```
/* ************************  Function Implementations ***************** */

FUNCTION AssignCountry RETURNS INTEGER
            ( INPUT pcCountry AS CHARACTER ):

/*------------------------------------------------------------------------------
Purpose: Assigns a value to all blank EmployeeCountry fields.
Notes:   Returns the number of Employees assigned.

------------------------------------------------------------------------------*/

        DEFINE VARIABLE iCount AS INTEGER NO-UNDO.

        FOR EACH ttEmployee WHERE ttEmployee.EmployeeBirthCountry = "":
            ttEmployee.EmployeeBirthCountry = pcCountry.
            iCount = iCount + 1.
         END.

         cAssignedCountry = pcCountry.

        RETURN iCount.

END FUNCTION.
```

The second function returns the count of modified employee rows to the caller.

```
FUNCTION GetEmployeeCount RETURNS INTEGER
            (  ):

/*------------------------------------------------------------------------------
Purpose:       Return the total number of employee in the temp-table.

------------------------------------------------------------------------------*/

        RETURN QUERY qEmployee:NUM-RESULTS.

END FUNCTION.
```

The persistent procedure is then converted into a class with the equivalent behavior.

```
/*------------------------------------------------------------------------
    File        : SampleClass.cls
    Description : Sample class to use to compare procedural coding to class-based

    Author(s)   : john
    Created     : Wed Feb 17 14:53:32 EST 2010
    Notes       :
  ----------------------------------------------------------------------*/
```

The **CLASS** statement identifies the name of the class and its location – the *package* – to the compiler:

```
CLASS OOSamples.SampleClass:
```

A class can have a Definitions section just as a procedure can. In addition, variables in the Main Block's definitions can be made **PUBLIC**, **PRIVATE**, or **PROTECTED**, and are referred to as *data members* to emphasize these special characteristics.

```
/* ************************  Definitions  ********************** */

DEFINE TEMP-TABLE ttEmployee
    FIELD EmployeeID           AS CHARACTER
    FIELD EmployeeFirstName    AS CHARACTER
    FIELD EmployeeLastName     AS CHARACTER
    FIELD EmployeePosition     AS CHARACTER
    FIELD EmployeeStartDate    AS DATE
    FIELD EmployeeNotes        AS CHARACTER
    FIELD EmployeeBirthCountry AS CHARACTER
    FIELD EmployeeGender       AS CHARACTER.

DEFINE DATASET     dsEmployee  FOR ttEmployee.
DEFINE DATA-SOURCE srcEmployee FOR AutoEdge.Employee.

DEFINE QUERY qEmployee FOR ttEmployee.

DEFINE VARIABLE mcAssignedCountry AS CHARACTER NO-UNDO.
```

A function that returns a value can be turned into a property of a class, which allows it to be referenced from another procedure or class as if it were a simple variable, but with supporting blocks of code to get or set the property value.

```
DEFINE PUBLIC PROPERTY EmployeeCount AS INTEGER
    GET():
        RETURN QUERY qEmployee:NUM-RESULTS.
    END GET.
    PRIVATE SET.
```

The Main Block of a class can't have executable statements. Startup code goes into the class's constructor, a special method with the same name as the class.

```
CONSTRUCTOR PUBLIC SampleClass():
    BUFFER ttEmployee:ATTACH-DATA-SOURCE (DATA-SOURCE srcEmployee:HANDLE ).
    DATASET dsEmployee:FILL ().   /* All 18 Employees */
    OPEN QUERY qEmployee PRESELECT EACH ttEmployee.
END.
```

Cleanup code can go into the destructor, which is reliably executed when the running class instance is deleted.

```
DESTRUCTOR PUBLIC SampleClass():
   CLOSE QUERY qEmployee.
   EMPTY TEMP-TABLE ttEmployee.
END.
```

An internal procedure in a procedure becomes a method in a class with a return type of **VOID**.

```
METHOD PUBLIC VOID InitializeNotes ( INPUT pcPosition AS CHARACTER ):

/*------------------------------------------------------------------------------
Purpose:       Assign a value to all empty EmployeeNotes.
Notes:          Done for a specific EmployeePosition.

------------------------------------------------------------------------------*/

    FOR EACH ttEmployee WHERE ttEmployee.EmployeePosition = pcPosition:
        ttEmployee.EmployeeNotes = "Initial note for this " + pcPosition.
    END.
END METHOD.
```

A user-defined function in a procedure becomes a method with a return type in a class.

```
METHOD PUBLIC INTEGER AssignCountry  ( INPUT pcCountry AS CHARACTER ):

/*------------------------------------------------------------------------------
        Purpose: Assigns a value to all blank EmployeeCountry fields.

        Notes:   Returns the number of Employees assigned.

------------------------------------------------------------------------------*/

        DEFINE VARIABLE iCount AS INTEGER NO-UNDO.

        FOR EACH ttEmployee WHERE ttEmployee.EmployeeBirthCountry = "":
            ttEmployee.EmployeeBirthCountry = pcCountry.
            iCount = iCount + 1.
         END.

         mcAssignedCountry = pcCountry.

        RETURN iCount.

END METHOD.
```

The class ends with an **END CLASS** statement to balance the **CLASS** header statement.

```
END CLASS.
```

The presentation then shows a wrapper procedure to run an instance of the persistent procedure and invoke the internal procedure and the functions it contains. Any errors in the run statements are not detected until runtime, because the compiler does not cross-check the validity of calls to other procedures.

```
/*------------------------------------------------------------------------
    File        : SampleProcRunner.p
    Purpose     : Wrapper procedure to run SampleProc.p

------------------------------------------------------------------------*/

DEFINE VARIABLE iEmployeeCount AS INTEGER NO-UNDO.
DEFINE VARIABLE iCountryCount  AS INTEGER NO-UNDO.
DEFINE VARIABLE hSampleProc    AS HANDLE  NO-UNDO.

RUN OOSamples/SampleProc.p PERSISTENT SET hSampleProc.

RUN InitializeNotes IN hSampleProc (INPUT "Admin").

iEmployeeCount = DYNAMIC-FUNCTION ("GetEmployeeCount" IN hSampleProc).

iCountryCount  = DYNAMIC-FUNCTION ("AssignCountry" IN hSampleProc,
  INPUT "France").

DELETE PROCEDURE hSampleProc.

MESSAGE "There are" iEmployeeCount "employees, of which" SKIP
        iCountryCount "have just been assigned to France."
        VIEW-AS ALERT-BOX.
```

A similar wrapper procedure can create an instance of the class and invoke its methods and reference its properties. Any errors in references to the class are detected by the compiler, because the variable that holds the reference to the class