Milestone 5

Ian M. McConihay

College of Science, Engineering and Technology, Grand Canyon University

CST-150: C# Programming I

Mark Smithers

September 22, 2024

Video Link:

https://www.loom.com/share/9fe6b6dc553342f4bbf32134623bb356?sid=becfe61d-7686-4dc4-9d8c-c9d78eb5fc64

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

What was challenging?

Implementing my logic I had in the main form into a service class.

What did you learn?

N-layer architecture.

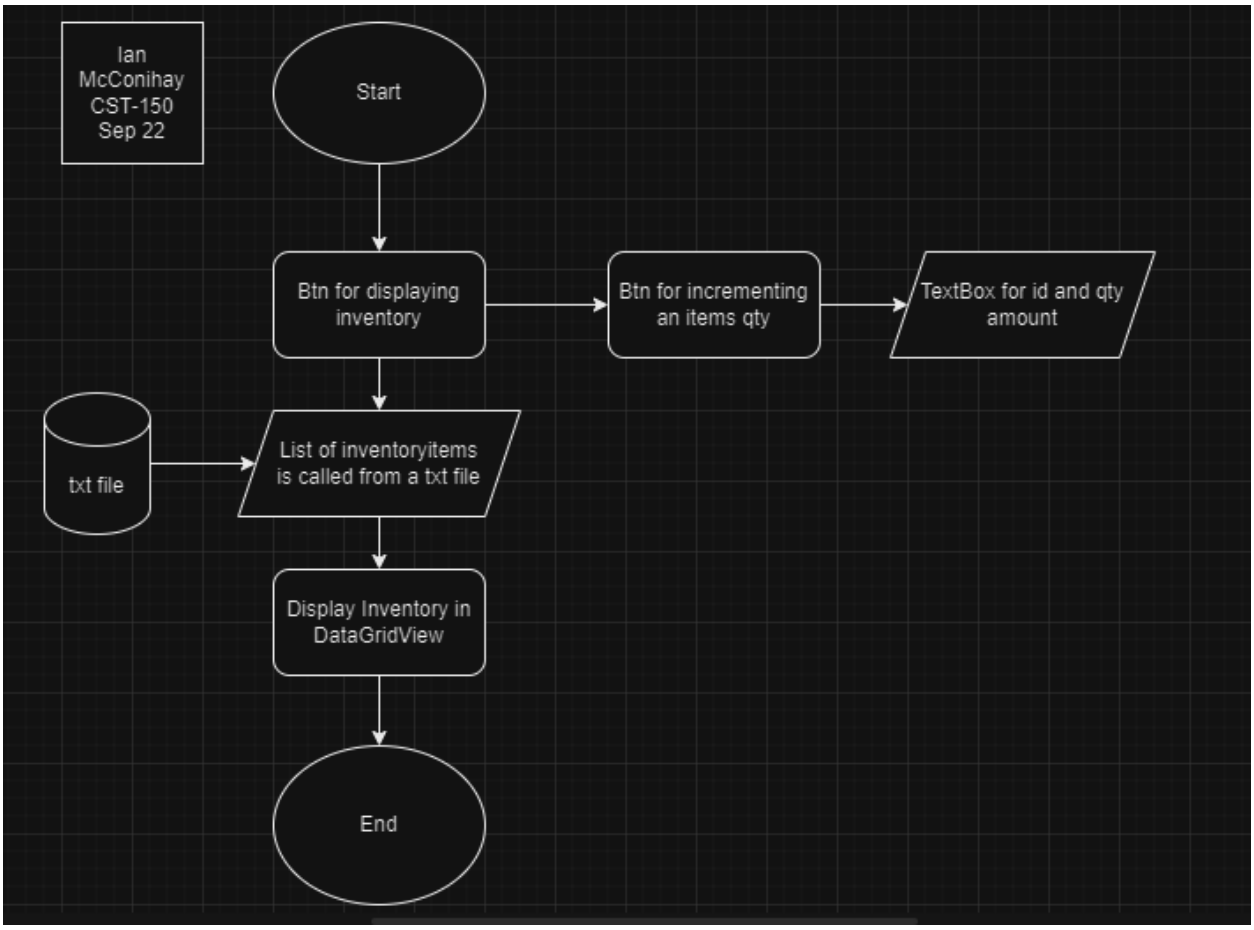
How would you improve on the project?

Change how the text file is read or just the data source.

How can you use what you learned on the job?

Architecture can make cleaner code and makes it easier to break down applications.

Figure 1: FlowChart



At the start of this application will open to a button for the user to click and persist a grid view of inventory items from a text file. Another button will allow the user to increment one of the items quantity. The user will be able to enter the id and the amount they want to increment an item. The datagridview provides table functions for the user to view the inventory.

Figure 2: UML InventoryItem

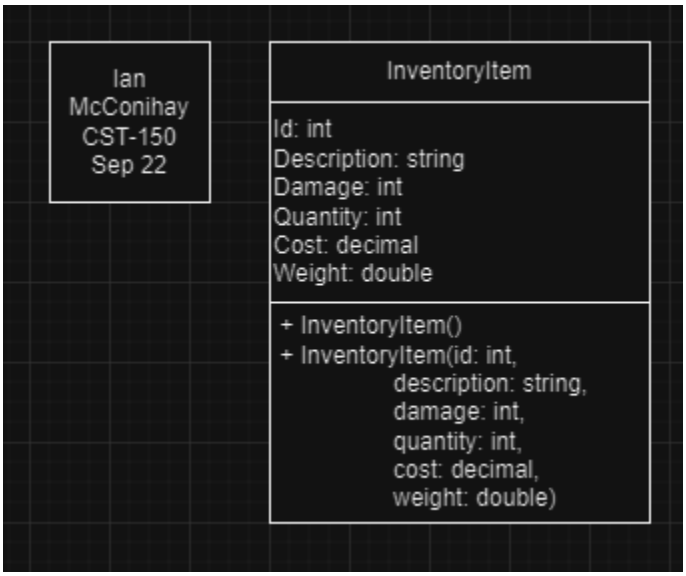
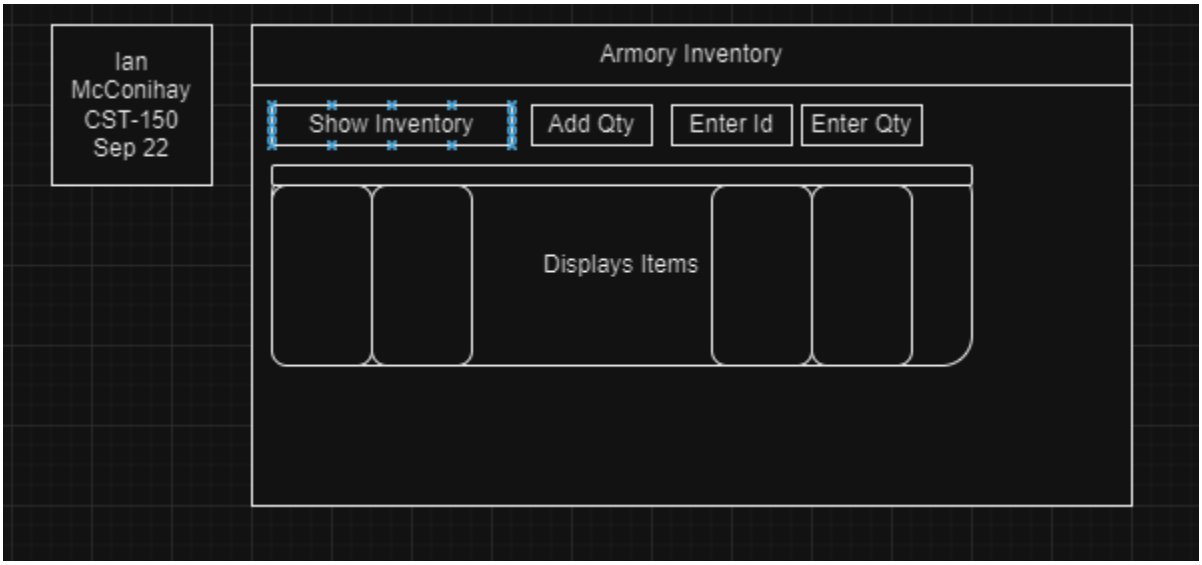


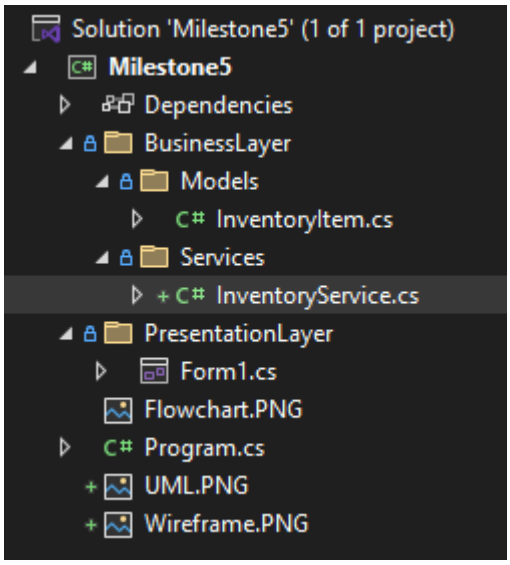
Figure 3: Wireframe



The updated wire frame has a handful of changes. The form has a name when displayed for the user. The button shows inventory is located above where the list will be displayed using a data grid view. The user will have columns to display the information. Add Qty button for

incrementing an items quantity now has two textboxes for the user to enter the id and qty amount.

N-Layer



Here is a screenshot of the file structure for the application. N-layer was required for milestone. InventoryItem and InventoryService has been moved to the BusinessLayer and The PresentationLayer contains the Main form for design.

Application Screenshots

Figure 4: Code

```
DataService.cs   Form1.cs   Form1.cs [Design]   InventoryItem.cs
Milestone5   Milestone5.Form1   filePath

3 references
public partial class Form1 : Form
{
    string filePath = "C:\\Users\\nmcco\\Desktop\\CST-150\\Milestone3\\Milestone3\\bin\\Debug\\net8.0-windows\\Data\\Inventory.txt"; // Ensure this path is correct
    InventoryService inventoryService;
    1 reference
    public Form1()
    {
        InitializeComponent();
        LoadInventoryData();

    }

    /// <summary> Event to display Inventory when clicked.
    1 reference
    private void btnShowInventory_Click(object sender, EventArgs e)
    {
        dataGridViewInventory.Visible = true;

    }

    /// <summary> Event to increment quantity when clicked.
    1 reference
    private void btnIncrement_Click(object sender, EventArgs e)
    {
        var inventoryService = new InventoryService();
        inventoryService.LoadInventoryData(filePath);
        inventoryService.IncrementInventory(int.Parse(txtBoxIncQty.Text), int.Parse(txtBoxQty.Text));
        DataTable inventoryData = inventoryService.PopulateDataGridView();
        dataGridViewInventory.DataSource = inventoryData;

    }


    1 reference
    private void LoadInventoryData()
    {
        var inventoryService = new InventoryService();
        inventoryService.LoadInventoryData(filePath);

        // Populate the DataGridView
        DataTable inventoryData = inventoryService.PopulateDataGridView();
        dataGridViewInventory.DataSource = inventoryData;
        dataGridViewInventory.Visible = true; // Show the DataGridView
    }
}
}
```

For figure 4 we start off with the citation at the top. The main changes here is the majority of the logic even for the data has been moved to the business layer. Creating the inventory service has allowed this.

Figure 5: Code

```
DataService.cs   Form1.cs   Form1.cs [Design]   InventoryItem.cs
Milestone5                                      Milestone5.BusinessLayer.Se

/*
* Ian McConihay
* CST-150
* Milestone
* Sep 22 2024
*
*/

using ...

namespace Milestone5.BusinessLayer.Services
{
    3 references
    public class InventoryService
    {
        // Hard-coded initial fantasy inventory
        private InventoryItem[] armoryInventory;

        2 references
        public void LoadInventoryData(string filePath)...

        3 references
        public DataTable PopulateDataGridView()...

        /// <summary>
        /// Increments Inventoryitems quantity
        /// </summary>
        /// <param name="itemId"></param>
        /// <param name="incrementAmount"></param>
        1 reference
        public void IncrementInventory(int itemId, int incrementAmount)...

    }
}
```

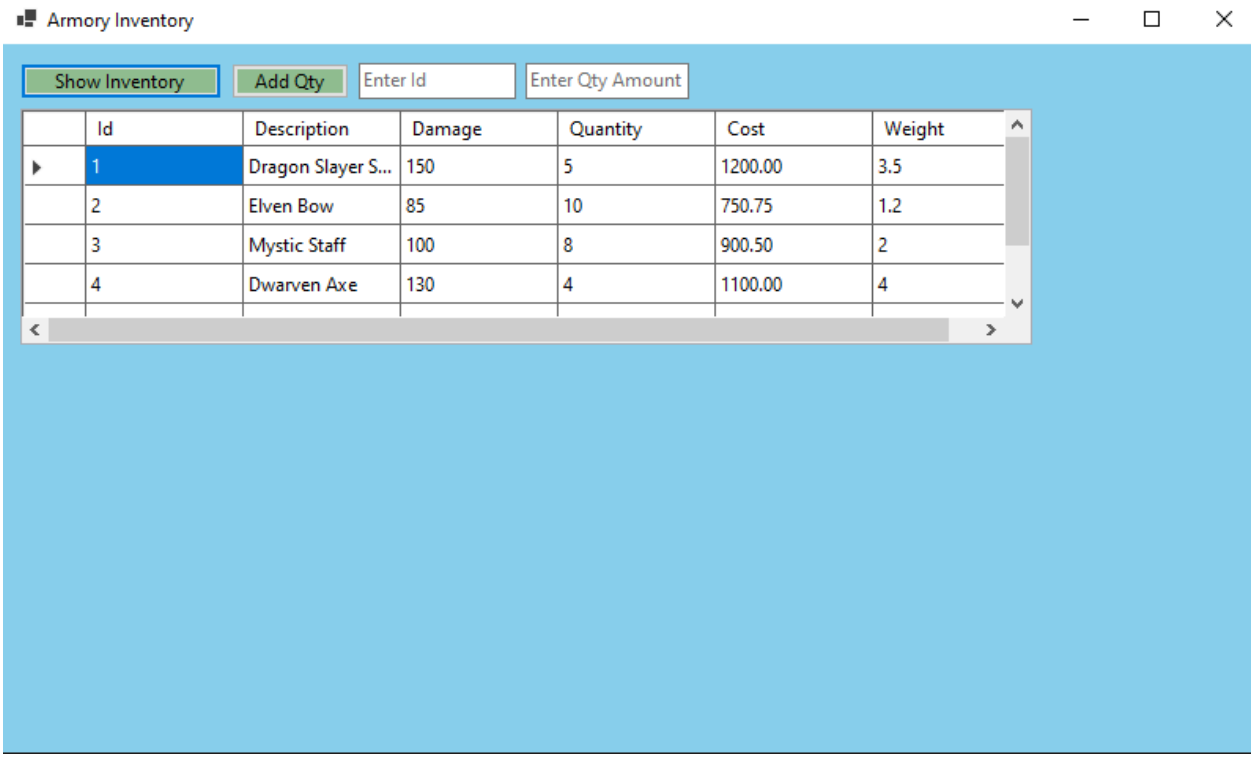
Figure 5 has the new class InventoryService. This class took the persisting of data logic and moved it into a service class. This was to focus on making what I have incorporate more architecture.

Figure 6: Code

```
DataService.cs   Form1.cs   Form1.cs [Design]   InventoryItem.cs
Milestone5   Milestone5.BusinessLayer.Models.InventoryItem   Id

/*
* Ian McConihay
* CST-150
* Milestone
* Sep 22 2024
*
*/

namespace Milestone5.BusinessLayer.Models
{
    /// <summary> Items for the Inventory.
    6 references
    public class InventoryItem
    {
        3 references
        public int Id { get; set; }
        2 references
        public string Description { get; set; }
        2 references
        public int Damage { get; set; }
        3 references
        public int Quantity { get; set; }
        2 references
        public decimal Cost { get; set; }
        2 references
        public double Weight { get; set; }

        /// <summary> /
        1 reference
        public InventoryItem(int id, string description, int damage, int quantity, decimal cost, double weight)
        {
            Id = id;
            Description = description;
            Damage = damage;
            Quantity = quantity;
            Cost = cost;
            Weight = weight;
        }
    }
}
```

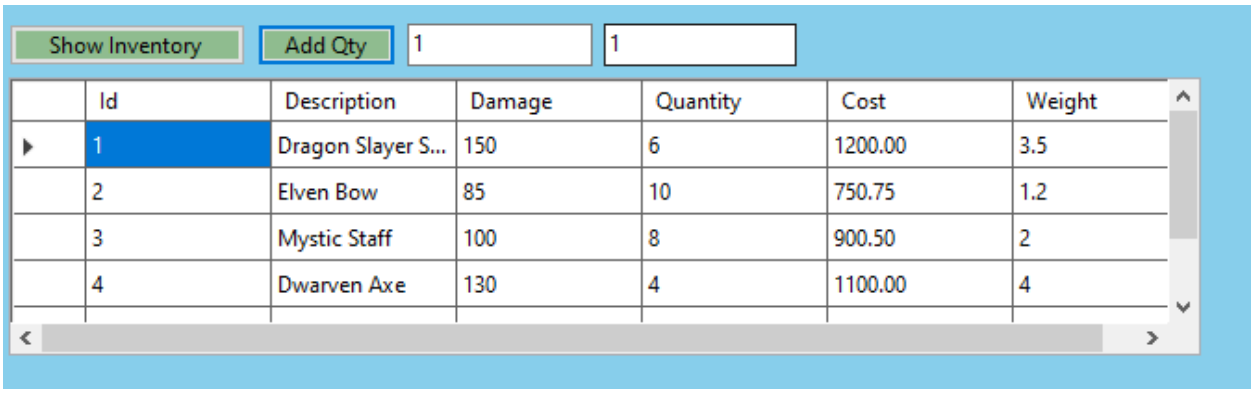
Figure 6 shows that I have created a Models folder. In the model folder I now will store my InventoryItem and any other class objects to be added. The InventoryItem has remained unchanged.

Figure 7: Application Start



The start of the application displays the show inventory button and add qty button. From here I can enter in an Id and qty amount. I freshened up the display using the color scheme tool.

Figure 8: Application Add Inventory



Entering a 1 in and 2 the item with an Id of 1 was incremented positively by 2. This give the user a lot more freedom to make adjustments to the items in stock. Adjusting the rest of the items would be the next step.

Figure 9: Application subtract Inventory item

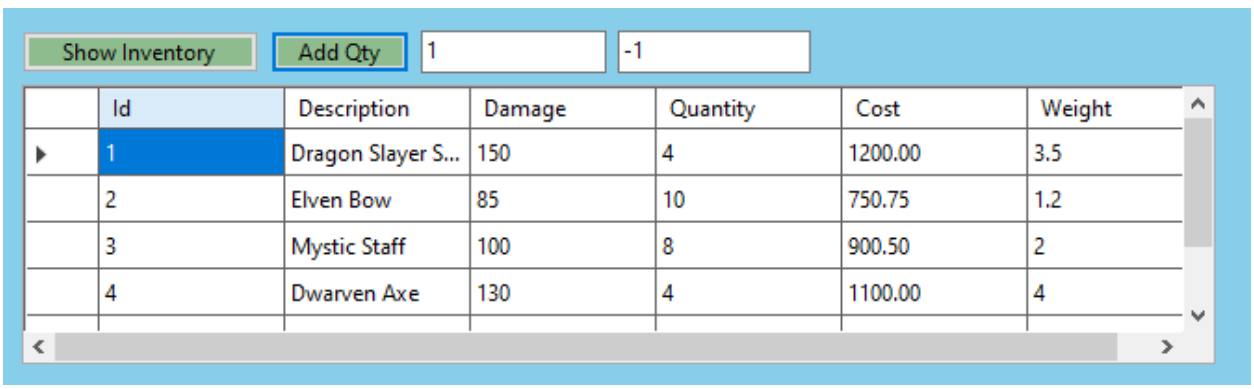


Figure 10 is post pushing the Add Qty button with a negative number. As we can see the Quantity entered was -2 so the stock item went from 8 to 6 for the id 3 item. This is not a good practice but it does add loads of functionality.

Bug Reports

Bug Report: NONE

Class name

Method name :

Steps to reproduce the bug:

Expected results

Actual results

details: N/A

Solution

1. List your computer specs (type of computer, OS, memory, etc)

Device name	DESKTOP-IAQ5CCD
Processor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Installed RAM	8.00 GB (7.88 GB usable)
Device ID	A0AC8D02-4885-4491-B27B-B40F0A0D2E35
Product ID	00356-02139-31547-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	Touch support with 10 touch points

2. Create 3 test cases

Valid File with Proper Data: The method should correctly populate the armoryInventory array with InventoryItem objects based on properly formatted data in the file.

File is Empty: The method should display a warning message indicating the file is empty and leave the armoryInventory array uninitialized.

File with Incorrect Data Format: The method should show an error message indicating an issue with loading data and only initialize valid InventoryItem objects in the armoryInventory array.

3. List 3 Programming conventions that will be used all milestones

Naming, Format, and Documentation Conventions

4. Create Use case diagram

System Boundary: Representing the WinForms application.
Use Case: "View Inventory" indicating the functionality provided by the application.
Actor: "User" who interacts with the system to view the inventory.

Monday
Start: 900pm End: 9:30pm Activity: Read announcements
Start: 930pm End: 1030 Activity: DQ1 and DQ 2
Start: 1030pm End: 1100pm Activity: Read Book
Tuesday
Start: 900pm End: 9:30pm Activity: Participation post
Start: 930pm End: 1030 Activity: Activity 4
Start: 1030pm End: 1100pm Activity: Read Book
Wednesday
Start: End: Activity: N/A
Start: End: Activity: N/A
Start: End: Activity: N/A
Thursday
Start: 900pm End: 9:30pm Activity: Participation post
Start: 930pm End: 1030 Activity: Activity 5
Start: 1030pm End: 1100pm Activity: Read Book
Friday
Start: 900pm End: 9:30pm Activity: Participation post
Start: 930pm End: 1030 Activity: Milestone
Start: 1030pm End: 1100pm Activity: Read Book
Saturday
Start: 900pm End: 9:30pm Activity: Activity 5
Start: 930pm End: 1030 Activity: Milestone
Start: 1030pm End: 1100pm Activity: Milestone
Sunday
Start: 900pm End: 9:30pm Activity: Activity 5
Start: 930pm End: 1030 Activity: Milestone
Start: 1030pm End: 1100pm Activity: Milestone