Application Development ToolSet/Client Server
VisualAge for RPG

Student Exercises

Course code 22LA

www.software.ibm.com/ad/varpg

Version 3 Release 2 Modification 1

First Edition (February, 1998)

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Comments concerning this notebook and its usefulness for its intended purpose are welcome. You may send written comments to:

IBM Canada Software Solutions

1150 Eglinton Ave. East, Norh York, Ontario, M3C 2H7

Attention: Claus Weiss, VisualAge for RPG Introduction (37LA).

or email to**: weiss@ca.ibm.com**

Overall Lab Guide

The objective of the VisualAge for RPG lab is to have the students work with the VisualAge for RPG development environment to create, build and execute VisualAge for RPG programs. At the end of the lab, the student should know how to create a simple VisualAge for RPG application by creating a graphical user interface, writing RPG code to handle the interface events, and building the application.

All the exercises require that the previous labs are successfully completed. Ensure that all students have finished the labs and understand the background so they can successfully develop larger VisualAge for RPG applications at home.

The first labs will have very detailed instructions on how to proceed. This will change when proceeding through the course because it is assumed that students will get familiar with the VisualAge for RPG environment and will not require the detailed instructions.

It is however, important that students go through the beginning labs to get familiar with VisualAge for RPG and its terminology.

Note: *The pictures in these labs show a similar application being built. Some of the names and icons may be different than the environment you are working with.*

Prerequisites

Although not required, familiarity with the RPG language is preferred.

Also, it is helpful if the student is familiar with basic Windows 95 operations such as working with the desktop and basic mouse operations such as opening folders and performing drag-and-drop operations.

Labs

Trademarks

IBM is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation.

AS/400
Application System/400
VisualAge
DB2/400
ILE
Integrated Language Environment
IBM
OS/400
RPG/400
VRPG Client
VisualAge

Trademarks of other companies as shown

'Lotus'   'Lotus Development Corporation'

'Microsoft'   'Microsoft Corporation'

'Windows'   'Microsoft Corporation'

VisualAge for RPG Hands on Lab

Create a Simple VisualAge for RPG Application

Exploring the GUI Designer and Creating Action Subroutines

During this lab you will learn more about the VisualAge for RPG GUI designer. In doing so you will be performing the following:

1. Exploring the GUI designer by:

   ◆ Starting the GUI designer

   ◆ Getting familiar with the components of the GUI designer

   ◆ Using parts and GUI designer components to create a graphical user interface

   ◆ Saving the graphical user interface

2. Creating Action Subroutines by:

   ◆ Selecting events

   ◆ Working with the LPEX editor

   ◆ Creating action subroutines to respond to events

   ◆ Building the application and running a simple application

What You Should be Able to Do

As a result of this exercise you will create the first window of your GUI application. In doing so you will:

◆ Create a graphical user interface

◆ Create action subroutines to respond to events

◆ Save your application

◆ Build and run the application

Starting the GUI Designer

To start the GUI designer for the first time press the **Start** button on the task bar and choose **Start→Programs→ADTS Client Server for AS/400→ VisualAge for RPG→GUI Designer**. A status window indicating the progress in the initialization process of the **GUI designer** will appear.

The GUI designer including the **parts palette** and the first design window appears. Rearrange the windows if necessary so they are all visible. Resize the **parts palette** so that all parts are visible.
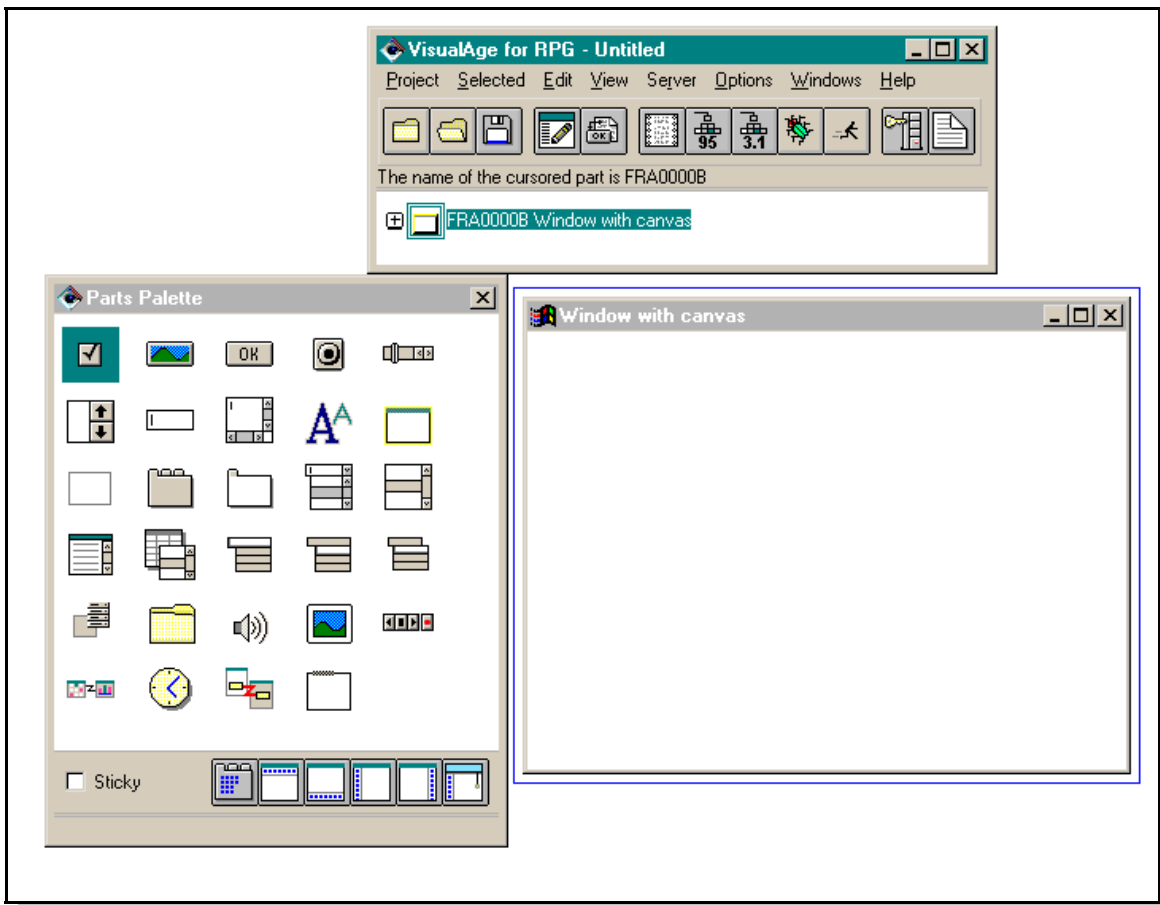
GUI designer Components

The VisualAge for RPG GUI designer consists of these major components:
◆ The Menu Bar

◆ The Tool Bar

◆ The Project View

◆ The Parts Palette

The Menu Bar

The **Menu Bar** is located just below the title bar of the GUI designer window. It provides a variety of tasks you can use to create and modify your GUI application, to customize the GUI designer, and to get online help.

Tip: To get additional information on a particular menu item, press F1 while the desired menu bar item is highlighted. This will display a help window with detailed information on the selected menu item.

The Tool Bar

The **tool bar** is a menu of icons. It provides a fast path access to many of the menu items on the menu bar. Move the mouse pointer over the icons of the tool bar. A short description of the icons function is displayed in the form of 'flyover' help next to the pointer**.**
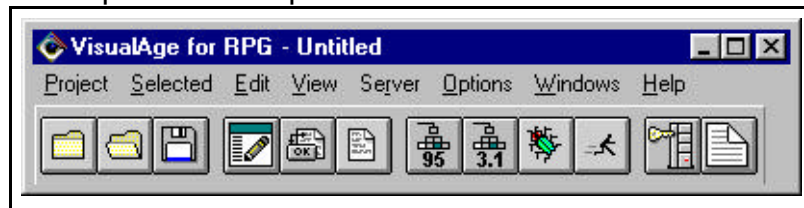
The Project View

The **Project View** is used to hold all the parts you will create during your GUI designer session. The project view can be changed to display the parts of a project in different views.
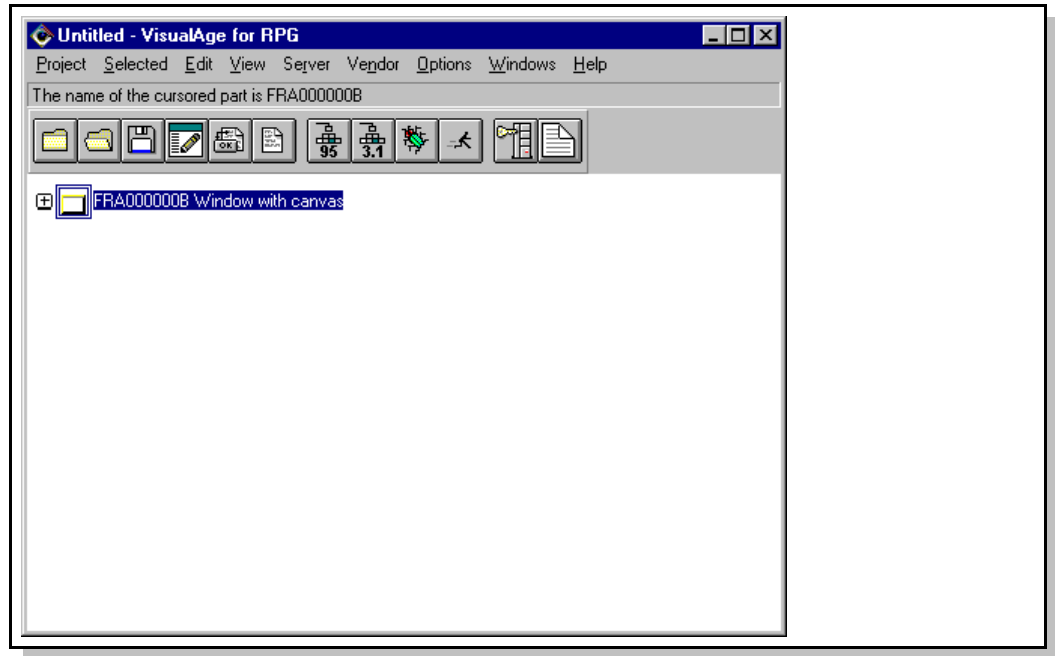
The Parts Palette

The **parts palette** contains parts that you will use to create the graphical user interface (GUI) of your application.
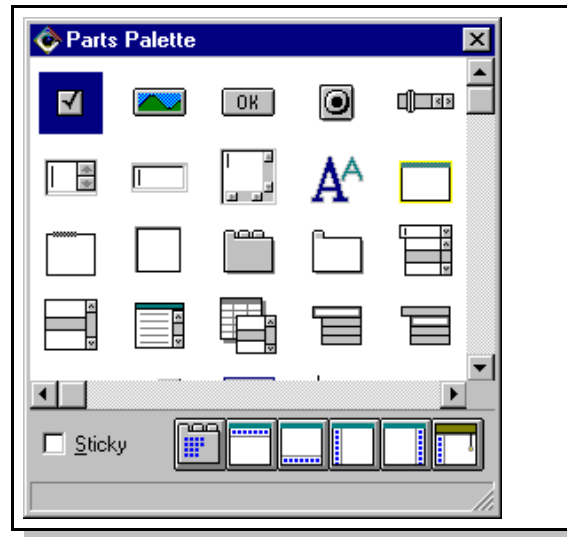
1. Move the mouse pointer over the parts on the **parts palette.** Notice the information area at the bottom of the parts palette. It displays a short description of the part.

Creating a Graphical User Interface

During this exercise you will learn how to create a graphical user interface by creating windows and adding parts to windows. The graphical user interface you are creating now will be the first window of a Customer Inquiry application. It will be used to prompt for a customer number.

When the GUI designer is started for a new project it creates the first design window for you. Notice also that a **Window** part has been added to the project view.

We will use this window as the first window in our Customer Inquiry application. As with all parts the initial part name is generated by the GUI designer with the first few characters indicating the part type. For example **FRA** indicates a Frame Window part.

Closing Windows

Several times throughout this lab you will be instructed to close a design window. In all there are four ways to close a design window. You can use which ever method you prefer:

1. Double click on the system icon in the upper left corner of the window

2. Single click on the system icon in the upper left corner of the window. From the menu choose *Close*.

3. Single click on the **close** button. This is the **X** icon in the upper right corner of the window

4. Pressing **Alt+F4** will close the window that currently has focus System icon

System icon   Close button

Window with canvas

Restore
Move
Size
Minimize
Maximize

Close   Alt+F4
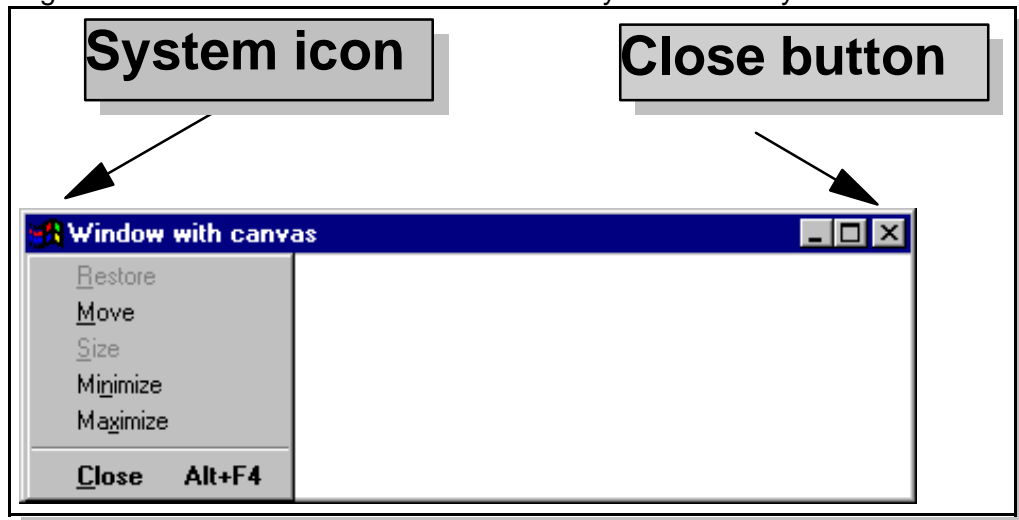
Drag-and-Drop

One method of moving objects in **VARPG** is by the way of dragging an object from one place and dropping it to another. This is referred to as drag and drop. To drag and drop an object click and hold the left mouse button with the mouse pointer over the object then move the mouse pointer so the object is at the desired location and release the left mouse button.

Point and Click

Point and click provides an alternative method to drag-and drop, for creating the user interface. Select a part on the parts palette by single clicking on it. Move the mouse pointer to the new location and click the left mouse button to release the part at that location.

Point-and-click is the preferred method of creating parts in VARPG and is the default.
This method will be used during the labs.

Pop-up Menus

Pop-up menus are provided for practically every object and area in the GUI designer allowing you to perform some action on that object or area. To invoke a pop-up menu position the cursor over the object or area and click and release the right mouse button. To choose a menu item from a pop-up menu single click on the menu item.

A Note About Notebooks

As mentioned earlier the parts palette contains parts that act as templates. Once the part has been created, its attributes such as its part name, can be changed from their default settings. These changes are made by invoking the parts **settings notebook.**

There are three methods of opening a settings notebook:
 1) Double click on the part,
 or
2) right mouse click on the part and choose ***Properties*** from the pop up menu
or
 3) select the part and from the Project menu bar, choose ***Selected→Properties***.

A settings notebook consists of **pages** as indicated by the labelled **Tabs**. You go to a notebook page by clicking with the left mouse button on its tab. Once you made your changes, press the **OK** push button at the bottom of the settings notebook.

Let's change some of the settings for the initial design window:

1. Double click on the **title bar** of the design window (see the tip below). The window part **settings notebook** appears.

> **Tip** *A **Window with canvas** part is made up of two parts. The Window, which consists of the title bar and frame, and the Canvas which is the white area enclosed by the window. Therefore, if you were to double click on the Canvas, you would get the settings notebook. for the **Canvas** part.*

```
Window Part Properties [11]                    ? X
General | Startup | Style | Style 2 | Font | Description |

   Part name                    Custinq

   Title                        Customer Inquiry


        Reset page

   OK      Reset    Cancel    Help
```
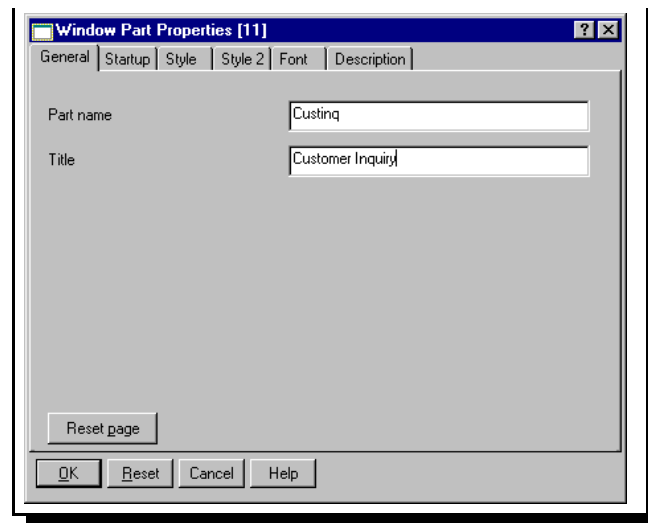
1.
Change the **Part name** setting to **CUSTINQ** and the **Title** to **Customer Inquiry**. Although you can type the part name in upper or lower case, it is always folded to upper case by the GUI designer

2 Click on the **Style** tab of the settings notebook

3 Deselect the **Minimize button** and **Maximize button** check boxes

4 Select **Thick** for the style of the window border

5 Close the settings notebook by pressing the **Ok** push button

Notice the changes to the window. The new window title is displayed, the border style has changed and **Minimize** and **Maximize** icons no longer appear on the title bar.

> **Tip** *To change the label and default text of parts you can also press the **Alt+left mouse button** instead of using the settings notebook. When you have made your change click the left mouse button again. This is called direct editing.*

The next step will be to add a **static text** part to the window to prompt for the customer number to be entered:

1. Find the **static text** part in the **parts palette**

> **Tip** *As you move the cursor over the icons on the parts palette a short description of the part type is shown in the parts palette information area.*

2 Use point-and-click to create a static text part on to the **Customer Inquiry** design window by clicking on a static text part in the parts palette with the left mouse button, moving the pointer to where you want it on the design window with the mouse, and then clicking the left mouse button again.

3 Invoke the settings notebook for this part by double-clicking on it (the text **static text**)

4 Change the **Text** setting on the **General** page to **Enter Customer Number**

5 Close the settings notebook by pressing the **OK** push button.

You might notice that not all of the changed text is displayed. This is because of the default size setting of the **static text** part. To change its size:

1 Click on the **static text** part with the left mouse button to select it. The borders of the part are marked by placing small squares, called sizing handles, around it

2 Move the mouse pointer to the middle of the rightmost three sizing handles. The pointer will change to a double-arrow ⇔ (pointing to the left and right)

3 Click and hold the left mouse button

4 Move the mouse pointer to the right until the entire text of the **static text** part is visible

5 Release the left mouse button

To add the entry field for the customer number:

1. Find the **entry field** part on the parts palette

2. Using point-and-click, create an entry field part on the **Customer Inquiry** window to the right of the static text part

3. Invoke the **settings notebook** for the entry field and

make the following changes to this settings notebook:

Page	Attribute	Value
General	Part name	CUSTNO
Data	Length	7

Close the settings notebook.

Most probably the static text and entry field parts are not aligned with each other. To align these parts with each other:

4. Move the mouse pointer above and to the left of the **static text** part

5. Click and hold the left mouse button

6. Drag the mouse down and to the right so that the two parts are covered by a grey rectangle. This grey rectangle is referred to as a **selection rectangle.**
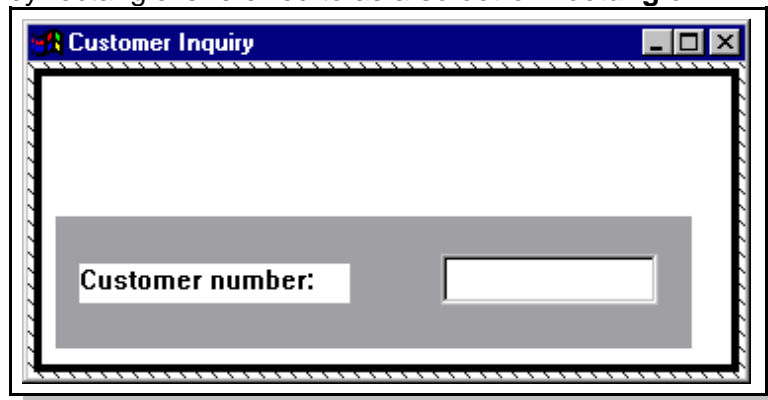
7. Release the left mouse button. The two selected parts will become highlighted with a dark border

8. Move the mouse pointer to the **static text** part

9. Click the right mouse button. The pop up menu for this part appears.

10. Choose the *Align...* choice on the pop up menu. A sub menu appears.

11. Choose the second alignment choice offered. As depicted by the choice icon, this aligns the two parts to an imaginary horizontal axis running through the **static text** part.

NOTE: *When more than one part is selected, invoking the pop up menu for one of the selected part makes that part the **Anchor** part. Any alignment*
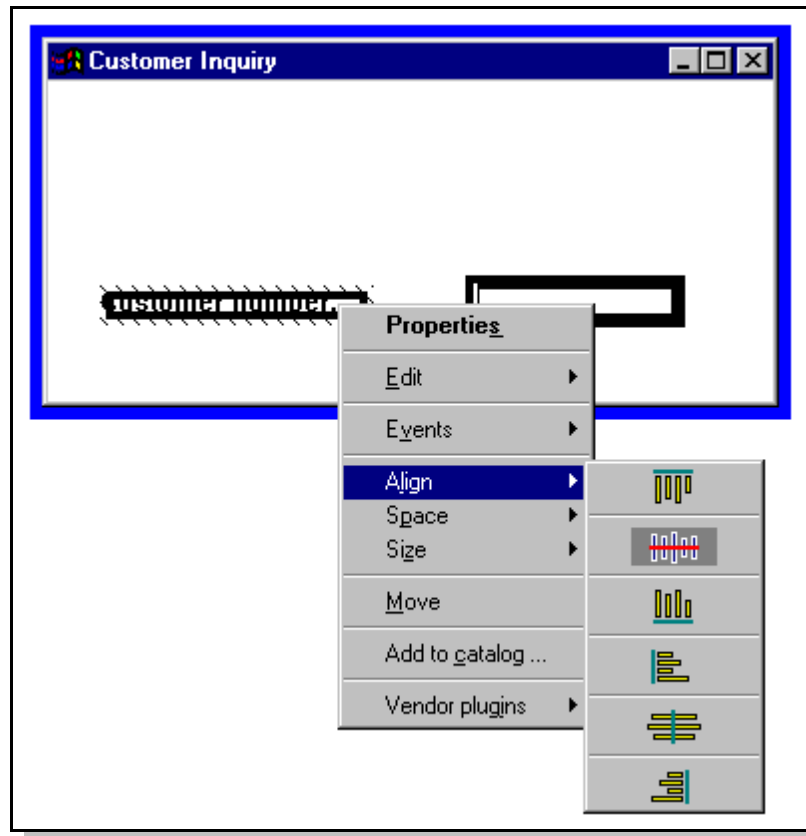
operations will be relative to that part.

To complete this window, we need to add two push buttons. An **OK** push button to process the customer number and an **Exit** push button to end the application. To add these push buttons:

1. Find the **Push button** part on the **parts palette**

2. Use point-and-click to create a push button part on the **Customer Inquiry** window below the **static text** part

3. Invoke the **settings notebook** for the push button part

4. Change the **Part name** setting on the **General** page to **PBOK** and the **Label** setting to **OK**

5. Go to the **Style** page and select the check box for **Default**. This makes it the part that gets selected when the Enter key is pressed.

6. Close the settings notebook

To create the second (Exit) push button, again pick the push button part from the parts palette and put it on your window.

Invoke the settings notebook for the **Exit** push button part by double-clicking on it.

1. Change the **Part name** setting on the **General** page to **PBEXIT** and change the label to **Exit**.

2. Close the settings notebook.

Now, the two **Push button** parts need to be aligned. The left border of the OK **Push button** part must be aligned to the left border of the **static text** part, while the right border of the Exit **Push button** part needs to be aligned to the right border of the **entry field** part. To do the alignment:

1. Move the mouse pointer above and to the left of the **static text** part.

2. Press and hold the left mouse button.

3. Drag the mouse, so that the **static text** and push button part are completely covered by the selection rectangle.

4. Release the left mouse button. The two selected parts will become highlighted.

5. Move the mouse pointer to the **static text.** part, click the right mouse button to get the pop up menu and choose the *Align...* menu choice from the pop up menu

6. Choose the fourth choice offered. This aligns the OK **Push button** part to the left border of the **static text** part.

To align the Exit **Push button** with the entry field:

1. Move the mouse pointer above and to the left of the **entry field** part.

2. Press and hold the left mouse button.

3. Drag the mouse, so that the **entry field and** push button part are covered by the selection rectangle.

4. Release the left mouse button. The two selected parts will become highlighted.

5. Move the mouse pointer to the **entry field.** part, click the right mouse button to get the pop up menu, and choose the *Align...* menu choice from of the pop up menu.

6. Choose the sixth choice offered. This aligns the Exit **Push button** part to the right border of the **entry field** part.

Now, align the two **Push button** parts to each other horizontally in the same way you aligned the **static text** and **entry field.** parts.

Finally, the size of the window has to be changed to contain the parts in it:

1. Move the mouse pointer over the upper right corner of the window. It will become a double-arrow.

2. Press and hold the left mouse button.

3. Drag the mouse to resize the window, until all parts are in the middle of the window. None of the parts created within the **Customer Inquiry** window are moved when the window is resized. This is because parts remain relative to the left and top edges of the **Window** part. You need to be aware of this, since some resizing operations will move parts so they are no longer visible.

The resulting window should look similar to the following:
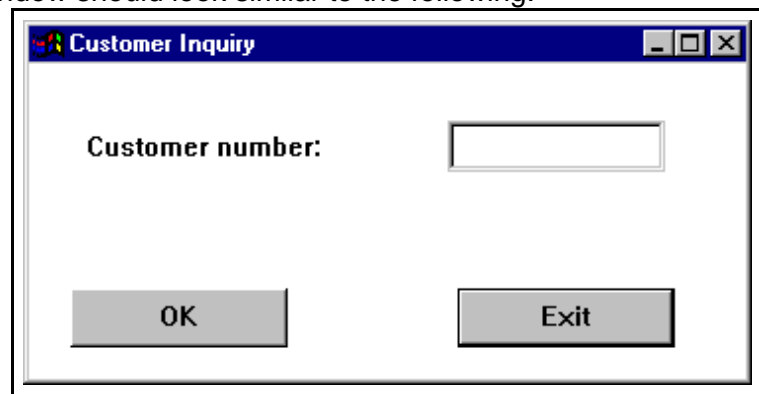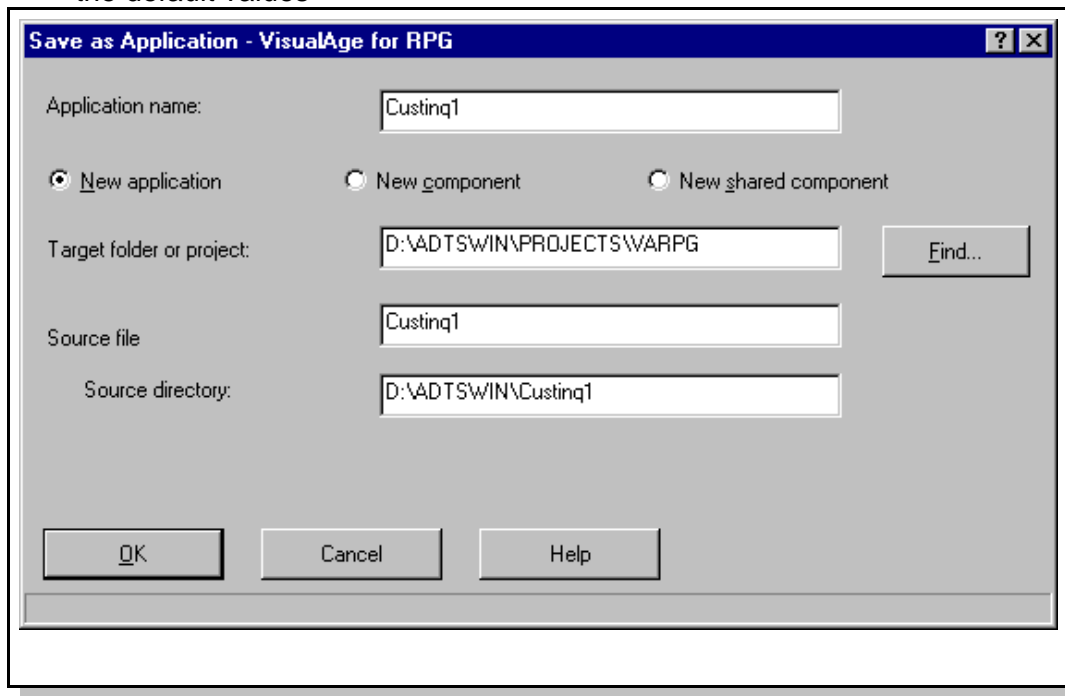

Saving Your Project

Now, that you have finished your user interface let's save it in a new project:

1. Move the mouse pointer to the **Save Project** icon on the tool bar.

2. Click on this icon and save your project. Since this is a new project and has not yet been named (notice the title bar on the GUI designer says **Untitled**) the **VisualAge RPG - Save as Application** window is displayed. This window is used to name your project, indicate in which folder it should be saved, and where the project files should be saved.

Make the following entries on this window:

- In the **Application name** entry field type a title for your application, for example *Custinq1*

- Leave the folders selection at **x:\adtswin\Projects\VARPG**. *(x represents the drive letter which may be different on your system)*

- Then just click in the **Source file** entry field, and the remaining fields get filled with the default values

| Save as Application - VisualAge for RPG	? ✕
Application name:	Custinq1
⦿ New application ○ New component ○ New shared component	
Target folder or project:	D:\ADTSWIN\PROJECTS\VARPG [Find...]
Source file	Custinq1
Source directory:	D:\ADTSWIN\Custinq1
[OK] [Cancel] [Help]	

Press the **OK** push button to save your project. Notice that after the project has been saved, the GUI designer title bar has been updated to show the project name.

This ends Part 1 of this lab. You will be using this project in the next part of this lab so please do not exit the GUI designer at this time.

Creating an Action Subroutine

Now that you have finished designing the fist window of your project, you will add some RPG logic to give it some functionality. In this exercise you will create the **action subroutine** for the Exit push button:

1. If the CUSTINQ design window is not open, open it by double-clicking on its icon in the GUI designer Project View.

2. Select the **Exit** push button using left mouse button. Click the right mouse button to get this parts pop up menu. Choose **Events**→**Press**
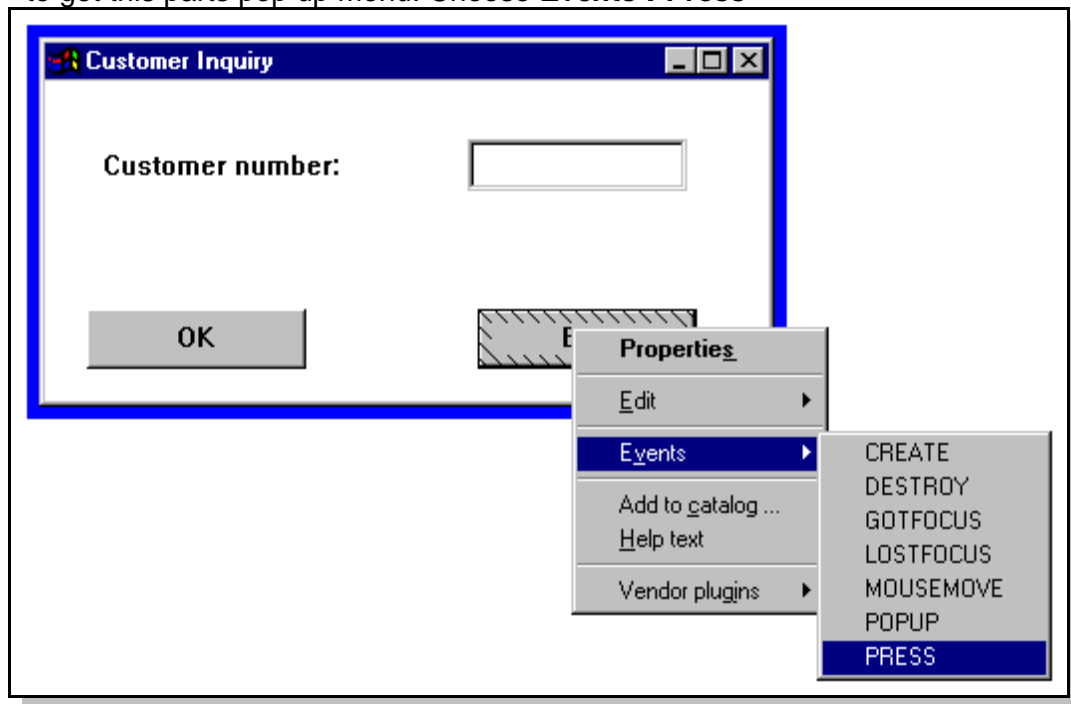
3. This will start the LPEX editor.

When the LPEX editor first appears, you will notice that the **BEGACT** and **ENDACT** statements have already been inserted into the action subroutine. As well, there are a number of comment lines included to encourage documentation. You may change the comments to include any further information that you might think necessary.

Although the framework for the RPG code has been built, the application logic is not

there. It must be added. In this example, the code to terminate the application must be added.

Position the mouse pointer anywhere on the line with the BEGACT operation code and click left mouse button. This will position the cursor.

Choose **Edit→ Insert with Prompt** from the editor menu bar. Tip *You can also perform an Insert with Prompt, by pressing **Ctl+F4***

A prompt window will be displayed. Type in the VisualAge for RPG statement as shown which will cause the program to terminate by setting on the last record indicator, LR.

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
 *
C                   MOVE      *ON           *INLR
```

There is extensive online help available for the VisualAge for RPG language. Before leaving this prompt screen position the cursor back on the **Operation and Extender** field. Press F1 (HELP). A help screen for the current value of the Operation code field is displayed. In this case, help for the **MOVE** operation code is shown.

Close the help window.

Press the **OK** push button to add this line of code to the source.
To end prompting, press the Cancel push button on the Prompt screen.

> **Tip** *Most dialogues that have a cancel push button can also be closed by pressing the **Esc** key.*

You will notice that this line of code was added directly below the BEGACT statement. To save your changes choose **File→ Save**. Close the LPEX editor by single-clicking on its **Close** button.

Let's recap what an action subroutine is: In the Event-driven paradigm, what happens in the program is controlled by the **Events** occurring on the GUI. An Event can be linked to an action subroutine. Action subroutines are easily found in the code when looking at the complete VisualAge for RPG source. An action subroutine begins with the BEGACT operation code and ends with an ENDACT operation code. The logic between these two statements is the business logic that is executed when the event is triggered.

Creating an Action Subroutine for the OK Push Button

Now you will create an action subroutine for the **OK** push button.

This action subroutine will toggle the background colour of the **CUSTNO** entry field from **red** to **pale grey** each time the push button is pressed.

This might not seem to be the most useful code but it will give you the opportunity to experiment with some of the new operation codes, namely:

   SETATR - Set an attribute

   GETATR - Get an attribute

Later on you will add some more meaningful logic for the **Press** event of this push button.

Create the action subroutine:

1. Select the **OK** push button with the left mouse button. Use right mouse button to get the pop up menu for the push button. Choose **Events→Press** which will invoke the LPEX editor.

2. You want to write logic to get the background colour attribute of the entry field. You will do this by using the **GETATR.** Your logic will then determine the new colour to set it to, and use the **SETATR** operation code to change it.

3. This logic will set the background colour to red if it is not currently red, otherwise it will set the background colour to grey. Your logic should look similar to the following figure.

> **Tip:** *VisualAge for RPG statements can be entered in upper, lower or mixed case, including part names and attribute names.*

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     'CUSTNO'       Getatr    'BackColor'  BCOLOR              2 0
C                    If        BCOLOR <> *RED
C     'CUSTNO'       Setatr    *RED         'BackColor'
C                    Else
C     'CUSTNO'       Setatr    *PALEGRAY    'BackColor'
C                    EndIf
```

Here are some editor line commands you can use while creating this code:

- **ALT+L** will mark a line

- **ALT+U** will unmark a line

- **ALT+D** will delete a marked line

- **ALT+C** will copy a marked line

- **ALT+M** will move a marked line Press Enter to insert a new line

RPG IV: The base for the VisualAge for RPG language definition

The VisualAge for RPG language is based on the RPGIV language syntax. Both upper and lower case letters are allowed in the source code. This greatly enhances the readability of the code, as well as the ease of coding. The second RPGIV feature that is notable in the VisualAge for RPG is the 10 character names. Not only does this make the code more legible and the names more meaningful, it also means that the names can be the same as the names used in other AS/400 languages as well as names defined in Data Description Specifications (DDS) without the need to rename them.

Some LPEX Features

Now that you have added the second action subroutine let's look at some of the features of the LPEX editor, so you can later easily find your way around and utilize them.

Token Highlighting

The LPEX editor highlights the **Tokens** (specification fields) of your VisualAge for RPG program source, providing a separate colour for each improving readability.

When you make changes to a line, the token colours updated only after you move your cursor off the line.

To see how token highlighting works:

1. Move the cursor to the Calculation statement you just created to set on LR.

2. Position the cursor to column 7 (right next to the 'C').

3. Type an asterisk (*).

4. Move the cursor off the line and watch what happens. The line where the '*' was typed has become a comment line and its colour changes accordingly.

5. Move the cursor back to column 7 and remove the '*'. Move the cursor off that line of code and the statement is tokenized.

6. The token highlighting changes to reflect that this is a non-commented 'C' specification.

Displaying Types of Lines

Using the LPEX editor. it is possible to have only particular types of source lines displayed at a time. To do this:

1. Choose *View→Show* from the LPEX editor menu bar. The options available on the pull down menu lists the types of line selections that can be made.

2. Choose **Comments**. The LPEX editor now contains only those VisualAge for RPG statements that are comments.

3. Choose *View→ Show all*. All statements types are displayed.

In addition, the choices in the **View** pull-down can be used to include only control specifications, user subroutines, and action subroutines. The **Include** choice will allow the selection of only those lines containing a particular character string. As the LPEX editor is fully-programmable, it is possible to create macros to include/exclude all types of source statements, based on specific criteria.

Syntax Checking

Syntax checking is available for VisualAge for RPG code, and by default will be active. The syntax of the VisualAge for RPG code is checked automatically when a change is made to a line, and the cursor is moved off the line.

When errors are found, they are displayed following the statement with the error. To see how syntax checking works:

1. Move the cursor to the statement containing the **MOVE**.

2. Make sure the LPEX editor is in Replace mode (the mode indicator is directly to the right of the row and column information). Use the **Insert** key to change modes.

3. Create an error by removing the 'E' from the MOVE operation code

4. Move the cursor off of this statement. An error line is displayed after the statement

5. Correct the error by typing the character 'E' back into the operation code 'MOVE'.

6. Move the cursor to the next line. The message area is cleared.

Using Format Lines

The format line is at the top of the LPEX editor window, just above the first statement. A format line is used to help keep track of the columns in a particular

specification line. The content of the format line can vary to reflect the particular type of specification being keyed such as F specs, C specs, D specs and so on.

To display a format line:

1. Position the cursor to any uncommented Calculation specification line by clicking on the line with the left mouse button, or by using the arrow keys and clicking the left mouse button.

2. Choose *View*→*Refresh format line*.

3. The format line changes to reflect an VisualAge for RPG Calculation specification, since the cursor was on a C-specification when the format line was requested.

> **Tip** *The format line can also be refreshed by pressing **Ctrl+R**. with the cursor on a statement.*

1. You can move the cursor right or left with the arrow keys to go from character to character, or with the Tab key to go from field to field. An indicator on the format line moves with the cursor to show in which column the cursor is positioned. Try moving the cursor and watching the indicator on the format line.

2. Move the cursor to a comment line (an asterisk in column 7).

3. Choose the *View*→*Refresh format line* (or press Ctrl+R). A format line for comment specifications appears, since the cursor was on that type of specification when you requested a refresh.

4. To select a format line for any specification you want, choose the *View* menu bar choice, then choose *Select format line*. The **VisualAge for RPG Format Line Selection** window appears.

5. Select the *C-Calculation* radio button, and then press the **OK** push button. The format line changes to reflect a calculation specification.

Now that you have created the push button logic, and explored some of the features of the LPEX editor., save your work by choosing *File*→*Save* from the LPEX editor menu bar. Close the LPEX editor by clicking on its **Close** button.

The next step is to build your application.

First close the design window (the **Customer Inquiry** window) by positioning the

mouse pointer anywhere on the window title bar and click with the right mouse button. From the pop up menu choose **Close**.

Building the Application

Before the application can be run, it must be built:

1. Choose **Project→Build→Windows NT/95**

   or press the **Build WIN32 push button** on the toolbar

1. The **VisualAge for RPG - Save project** window appears. Press the **Save** push button to save your project.

2. A status window will now appear to indicate that the project is being built. When the build has completed, another window will appear to indicate if the build was successful or not.

Compile Time Errors?

If you have compile time errors, the **Error List** window will be displayed. The error list window shows one line for each error found during the compile. Notice that the error number (e.g. RNF7030E) is followed by a character indicating the severity of the error. **I** type errors are informational and can usually be ignored. **E** errors must be fixed for a successful compile. To locate the line of code causing a specific error, double click on an error line in the Error List window and the editor will position your RPG source to the offending statement.

Correct any errors and recompile until you have a successful build.

Running the Application

Once you have a successful build you are ready to run the application. There are two ways to do this:

1. From the **Project** pull-down menu choose **Run** or

   press the **Run push button** on the toolbar

2. The window you designed will be displayed.

3. Press the **OK** push button and check whether your logic is right and the background colour of the Entry field changes.

4. To terminate the program press the **Exit** push button.

Lab Summary

This concludes the first lab.
Congratulations your first **VisualAge for RPG** application is completed:

You have learned:
- How to create and design a **GUI window**
- How to create **action subroutines**
- How to **Get/Set attributes** in the GUI
- How to **build** a VARPG application
- How to **run** the application

Now you are ready to build a VisualAge application that accesses data on the AS/400.

Just experience how easy it is.

Go ahead to the next Lab.

Enhance Your Application with DB2/400 Access

Exploring the GUI Designer and Accessing the AS/400

During this exercise you will work on the application that you created in the previous lab. The following tasks will be covered in this lab:

⬥ Adding a Window to an existing application

⬥ Referencing fields in a DB2/400 database

⬥ Performing more **Alignment**, **Sizing**, and **Spacing**

⬥ Using external described files in a VARPG application

⬥ Accessing DB2/400 files

As a result of this exercise you will create a window that is used to display customer information given the input from the window you created in the previous lab.

What You Should be Able to Do

As a result of this exercise you will customize the GUI designer to suit your needs. Also, you will be able to access an AS/400 to:

⬥ Use existing DB2/400 field definitions

⬥ Use externally described AS/400 files in your VisualAge for RPG program

⬥ Read data from the AS/400

⬥ Read and write data from/to a window

⬥ Build a VisualAge for RPG application accessing AS/400 data

⬥ Run a VisualAge for RPG application accessing AS/400 data

Adding a New Window to your Project

During this exercise you will create a new **Window with Canvas**, change its attributes and add parts to it. This window will display the customer information for the customer whose number was typed in the **CUSTINQ** window:

To create the new window:
In these steps we will create a new window that will display detailed information about the customer number selected from the first window:

1. Find the **Window with Canvas** part on the parts palette and position the mouse pointer on it.

2. Use point-and-click to create a new window. Windows are created by placing them on the Project view.

3. A new design window will appear and an icon is placed in the Project View.

To change the settings for the new window:

1 Double click on the window title bar to open its **settings notebook** and make the following changes:

| Page	Attribute	Value
General	Part name	CUSTINFO
	Title	Customer Information
Startup	Open immediately	OFF

2 Close the settings notebook.

Using an AS/400 File as a Field Reference File

During this exercise you will see how to add entry fields to a window using an AS/400 file as a **Field Reference File** using the **Define Reference Field** function of VisualAge for RPG. You will be using a file on the AS/400 that contains the necessary fields required for the **Customer Information** window.

Adding the Entry Fields

For the field reference file, a file named **CUSTOML3** will be used. It is located in library **GUIDES2.** To invoke the Define Reference Fields dialogue choose *Server→Define Reference fields* from the menu bar.

On the Define Reference Fields dialogue:
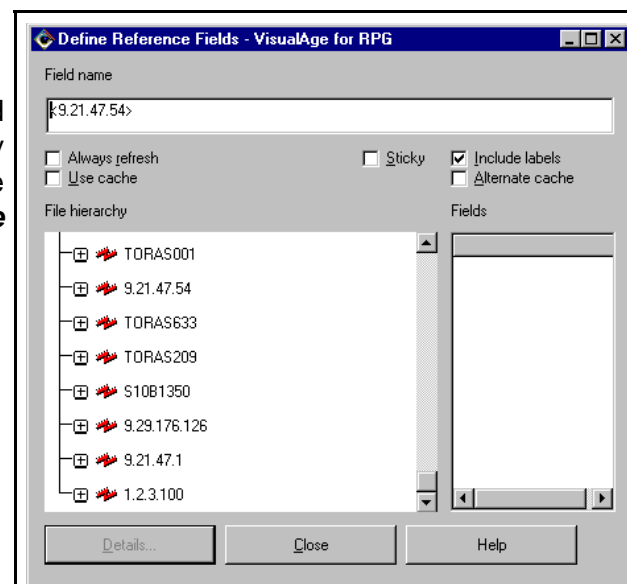
1. Expand the server name as provided by the instructor in the server tree by clicking on the **+** sign next to its name in the **File hierarchy** list. As this is the first time you are accessing this server, the **VisualAge for RPG - Logon** dialogue appears asking you for your AS/400 User ID and Password.

2. Type your

userid:	**VARLABxx**
password.	**VARLABxx**

   where **xx** is your workstation number, for example if your workstation number is **50** use:

<div align="center">

VARLAB50

</div>

3. Press the **OK** push button to log on to the AS/400. If the log on is successful, the libraries of your initial library list are displayed (*LIBL).

4. Find the library GUIDES2 in this list and expand it by clicking on the '+' sign. The logical and physical files in this library are displayed. If the library GUIDES2 does not appear in the list of libraries, it is not in your library list. In that case, type GUIDES2 in the entry field at the top of this window next to the server name, and press enter. (e.g. <TORAS016>GUIDES2)

5. Click on the **'+'** next to **CUSTOML3**. A list of the record formats for this file will appear.

6. double click on the only available format name **CUSTOM01**. All fields in this record format are shown in the **Fields** list.
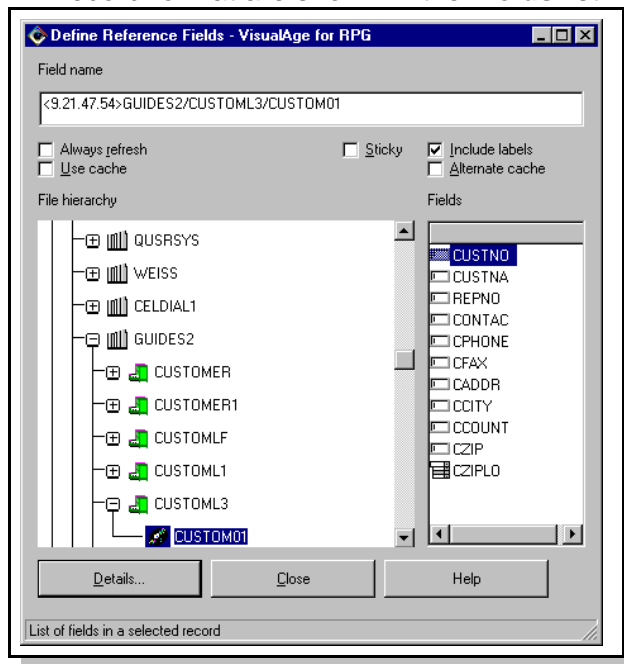
7. Double click on the first field **CUSTNO**. The **VisualAge for RPG - Database Field Description** window appears with detailed information about the field.

8. Press the **OK** push button to close this window.

You will now create the necessary entry field parts for the **Customer Information** window using some of the fields of the **CUSTOM01** record format as references:

1. Rearrange the **Customer Information** window and the **VisualAge for RPG - Define Reference Field** window so that you can see them both.

2. Move the mouse pointer over the **CUSTNO** entry in the **Fields** list.

3. Click on the **CUSTNO** entry field, and then point to the **Customer Information** window where you want to place the **CUSTNO** field, click there and the **entry field** plus **static text** containing the header text will be created.

   The **Static Text** part will be automatically created with the field if the COLHDG DDS keyword is defined and the **Include Labels** option is checked.

4. Double click on this new **entry field** part to invoke its **settings notebook**.

5. Go to the **Reference** page of the settings notebook. This page shows from which AS/400 database file this field is referenced.

6. Go to the *Style* page of the settings notebook and select the **Read only** check box to prevent the customer number from being changed.

7. Go to the *Data* page of the settings notebook and verify, that these attributes have been adopted from the referenced field.

8. Close the **settings notebook**.

TIP: Several fields can be created from the Define Reference Field window at one time by selecting each field that is required with the **CTRL key** pressed, and then using **point and click** to move the fields to the design window. The point and click icon will indicate several fields are being created. An example of the resulting window is displayed below. Note that the labels on your window may not exactly match the diagram below.

Create the other entry field parts for the **Customer Information** window in the same way using the following AS/400 fields as a reference:

- ◆   CUSTNA
- ◆   CONTAC
- ◆   CPHONE
- ◆   CADDR
- ◆   CCITY
- ◆   CZIP
- ◆   CZIPLO
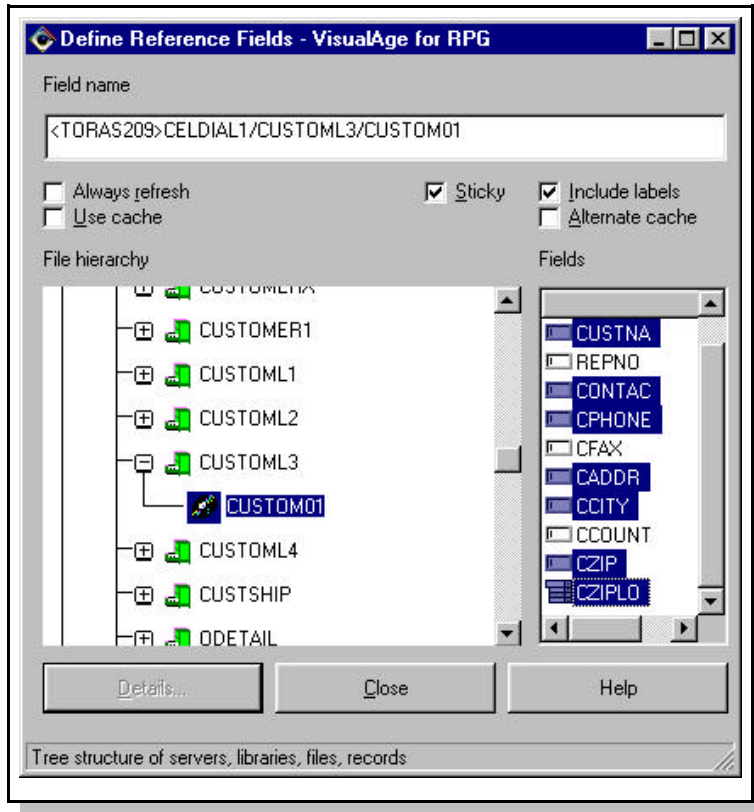
Your design window will look similar to the following picture:
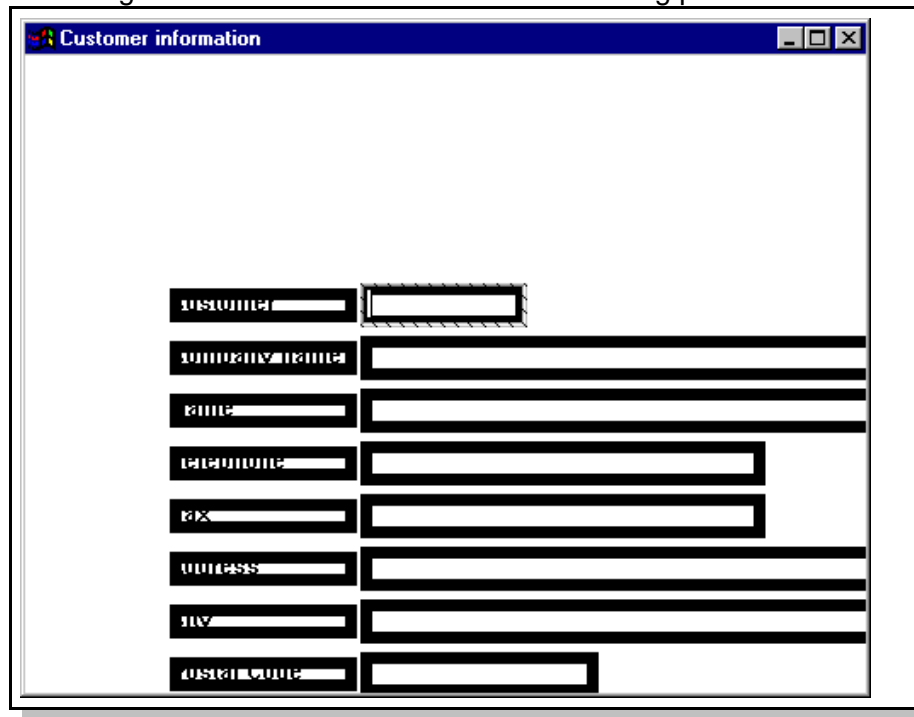
Note that the part created from the ZIP location field **CZIPLO** is not an entry field part, but a **Combination box** part. This is because the **VALUES** DDS keyword was specified for this field. You can check this by looking at the details for this field (double click on **CZIPLO** in the **Fields** list of the **VisualAge for RPG - Database Reference Field** window).

Close the **VisualAge for RPG - Database Reference Field** window by pressing the **Close** push button.

More Alignment, Sizing, and Spacing

Feel free to skip some of the alignment and spacing exercises suggested here if you feel comfortable with using these tools.
Now that all the parts required to make up the **Customer Information** window have been created, they still need to be arranged within the window. First, you will space and realign all static text parts containing the description of the information shown in the **Customer Information** window to the same vertical axis:

1. Move the mouse pointer above the static text part **Customer Number** and to the left of all of static text parts.

2. Press and hold the left mouse button and move the mouse pointer down and to the right to create a selection rectangle so that it covers all of the static text parts.

3. Release the left mouse button. All static text parts become highlighted.

4. Get the pop-up menu of the **Customer Number** static text part by clicking on the right mouse button with the mouse pointer located somewhere over this part.

5. Choose *Align* and then the fourth alignment choice offered. All static text parts are aligned to the left.

Tip: *In case you selected the wrong alignment option, and the selected parts are overlaying each other, you can use the **Undo** function in the GUI designer to restore the parts to their original position. To undo the alignment, choose **Edit** from the menu bar, then choose **Undo**.*

The next few steps will even out the spaces between the static text parts **Customer Name Customer Contact , Phone Number,** and **Address:**

1. Select these static text parts in the same way you just did for vertical alignment.

2. Invoke the pop-up menu of one of the parts.

3. Choose *Space* and then the fourth menu item. This will even out the spaces between the selected parts.

Now you will align each entry field part with its corresponding static text part:

1. Draw a selection rectangle around the **Customer Number** static text part and the **CUSTNO** entry field part.

2. Invoke the pop-up menu for the **Customer Number** static text part.

3. Choose *Align*, then the second alignment choice. The selected parts will be aligned to an imaginary horizontal line running through the centre of the **Customer Number** static text part.

4. Repeat these steps for the **Company Name** static text part and the **CUSTNA** entry field part as well as for the **Address** static text part and the **CADDR** entry field.

5. The combination box requires some consideration before aligning. Click on the combination box and you will notice that selected area includes the entry field portion and the drop down portion of the combination box. To align the

combination box with the **ZIP Location** Align the two parts and then stretch the combination box vertically so it will hold 2 or 3 entries.

6. To align the **CONTAC** entry field part as well as the **CPHONE** entry field part to their corresponding static text parts you can use the **Spacing** tool again.

7. Draw a selection rectangle around the **entry field** parts **CUSTNA**, **CONTAC**, **CPHONE**, and **CADDR** parts.

8. Get the pop-up menu of any of the selected parts and choose *Space* and then the fourth choice offered. The spaces between the entry field parts are evened out.

Now, align all entry field parts except **CCITY** to one vertical axis:

1. Draw a selection rectangle around all the entry field parts with the exception of the **CCITY** part

2. Choose the fourth choice out of the **Align** menu item from the pop-up menu of the **CUSTNO** entry field part.

3. If the **CCITY** entry field part overlaps the **CZIP** entry field part, move the **CCITY** entry field part to the right of the **CZIP** entry field part.
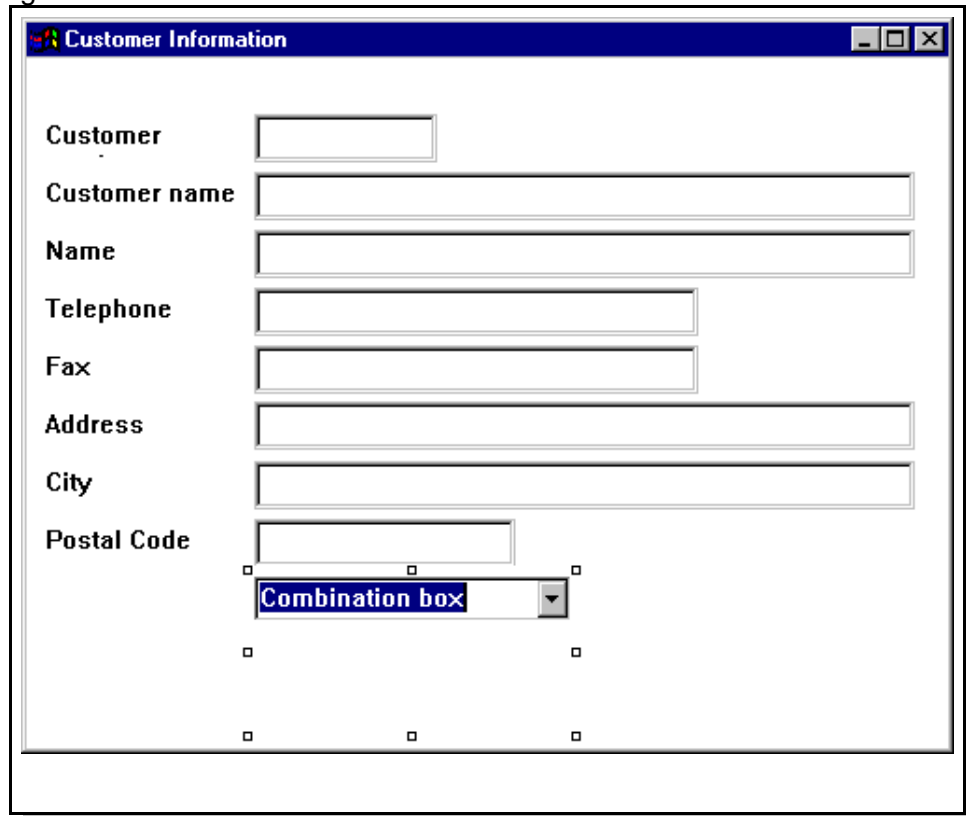

To change the horizontal spaces between the entry field part **CZIP** and the **Combination box** part **CZIPLO** on one hand and the entry field part **CADDR** on the other hand to be equal to the space between the entry field parts **CPHONE** and **CADDR**:

 1 Draw a selection rectangle around the entry field parts **CPHONE**, **CADDR**, **CZIP**, and **CZIPLO**.

 2 Get the pop-up menu of any of the selected parts and choose *Space* and then the fifth choice offered. This uses the space between the entry field parts **CPHONE** and **CADDR** and applies it to the other selected parts so they are evenly spaced.

Finally, the entry field part **CCITY** and the **static text** part **ZIP Location** need to be horizontally aligned to the entry field parts **CZIP** and **CZIPLO**:

1. Draw a selection rectangle around the entry field parts **CZIP** and **CCITY**.

2. Invoke the pop-up menu of the **CZIP** entry field part.

3. Choose *Align* and then the second alignment choice offered.

When you are done your window should look similar to the following. Don't worry if it is not exactly the same. We just wanted to show off more of the aligning and spacing tools.

Changing the Size of Some Fields

You might also want to change the size of some entry field parts, that are supposed to display strings, that are too long to be displayed using the default size of the entry field part on the screen. You will do this now for the entry field parts **CUSTNA**, **CONTAC**, and **CADDR**

Even though when using the Define Field Reference feature, that sizes the fields according to their definition, you might want to go through this exercise to get familiar with the **sizing tools**.

1. Click on the **CUSTNA** entry field part to make it the active part.

2. Drag the middle of the right sizing handles until the field size is enlarged to the right border of the **CCITY** entry field part.

3. Use the arrow keys to get to the **CONTAC** entry field part.

4. Press and hold the **CTRL** key and press the **space bar** to select it.

5. Use the arrow keys to get to the **CADDR** entry field part.

6. Select this part also by using the CTRL key and the space bar.

7. Invoke the pop-up menu for the **CUSTNA** entry field part.

8. Choose *Size* and the first choice offered.   The size of the entry field parts **CONTAC** and **CADDR** will be enlarged to the size of the **CUSTNA** entry field part.

End of Alignment, Sizing, and Spacing exercise

Add Ok push button to Customer Information window

Now, you will add a **Push button** part to the **Customer Information** window, that allows the user to close this window and return to the **Customer Inquiry** window.
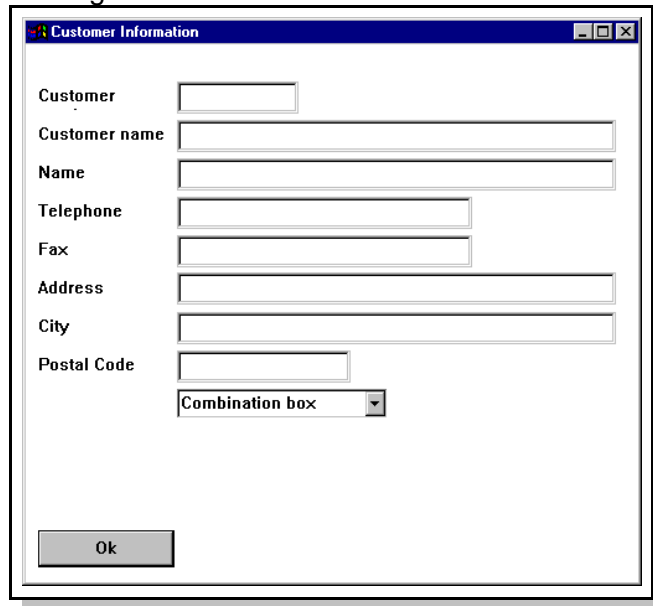
1 Move the mouse pointer to the bottom corner of the **Customer Information** window, so that it becomes a double-arrow.

2 Press and hold the left mouse button and resize the window so it can contain the additional Push button part.

Now let's add the push button:

1. Use point-and-click to create a push button below the **ZIP Location** combo box.

2. To align the push button with the static text part above, select **Postal Code** static text part and the **Push button**.

3. Invoke the pop-up menu of the static text part

4. Choose *Align* and then the fourth alignment choice offered. The Push button part gets vertically aligned to the static text parts of the window.

5. Double click on the Push button part to get the **settings notebook** for this part.

6. Change the **Part name** setting to *INFOOK* and its **Text** setting to *OK*.

7. Go to the **Style** page of the settings notebook.

8. Select the **Default** check box. This will cause this push buttons' **Press** event
   action subroutine to be invoked when the **Enter** key is pressed.

9. Close the settings notebook.

The resulting window should look similar to the following illustration:

To save the changes to your project, press the **Save Project** icon on the tool bar
(the third icon if you have not customized it) and your project will be
saved.

Adding More Logic to your Project

The objective of this exercise is to add the necessary logic to fill the Customer
Information window you have just designed with data from the AS/400.

You will perform the following steps:

1. Add a File specification to your program

2. Change code for the **OK** push button **PRESS** event on the **CUSTINQ** window to display the **CUSTINFO** window

3. Create code for the **OK** push button **PRESS** event on the **CUSTINFO** window to close it

4. **Build** the application and **run** the application

Let's get started by adding a File specification to your program source:

1. Choose *Project→Edit source code* or

   use the Edit push button on the toolbar

2. Scroll to the **H** specification

3. Position the mouse cursor on the H specification

4. Choose *Edit→Insert prompt* from the LPEX menu bar.

5. Press the **Select** push button and select the **F** specification radio button. Press the **OK** push button and you will see the prompt window for a File specification.

6. Fill in the following values:

```
    File name              CUSTOML3
    File type              I
    File designation       F
    File Format            E
    Record address type    K
    Device                 Disk
    Keywords               Remote
```

Note that with the exception of the **REMOTE** keyword all entries are the same as RPGIV on AS/400. Save your changes by choosing *File→Save* from the LPEX menu bar. Close the editor.

Changing the OK Push Button Action Subroutine

You will now add the logic for the **Press** event of the **OK** push button on the **CUSTINQ** design window. Before you go ahead and code the **real** logic for this

action subroutines, you may want to remove the 'colour' code from the previous lab **Lab 1**. That code was only created to demonstrate events and action subroutines. If you prefer the entry field to change colour just leave this code in. The RPG logic you will be coding now should perform the following functions:

◆ Get the customer number from the **CUSTNO** entry field on the **CUSTINQ** window

◆ Retrieve customer detail from the AS/400 database file

◆ Display the second window **CUSTINFO**

◆ Update the **CUSTINFO** window with the data just read

Let's get started:

1. Using the pop-up menu from the **OK** push button, bring up the action subroutine for the **PRESS** event. You should see the code you created in an earlier lab that changed the entry field colour.   At this point you can remove this colour code.

2. The first step is to get the customer number entered in the **CUSTNO** entry field.

3. There are 2 methods of getting this value. You could use the **GETATR** operation code to get the TEXT attribute of the CUSTNO entry field, or do a **READ** of the **window** to get the TEXT attribute of all entry fields. In this case let's use the **READ** operation code:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C                   READ      'CUSTINQ'
```

With the value of the field **CUSTNO** you now have to chain to the database file **CUSTOML3.**

Add the **CHAIN** statement with an indicator in the **HI** value column (record not found).

You might want to use F4 for prompting to get the right column for this indicator, or try out the syntax checker in the LPEX editor.

Your **CHAIN** statement should look like this:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     CUSTNO        CHAIN     CUSTOML3                          50
```

Your completed code should look similar to the following:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
C     PSBOK        BEGACT    PRESS          CUSTINQ
C                  Read      'CUSTINQ'
C     CustNo       CHAIN     CUSTOML3                              50
C                  If        *IN50 = *OFF
C                  ShowWin   'CustInfo'
C                  Write     'CustInfo'
C                  EndIf
C                  ENDACT
```

This completes this action subroutine for now. To save your changes, choose **File→Save** from the LPEX editor menu bar. Close the LPEX editor.

Adding an Action Subroutine to the CUSTINFO Window

In this exercise you will add logic to hide the **CUSTINFO** window when its **OK** button is pressed. The instructions in the following exercises will not be as detailed, because you are now familiar with the **VisualAge for RPG GUI designer** development environment. Perform the following steps:

1. Create an action subroutine for the **PRESS** event of the **OK** push button on window **CUSTINFO**.

2. Add a statement to set the **Visible** attribute for the window **CUSTINFO** to off.

> *Tip: All of the VisualAge for RPG manuals are on-line and can be accessed by choosing **Help** from the GUI designer menu bar.*

1. Another way to get information on part attributes is to select the part you are interested in the parts palette and pressing **F1**. This will display a help window with details on this part along with allowed attributes and sample code.

2. Save this action subroutine

You might want to close the design windows at this point. To close a design window, position the mouse pointer on the windows title bar and click the right mouse button. From the pop-up, choose **Close**.
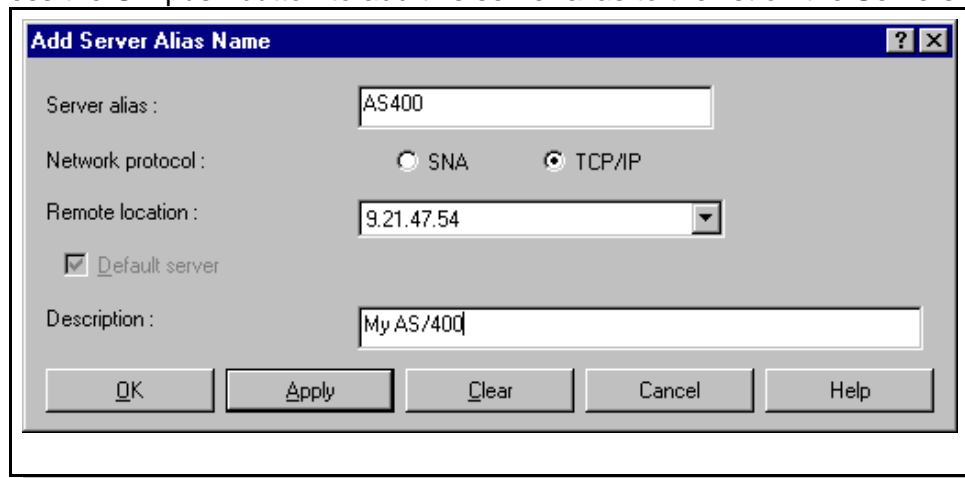
Building the Application

Before you build the application you have to specify where the compiler can find the AS/400 information it will need at compile time. In this case you have to define on which server (AS/400) the file used your program is located.

To enter this information, choose *Server→Define AS/400 information* from the GUI designer menu bar. The **Define AS/400 Information** notebook will be displayed. This notebook has several pages on which you define information regarding your server and files.

For this lab you need to define a server.

1. On the **Servers** page, press the **Add...** push button. The **Add Server Alias Name** dialogue appears. Type an alias name in the **Server alias** entry field. From the drop-down combination box, select the same server you used when using the Define Reference Fields function.

2. Press the **OK** push button to add this server alias to the list on the Servers page.

3. Close the **Define AS/400 Information** dialogue by pressing the **Ok** button.

Time to build the project:

1. Choose *Project→Build→Windows NT/95* from the GUI designer menu bar.

2. If you have made changes to any of the design windows, you will be prompted to save or discard your changes. Answer **Yes** to this message.

3. A status window will be displayed indicating that the build is in progress.

4. Once the build is complete a completion window is displayed indicating if the build was successful.

Viewing the Compile Listing

Recall that the VisualAge for RPG compiler is resident on the workstation. Therefore, the compile listing is also resident on the workstation. At any time, it is possible to view the files that make up your project including the compile listing.

To display the compile listing for this latest build choose **Project→Browse compile listing**. The compile listing will now be displayed. Note the similarities between the VisualAge for RPG compile listing and the compiler output of the AS/400 RPG compilers.

Testing Your Application

Let's see if the application runs:

1. There are two ways to run your project. Choose from one of the following:

    ◆ Choose **Project→Run** from the GUI designer menu bar

    ◆ Press the **Run** icon on the tool bar (the running man).

    The first window should be displayed.

2. **Type**

<div style="border:1px solid;">

<u>0010100</u>

</div>

as the **<u>customer number</u>**.

3. Press the **OK** push button. This should bring up the second screen with the customer information filled in.

4. If you type in a **wrong** customer number, nothing will happen since you did not code any error handling.

5. Go to the **CUSTINFO** window and press the OK push button; the window should be hidden.

6. Press the **Exit** push button on the **CUSTINQ** window to terminate the application.

VisualAge for RPG Lab Summary

See how easy it is to create a Client/Server application by using your RPG skills to produce these GUI applications   your endusers are begging for .

This concludes the VisualAge for RPG hands on lab. We hope you **enjoyed** the experience. Using what you have learned from this lab you can build on this experience and create your own **VisualAge for RPG** applications by exploring many of the other features of **VisualAge for RPG** that were not covered here.

For more details, including **downloadable** sample programs, tips and techniques, and service and support information, be sure to visit the **VisualAge for RPG** web site at

www.software.ibm.com/ad/varpg.

Have fun with VisualAge for RPG!

If time permits feel free to go on to the next Lab.

Using Subfiles

Working with Subfiles

In this lab you will enhance your application by creating a new component with a subfile part. During this lab you will:

- ◆ Create a new component
- ◆ Create a subfile and add fields to it
- ◆ Write the VisualAge for RPG logic to write records to the subfile part
- ◆ Build the application and run the application

What you should be able to do

As a result of this lab you will be able to write applications that use subfiles by:

- ◆ Creating an application
- ◆ Creating a subfile
- ◆ Writing VisualAge for RPG logic to write data to the subfile

Creating a new application

In this lab, you will create another VisualAge for RPG component that will display in a subfile a list of customers.

To begin, start the GUI designer if it is not already running. If it is running, choose **Project→ New** from the menu bar to open a new project.

You will add a subfile part that will display records from a logical file on the AS/400. This file is named **CUSTOML1**, and will be in library **GUIDES2**. The record format name is **CUSTOM01**. The subfile you will create displays data from the following fields:

```
CUSTNO
CUSTNA
CCITY
CADDR
```

Perform the following steps to add the subfile part:

1. Change the title of the design window to *Customer list* (or something equivalent).

2. Name the design window *CUSTLIST*.

3. Locate the **Subfile** part on the parts palette and use point-and-click to create a subfile on the design window. Position and size the subfile as required.

4. Change the name of the subfile to *SFL1*.

Now you will add fields to the subfile. There are three ways to add fields to a subfile part:

   a. by creating entry fields from the parts palette.

   b. by opening the Subfile parts settings notebook. and going to the **Field list** page.

   c. by creating fields from a reference file.

For this exercise, we will use the Define Reference Fields method. Choose *Server→Define reference fields* from the GUI designer menu bar to display the field list for file **CUSTOML1** in library **GUIDES2**. Use **point-and-click** to add the following fields to your **subfile** part:

    ◆ CUSTNO

    ◆ CUSTNA

    ◆ CADDR

    ◆ CCITY

If you want to change the order of the fields in the subfile, open the settings notebook. for the subfile, go to the **Fields** page and use the **Move up** and **Move down** push buttons.

Once you have completed the previous steps, you should have a design window that looks similar to the following:
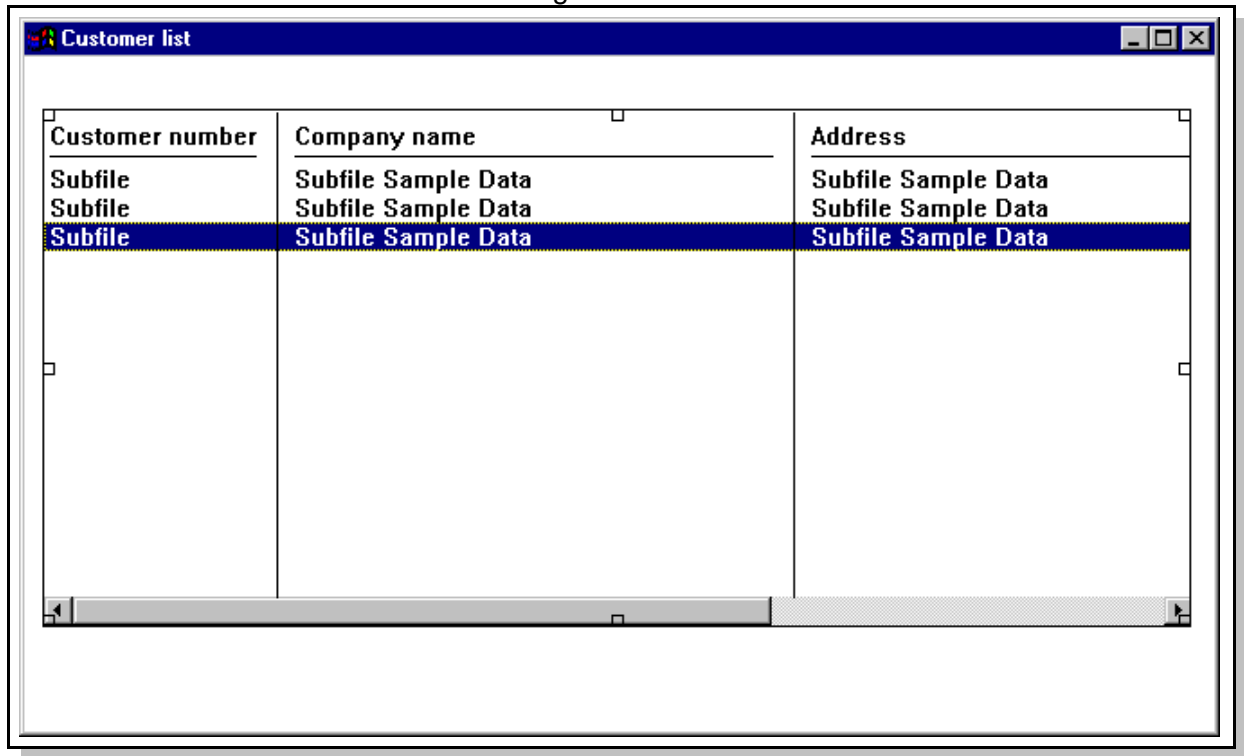
Adding the Component Logic

Now you need to add code to fill the subfile. In this example, you are going to fill the subfile when the window is created. Therefore, you need to create an action subroutine for the windows **CREATE** event.

To create the action subroutine, use the right mouse button to invoke the pop-up menu for the window (click on the windows title bar). Choose *Events→Create*.

When the template action subroutine is displayed, add the following statements between the **BEGACT** and **ENDACT** statements to fill the subfile:

```
CL0N01Factor1+++++++Opcode(E)+Factor2+++++++Result++++++++Len++D+HiLoEq
 * Read a record from database
C                   Read      Customl1                               99
 * Do until end of file
C     *in99         DowEq     *OFF
 * Add a record to the subfile
C                   Write     SFL1
 * Read a record from database
C                   Read      Customl1                               99
C                   EndDo
```

Now, you must add a **File** specification to define the database file that will be used to fill the subfile.

Adding the File Specification

1. File specification must come before any **C** specifications and after any **H** specifications, so scroll to the top of your program source.

2. Enter the following File specification for the CUSTOML1 logical file:

```
FFilename++IPEASFRlen+LKlen+AIDevice+.Keywords+++++++++++++++++++++++++
FCustomL1  IF   E          K DISK    REMOTE
```

The following values were used to define this 'F' specification:

    Customl1   - The file name used by the program
    I   - File is Input only
    F   - File is being processed as Full-Procedural
    E   - The file is Externally described
    K   - The file is being processed by Key
    DISK   - This is a Disk file
    REMOTE   - The file resides on the AS/400

> **Tip** *You might want to add the file keyword **BLOCK(*YES)** in applications that use SETLL or CHAIN. This will allow the RPG compiler to define blocking even if these operation codes are used to access the file.*

Defining AS/400 Information

Before you can build the application, you must indicate which file on the AS/400 is being used, and on which AS/400, or Server, the file resides. Do this by entering information in the **Define AS/400 information** notebook. Note that you must define this information for each VARPG project that accesses the AS/400 database.

Defining the Server

Display the **Define AS/400 information** window by choosing ***Server→Define AS/400 information***, from the GUI designer menu bar. On the server page, press the ***Add...*** push button. On the ***Add Server Alias Name*** type any value for the ***Server alias*** name, and from the ***Remote location*** combination box select the server name provided by the instructor. Press the ***OK push button to save your changes.***

Building the Subfile Component

Before building this component you must first save it. Choose *Project→Save*. The **Save As** window appears. For the **Application Name** specify *Customer List*. Change the **Source file and Source directory** to COMPLIST. Press the **OK** push button to save the project.

Now build the project by choosing *Project→Build→ Windows NT/95*.

If the project built successfully test it by choosing *Project→Run*. The subfile should be displayed with a list of all customers from the data base.

Customer number	Company name	Address
0010100	Meridien Electronics Limited	10423 S.E. 30th Place
0010200	Royal Hardware Supplies	Maple View Plaza, 256 New St
0010300	Webster Appliances	7350 Miramar Road, Suite 247
0010400	ProLine Building Supplies	73 Marchwood Road
0010500	Donnamora Construction	Woodbridge Plaza, 276 Simco
0010700	Universal Communications Ltd.	720 Harrison Street
0010800	Baker Electronics	17150 Koll Center Parkway
0010900	Village Telephone	2752 Betsy Ross Drive
0011000	Bayview Direct Sales	201 Bloor Street East
0011100	BelAir Communications incorp.	302 Washington Road
0011300	Burnham Trading Inc	91 Baseline Road
0011400	Calderone Imports	702 S.W. 15th Street
0011500	The Communications Specialists	3011 E. Georgia St.
0011600	Sudbury Radio and TV	7 North Road
0011700	Christies Electronics Limited	70223 Agoura Road, Suite 13

.

> **Tip** *Getting tired of entering your userid and password each time you do a build? You can define a userid and password in the GUI designer so you are not prompted every time. To define a userid and password choose* **Server→Define server logon**. *When the* **Define Server Logon** *dialogue appears press the* **Add** *and fill in the information. This information is defined for the workstation and needs to only be defined one time; not once for each application*

Lab summary

This Lab guided you through:
◆ Creating a new application

◆ Creating a subfile

◆ Filling the subfile with AS/400 data

You might have noticed:

Amazingly, GUI subfiles are even easier to program than 5250 subfiles

This ends this lab.

Thanks for joining us in the *VisualAge for RPG* Lab.

Enjoy the rest of the conference