Name: **InventoryRecordsGUI**

Description: **Chapter 14 – Programming Exercises 4a, b, and c**

**Enter and save inventory records – View saved inventory records**

Write a C# GUI application that contains three different Forms. Form1 allows the user to enter data for items that are for sale and then saves those data items to a text file when a Button is clicked. Name the text file “Inventory.txt”. Form2 allows the user to enter an item number and then reads the “Inventory.txt” file and displays the data items associated with that specific item number when a Button is clicked. Form3 displays all records contained in the “Inventory.txt” file when a Button is clicked. On Form1, the “Save to File” button is to be designated as the Accept button and the “Exit” button is to be designated as the Cancel button. On Form2, the “Find Item” button is to be designated as the Accept button and the “Close Form” button is to be designated as the Cancel button. On Form3, the “Display Records” button is to be designated as the Accept button and the “Close Form” button is to be designated as the Cancel button. Access Keys are to be assigned to all buttons on all Forms. Set the Tab index to a logical order on each form. The application is to start in the center of the screen. Forms 2 and 3 are to start in Center of Parent. Design your GUI’s as shown.

When the application starts, enter each record, clicking the “Save to File” button after each.

Item Number Description Price

A111 RiteLite Server Rk 599.99

B222 Wire Harness Track 249.49

C333 Console Tray Mount 175.75

D444 Router Mt Brackets 125.25

E555 Floor Track Mounts 100.00

F666 Rockland Toolkit 325.25

*External* Class Definition

Create a class named InventoryItems. This class *gets* and *sets* the fields for Item Number, Description, and Price. All three of these fields are *string* data items. The InventoryItems class also contains a public new string ToString() method. Within the method, *return* the Item Number, Description, and Price fields each separated by a comma (,) delimiter.

Form1 Code Window

Create a Click event method for the “Save to File” button. Within the method, declare a *string* constant to hold text file name. Instantiate an object based on the InventoryItems class. Instantiate an object based on the FileStream class (file name constant, FileMode is Append, and FileAccess is Write). Instantiate an object based on the StreamWriter class. Reference Figure 14-11 on Page 689 for FileStream and StreamWriter instantiations. Populate the fields contained in the InventoryItems class with the data items entered by the user. Write those fields to the text file. Clear all textboxes of their contents. Set the focus to the Item Number textbox. Close the FileStream and StreamWriter objects. Create a Click event for the “Select an Item” button that displays Form2 when clicked. Create a Click event for the “View all Items” button that displays Form3 when clicked. Create a Click event method for the “Exit” button that terminates the application when clicked.

Form2 Code Window

Create a Click event method for the “Close Form” button that closes the form.

Create a Click event method for the “Find Item” button. Within the method, create a constant that contains the comma (,) delimiter character. Declare a constant to hold text file name. Instantiate an object based on the InventoryItems class. Instantiate an object based on the FileStream class (file name constant, FileMode is Open, and FileAccess is Read). Instantiate an object based on the StreamReader class. Declare a string variable to hold each record contained in the text file. Declare a string variable to hold the item number the user enters. Declare a string array to hold each field contained in the record that is read. Declare a Boolean variable used to indicate a record is found (initialize to false). Reference Figure 14-14 on Page 691 for constant declarations, object instantiations, array declaration, and variable declarations. Capture the Item Number the user enters and store it in the appropriate variable. Read a record from the file and store it in the appropriate variable. Reference Figure 14-14 on Page 691 (ReadLine).

Use a while loop to process the text file. Within the loop, populate the array with the splitting of the record read into fields. Store the array values for Item Number, Description, and Price in the class fields. Reference Figure 14-14 on Page 691 for record splitting and assigning array values to class fields. Use an if statement to determine if the Item Number from the class equals the item number entered by the user. If they are equal (found), display the required items to the user. Populate the Boolean variable with true. Set the focus to the Item Number textbox. After the if statement, read the next record in the text file.

After the while loop, use an if statement to determine if the record was not found. If it was not found, display the entire “No record found…” message, set the focus to the Item Number textbox, and select all text in the Item number textbox.

After the if statement, close the FileStream and StreamReader objects.

Form3 Code Window

Create a Click event method for the “Close Form” button that closes the form.

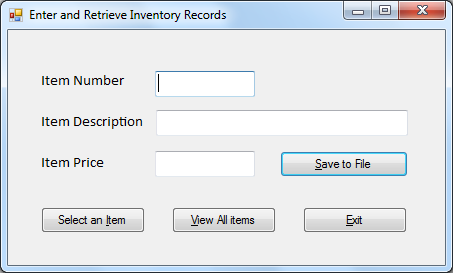
Create a Click event method for the displayButton. Within the method, create a constant that contains the comma (,) delimiter character. Declare a constant to hold text file name. Instantiate an object based on the InventoryItems class. Instantiate an object based on the FileStream class (file name constant, FileMode is Open, and FileAccess is Read). Instantiate an object based on the StreamReader class. Declare a string variable to hold each record contained in the text file. Declare a string array to hold each field contained in the record that is read. Read a record from the file and store it in the appropriate variable.

Use a while loop to process the text file. Within the while loop, populate the array with the splitting of the record read into fields. Store the array values for Item Number, Description, and Price in the class fields. Display the required items to the user. Read the next record in the text file.

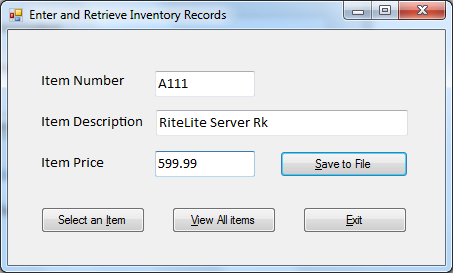
After the while loop, close the FileStream and StreamReader objects.

No Pseudocode document required for this programming assignment.

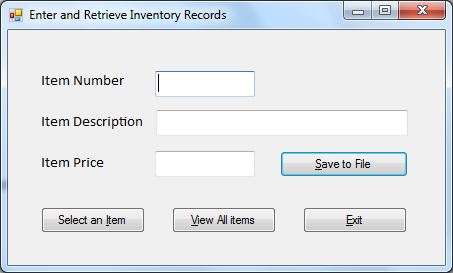
Application Started



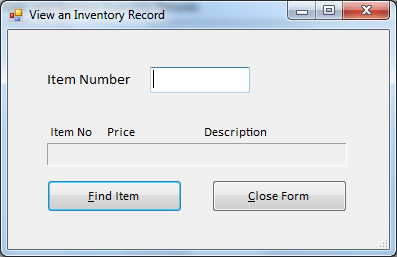
Application Tested (1) – Save to File clicked, write data items to the “Inventory.txt” file…



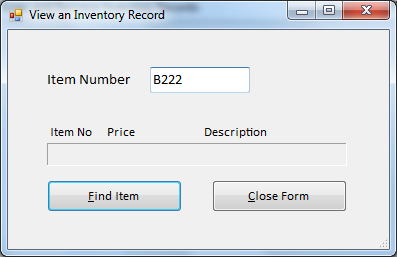
… and clear (all) textbox contents and set focus to item number textbox



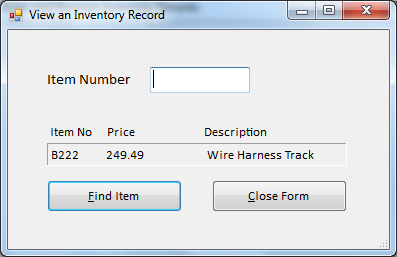
Application Tested (2) – Select an Item clicked on Form1, Form2 appears



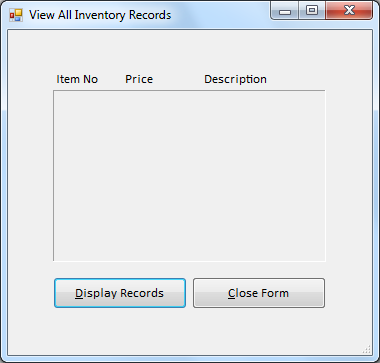
Item number entered



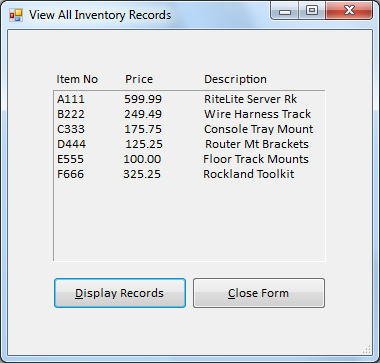
Find Item clicked, display inventory information, clear and set focus to the item number textbox



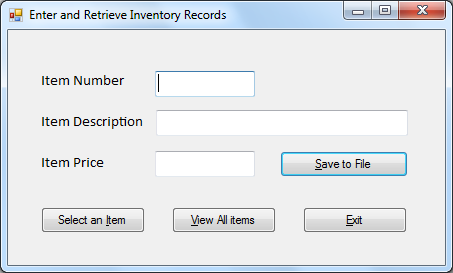
Application Tested (3) – View All Items button clicked on Form1, Form3 appears



Display Records button clicked



Application Tested (4) – Select an Item clicked on Form1, Form2 appears



Invalid item number – not found displayed, set focus and select all text in item number textbox

