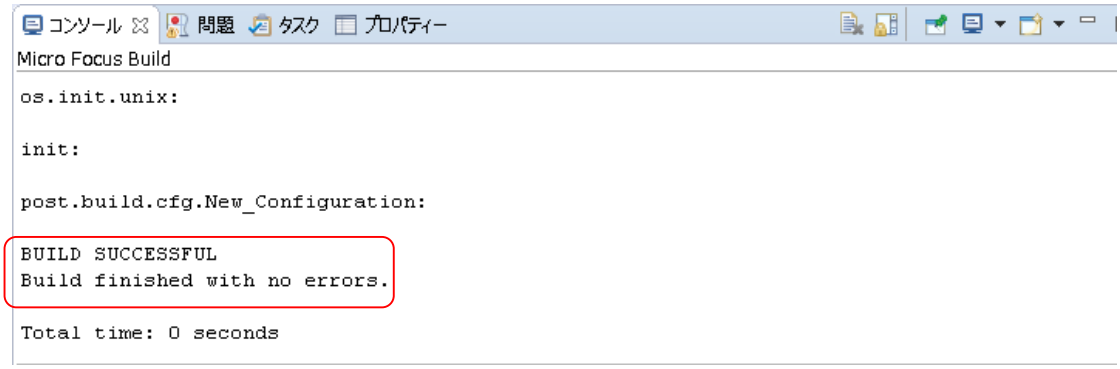
最後に、手続き部の goback 文を削除し、以下の手続き文を追加します。

```
PROC1.
    OPEN  INPUT   IN-FILE.
    OPEN  OUTPUT  OUT-FILE.

PROC2.
    READ  IN-FILE  AT END   GO TO PROC9.
    INITIALIZE  OUT-REC.
    UNSTRING  IN-REC  DELIMITED  BY  ","
      INTO  OUT-EMPNO
            OUT-JNAME1
            OUT-JNAME2
            OUT-NAME1
            OUT-NAME2
            OUT-GENDER
            OUT-DIV
            OUT-EMPDATE
    END-UNSTRING.
    WRITE  OUT-REC.
    GO  TO  PROC2.

PROC9.
    CLOSE  IN-FILE  OUT-FILE.
    STOP  RUN.
```



5　COBOL アプリケーションをビルドします。

終止符(ピリオド)を含めてスペルミスがなければ、エディタービュー上の LoadCSV.cbl をアクティブにしたまた**ファイル(F)** メニューの**保管(S)** 或いは Ctrl + S キーで保存します。コンソールビューに正常にビルドできた旨のメッセージが出力されていることを確認します。



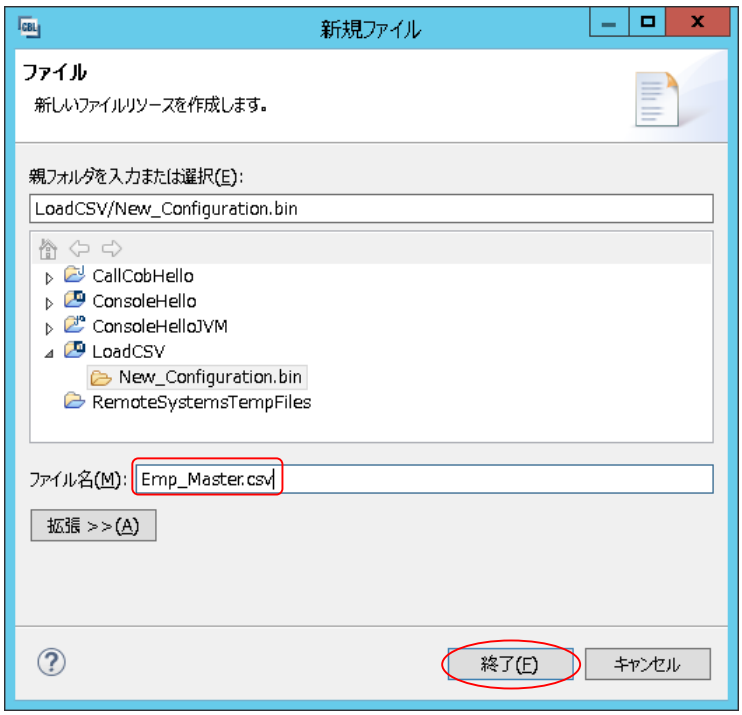
6　CSV ファイルを作成します。

LoadCSV プロジェクト配下の **New_Configuration.bin** フォルダを右クリックし

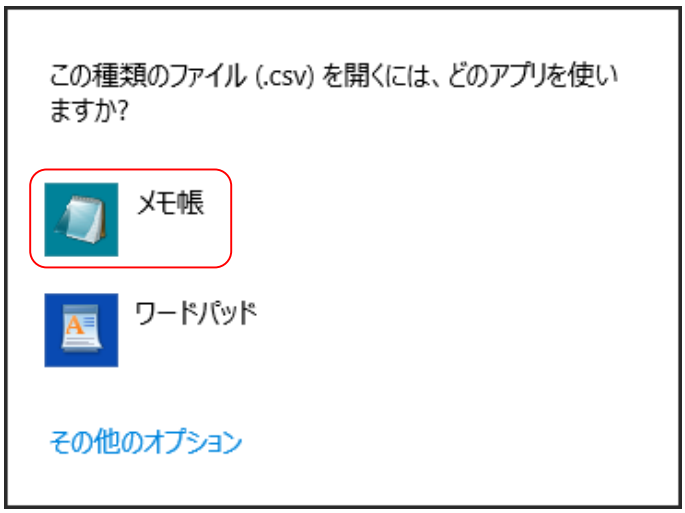
新規(N) → ファイル

を選択します。

ファイル名欄に **Emp_Master.csv** を指定し［**終了(F)**］ボタンを押下します。



「この種類のファイル(.csv)を開くには、どのアプリを使いますか？」の問いがポップアップされた場合は「メモ帳」を選択します。



以下のデータを入力し、ファイルを保存の上、閉じます。

11111113,佐藤,隆,ｻﾄｳ,ﾀｶｼ,M,営業部,19980401,0
22222226,鈴木,尚之,ｽｽﾞｷ,ﾅｵﾕｷ,M,技術部,19981015,0
33333339,田中,直美,ﾀﾅｶ,ﾅｵﾐ,F,総務部,19990401,0
44444442,山田,洋一,ﾔﾏﾀﾞ,ﾖｳｲﾁ,M,営業部,20000701,0
55555555,伊藤,弘子,ｲﾄｳ,ﾋﾛｺ,F,技術部,20010401,0
66666668,木村,貴弘,ｷﾑﾗ,ﾀｶﾋﾛ,M,営業部,20021220,0
77777771,中村,慎司,ﾅｶﾑﾗ,ｼﾝｼﾞ,M,技術部,20030401,0
88888884,橋本,悦子,ﾊｼﾓﾄ,ｴﾂｺ,F,総務部,20040805,0
99999997,三井,薫,ﾐﾂｲ,ｶｵﾙ,F,営業部,20050401,0

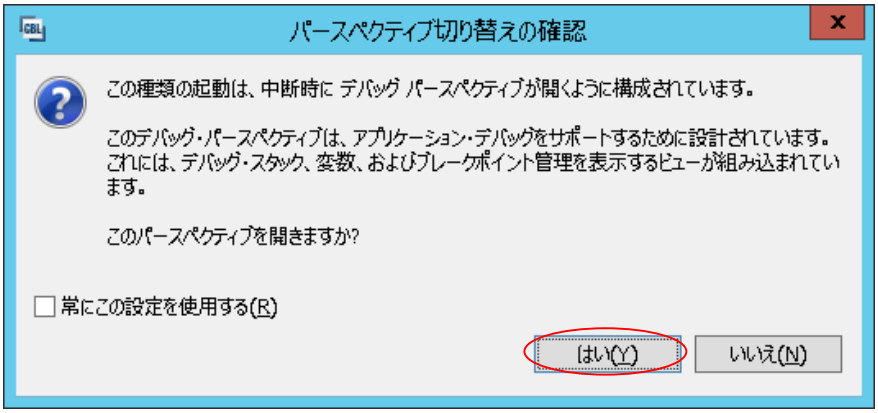
7　COBOL アプリケーションをデバッグ実行します。

New_Configuration.bin フォルダ配下の **LoadCSV.exe** を右クリックし

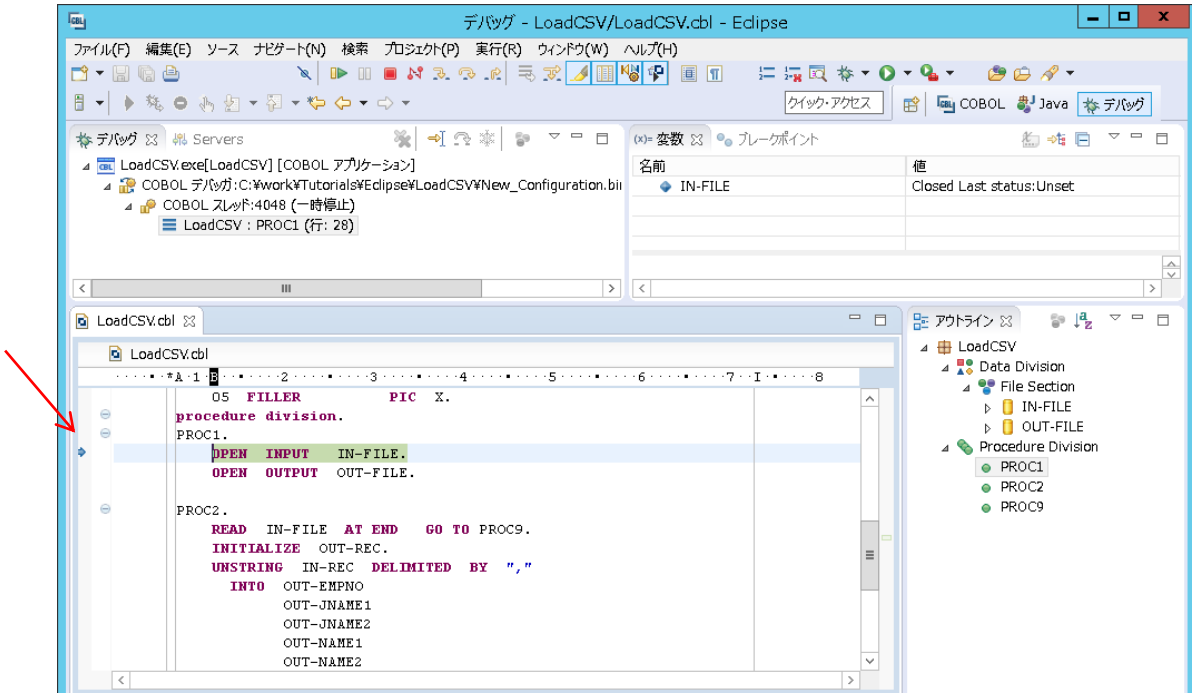
デバッグ(D) → COBOL アプリケーション

を選択します。

パースペクティブ切り替えの確認メッセージには［**はい(Y)**］を選択します。



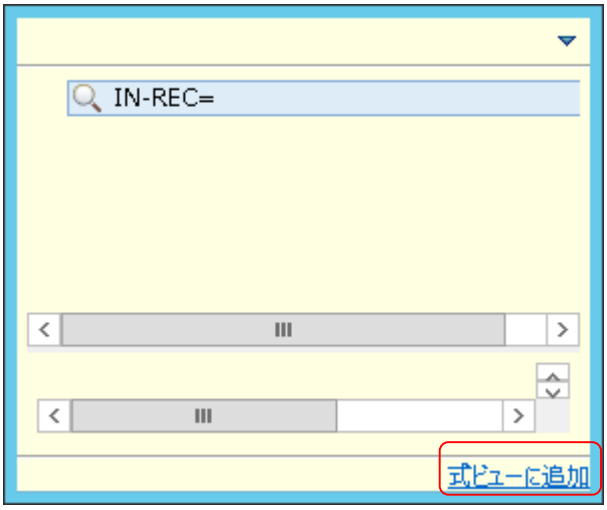
デバッガーは手続き部の最初の COBOL 文である open 文の処理前に一時停止している状態となっています。



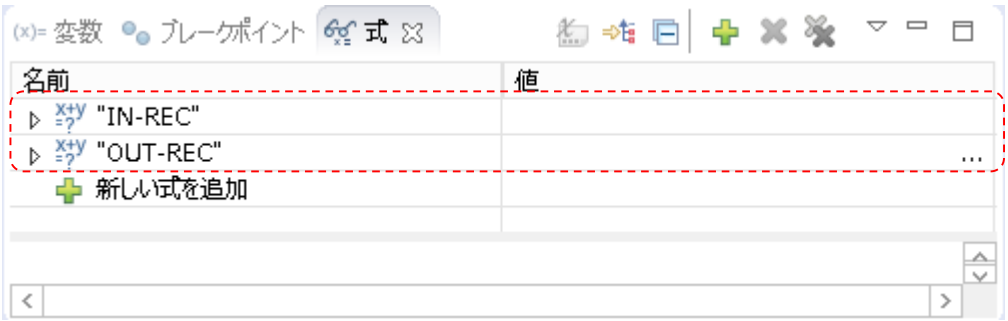
入力ファイルから読み込んだレコードの内容を確認するため、**IN-REC** に格納される値の変遷を追います。UNSTRING 文の **IN-REC** を選択します。

この状態で右クリックし、「**検査**」を選択します。

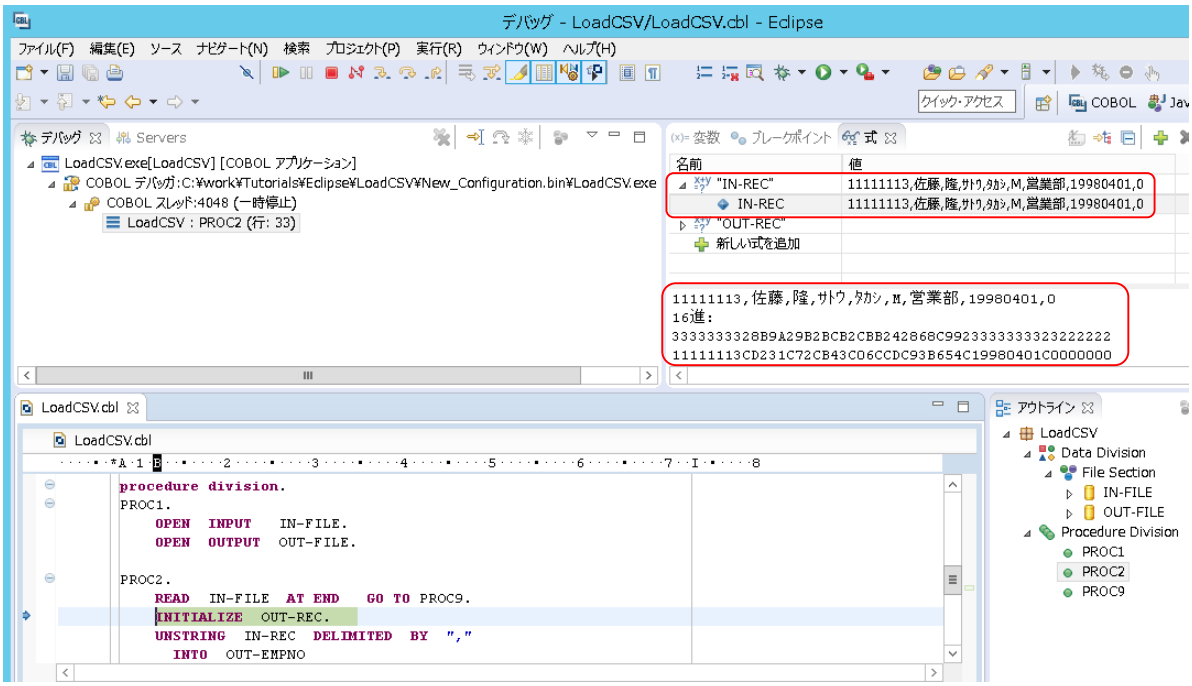
「**式ビューに追加**」をクリックします。



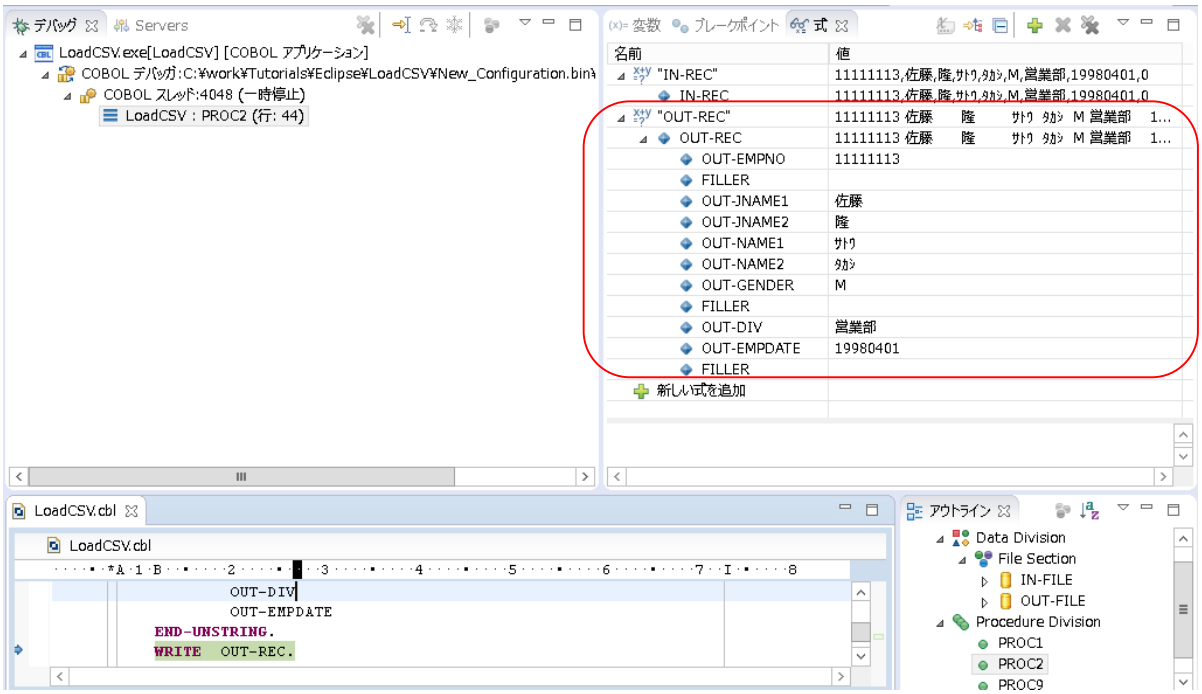
同様に出力ファイルに書き出すレコードの内容もトレースするため、INITIALIZE 文の **OUT-REC** を同様に式ビューに追加します。



F5 キー(ステップイン)を 3 回押すと、デバッガーは READ 文実行後、処理を中断します。
式ビューの IN-REC の値には CSV ファイルから読み込んだ最初のレコードが表示されます。

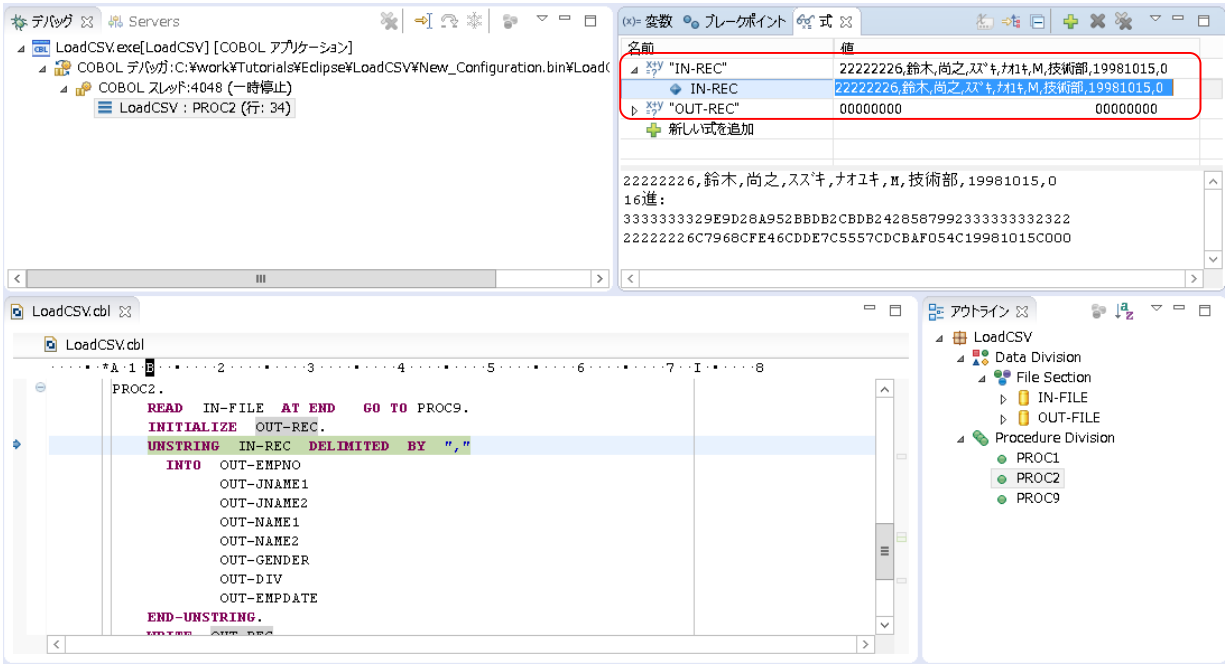


さらに **F5** キーを 2 回押すと、デバッガーは UNSTRING 文を実行後、処理を中断します。
式ビューの OUT-REC の値には出力ファイルへ書き出す最初のレコードが表示されます。

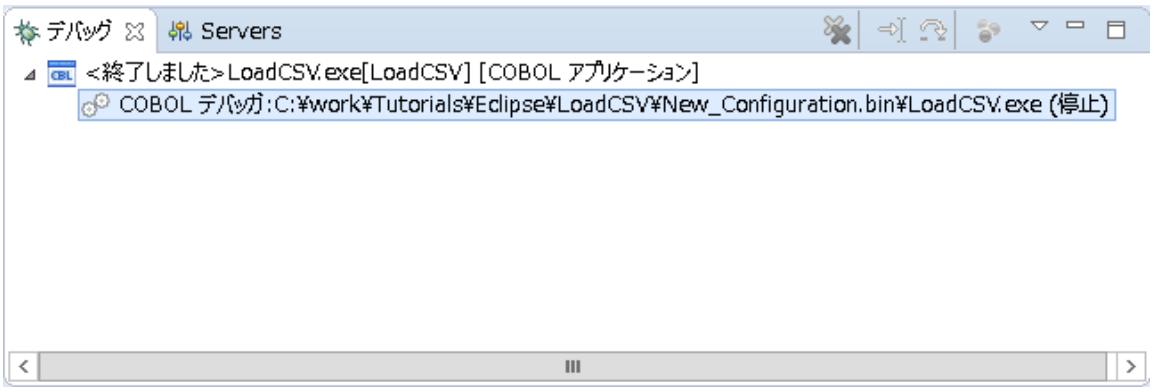


さらに **F5** キーを 4 回押すと、デバッガーは INITIALIZE 文を実行後、処理を中断します。

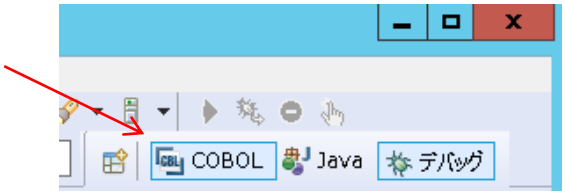
ウォッチ式の IN-REC の値には CSV ファイルから読み込んだ 2 番目のレコードが表示され、OUT-REC の値は INITIALIZE 文で初期化されています。



F8(再開)キーを打鍵するか CSV ファイルからすべてのレコードを読み込むまで **F5** キーを押すと、デバッガは終了します。

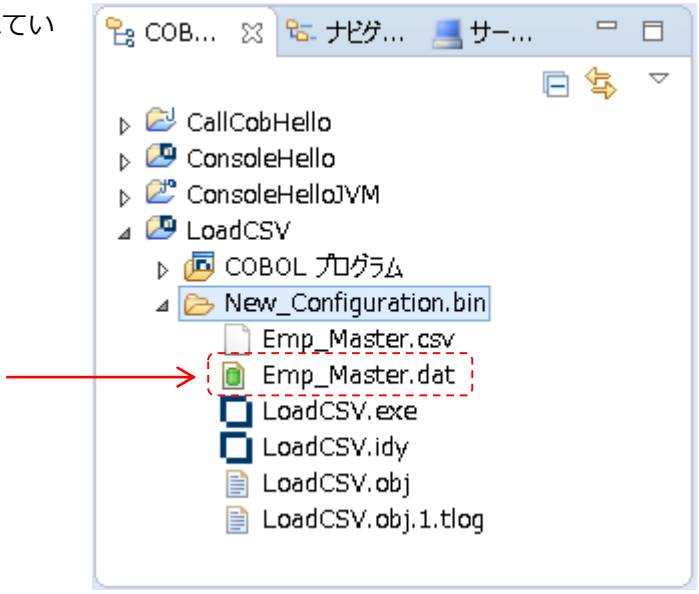


ワークスペース右上コーナにあるパースペクティブ切り替えボタンエリアにて「**COBOL**」を選択し、COBOL パースペクティブに戻ります。

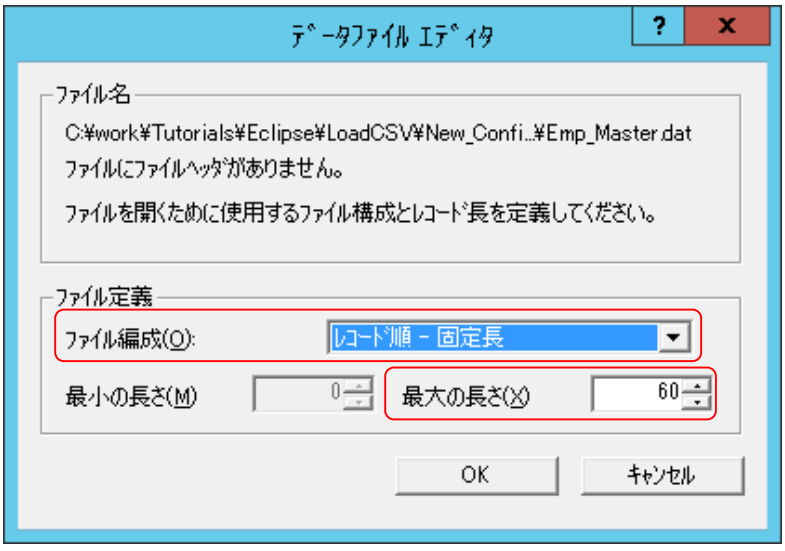


COBOL エクスプローラビューにて **LoadCSV** プロジェクト配下の **New_Configuration.bin** フォルダを右クリックし「**更新(F)**」を選択します。

Emp_Master.dat ファイルが作成されていることを確認します。



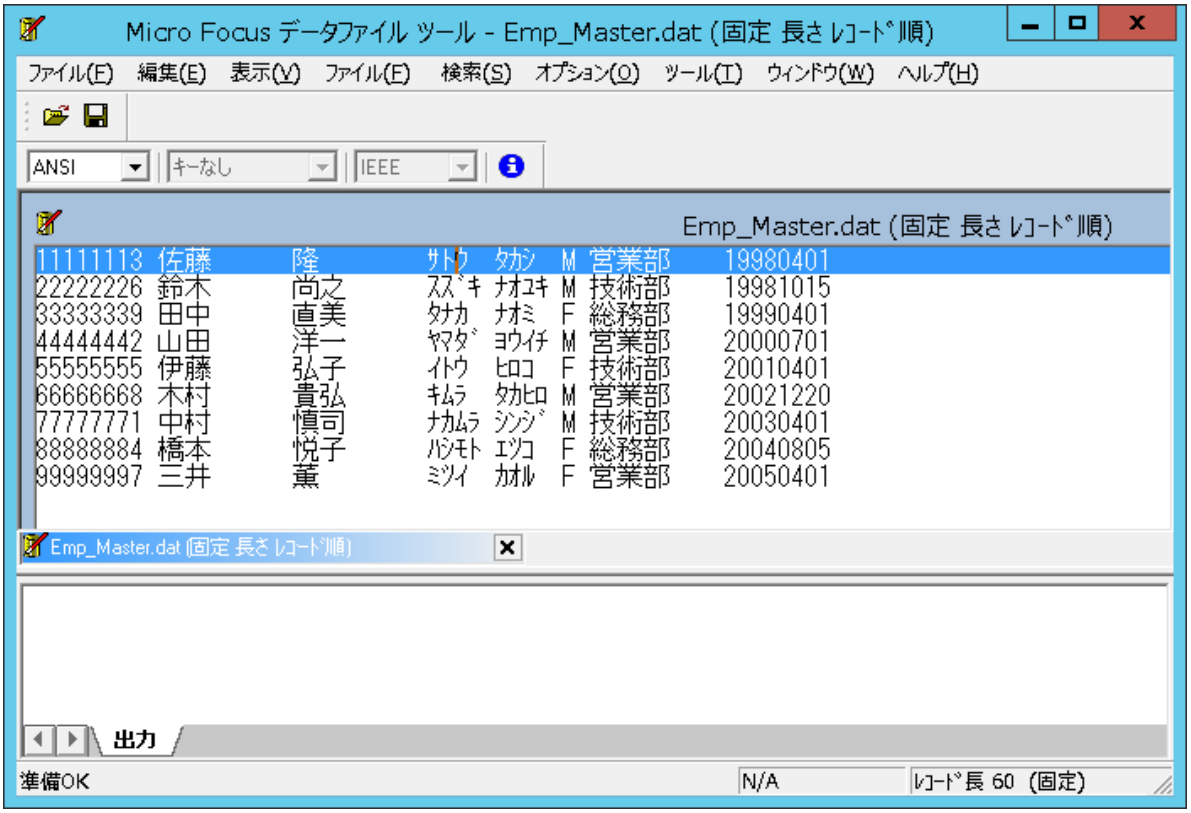
Emp_Master.dat をダブルクリックします。Visual COBOL に付属する **Data File Editor ユーティリティ**が起動します。[**ファイル編成**] 欄はデフォルトの「**レコード順 – 固定長**」のままにしておきます。[**最大の長さ**] 欄には「**60**」を指定します。



[**OK**] ボタンを押下します。

プロファイルファイルを作成するかどうかの問いには [**はい(Y)**] を選択します。

社員９名分のデータが正常に固定長順編成ファイルに書き込まれていることを確認します。



第6章　Visual COBOL のバッチアプリケーション

本章では、第 5 章で作成した固定長順編成ファイルを読み込んでレポートファイルを作成するバッチアプリケーションを Visual COBOL for COBOL で作成します。

1　Visual COBOL for Eclipse を起動します。

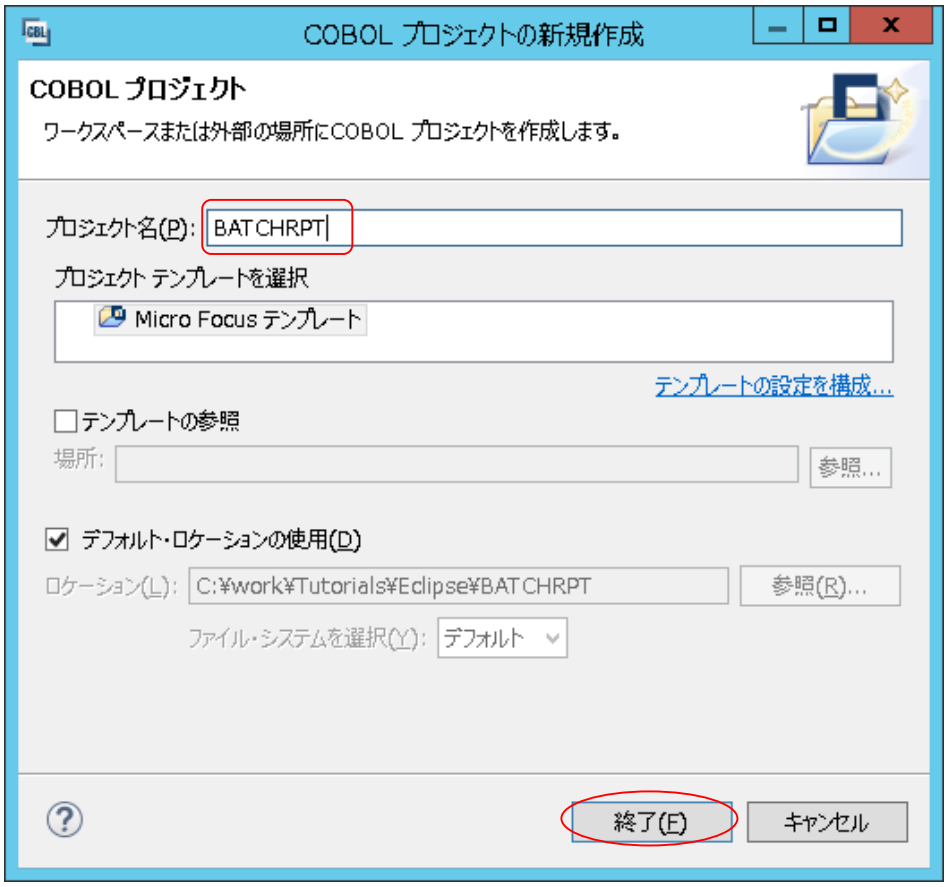
Windows のスタートメニューから、**Visual COBOL for Eclipse** をクリックします。

ワークスペースは前章までで使用されたものをそのまま使用します。

2　COBOL プロジェクトを作成します。

ファイル(F)メニューから **新規(N) → COBOL プロジェクト** を選択します。

プロジェクト名欄に **BATCHRPT** と入力し[**終了(F)**] ボタンを押下します。



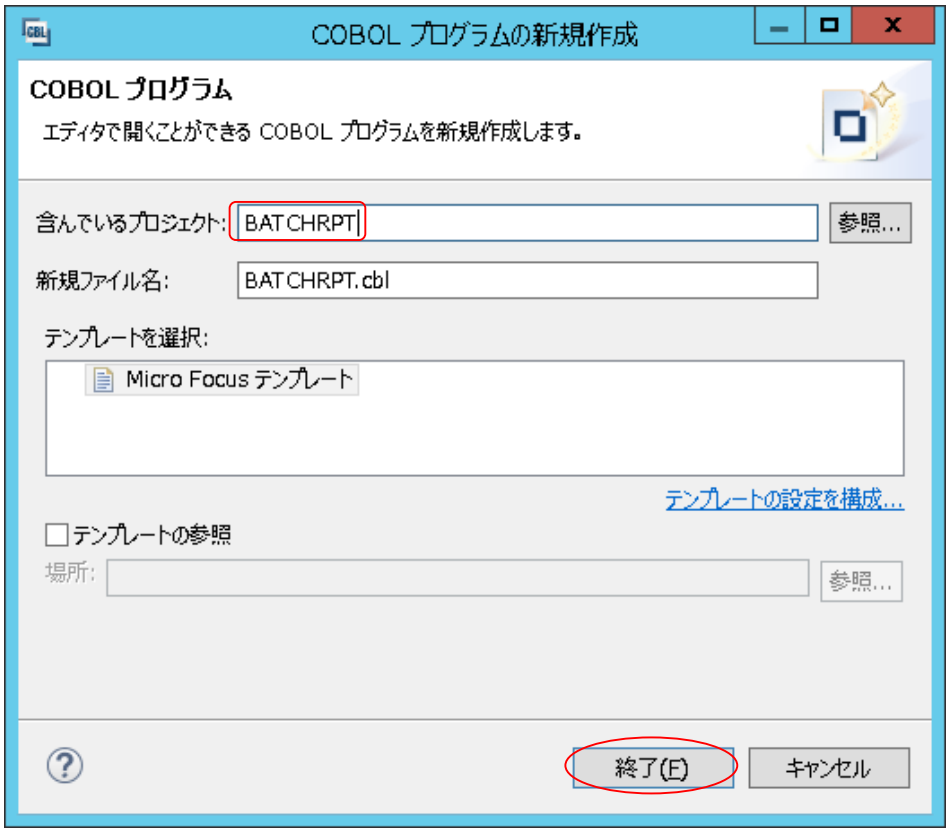
3　COBOL プログラムを追加します。

COBOL エクスプローラビューにて **プロジェクト** フォルダを右クリックし

新規(N) → COBOL プログラム

を選択します。

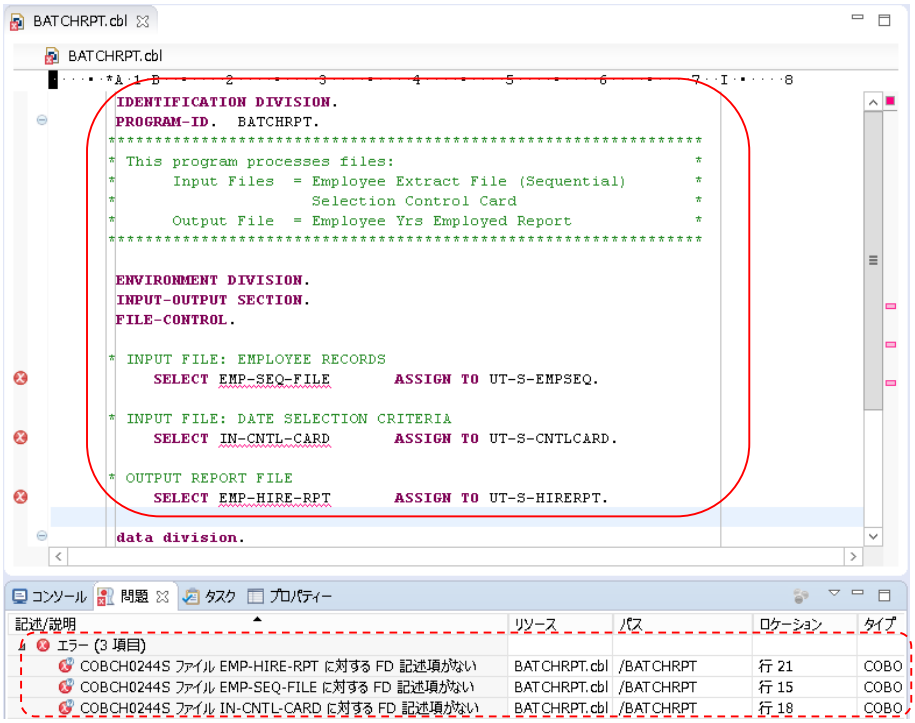
新規ファイル名欄には **BATCHRPT.cbl** を指定します。[**終了(F)**] ボタンを押下します。



4　コードエディターで COBOL ソースコードを入力します。

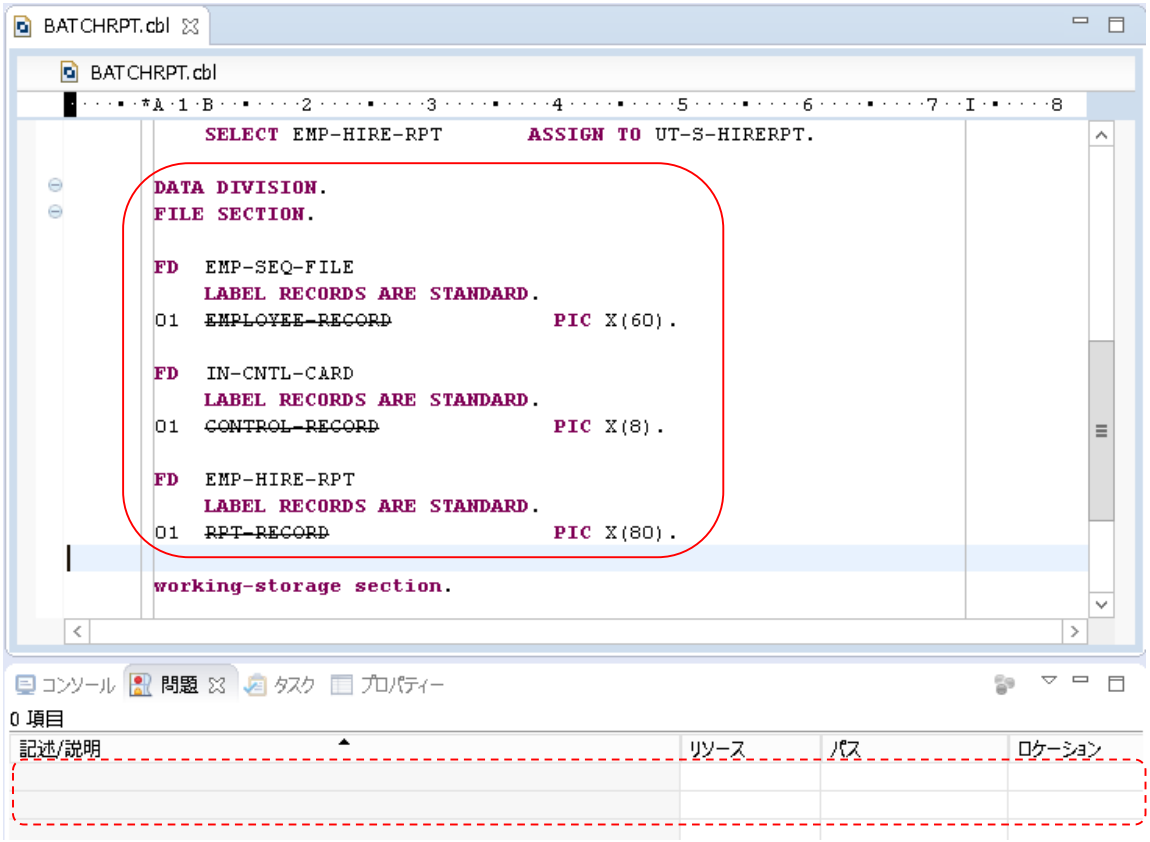
本章では既存資産の流用を想定して COBOL 正書法に従った伝統的スタイルのソースコードを入力しますので、アスタリスクで始まるコメント行が 7 列目(エディタービュー左側のグレー領域の右端)から始まるよう注意して、以下の見出し部と環境部を入力します。 まだ、データ部のファイル定義が未入力なので 3 件のエラーとなりますが、ここでは無視して構いません。

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  BATCHRPT.
*****************************************************************
* This program processes files:                                 *
*      Input Files  = Employee Extract File (Sequential)        *
*                     Selection Control Card                    *
*      Output File  = Employee Yrs Employed Report              *
*****************************************************************

 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

* INPUT FILE: EMPLOYEE RECORDS
     SELECT EMP-SEQ-FILE       ASSIGN TO UT-S-EMPSEQ.

* INPUT FILE: DATE SELECTION CRITERIA
     SELECT IN-CNTL-CARD       ASSIGN TO UT-S-CNTLCARD.

* OUTPUT REPORT FILE
     SELECT EMP-HIRE-RPT       ASSIGN TO UT-S-HIRERPT.
```



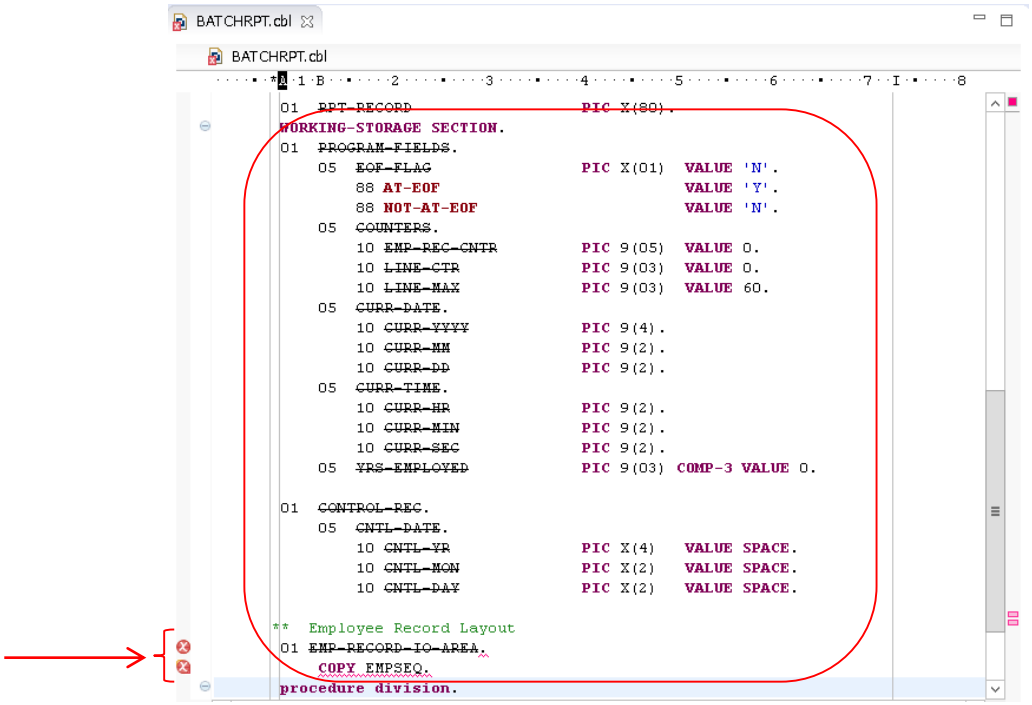
データ部のファイル節を入力します。 なお、データ部のファイル定義を入力したので、環境部のエラーは無くなります。

```
DATA DIVISION.
FILE SECTION.

FD  EMP-SEQ-FILE
    LABEL RECORDS ARE STANDARD.
01  EMPLOYEE-RECORD              PIC X(60).

FD  IN-CNTL-CARD
    LABEL RECORDS ARE STANDARD.
01  CONTROL-RECORD               PIC X(8).

FD  EMP-HIRE-RPT
    LABEL RECORDS ARE STANDARD.
01  RPT-RECORD                   PIC X(80).
```



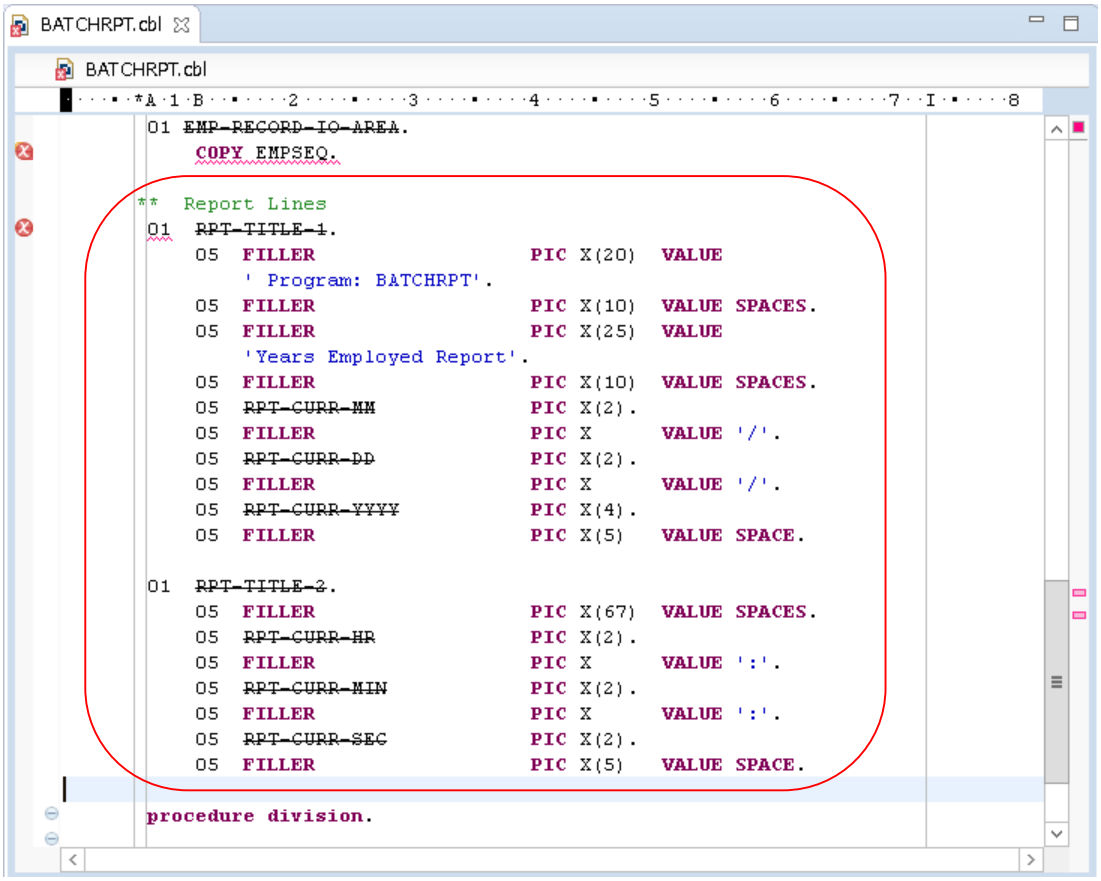
データ部の作業場所節で PROGRAM-FIELDS、CONTROL-REC データ項目を入力します。 COPY 文で外部参照する EMP-RECORD-IO-AREA データ項目はエラーとなりますが、無視して構いません。

```
WORKING-STORAGE SECTION.
01  PROGRAM-FIELDS.
    05  EOF-FLAG                PIC X(01)   VALUE 'N'.
        88 AT-EOF                            VALUE 'Y'.
        88 NOT-AT-EOF                        VALUE 'N'.
    05  COUNTERS.
        10 EMP-REC-CNTR          PIC 9(05)   VALUE 0.
        10 LINE-CTR              PIC 9(03)   VALUE 0.
        10 LINE-MAX              PIC 9(03)   VALUE 60.
    05  CURR-DATE.
        10 CURR-YYYY             PIC 9(4).
        10 CURR-MM               PIC 9(2).
        10 CURR-DD               PIC 9(2).
    05  CURR-TIME.
        10 CURR-HR               PIC 9(2).
        10 CURR-MIN              PIC 9(2).
        10 CURR-SEC              PIC 9(2).
    05  YRS-EMPLOYED            PIC 9(03) COMP-3 VALUE 0.

01  CONTROL-REC.
    05  CNTL-DATE.
        10 CNTL-YR               PIC X(4)    VALUE SPACE.
        10 CNTL-MON              PIC X(2)    VALUE SPACE.
        10 CNTL-DAY              PIC X(2)    VALUE SPACE.

**  Employee Record Layout
 01 EMP-RECORD-IO-AREA.
    COPY EMPSEQ.
```



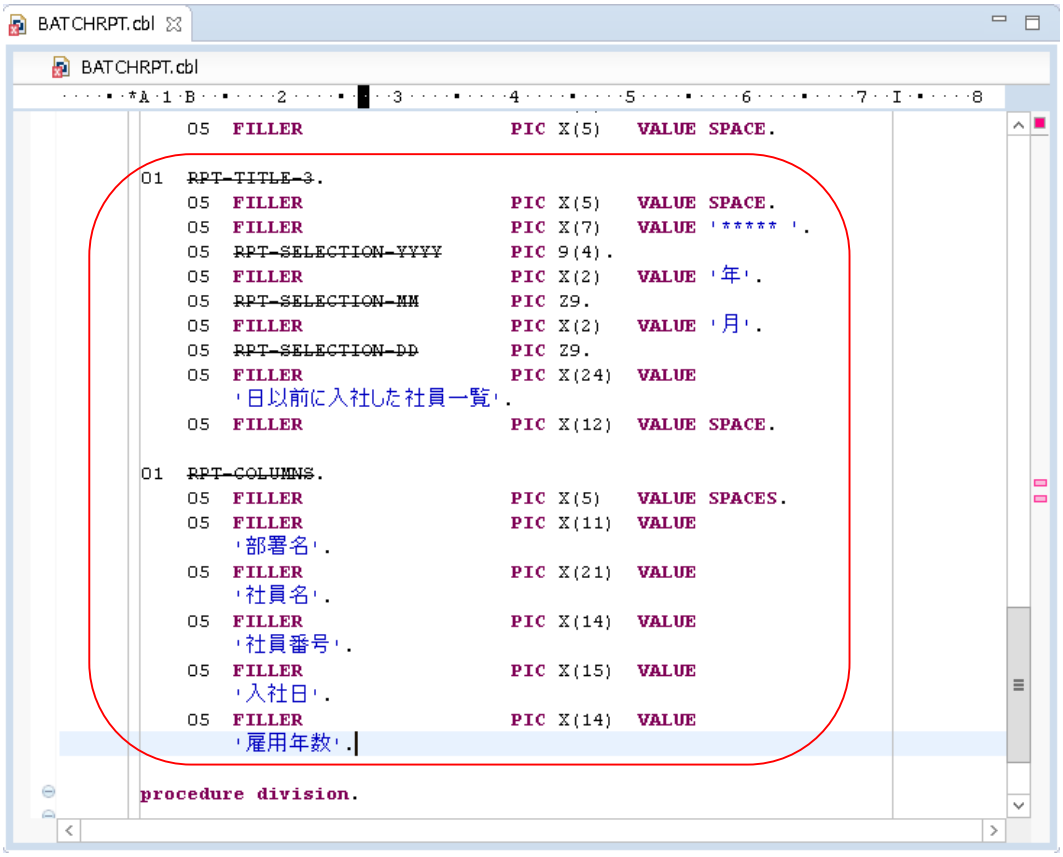
データ部の作業場所節で RPT-TITLE-1 と RPT-TITLE-2 データ項目を入力します。

```
**  Report Lines
 01  RPT-TITLE-1.
     05  FILLER                 PIC X(20)  VALUE
         ' Program: BATCHRPT'.
     05  FILLER                 PIC X(10)  VALUE SPACES.
     05  FILLER                 PIC X(25)  VALUE
         'Years Employed Report'.
     05  FILLER                 PIC X(10)  VALUE SPACES.
     05  RPT-CURR-MM            PIC X(2).
     05  FILLER                 PIC X      VALUE '/'.
     05  RPT-CURR-DD            PIC X(2).
     05  FILLER                 PIC X      VALUE '/'.
     05  RPT-CURR-YYYY          PIC X(4).
     05  FILLER                 PIC X(5)   VALUE SPACE.

 01  RPT-TITLE-2.
     05  FILLER                 PIC X(67)  VALUE SPACES.
     05  RPT-CURR-HR            PIC X(2).
     05  FILLER                 PIC X      VALUE ':'.
     05  RPT-CURR-MIN           PIC X(2).
     05  FILLER                 PIC X      VALUE ':'.
     05  RPT-CURR-SEC           PIC X(2).
     05  FILLER                 PIC X(5)   VALUE SPACE.
```



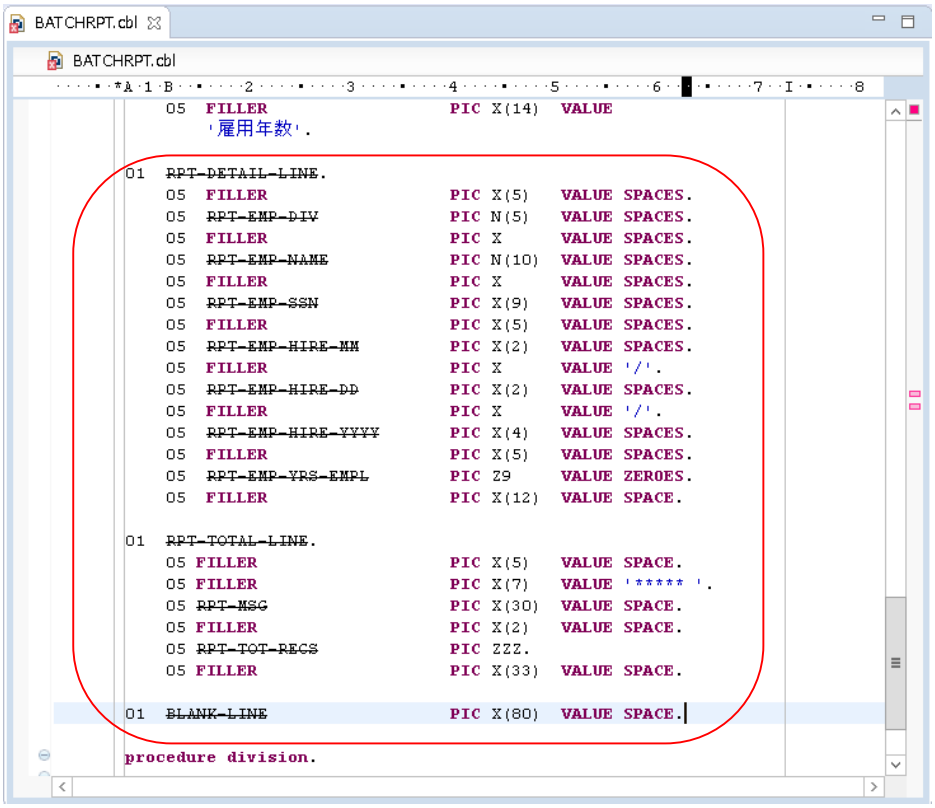
作業場所節で RPT-TITLE-3 と RPT-COLUMNS データ項目を入力します。

```
01  RPT-TITLE-3.
    05  FILLER                 PIC X(5)   VALUE SPACE.
    05  FILLER                 PIC X(7)   VALUE '***** '.
    05  RPT-SELECTION-YYYY     PIC 9(4).
    05  FILLER                 PIC X(2)   VALUE '年'.
    05  RPT-SELECTION-MM       PIC Z9.
    05  FILLER                 PIC X(2)   VALUE '月'.
    05  RPT-SELECTION-DD       PIC Z9.
    05  FILLER                 PIC X(24)  VALUE
        '日以前に入社した社員一覧'.
    05  FILLER                 PIC X(12)  VALUE SPACE.

01  RPT-COLUMNS.
    05  FILLER                 PIC X(5)   VALUE SPACES.
    05  FILLER                 PIC X(11)  VALUE
        '部署名'.
    05  FILLER                 PIC X(21)  VALUE
        '社員名'.
    05  FILLER                 PIC X(14)  VALUE
        '社員番号'.
    05  FILLER                 PIC X(15)  VALUE
        '入社日'.
    05  FILLER                 PIC X(14)  VALUE
        '雇用年数'.
```



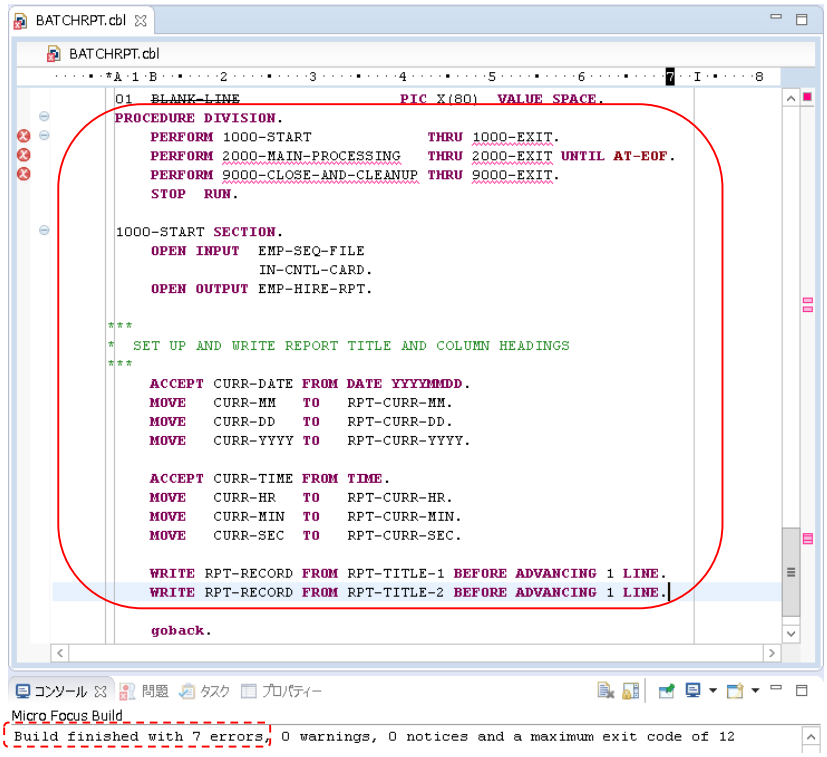
作業場所節で RPT-DETAIL-LINE、RPT-TOTAL-LINE と BLANK-LINE データ項目を入力します。

```
01  RPT-DETAIL-LINE.
    05  FILLER                  PIC X(5)    VALUE SPACES.
    05  RPT-EMP-DIV             PIC N(5)    VALUE SPACES.
    05  FILLER                  PIC X       VALUE SPACES.
    05  RPT-EMP-NAME            PIC N(10)   VALUE SPACES.
    05  FILLER                  PIC X       VALUE SPACES.
    05  RPT-EMP-SSN             PIC X(9)    VALUE SPACES.
    05  FILLER                  PIC X(5)    VALUE SPACES.
    05  RPT-EMP-HIRE-MM         PIC X(2)    VALUE SPACES.
    05  FILLER                  PIC X       VALUE '/'.
    05  RPT-EMP-HIRE-DD         PIC X(2)    VALUE SPACES.
    05  FILLER                  PIC X       VALUE '/'.
    05  RPT-EMP-HIRE-YYYY       PIC X(4)    VALUE SPACES.
    05  FILLER                  PIC X(5)    VALUE SPACES.
    05  RPT-EMP-YRS-EMPL        PIC Z9      VALUE ZEROES.
    05  FILLER                  PIC X(12)   VALUE SPACE.

01  RPT-TOTAL-LINE.
    05 FILLER                   PIC X(5)    VALUE SPACE.
    05 FILLER                   PIC X(7)    VALUE '***** '.
    05 RPT-MSG                  PIC X(30)   VALUE SPACE.
    05 FILLER                   PIC X(2)    VALUE SPACE.
    05 RPT-TOT-RECS             PIC ZZZ.
    05 FILLER                   PIC X(33)   VALUE SPACE.

01  BLANK-LINE                  PIC X(80)   VALUE SPACE.
```



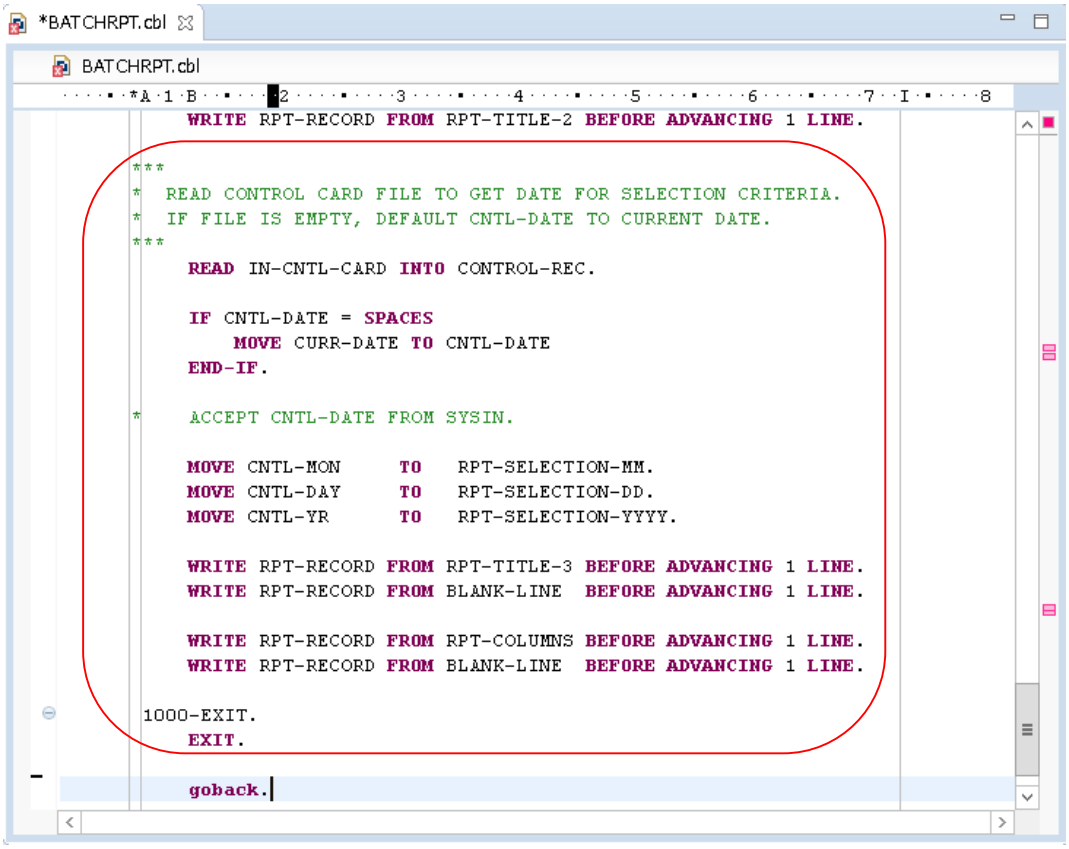
最後に、手続き部の 1000-START 節の前半部分を入力します。PERFORM 文で参照する手続き名が未定義なのでエラーが 5 件増えますが、気にせず先に進んでください。

```
PROCEDURE DIVISION.
    PERFORM 1000-START              THRU 1000-EXIT.
    PERFORM 2000-MAIN-PROCESSING    THRU 2000-EXIT UNTIL AT-EOF.
    PERFORM 9000-CLOSE-AND-CLEANUP THRU 9000-EXIT.
    STOP  RUN.

1000-START SECTION.
    OPEN INPUT  EMP-SEQ-FILE
                IN-CNTL-CARD.
    OPEN OUTPUT EMP-HIRE-RPT.

***
*  SET UP AND WRITE REPORT TITLE AND COLUMN HEADINGS
***
    ACCEPT CURR-DATE FROM DATE YYYYMMDD.
    MOVE   CURR-MM   TO   RPT-CURR-MM.
    MOVE   CURR-DD   TO   RPT-CURR-DD.
    MOVE   CURR-YYYY TO   RPT-CURR-YYYY.

    ACCEPT CURR-TIME FROM TIME.
    MOVE   CURR-HR   TO   RPT-CURR-HR.
    MOVE   CURR-MIN  TO   RPT-CURR-MIN.
    MOVE   CURR-SEC  TO   RPT-CURR-SEC.

    WRITE RPT-RECORD FROM RPT-TITLE-1 BEFORE ADVANCING 1 LINE.
    WRITE RPT-RECORD FROM RPT-TITLE-2 BEFORE ADVANCING 1 LINE.
```



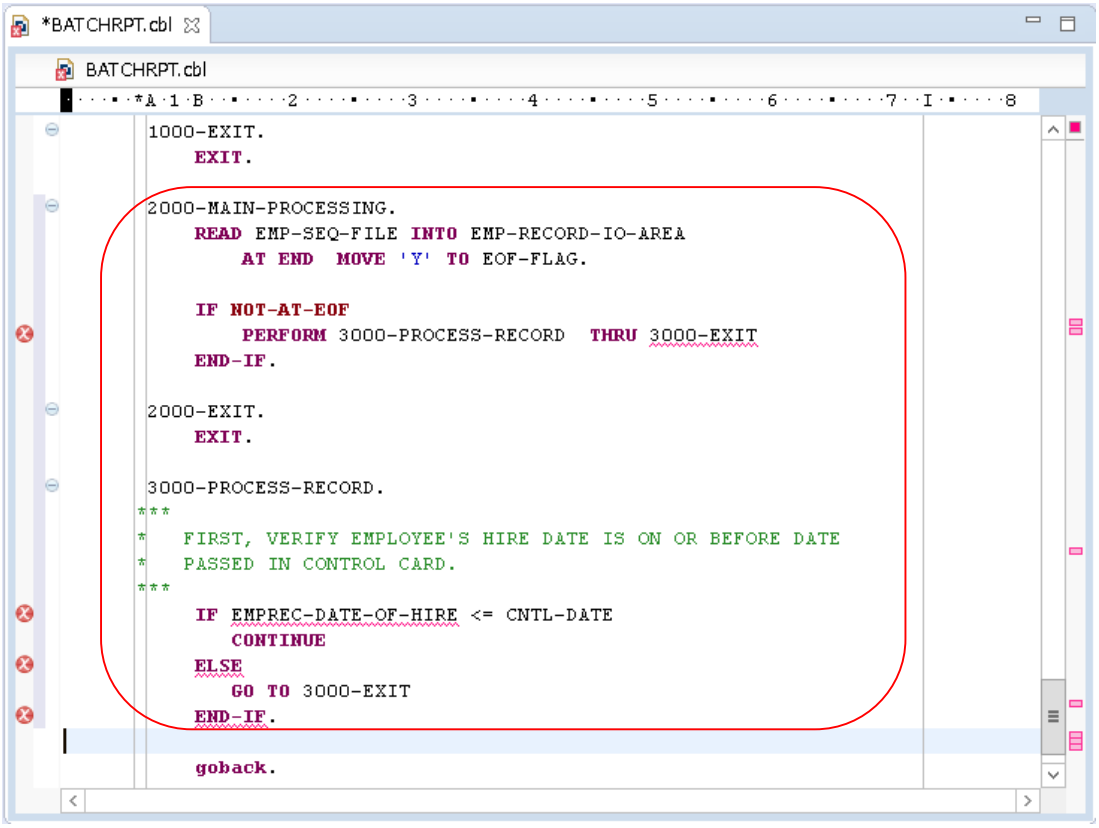
手続き部の 1000-START 節の後半部分を入力します。

```
***
*  READ CONTROL CARD FILE TO GET DATE FOR SELECTION CRITERIA.
*  IF FILE IS EMPTY, DEFAULT CNTL-DATE TO CURRENT DATE.
***
     READ IN-CNTL-CARD INTO CONTROL-REC.

     IF CNTL-DATE = SPACES
         MOVE CURR-DATE TO CNTL-DATE
     END-IF.

*    ACCEPT CNTL-DATE FROM SYSIN.

     MOVE CNTL-MON      TO   RPT-SELECTION-MM.
     MOVE CNTL-DAY      TO   RPT-SELECTION-DD.
     MOVE CNTL-YR       TO   RPT-SELECTION-YYYY.

     WRITE RPT-RECORD FROM RPT-TITLE-3 BEFORE ADVANCING 1 LINE.
     WRITE RPT-RECORD FROM BLANK-LINE  BEFORE ADVANCING 1 LINE.

     WRITE RPT-RECORD FROM RPT-COLUMNS BEFORE ADVANCING 1 LINE.
     WRITE RPT-RECORD FROM BLANK-LINE  BEFORE ADVANCING 1 LINE.

 1000-EXIT.
     EXIT.
```



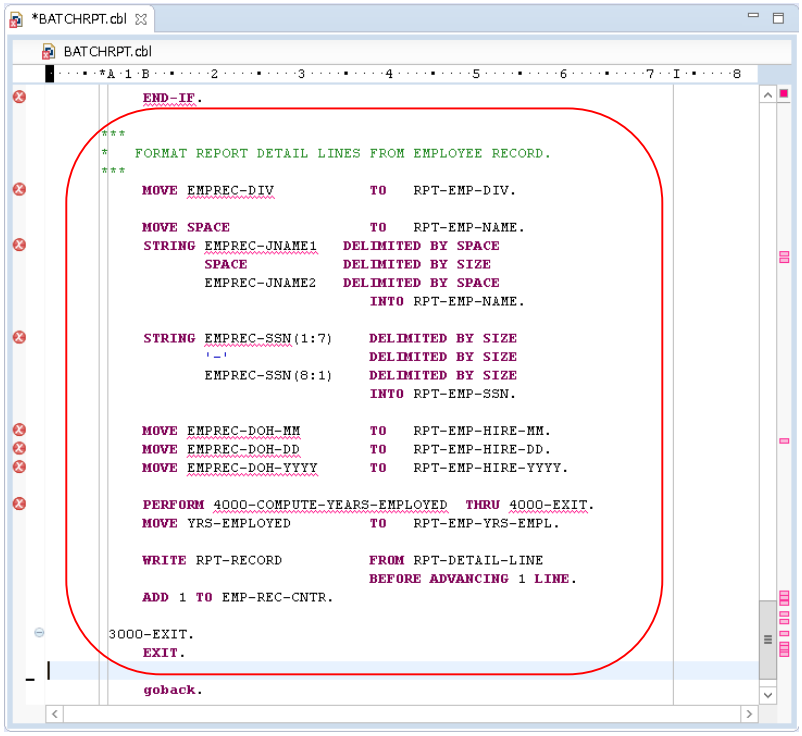
手続き部の 2000-MAIN-PROCESSING 段落と 3000-PROCESS-RECORD 段落の前半部分を入力します。

```
 2000-MAIN-PROCESSING.
     READ EMP-SEQ-FILE INTO EMP-RECORD-IO-AREA
         AT END  MOVE 'Y' TO EOF-FLAG.

     IF NOT-AT-EOF
         PERFORM 3000-PROCESS-RECORD  THRU 3000-EXIT
     END-IF.

 2000-EXIT.
     EXIT.

 3000-PROCESS-RECORD.
***
*   FIRST, VERIFY EMPLOYEE'S HIRE DATE IS ON OR BEFORE DATE
*   PASSED IN CONTROL CARD.
***
     IF EMPREC-DATE-OF-HIRE <= CNTL-DATE
        CONTINUE
     ELSE
        GO TO 3000-EXIT
     END-IF.
```



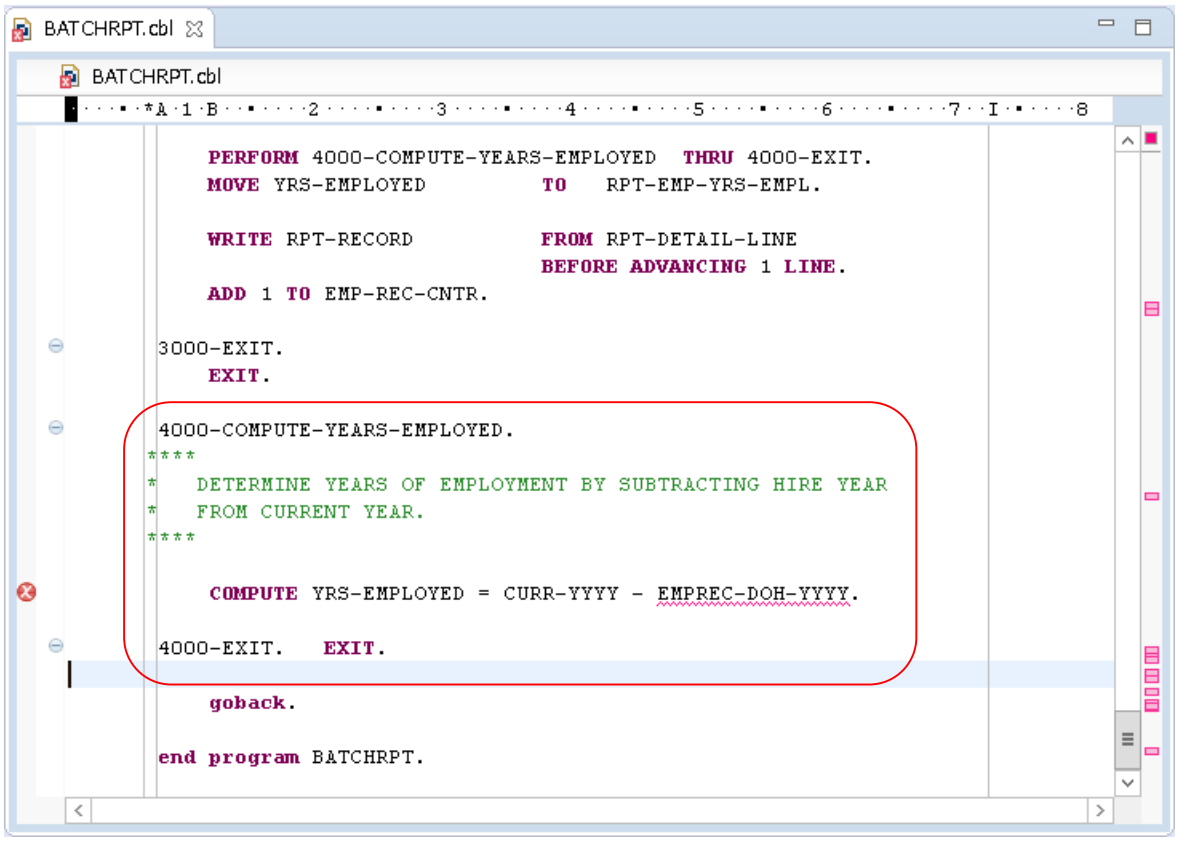
手続き部の 3000-PROCESS-RECORD 段落の後半部分を入力します。

```
***
*   FORMAT REPORT DETAIL LINES FROM EMPLOYEE RECORD.
***
     MOVE EMPREC-DIV           TO   RPT-EMP-DIV.

     MOVE SPACE                TO   RPT-EMP-NAME.
     STRING EMPREC-JNAME1   DELIMITED BY SPACE
            SPACE           DELIMITED BY SIZE
            EMPREC-JNAME2   DELIMITED BY SPACE
                            INTO RPT-EMP-NAME.

     STRING EMPREC-SSN(1:7)    DELIMITED BY SIZE
            '-'                DELIMITED BY SIZE
            EMPREC-SSN(8:1)    DELIMITED BY SIZE
                               INTO RPT-EMP-SSN.

     MOVE EMPREC-DOH-MM        TO   RPT-EMP-HIRE-MM.
     MOVE EMPREC-DOH-DD        TO   RPT-EMP-HIRE-DD.
     MOVE EMPREC-DOH-YYYY      TO   RPT-EMP-HIRE-YYYY.

     PERFORM 4000-COMPUTE-YEARS-EMPLOYED  THRU 4000-EXIT.
     MOVE YRS-EMPLOYED         TO   RPT-EMP-YRS-EMPL.

     WRITE RPT-RECORD          FROM RPT-DETAIL-LINE
                               BEFORE ADVANCING 1 LINE.
     ADD 1 TO EMP-REC-CNTR.

 3000-EXIT.
     EXIT.
```



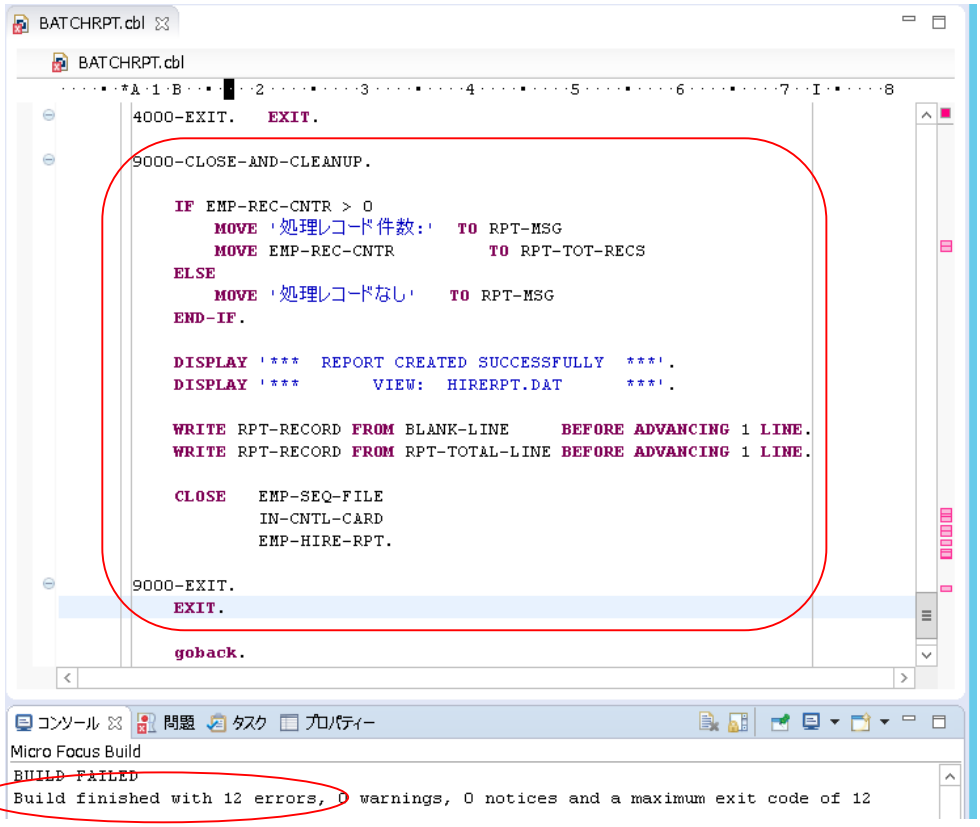
手続き部の 4000-COMPUTE-YEARS-EMPLOYED 段落を入力します。

```
 4000-COMPUTE-YEARS-EMPLOYED.
****
*   DETERMINE YEARS OF EMPLOYMENT BY SUBTRACTING HIRE YEAR
*   FROM CURRENT YEAR.
****

     COMPUTE YRS-EMPLOYED = CURR-YYYY - EMPREC-DOH-YYYY.

 4000-EXIT.    EXIT.
```



手続き部の 9000-CLOSE-AND-CLEANUP 段落を入力します。

```
9000-CLOSE-AND-CLEANUP.

    IF EMP-REC-CNTR > 0
        MOVE '処理レコード件数:'  TO RPT-MSG
        MOVE EMP-REC-CNTR         TO RPT-TOT-RECS
    ELSE
        MOVE '処理レコードなし'   TO RPT-MSG
    END-IF.

    DISPLAY '***  REPORT CREATED SUCCESSFULLY  ***'.
    DISPLAY '***       VIEW:  HIRERPT.DAT      ***'.

    WRITE RPT-RECORD FROM BLANK-LINE     BEFORE ADVANCING 1 LINE.
    WRITE RPT-RECORD FROM RPT-TOTAL-LINE BEFORE ADVANCING 1 LINE.

    CLOSE   EMP-SEQ-FILE
            IN-CNTL-CARD
            EMP-HIRE-RPT.

9000-EXIT.
    EXIT.
```



以上で BATCHRPT.cbl ソースプログラムの入力は終了です。 この時点でエラー件数が 12 であれば、問題ありません。先に進んでください。

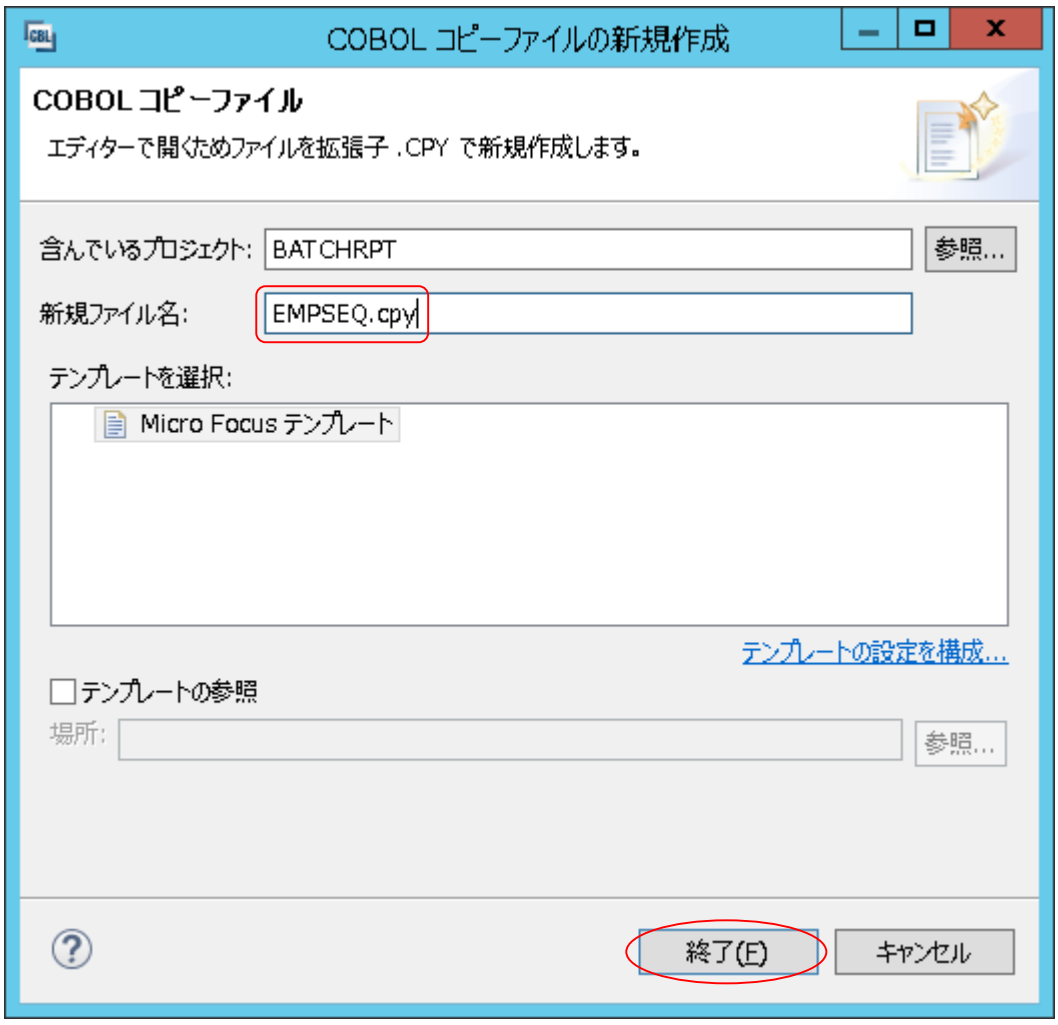
5　コードエディターで COBOL コピーファイルを入力します。

COBOL エクスプローラビューにて **COBOL プログラム** フォルダを右クリックし

新規(N) → COBOL コピーファイル

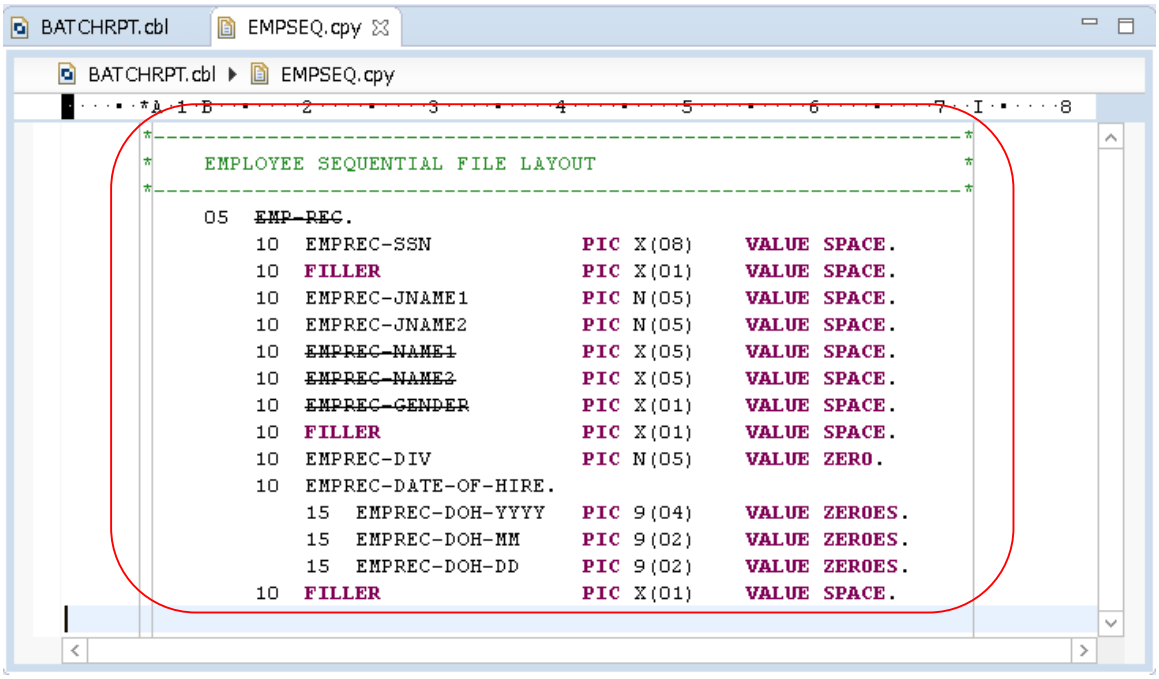
を選択します。

新規ファイル名欄には **EMPSEQ.cpy** を指定します。[**終了(F)**] ボタンを押下します。



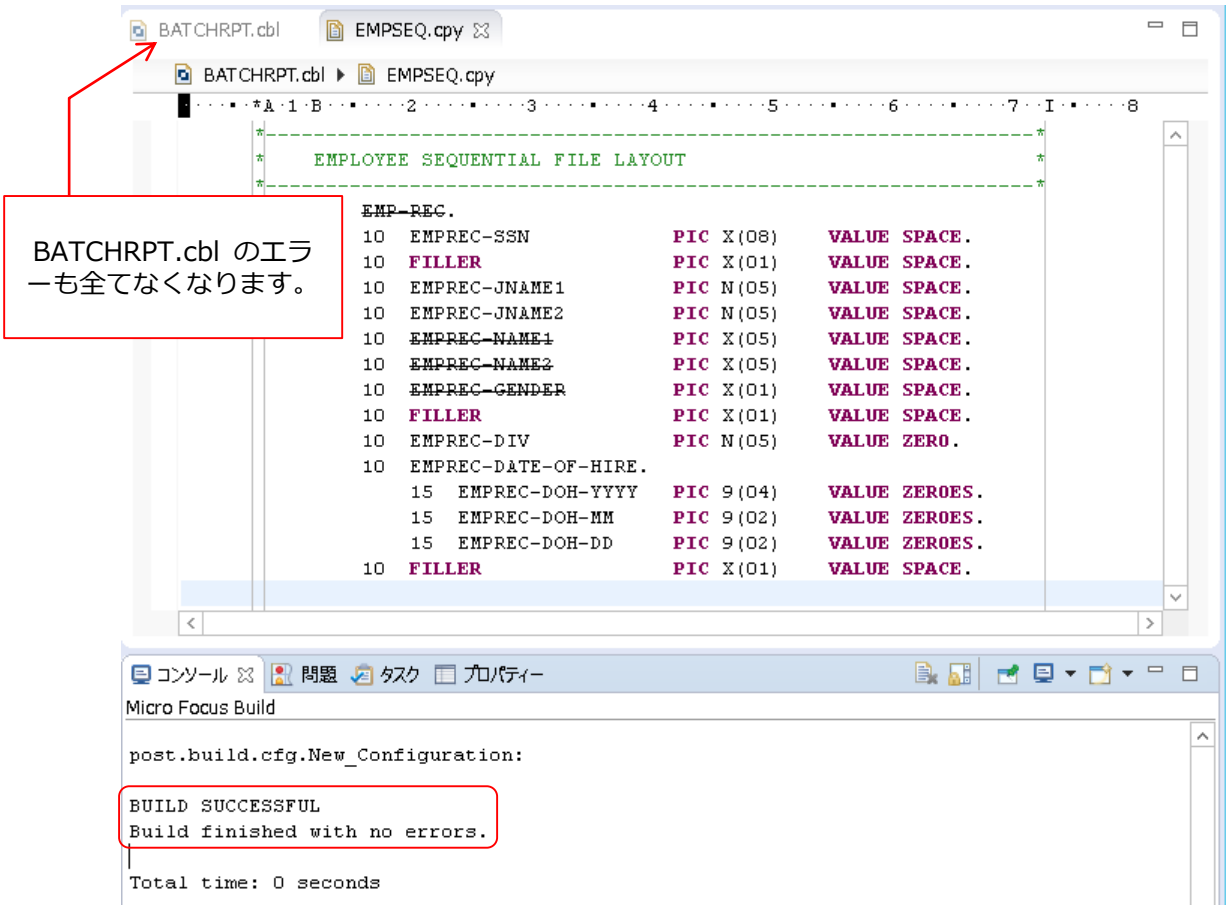
EMP-RECORD-IO-AREA データ項目のレコード記述を入力します。

```
*-----------------------------------------------------------------*
*    EMPLOYEE SEQUENTIAL FILE LAYOUT                              *
*-----------------------------------------------------------------*
     05  EMP-REC.
         10  EMPREC-SSN              PIC X(08)    VALUE SPACE.
         10  FILLER                  PIC X(01)    VALUE SPACE.
         10  EMPREC-JNAME1           PIC N(05)    VALUE SPACE.
         10  EMPREC-JNAME2           PIC N(05)    VALUE SPACE.
         10  EMPREC-NAME1            PIC X(05)    VALUE SPACE.
         10  EMPREC-NAME2            PIC X(05)    VALUE SPACE.
         10  EMPREC-GENDER           PIC X(01)    VALUE SPACE.
         10  FILLER                  PIC X(01)    VALUE SPACE.
         10  EMPREC-DIV              PIC N(05)    VALUE ZERO.
         10  EMPREC-DATE-OF-HIRE.
             15  EMPREC-DOH-YYYY     PIC 9(04)    VALUE ZEROES.
             15  EMPREC-DOH-MM       PIC 9(02)    VALUE ZEROES.
             15  EMPREC-DOH-DD       PIC 9(02)    VALUE ZEROES.
         10  FILLER                  PIC X(01)    VALUE SPACE.
```



エディタービュー上の **EMPSEQ.cpy** をアクティブにしたまま [**ファイル(F)**] メニューの [**保管(S)**] 或いは **Ctrl + S** キーで保存します。これによりアプリケーションがビルドされます。

コンソールビューにエラーなくビルドできた旨が出力されていることを確認して、次に進んでください。



6　COBOL コンパイル指令を追加します。

本項ではファイル名の割り当てを EXTERNAL(外部割り当て)に変更するためのコンパイラ指令を指定します。

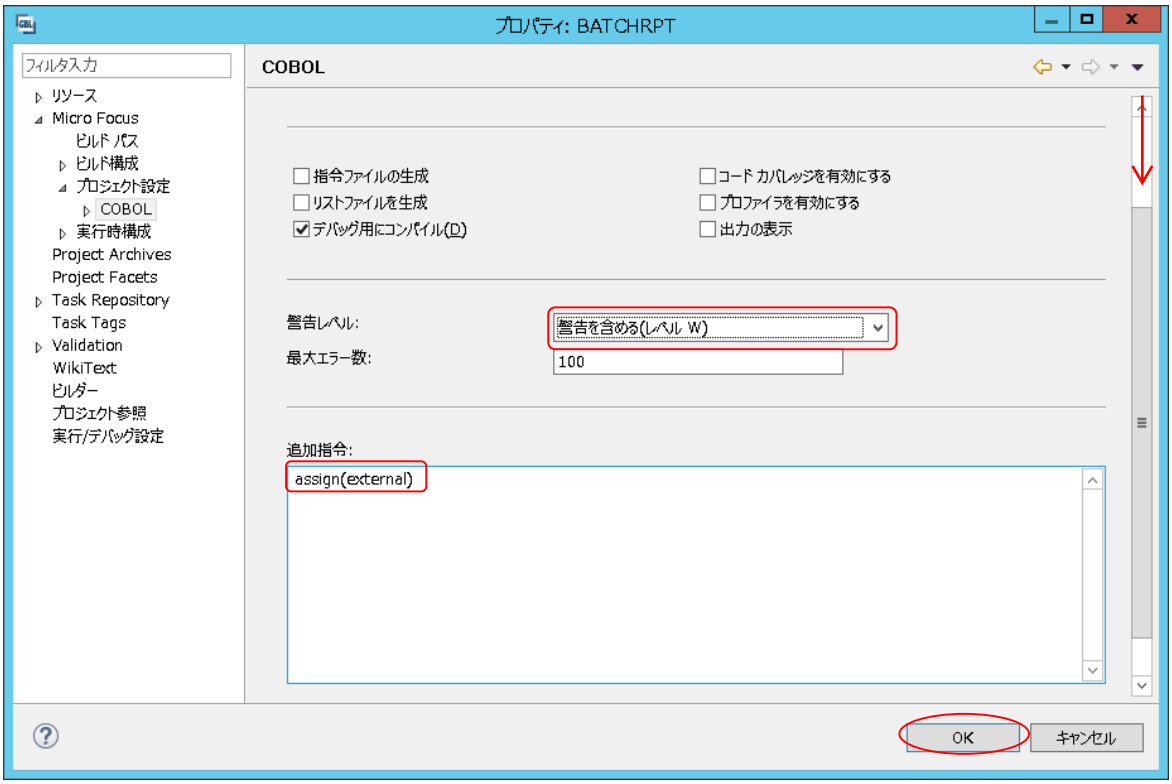
COBOL エクスプローラビューにて **BATCHRPT** プロジェクトを右クリックし [**プロパティ(R)**] を選択します。

Micro Focus COBOL > Project 設定 > COBOL
とナビゲートし表示される画面では

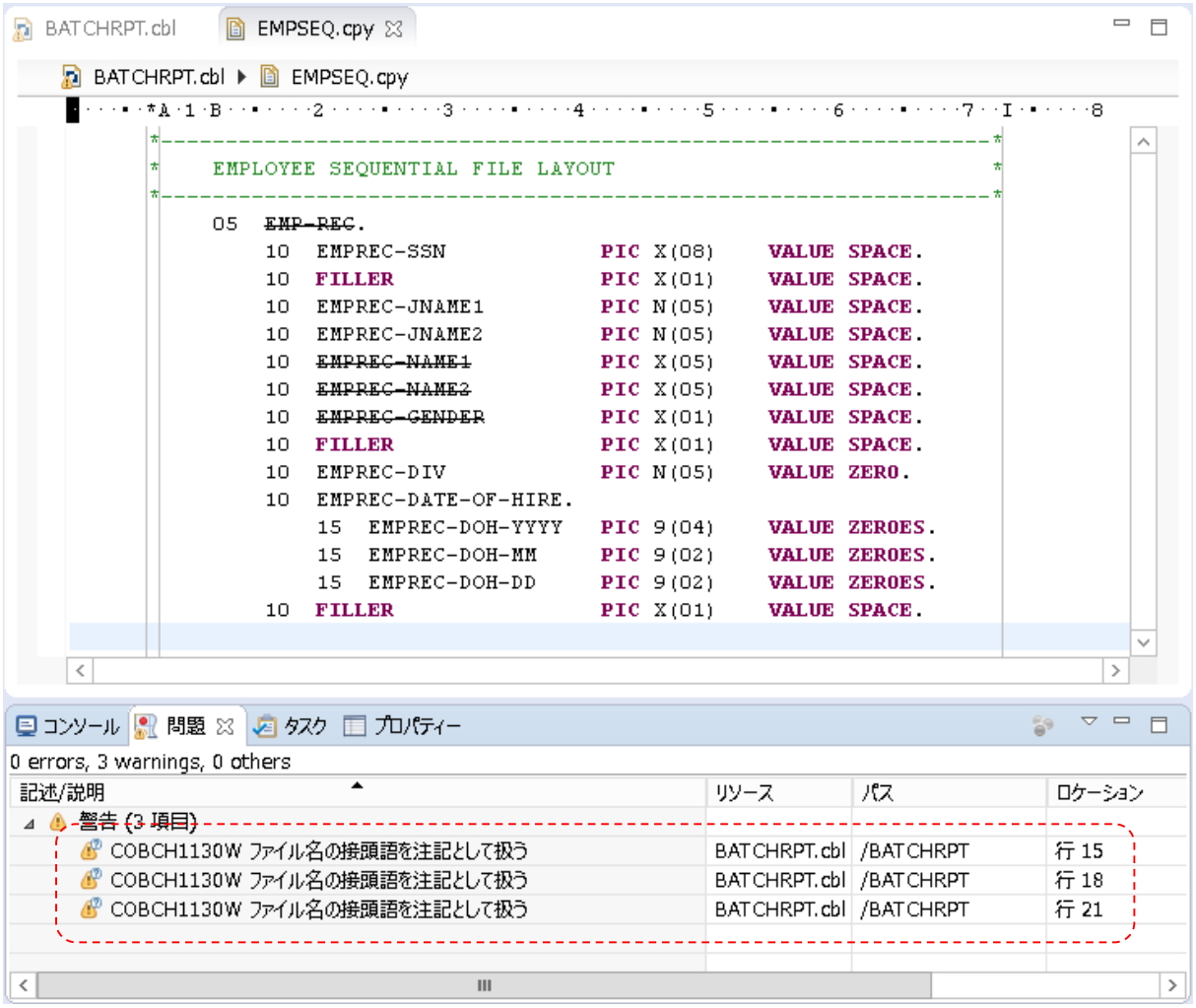
警告レベル　　警告を含める(レベル W)

追加指令欄　　assign(external)

のように選択／入力し、[**OK**] ボタンをクリックします。



問題ビューを選択し「ファイル名の接頭語を注記として扱う」警告が 3 件表示されることを確認して、先に進んでください。



7　環境変数を構成します。

COBOL エクスプローラビューにて **BATCHRPT** プロジェクトを右クリックし[**プロパティ(R)**]を選択します。

Micro Focus COBOL > 実行時構成 > 環境変数

とナビゲートします。

[**追加(A)**] ボタンを押下します。

変数欄　　dd_EMPSEQ

値欄　　Emp_Master.dat

のように入力し [**OK**] ボタンを押下します。

[**追加(A)**] ボタンを押下し更に追加します。

変数欄　　dd_CNTLCARD

値欄　　Cntl_Card.dat

のように入力し [**OK**] ボタンを押下します。

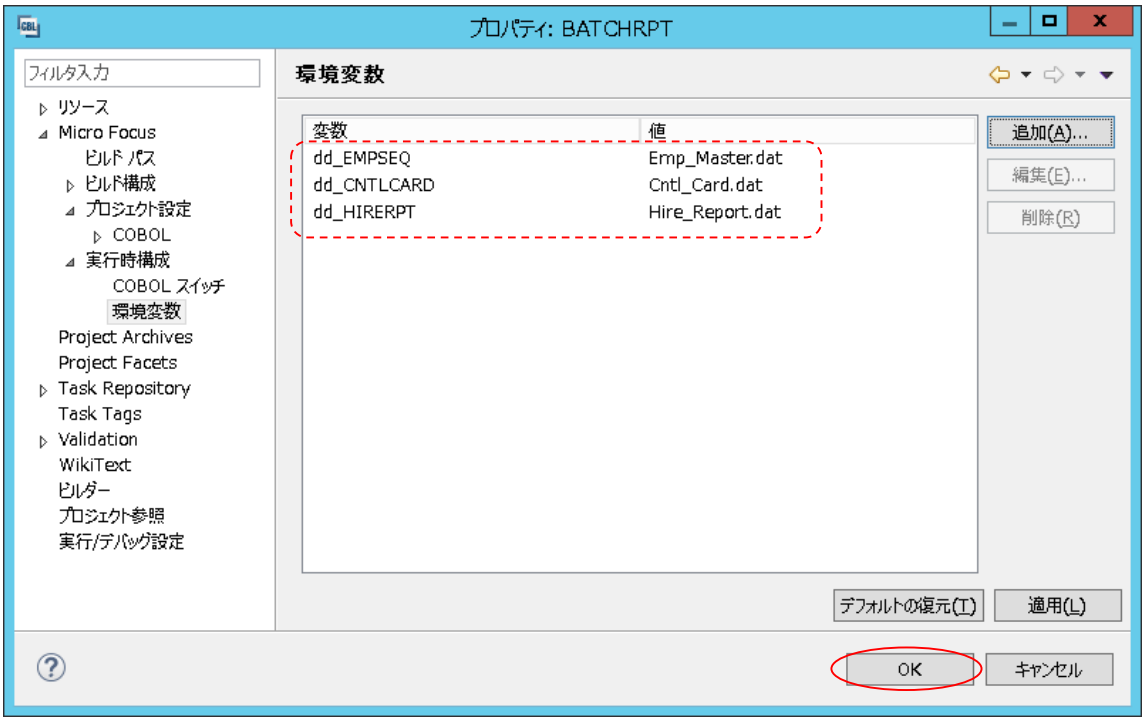
[**追加(A)**] ボタンを押下し更に追加します。

変数欄　　dd_HIRERPT

値欄　　Hire_Report.dat

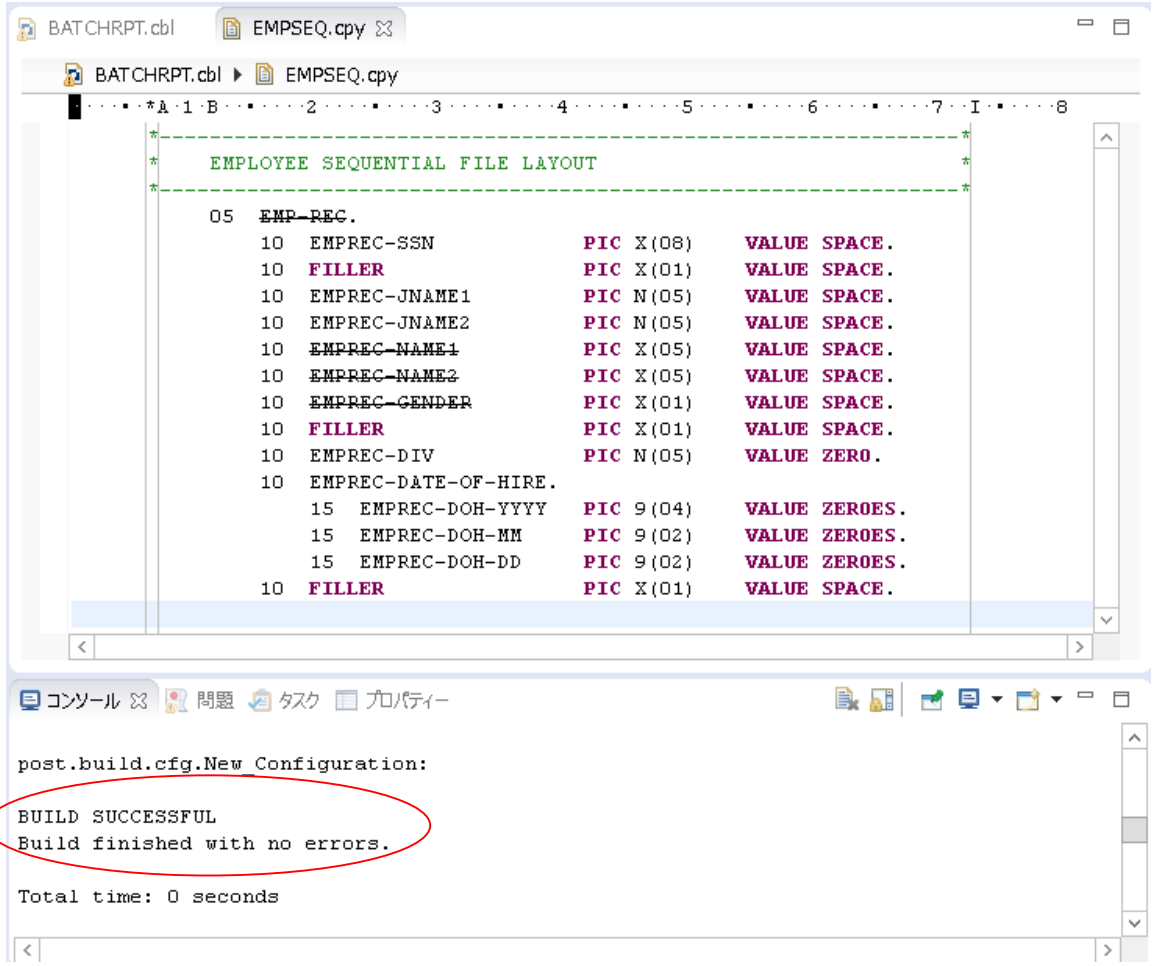
のように入力し [**OK**] ボタンを押下します。

下図のように指定が反映されていることを確認して、[**OK**] ボタンを押下します。



8　COBOL アプリケーションをビルドします。

プロパティー画面で［**OK**］ボタンを押下して自動ビルド機能によるビルド処理をキックします。



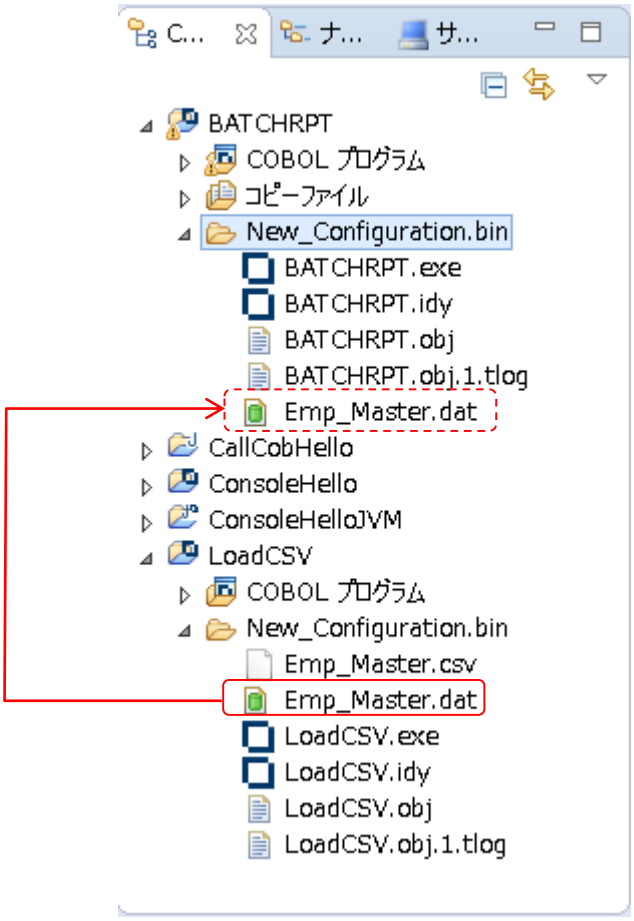
9　入力ファイルをコピーします。

前章で作成した **Emp_Master.dat** ファイルを、COBOL エクスプローラビューより選択します。

右クリックより「**コピー**」を選択します。

BATCHRPT プロジェクト配下の **New_Configuration.bin** フォルダを選択し右クリックから「**貼り付け**」を選択します。

COBOL エクスプローラ上でデータファイルコピーが反映されたことを確認します。



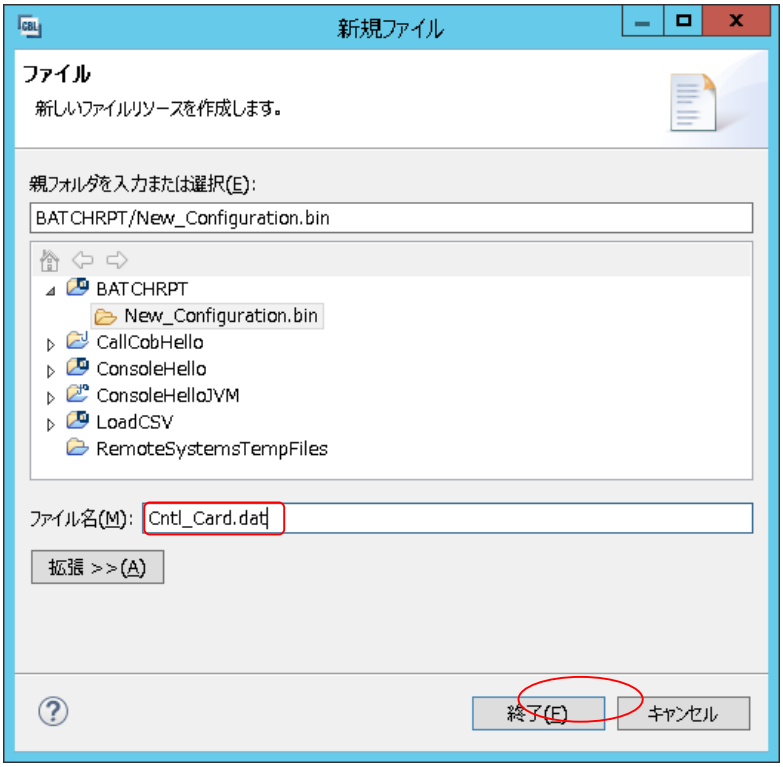
10　制御ファイルを作成します。

New_Configuration.bin フォルダを右クリックし

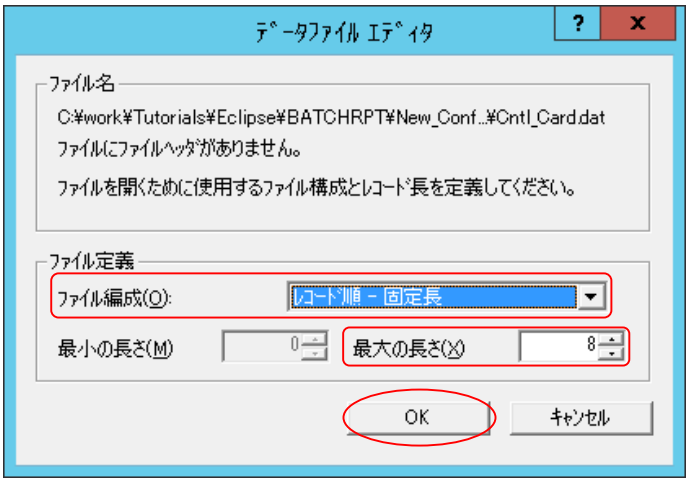
新規(N) → ファイル

を選択します。

[**ファイル名**] 欄にて **Cntl_Card.dat** を指定し [**終了(F)**] ボタンを押下します。

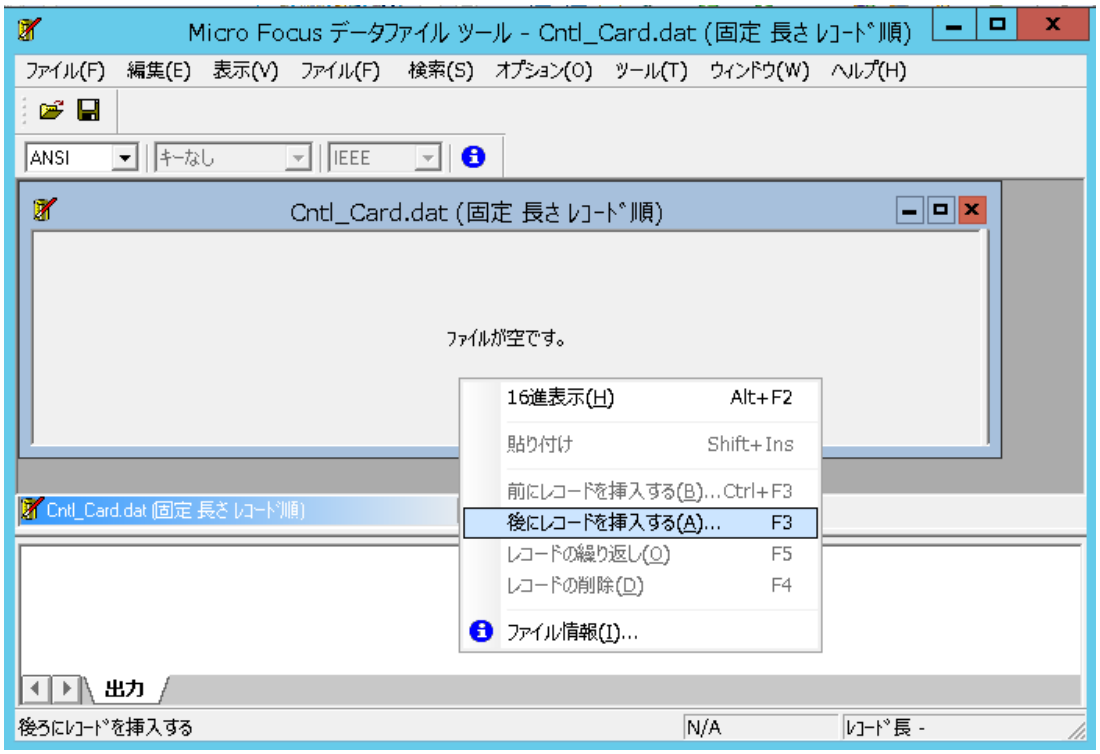


Data File Editor が起動します。[**ファイル編成**] 欄はデフォルトの「**レコード順 – 固定長**」のままにして、[**最大の長さ**] 欄には「**8**」を指定します。[**OK**] ボタンを押下します。

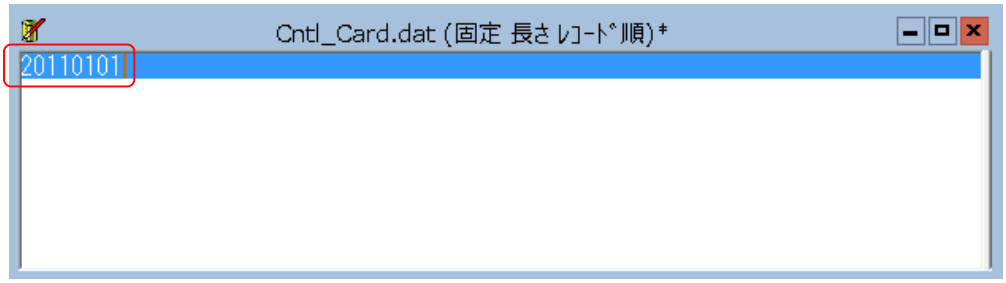


プロファイルファイルを作成するかどうかの問いには [**はい(Y)**] を選択します。

右クリックから [**後にレコードを挿入する**] を選択します。



「**20110101**」を入力します。



[**ファイル**] メニューから [**保存**] を選択してファイルを保存します。

Data File Editor を終了します。

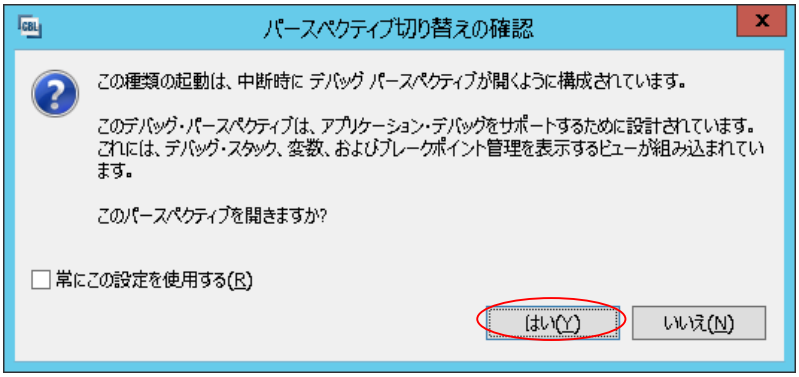
11 COBOL アプリケーションをデバッグ実行します。

New_Configuration.bin フォルダ配下の **BATCHRPT.exe** を右クリックし

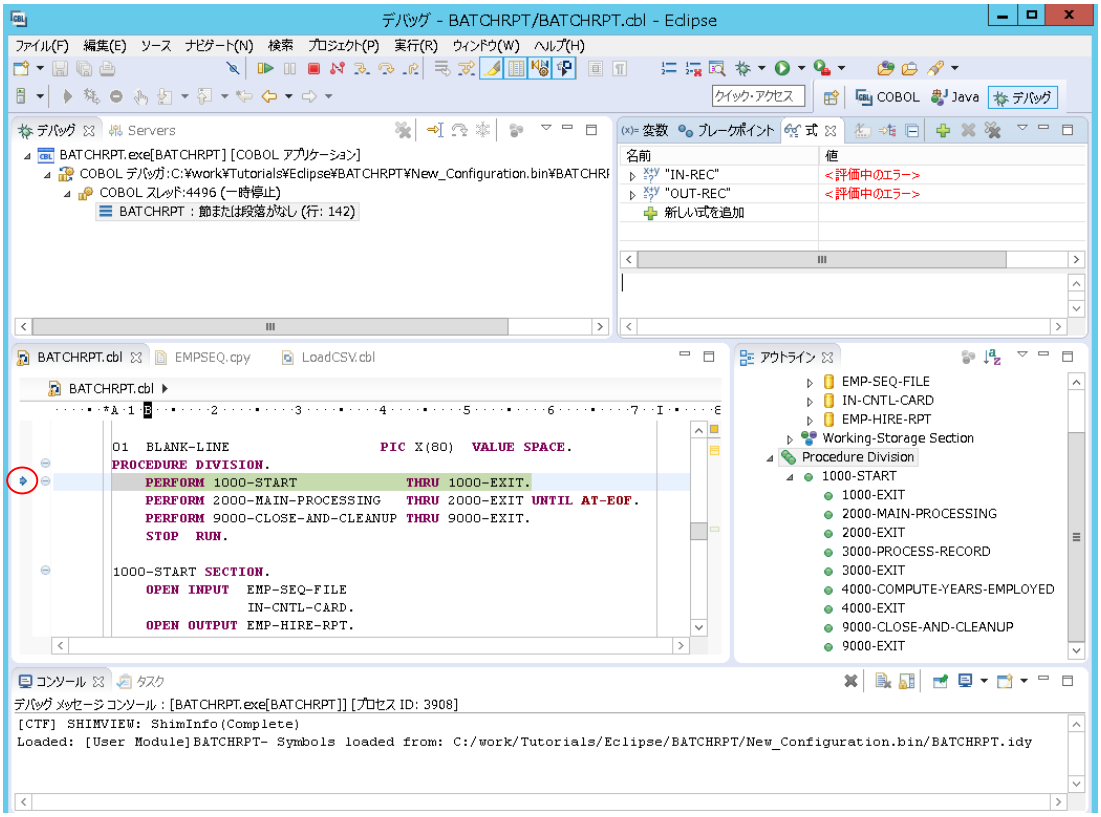
デバッグ(D) → COBOL アプリケーション

を選択します。

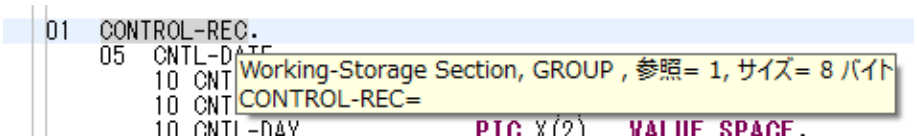
パースペクティブ切り替えの確認メッセージには［**はい(Y)**］を選択します。



デバッガーがステップ実行を開始します。 デバッガーは手続き部の最初の COBOL 文である PERFORM 文の処理前に一時停止している状態となっています。

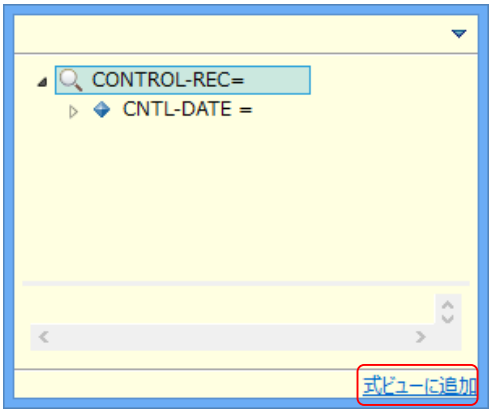


制御ファイルから読み込んだレコードの内容を確認するため、**CONTROL-REC** に格納される値の変遷を追います。データ部の **CONTROL-REC** を選択します。

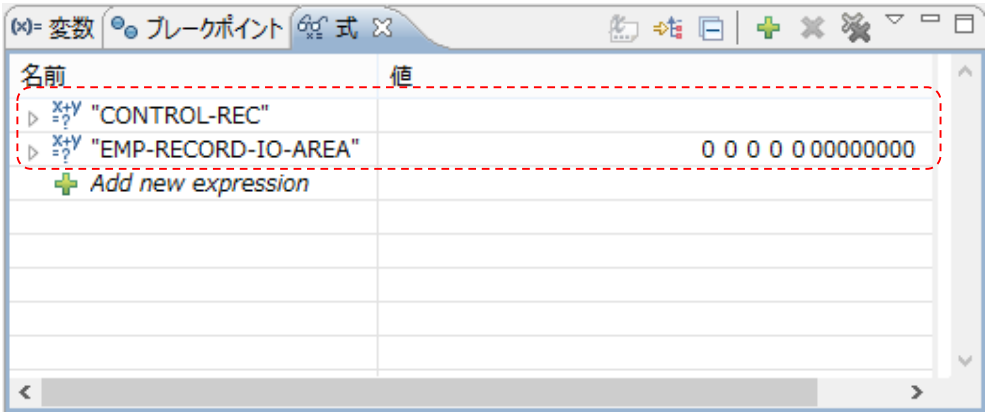


この状態で右クリックし、「**検査**」を選択します。

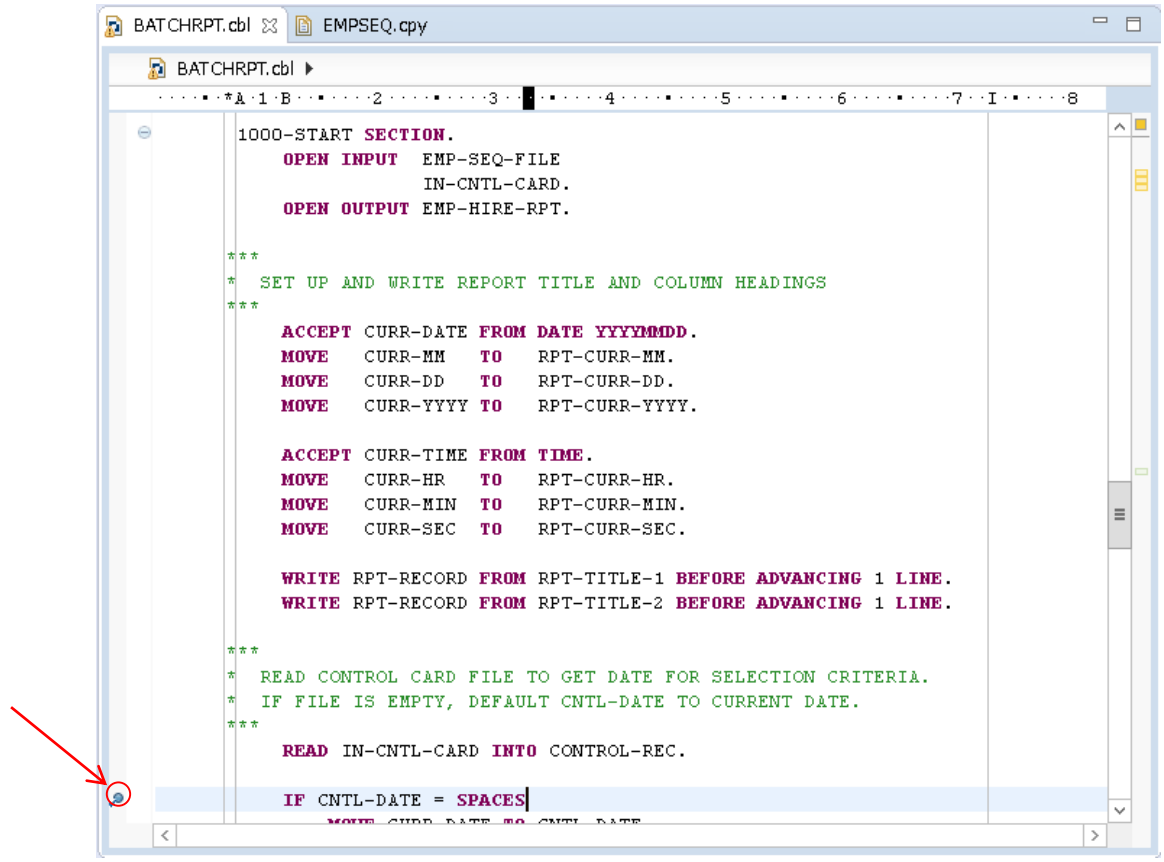
「**式ビューに追加**」をクリックします。



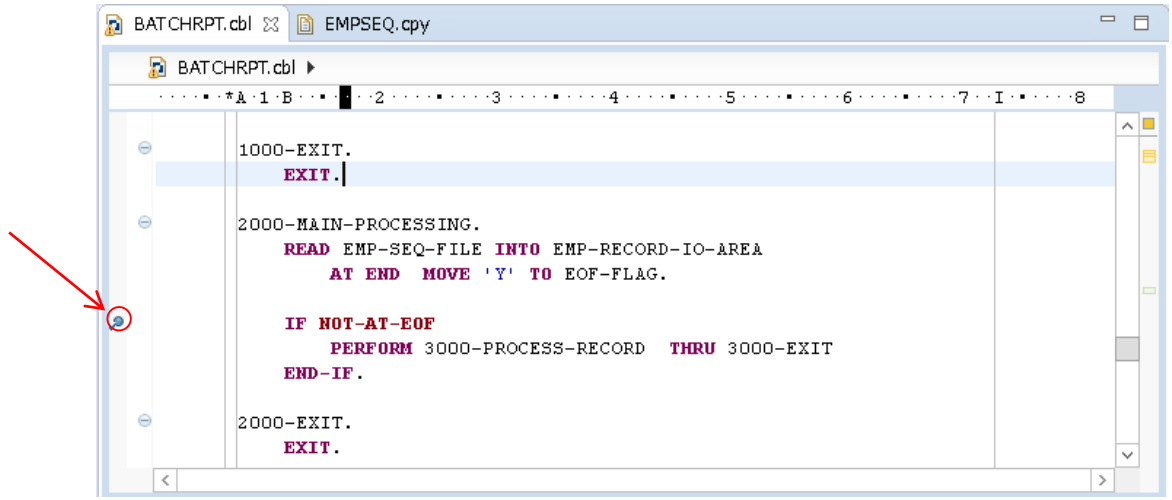
入力ファイルから読み込んだレコードの内容を確認するため、データ部の **EMP-RECORD-IO-AREA** を同じ要領で式ビューに追加します。(前章で追加した IN-REC 及び OUT-REC は式ビューから削除しても構いません。)



手続き部 **1000-START 節**の READ 文に続く IF 文でエディタービューの左端をクリックし、ブレークポイントを設定します。

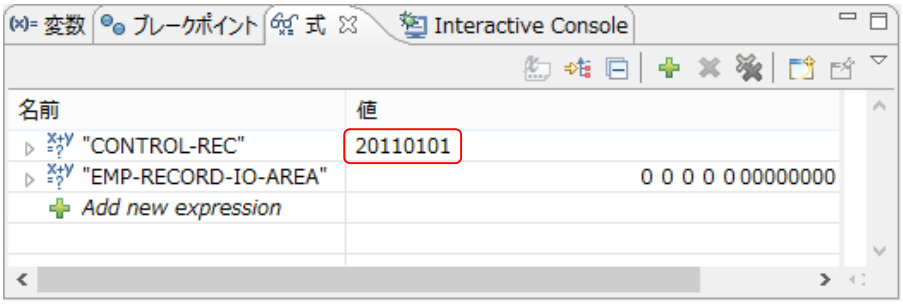


同様に手続き部 **2000-MAIN-PROCESSING 段落**の READ 文に続く IF 文でエディタービューの左端をクリックし、ブレークポイントを設定します。



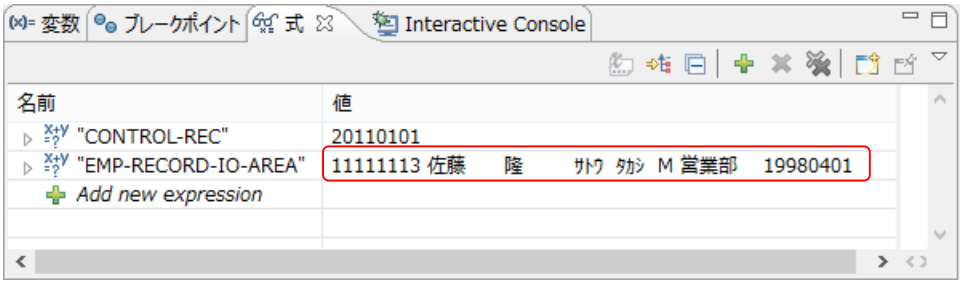
実行(R)メニューから **再開(M)** を選択するか **F8** キーを打鍵すると、デバッガーは最初のブレークポイントで実行を中断します。

式ビュー中の CONTROL-REC の値に制御ファイルから読み込んだレコードが表示されます。



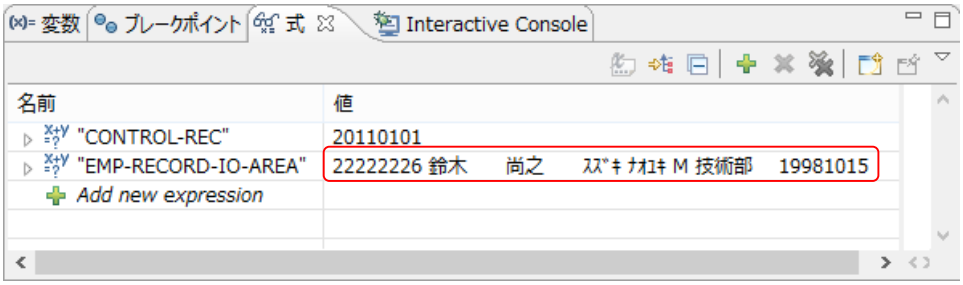
実行(R)メニューから **再開(M)** を選択するか **F8** キーを打鍵すると、デバッガーは 2 番目のブレークポイントで実行を中断します。

式ビューの EMP-RECORD-IO-AREA の値に入力ファイルから読み込んだ 1 番目のレコードが表示されます。



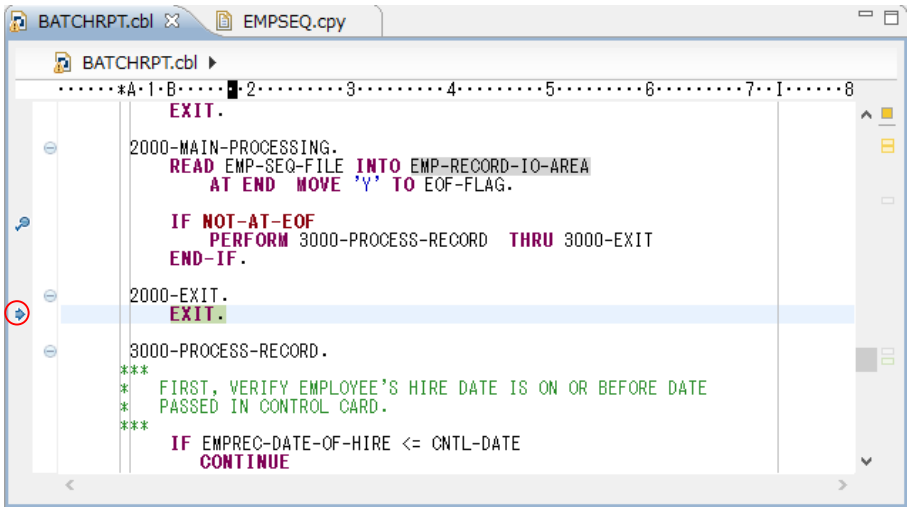
同様に**実行(R)**メニューから **再開(M)** を選択するか **F8** キーを打鍵すると、デバッガーは 2 番目のブレークポイントで実行を中断します。

ウォッチ式の EMP-RECORD-IO-AREA の値に入力ファイルから読み込んだ 2 番目のレコードが表示されます。



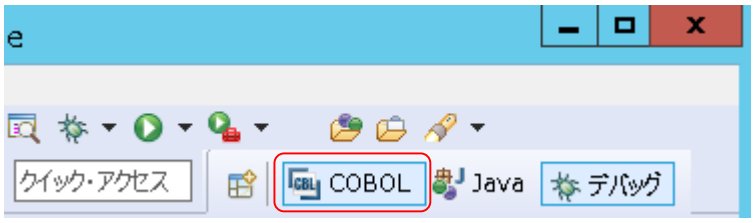
さらに **F8** キーを 8 回、 **F5(ステップイン)**キーを 1 回打鍵すると、デバッガーは 2 番目のブレークポイントに続く EXIT 文で実行を中断します。

IF 文の条件式は、入力ファイルがファイル終了状態であることを示しています。



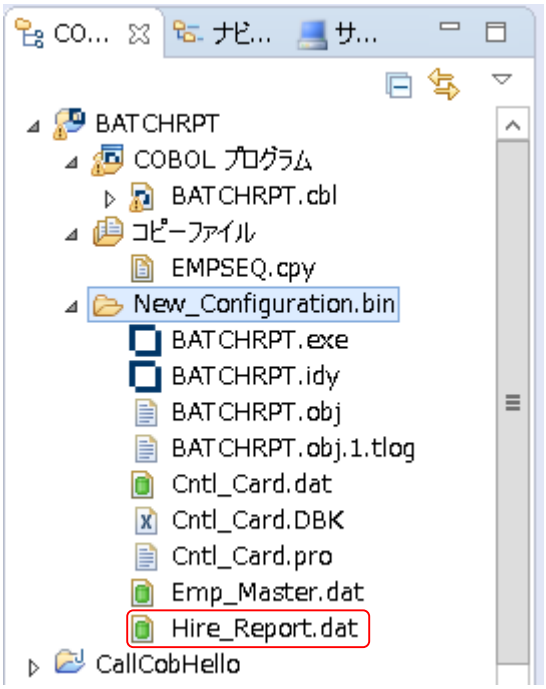
実行(R)メニューから **再開(M)** を選択するか STOP 文を実行するまで **F5** キーを押すと、デバッガーは終了します。

ワークスペース右上コーナにあるパースペクティブ切り替えボタンエリアにて「**COBOL**」を選択し、COBOL パースペクティブに戻ります。



COBOL エクスプローラビューにて **BATCHRPT** プロジェクト配下の **New_Configuration.bin** フォルダを右クリックし [**更新(F)**] を選択します。

New_Configuration.bin 配下に **Hire_Report.dat** が生成されていることを確認します。

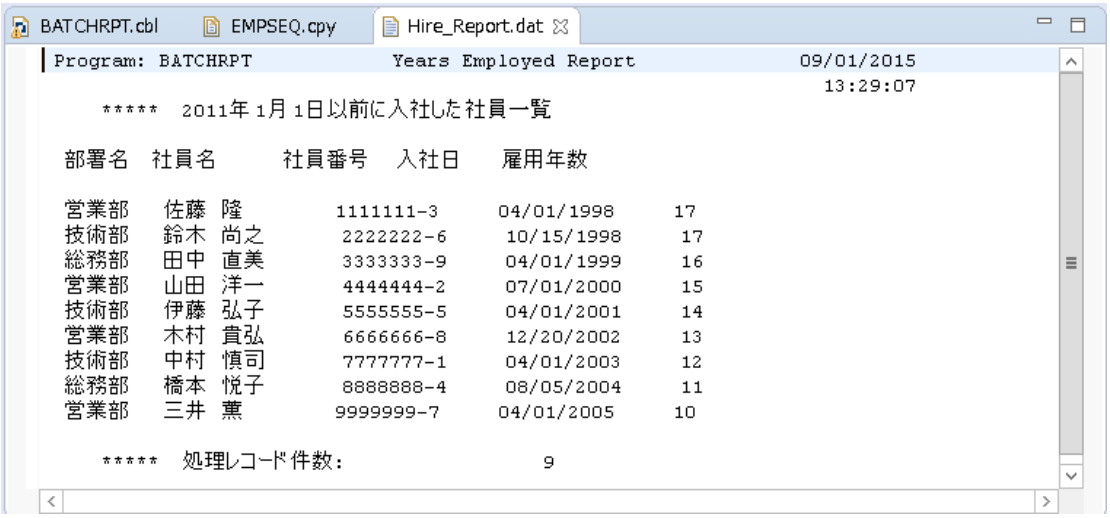


Hire_Report.dat を右クリックし

アプリケーションから開く → テキスト・エディター

を選択します。

社員 9 名分のデータが表示されることを確認します。



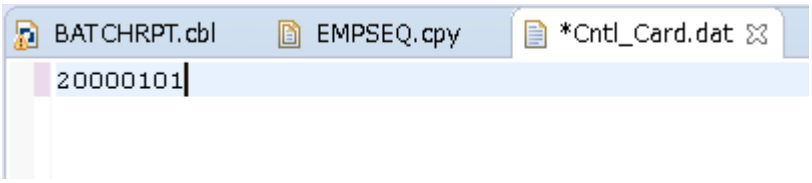
続いて、制御ファイルの値を変更してアプリケーションを実行します。

COBOL エクスプローラ上で **Cntl_Card.dat** ファイルを右クリックし

アプリケーションから開く → テキスト・エディター

を選択します。

Cntl_Card.dat ファイル中のデータを「**20000101**」に更新します。



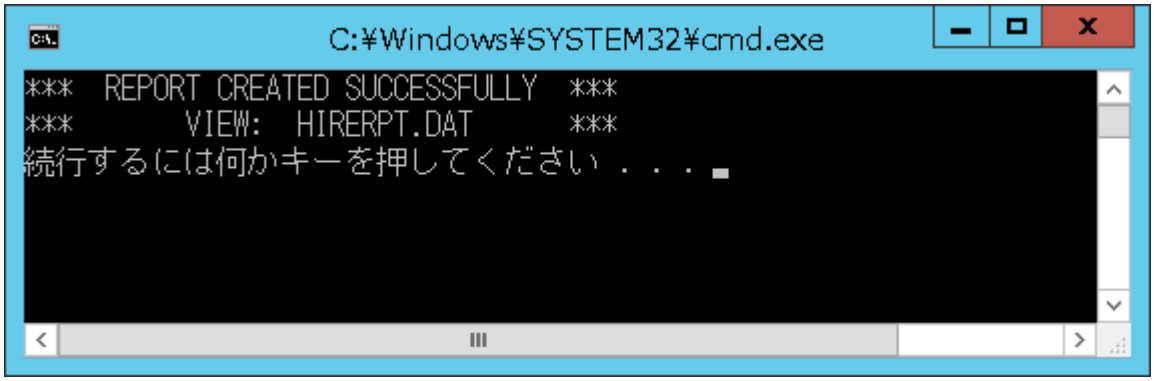
[**ファイル**] メニュー > [**保管**] もしくは **Ctrl + S** を打鍵して、ファイルを保存します。

New_Configuration.bin フォルダ配下の **BATCHRPT.exe** を右クリックし

実行 → COBOL アプリケーション

を選択しアプリケーションを実行します。

コンソール画面がポップアップされたら、メッセージに従い、何等かのキーを打鍵しアプリケーションを終了します。



Hire_Report.dat を右クリックし

アプリケーションから開く → テキスト・エディター

を選択します。

2000 年 1 月 1 日以前に入社した社員 3 名分のデータだけが表示されることを確認します。

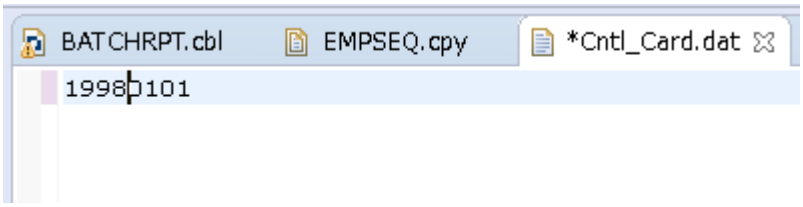
続いて、制御ファイルの値を変更してアプリケーションを実行します。

COBOL エクスプローラ上で **Cntl_Card.dat** ファイルを右クリックし

アプリケーションから開く → テキスト・エディター

を選択します。

Cntl_Card.dat ファイル中のデータを「**19980101**」に更新します。



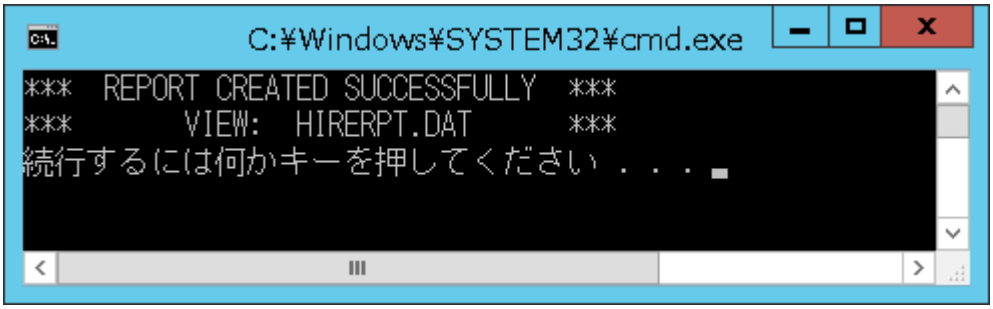
[**ファイル**] メニュー > [**保管**] もしくは **Ctrl + S** を打鍵して、ファイルを保存します。

New_Configuration.bin フォルダ配下の **BATCHRPT.exe** を右クリックし

実行 → COBOL アプリケーション

を選択しアプリケーションを実行します。

コンソール画面がポップアップされたら、メッセージに従い、何等かのキーを打鍵しアプリケーションを終了します。



Hire_Report.dat を右クリックし

アプリケーションから開く → テキスト・エディター

を選択します。

処理されたレコードない旨の出力が表示されることを確認します。

2015 年 11 月 01 日　　初版

マイクロフォーカス株式会社

〒106-0032 東京都港区六本木 7-18-18
住友不動産六本木通ビル 9F

電話　03-5413-4800
URL　http://www.microfocus.co.jp/