Activity 7

Ian M. McConihay

College of Science, Engineering and Technology, Grand Canyon University

CST-150: C# Programming I

Mark Smithers

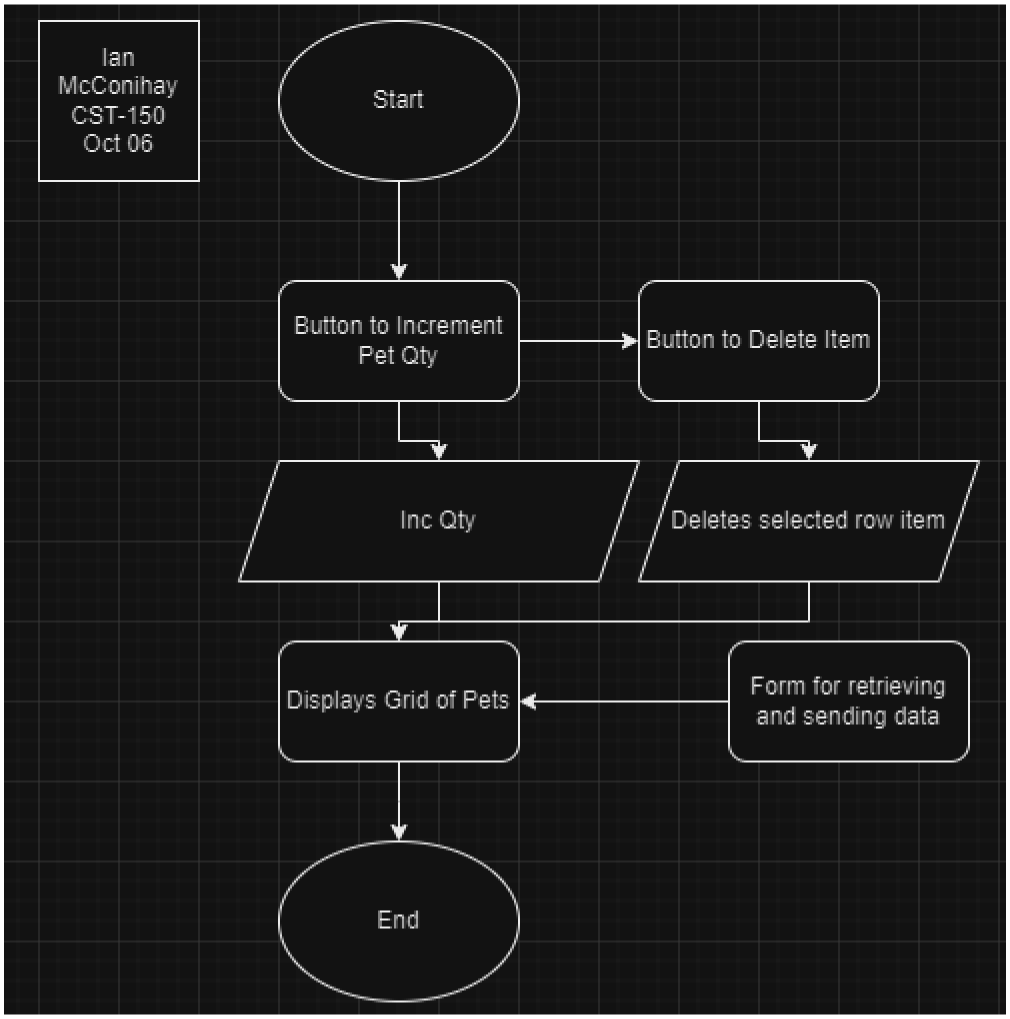
October 6, 2024

Video Link:

https://www.loom.com/share/2b776a138a0b4b3c84cf220972576584?sid=ba0f5657-a050-45a6-8d36-f6aef3df620b

Github: https://github.com/Ian-McConihay/CST-150

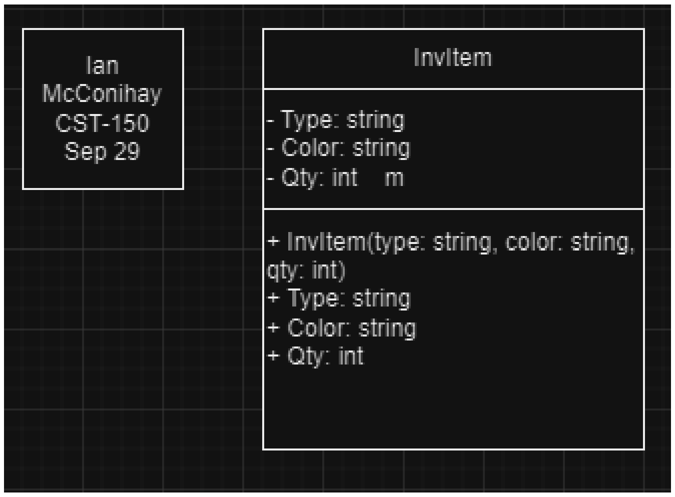
Flowchart


Ian McConihay CST-150 Oct 06
Start
Button to Increment Pet Qty
Button to Delete Item
Inc Qty
Deletes selected row item
Displays Grid of Pets
Form for retrieving and sending data
End

This is a continuation of Activity 6 part 1. This will be adding a second form to communicate data with the first form. This activity also added a delete item function.

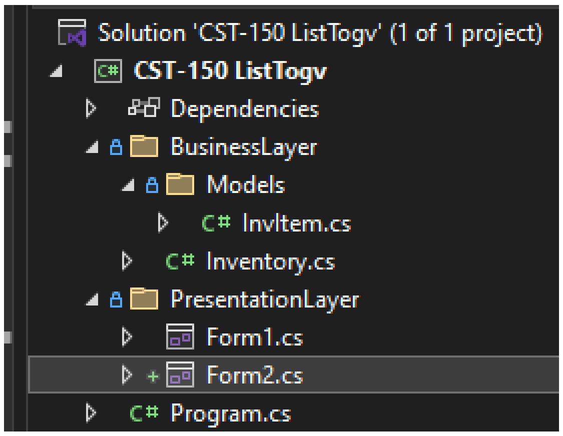
UML

Figure 1: InvItem



The UML InvItem has three attributes for Type Color and Quantity. Then is has a parameterized constructor. That is all the object consists of.

N-Layer

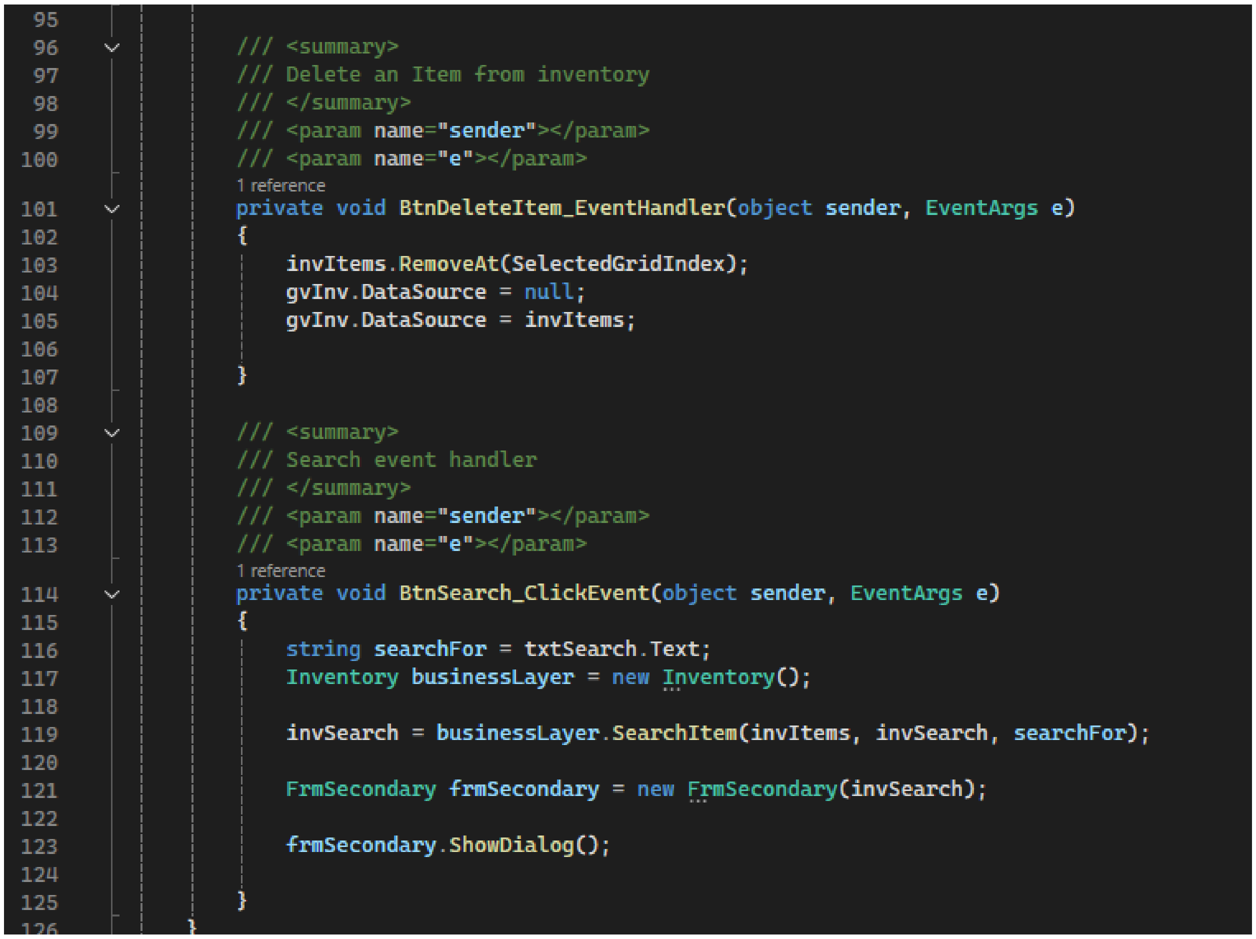


```
Solution 'CST-150 ListTogv' (1 of 1 project)
  CST-150 ListTogv
    Dependencies
    BusinessLayer
      Models
        InvItem.cs
      Inventory.cs
    PresentationLayer
      Form1.cs
      + Form2.cs
    Program.cs
```

Here is a screenshot of the file structure for the application. N-layer was required and demonstrated in the assignment. InvItem and Inventory have been moved to the BusinessLayer and The PresentationLayer contains the Main form and the Second Form for design.

Application Screenshots

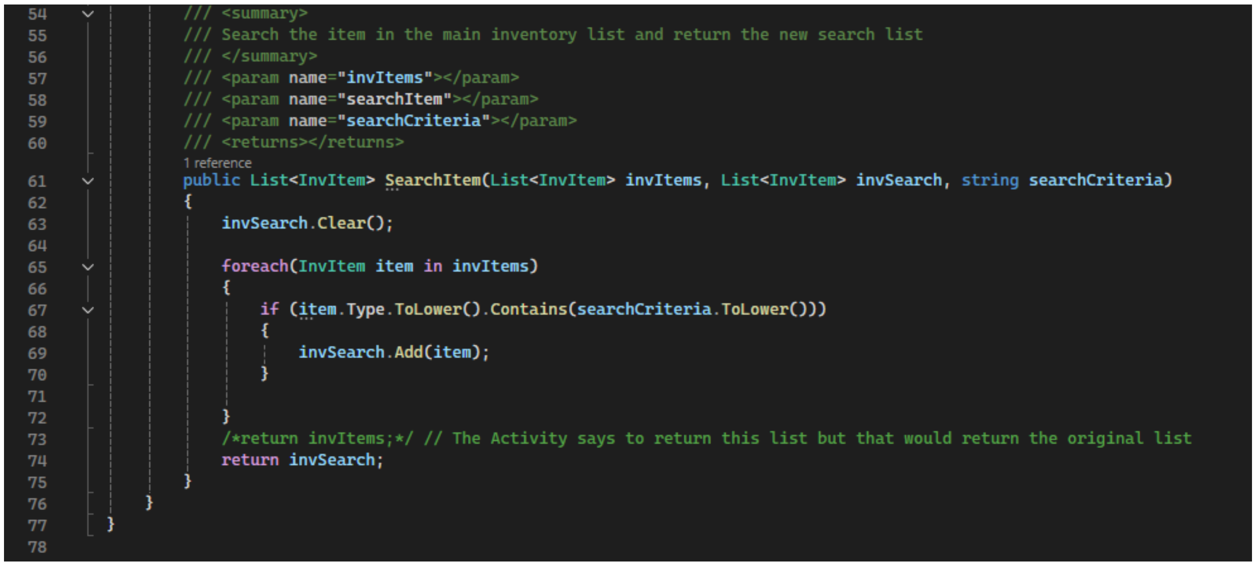
Figure 1: Code



```
        /// <summary>
        /// Delete an Item from inventory
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 reference
        private void BtnDeleteItem_EventHandler(object sender, EventArgs e)
        {
            invItems.RemoveAt(SelectedGridIndex);
            gvInv.DataSource = null;
            gvInv.DataSource = invItems;

        }

        /// <summary>
        /// Search event handler
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 reference
        private void BtnSearch_ClickEvent(object sender, EventArgs e)
        {
            string searchFor = txtSearch.Text;
            Inventory businessLayer = new Inventory();

            invSearch = businessLayer.SearchItem(invItems, invSearch, searchFor);

            FrmSecondary frmSecondary = new FrmSecondary(invSearch);

            frmSecondary.ShowDialog();

        }
    }
```

BtnDeleteItem_EventHandler deletes an Item from inventory. BtnSearch_ClickEvent is a Search event handler. This also calls on our Search Item method located in the Inventory class.

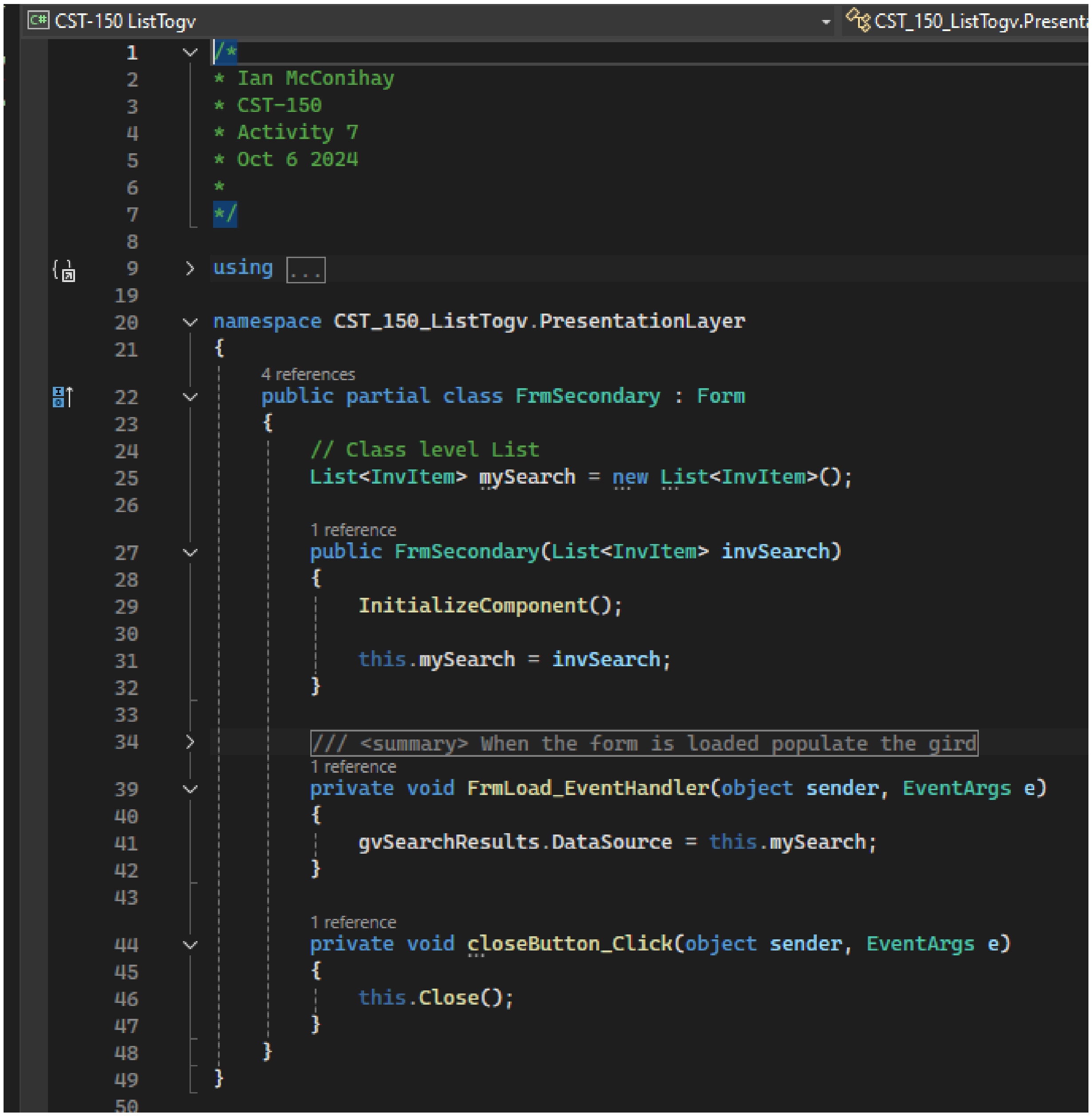
Figure 2: Code



```
        /// <summary>
        /// Search the item in the main inventory list and return the new search list
        /// </summary>
        /// <param name="invItems"></param>
        /// <param name="searchItem"></param>
        /// <param name="searchCriteria"></param>
        /// <returns></returns>
        1 reference
        public List<InvItem> SearchItem(List<InvItem> invItems, List<InvItem> invSearch, string searchCriteria)
        {
            invSearch.Clear();

            foreach(InvItem item in invItems)
            {
                if (item.Type.ToLower().Contains(searchCriteria.ToLower()))
                {
                    invSearch.Add(item);
                }

            }
            /*return invItems;*/ // The Activity says to return this list but that would return the original list
            return invSearch;
        }
    }
}
```

SearchItem searches the item in the main inventory list and returns the new search list. We also are clearing the original search so that the method can be called multiple times. Then we iterate through the list searching for Types that contain the search criteria text.

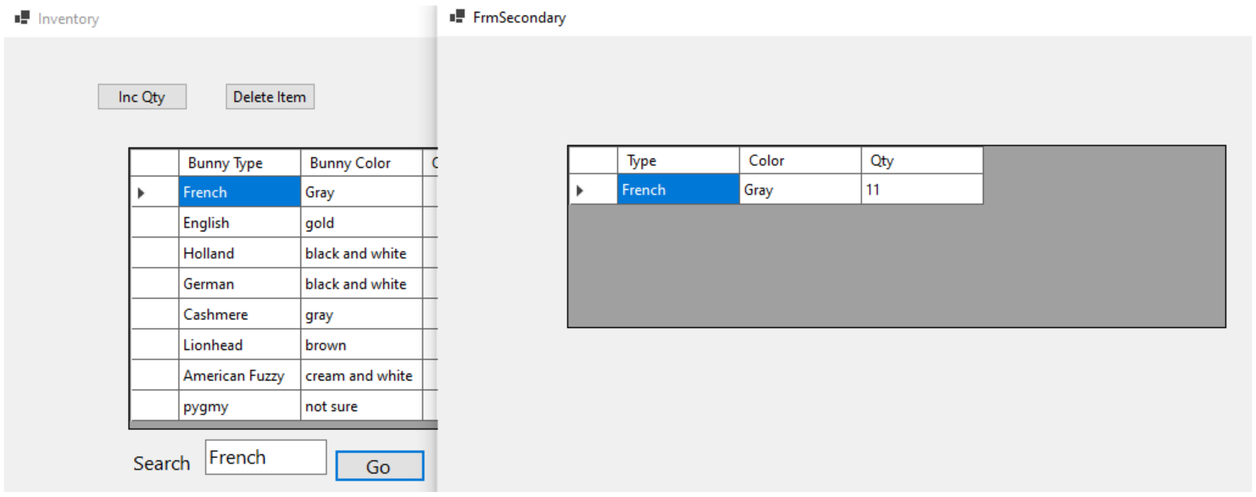
Figure 3: Second Form Code



```
/*
* Ian McConihay
* CST-150
* Activity 7
* Oct 6 2024
*
*/

using ...

namespace CST_150_ListTogv.PresentationLayer
{
    4 references
    public partial class FrmSecondary : Form
    {
        // Class level List
        List<InvItem> mySearch = new List<InvItem>();

        1 reference
        public FrmSecondary(List<InvItem> invSearch)
        {
            InitializeComponent();

            this.mySearch = invSearch;
        }

        /// <summary> When the form is loaded populate the gird
        1 reference
        private void FrmLoad_EventHandler(object sender, EventArgs e)
        {
            gvSearchResults.DataSource = this.mySearch;
        }

        1 reference
        private void closeButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

In this screenshot we start off with the citation. We initialize our search inventory invItems list. Then the form is loaded the data source for the dataGridView using the search list. We also have a click event to close out the form by clicking on the form itself.

Figure 5: Application



	Bunny Type	Bunny Color
▶	French	Gray
	English	gold
	Holland	black and white
	German	black and white
	Cashmere	gray
	Lionhead	brown
	American Fuzzy	cream and white
	pygmy	not sure

French

	Type	Color	Qty
▶	French	Gray	11

Here I have the application running. We have the data displayed in the second form based on the search criteria set in the text box. Everything design related is displayed.

1. What was challenging?
   Having the correct list persists in the second form.

2. What did you learn?
   How to use multiple forms.

3. How would you improve on the project?
   Design styling for the scond form to be set to one setting.

4. How can you use what you learned on the job?
   Multiple forms can provide a better experience for the user.