Activity 6

Ian M. McConihay

College of Science, Engineering and Technology, Grand Canyon University

CST-150: C# Programming I

Mark Smithers

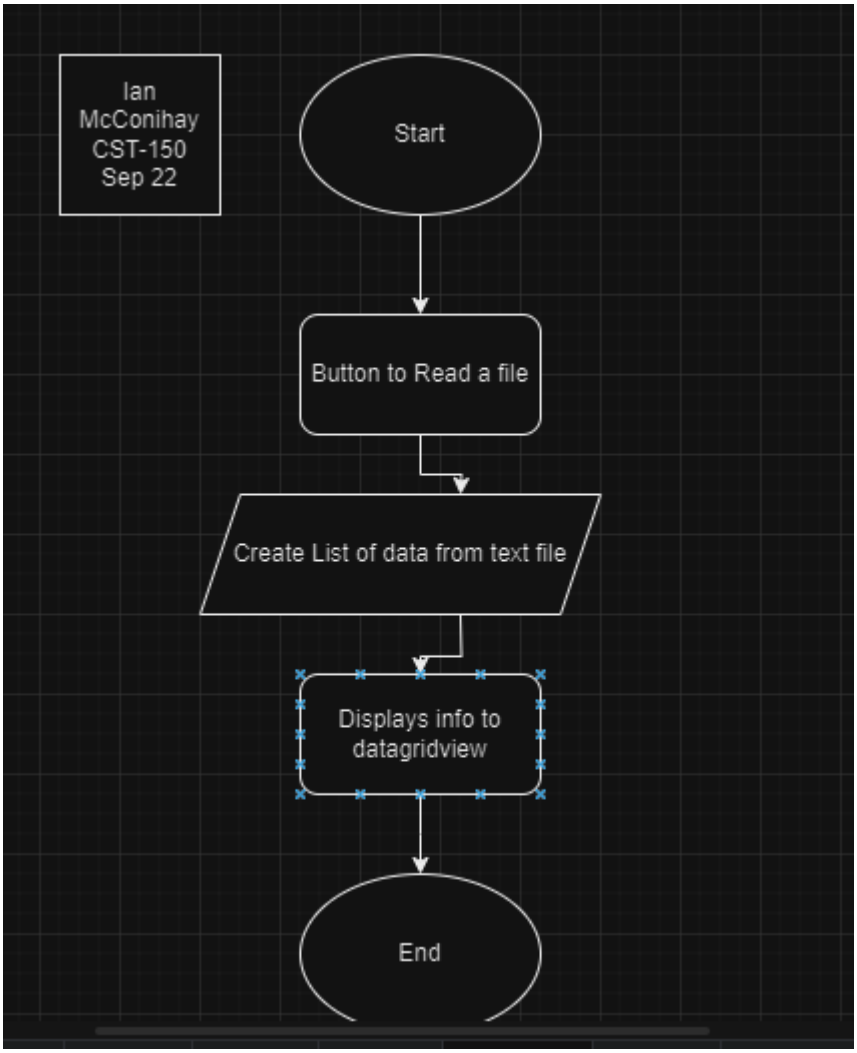
September 29, 2024

Video Link:

https://www.loom.com/share/850a1aaf51ef402b8ce1fb33f07bf29f?sid=da382b41-771e-4a81-b419-ff40c83310f6

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

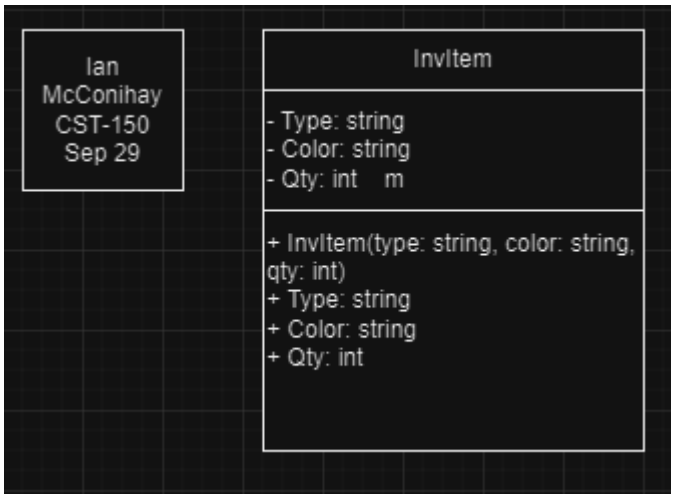
Flowchart



This application allows the user to read from a text file. Then it will display the text file persisted through a List into a datagridview. It has a button to increase the quantity.

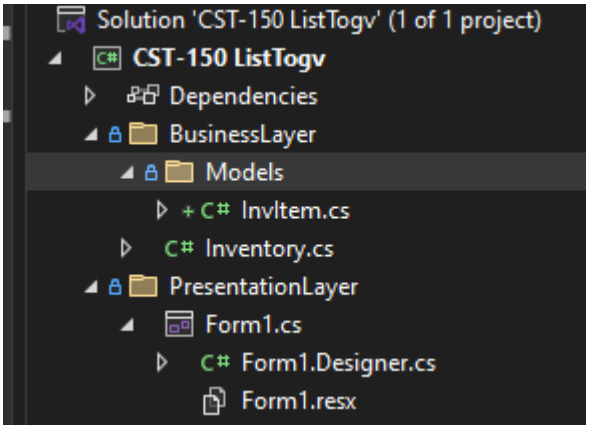
UML

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.

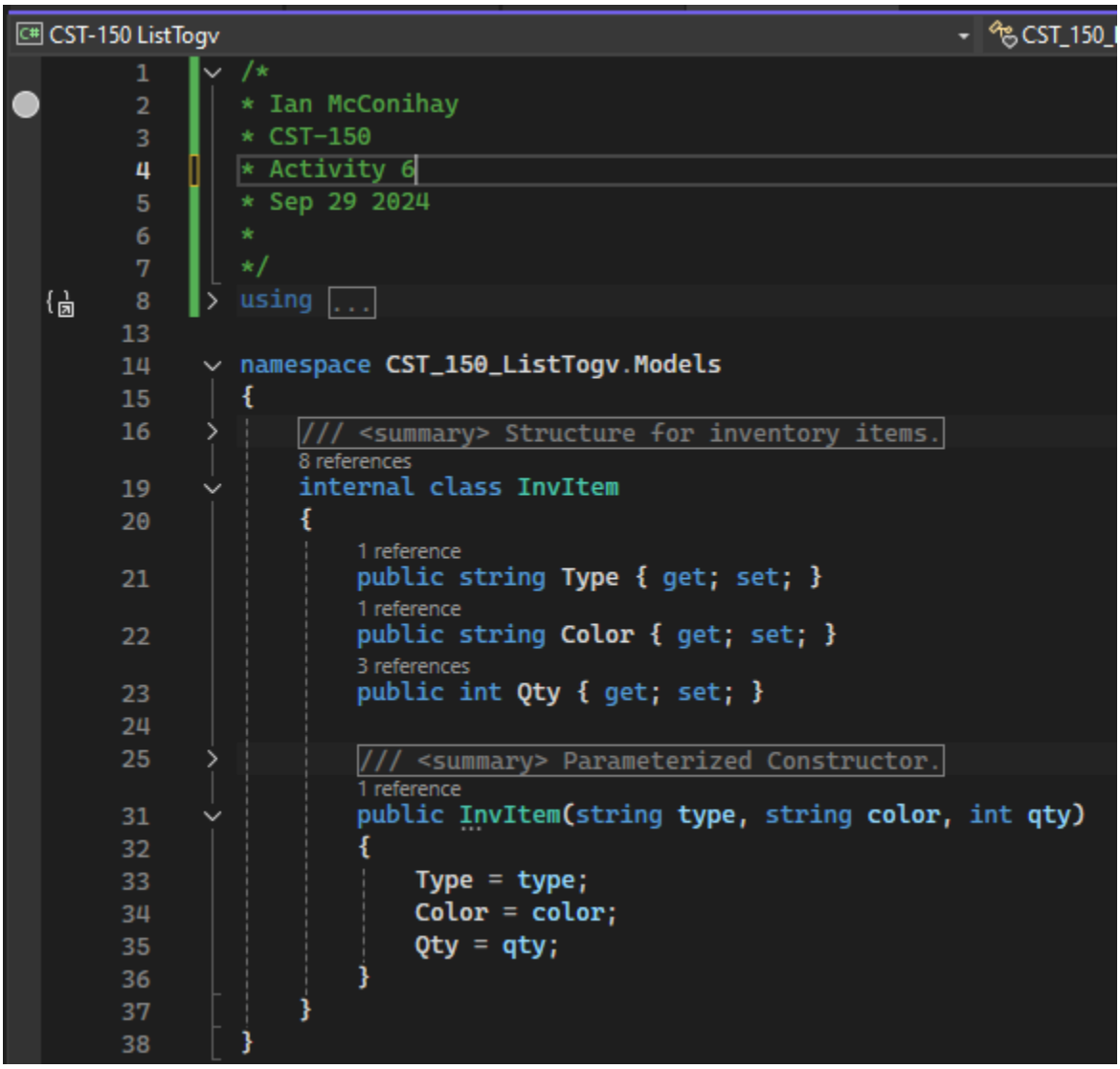
N-Layer



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 for design.

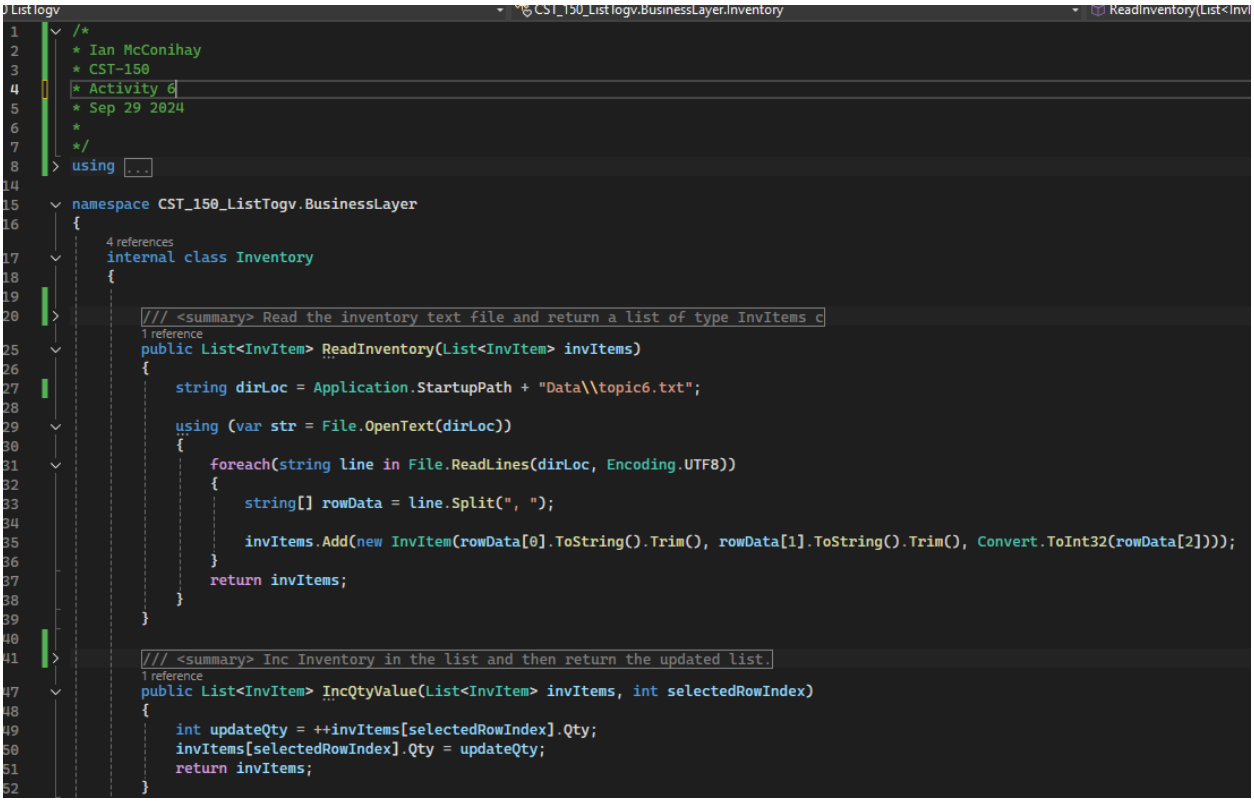
Application Screenshots

Figure 1: Code



Starting with the citation at the top of the class. The InvItem model is for establishing the contents of our list. It has the attributes and the parameterized constructor.

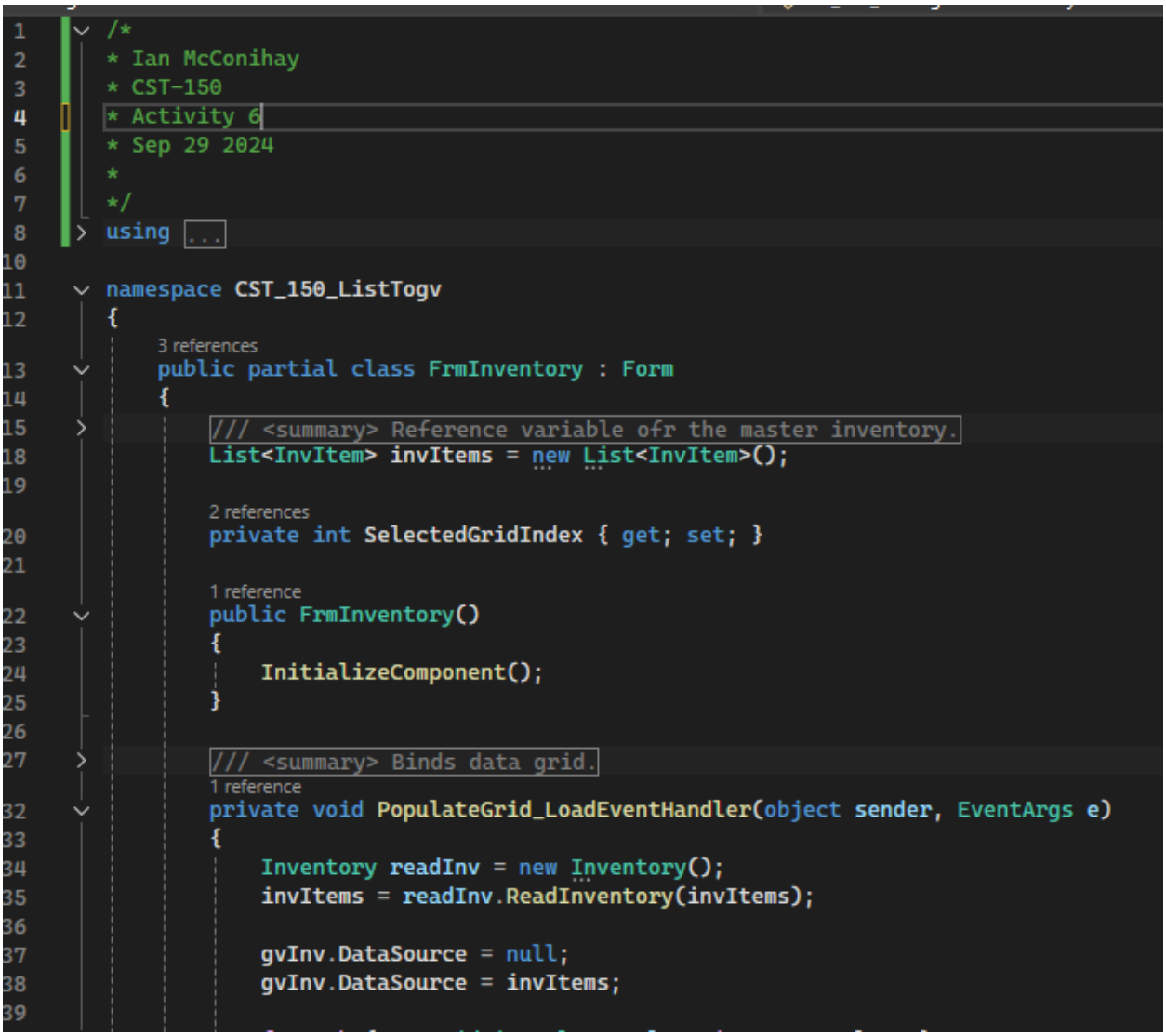
Figure 2: Code



```
/*
* Ian McConihay
* CST-150
* Activity 6
* Sep 29 2024
*
*/
using ...

namespace CST_150_ListTogv.BusinessLayer
{
    4 references
    internal class Inventory
    {

        /// <summary> Read the inventory text file and return a list of type InvItems c
        1 reference
        public List<InvItem> ReadInventory(List<InvItem> invItems)
        {
            string dirLoc = Application.StartupPath + "Data\\topic6.txt";

            using (var str = File.OpenText(dirLoc))
            {
                foreach(string line in File.ReadLines(dirLoc, Encoding.UTF8))
                {
                    string[] rowData = line.Split(", ");

                    invItems.Add(new InvItem(rowData[0].ToString().Trim(), rowData[1].ToString().Trim(), Convert.ToInt32(rowData[2])));
                }
                return invItems;
            }
        }

        /// <summary> Inc Inventory in the list and then return the updated list.
        1 reference
        public List<InvItem> IncQtyValue(List<InvItem> invItems, int selectedRowIndex)
        {
            int updateQty = ++invItems[selectedRowIndex].Qty;
            invItems[selectedRowIndex].Qty = updateQty;
            return invItems;
        }
```

In this screenshot we start off with the citation. Then we have the ReadInventory method to readthrough the provide file path. Then break up the rows into invItems. Also the IncQtyValue method that adds to the quantity of the selected row.

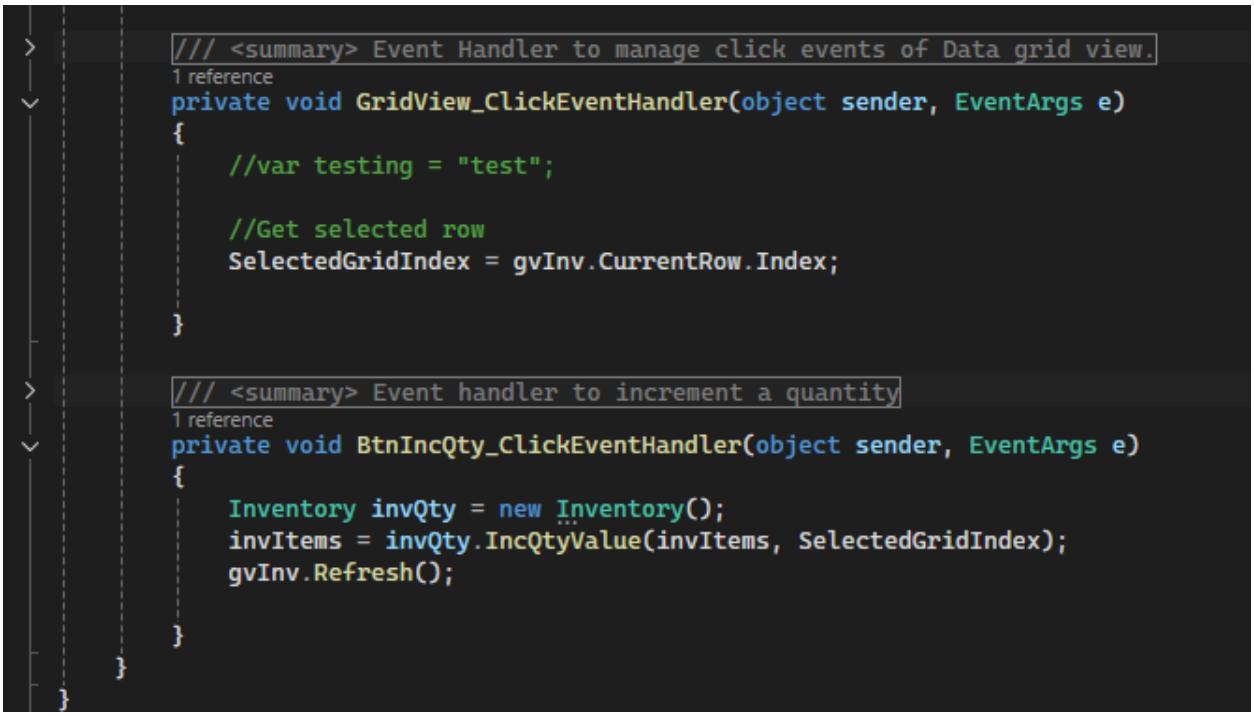
Figure 3: Dog Code



```
/*
* Ian McConihay
* CST-150
* Activity 6
* Sep 29 2024
*
*/
using ...

namespace CST_150_ListTogv
{
    3 references
    public partial class FrmInventory : Form
    {
        /// <summary> Reference variable ofr the master inventory.
        List<InvItem> invItems = new List<InvItem>();

        2 references
        private int SelectedGridIndex { get; set; }

        1 reference
        public FrmInventory()
        {
            InitializeComponent();
        }

        /// <summary> Binds data grid.
        1 reference
        private void PopulateGrid_LoadEventHandler(object sender, EventArgs e)
        {
            Inventory readInv = new Inventory();
            invItems = readInv.ReadInventory(invItems);

            gvInv.DataSource = null;
            gvInv.DataSource = invItems;
```

In this screenshot we start off with the citation. We initilize our main master inventory invItems list. We also have our SlectedGriIndex being created. PopulateGrid_LoadEventHandler manipulates the data source for the datagridview using a switch case for the column indexes.

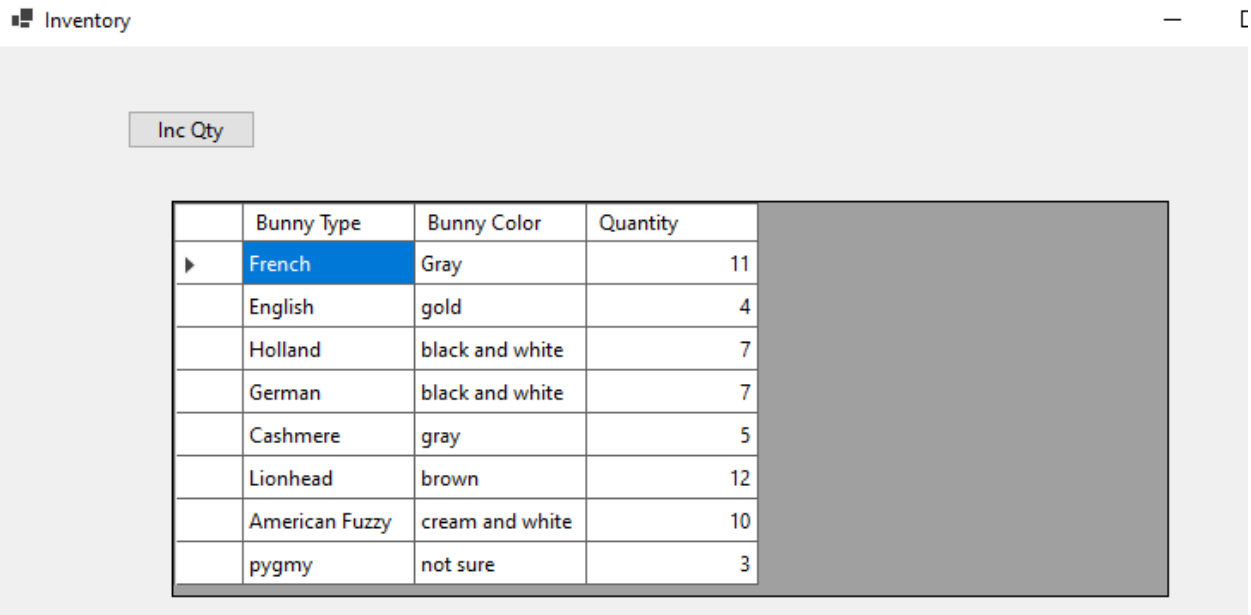
Figure 4: Dog Code



```
        /// <summary> Event Handler to manage click events of Data grid view.
        1 reference
        private void GridView_ClickEventHandler(object sender, EventArgs e)
        {
            //var testing = "test";

            //Get selected row
            SelectedGridIndex = gvInv.CurrentRow.Index;

        }

        /// <summary> Event handler to increment a quantity
        1 reference
        private void BtnIncQty_ClickEventHandler(object sender, EventArgs e)
        {
            Inventory invQty = new Inventory();
            invItems = invQty.IncQtyValue(invItems, SelectedGridIndex);
            gvInv.Refresh();

        }
    }
}
```

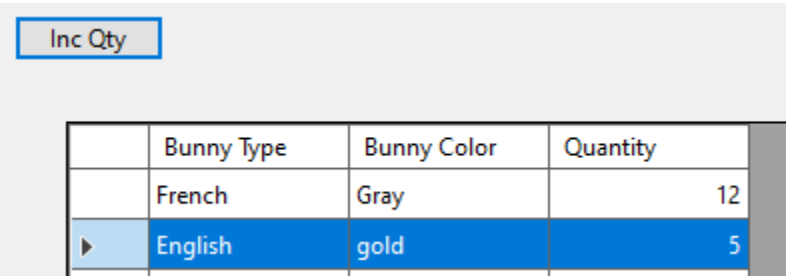
In this screenshot we have for event handling clicks. One method to select a row. The other method handles the incrementing of an items quantity.

Figure 5: Application



Here I have the application running. We have the columns displaying as instructed. Everything design related is displayed.

Figure 6: Application



Here I have clicked on the first two rows to display the new incremented quantities.

1. What was challenging?

Getting some of the column manipulation to work was tricky.

2. What did you learn?
   Using a List instead of an Aarry for capturing the text file data.

3. How would you improve on the project?
   I would create a writer to save the files new changes.

4. How can you use what you learned on the job?

List manipulation is more effective and can be joined with LINQ.