The default destructor and **END CLASS** statement end the class definition for **EmployeeManager**.

```
        DESTRUCTOR PUBLIC EmployeeManager ( ):

        END DESTRUCTOR.

END CLASS.
```

EmployeeClient.cls serves as a test class that creates an instance of **EmployeeManager** and invokes its methods.

```
/*------------------------------------------------------------------------
    File        : EmployeeClient

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

USING Progress.Lang.*.

CLASS OOSamples.EmployeeClient:
```

The variable **oEmpManager** holds the object reference to the instance of **EmployeeManager**. If this class needs to invoke **InitializeNotes**, which is not part of the **IDataManager** interface, then it must define **oEmpManager** as of type **EmployeeManager**. Otherwise it can define the variable as **IDataManager**.

```
    DEFINE PRIVATE VARIABLE oEmpManager AS OOSamples.EmployeeManager.
```

The constructor invokes the class's one method, **FetchEmployees**:

```
    DEFINE PRIVATE VARIABLE cFilter     AS CHARACTER NO-UNDO.
    DEFINE PRIVATE VARIABLE hEmpTable   AS HANDLE    NO-UNDO.

        CONSTRUCTOR PUBLIC EmployeeClient (  ):
                SUPER ().
                FetchEmployees().
        END CONSTRUCTOR.
```

Its **FetchEmployees** method creates an instance of **EmployeeManager**, executes the **InitializeNotes** and **FetchData** methods, and displays a message that uses the **RowCount** property.

```
METHOD PRIVATE VOID FetchEmployees(  ):

        DEFINE VARIABLE hEmpBuffer AS HANDLE NO-UNDO.

        oEmpManager = NEW OOSamples.EmployeeManagerAlt().
        oEmpManager:InitializeNotes("SalesRep").
        cFilter = "EmployeePosition = 'SalesRep' ".
        hEmpTable = oEmpManager:FetchData(INPUT cFilter).
        hEmpBuffer = hEmpTable:DEFAULT-BUFFER-HANDLE.
        hEmpBuffer:FIND-FIRST ().
        MESSAGE " Employee "
           hEmpBuffer:BUFFER-FIELD ("EmployeeFirstName"):BUFFER-VALUE
           hEmpBuffer:BUFFER-FIELD ("EmployeeLastName"):BUFFER-VALUE
           " is the first of " oEmpManager:RowCount " employees with "
           cFilter VIEW-AS ALERT-BOX.
        RETURN.
END METHOD.
```

The two-part session on ***Inheritance and Super Classes*** shows how to refactor the **EmployeeManager** and **CustomerManager** classes to extract common code from them and move it into a common super class, called **DataManager.cls**.

```
/*------------------------------------------------------------------------
    File        : DataManager

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

USING Progress.Lang.*.

CLASS OOSamples.DataManager:
```

The **RowCount** property is common to both employee and customer management, so it is moved to the super class.

```
DEFINE PUBLIC PROPERTY RowCount AS INTEGER
GET.
PRIVATE SET.
```

There are two data values that are used in the super class but must be set in the respective subclass: the buffer handle of the database table that data is retrieved from, and the handle of the temp-table buffer that data is copied into. To make these values available to the super class to use, and to the subclass to set, they are defined in the super class as protected properties.

```
DEFINE PROTECTED PROPERTY DB_BufferHandle AS HANDLE
GET.
SET.

DEFINE PROTECTED PROPERTY TT_BufferHandle AS HANDLE
GET.
SET.
```

Almost all the code from the **FetchData** method is moved to the super class, with adjustments to make its references to the database and temp-table buffers dynamic:

```
   METHOD PROTECTED HANDLE FetchData( INPUT pcFilter AS CHARACTER ):
        DEFINE VARIABLE hQuery    AS HANDLE NO-UNDO.

        CREATE QUERY hQuery.
        hQuery:SET-BUFFERS (THIS-OBJECT:DB_BufferHandle).
        hQuery:QUERY-PREPARE (pcFilter).
        hQuery:QUERY-OPEN ().
        hQuery:GET-FIRST ().
        DO WHILE NOT hQuery:QUERY-OFF-END:
            TT_BufferHandle:BUFFER-CREATE ().
            TT_BufferHandle:BUFFER-COPY (DB_BufferHandle ).
            hQuery:GET-NEXT ().
        END.
        RowCount = hQuery:NUM-RESULTS.
        hQuery:QUERY-CLOSE ().
        DELETE OBJECT hQuery.
        RETURN TT_BufferHandle.
    END METHOD.
```

Most of the executable code can then be removed from **CustomerManager.cls**, which is now a subclass of **DataManager.cls**. The **INHERITS** phrase tells the compiler that this subclass inherits common behavior from **DataManager.cls**.

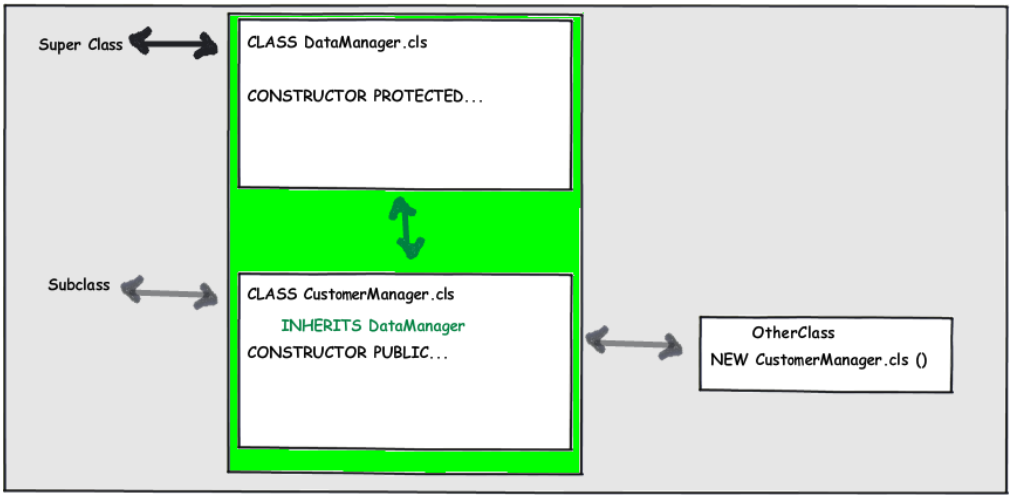
```
/*------------------------------------------------------------------------
    File        : CustomerManager

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

USING Progress.Lang.*.
USING OOSamples.IDataManager.

CLASS OOSamples.CustomerManager
    INHERITS OOSamples.DataManager IMPLEMENTS IDataManager:
```

Because the super class constructor is **PROTECTED**, some other class outside its class hierarchy cannot create an instance of **DataManager** directly ("NEW" it). However, when another class creates a new **CustomerManager**, an instance of **DataManager** is also created, and both constructors executed:



The static temp-table definition is specific to the table being managed, so that stays in **CustomerManager.cls**:

```
    DEFINE TEMP-TABLE ttCustomer
        FIELD CustomerFirstName AS CHARACTER
        FIELD CustomerLastName  AS CHARACTER
        FIELD CustomerBirthCountry  AS CHARACTER.
```

The subclass constructor sets the values of the two protected properties that the super class uses:

```
        CONSTRUCTOR PUBLIC CustomerManager (  ):
                SUPER ().
                DB_BufferHdl = BUFFER AutoEdge.Customer:HANDLE.
                TT_BufferHdl = BUFFER ttCustomer:HANDLE.
        END CONSTRUCTOR.
```

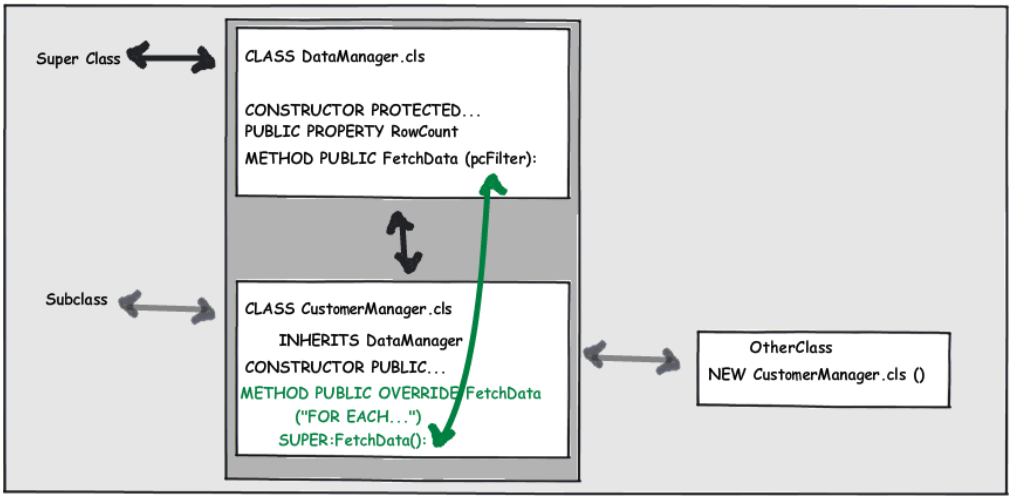
The subclass's **FetchData** method now has almost no code if its own. It defines itself as being an **OVERRIDE** of a method with the same name in its super class, and then invokes the common behavior in the super class.

```
METHOD PUBLIC OVERRIDE HANDLE FetchData( INPUT pcFilter AS CHARACTER ):

    SUPER:FetchData("FOR EACH AutoEdge.Customer WHERE " + pcFilter).
     RETURN TEMP-TABLE ttCustomer:HANDLE.

END METHOD.
```

The subclass implementation of **FetchData** can now invoke the common code in its super class, so that together they provide a complete implementation of its behavior:



The destructor is still there as a place to put any cleanup code required by **CustomerManager.cls**. If there is none, it could be dispensed with.

```
    DESTRUCTOR PUBLIC CustomerManager ( ):

     END DESTRUCTOR.
END CLASS.
```

In the test class **CustomerClient.cls**, which creates an instance of **CustomerManager.cls**, invokes its **FetchData** method, and references its **RowCount** property, nothing changes. It is unaware that most of the code for **FetchData** is now in a super class, and that the property **RowCount** is defined in that super class. Factoring out common code into a class hierarchy is transparent to all other classes, which instantiate and reference only the subclass.

```
    METHOD PRIVATE VOID FetchCustomers(  ):

        DEFINE VARIABLE hBuffer AS HANDLE NO-UNDO.

        oManager = NEW OOSamples.CustomerManager().
        cFilter = "CustomerBirthCountry = 'USA' ".
        hTable = oManager:FetchData(INPUT cFilter).
        hBuffer = hTable:DEFAULT-BUFFER-HANDLE.
        hBuffer:FIND-FIRST ().
        MESSAGE " Customer "
           hBuffer:BUFFER-FIELD ("CustomerFirstName"):BUFFER-VALUE
           hBuffer:BUFFER-FIELD ("CustomerLastName"):BUFFER-VALUE
           " is the first of " oManager:RowCount " Customers with "
           cFilter VIEW-AS ALERT-BOX.
        RETURN.

    END METHOD.
```

The same changes can be made to **EmployeeManager.cls** so that it now acts as another subclass of **DataManager.cls**. All the common code is now in one place.

```
/*------------------------------------------------------------------------
    File        : EmployeeManager
  ----------------------------------------------------------------------*/

USING Progress.Lang.*.
USING OOSamples.IDataManager.

CLASS OOSamples.EmployeeManager
    INHERITS OOSamples.DataManager IMPLEMENTS IDataManager:

    DEFINE TEMP-TABLE ttEmployee
        FIELD EmployeeFirstName AS CHARACTER
        FIELD EmployeeLastName  AS CHARACTER
        FIELD EmployeePosition  AS CHARACTER.

       CONSTRUCTOR PUBLIC EmployeeManager (  ):
              SUPER ().
              DB_BufferHdl = BUFFER AutoEdge.Employee:HANDLE.
              TT_BufferHdl = BUFFER ttEmployee:HANDLE.
       END CONSTRUCTOR.

       METHOD PUBLIC OVERRIDE HANDLE FetchData( INPUT pcFilter AS CHARACTER ):

           SUPER:FetchData("FOR EACH AutoEdge.Employee WHERE " + pcFilter).
           RETURN TEMP-TABLE ttEmployee:HANDLE.

       END METHOD.

    METHOD PUBLIC VOID InitializeNotes ( INPUT pcPosition AS CHARACTER ):

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

The presentation on ***Encapsulation and Overloading*** shows an alternative way to make the two buffer handles available to the super class. It defines them as **PRIVATE** variables (data members) in the main block of **DataManager.cls**, and accepts the values from the subclass as new parameters to its constructor: