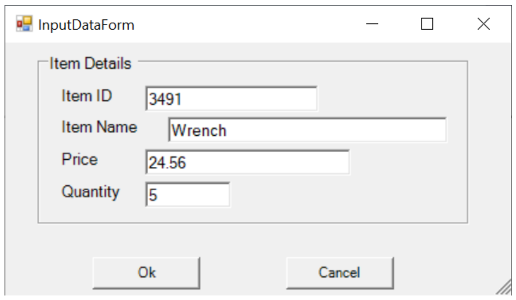
CMPS4143: Contemporary Programming Languages
Major Programming Assignment 8 **Due: Dec 3, 2022 100 points**

PURPOSE: To create an MDI app that uses menus; list boxes (sorted); and a user control; to input and output to serialized files; to use FileDialogs

PROBLEM: **Inventory Program**

You are the owner of two businesses – a toy store and a hardware store. You need a program to keep an inventory of the different items you sell in each store, how many of each item are currently in stock and the cost of each. Write a program that lets you load (input) data from a file relating to each item for a particular store. It should enable you to list all your items in that store, let you delete an item that you no longer have; let you add a new item; and let you update information. It should save the changes to a file (essentially rewriting over the original file). The program should be an MDI – the child windows will correspond to the two different inventory lists of your businesses.

You are to create a main menu with at least three menus: File menu with commands Open, Save, and Exit; an Edit menu with Insert, Delete, and Update commands; and an About menu. You should also add an About form to your project.



Item ID	3491
Item Name	Wrench
Price	24.56
Quantity	5

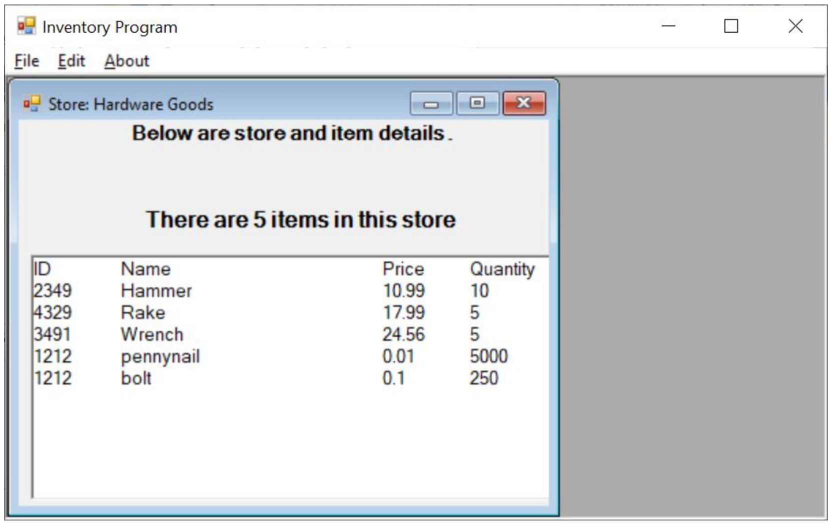
You will want to create an ItemForm UserControl that contains an item's details that you could "reuse" one day in another project. That control should contain labels and textboxes where the user inputs an item's name, cost and quantity.

For Extra Credit: Add a Update command to the Edit menu. This means you can open other businesses. (You'll be quite the entrepreneur!)

INPUT: User should be able to
- Open one or both child forms
  - open files, input business name(s) and load inventory (ies)
- Modify the inventory (ies)

OUTPUT:
- Child form should *display* 1) name of the business (get this from the file or the filename) and 2) an inventory list that was loaded from input
- Save the updatednventories to a file

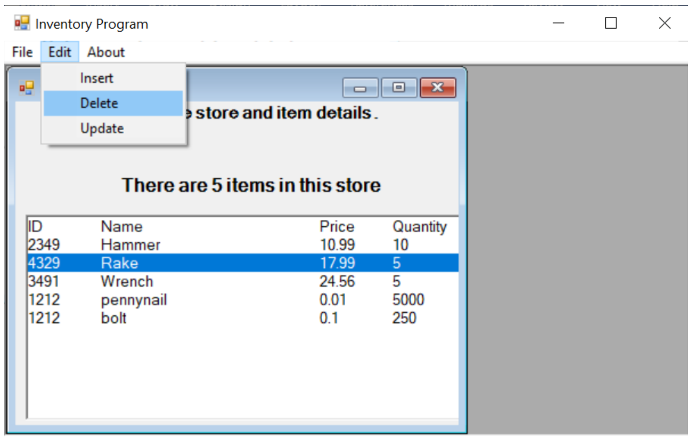
TURN IN (upload all project files):
- Zipped project folder with:
- Complete project files
- I/O files
- Screen dump(s) of image(s) when running



Below are store and item details.

There are 5 items in this store

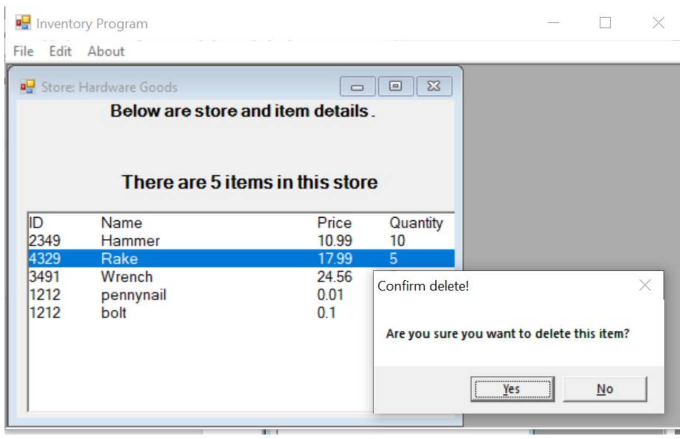
ID	Name	Price	Quantity
2349	Hammer	10.99	10
4329	Rake	17.99	5
3491	Wrench	24.56	5
1212	pennynail	0.01	5000
1212	bolt	0.1	250



store and item details.

There are 5 items in this store

ID	Name	Price	Quantity
2349	Hammer	10.99	10
4329	Rake	17.99	5
3491	Wrench	24.56	5
1212	pennynail	0.01	5000
1212	bolt	0.1	250

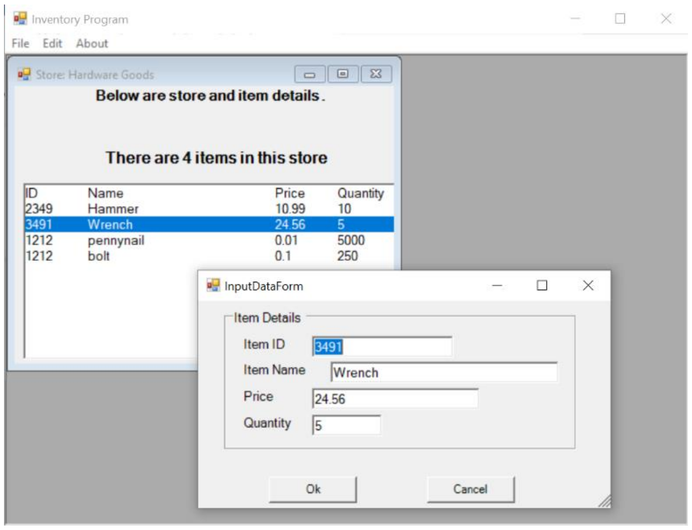


Below are store and item details.

There are 5 items in this store

ID	Name	Price	Quantity
2349	Hammer	10.99	10
4329	Rake	17.99	5
3491	Wrench	24.56	
1212	pennynail	0.01	
1212	bolt	0.1	

Are you sure you want to delete this item?



Below are store and item details.

There are 4 items in this store

ID	Name	Price	Quantity
2349	Hammer	10.99	10
3491	Wrench	24.56	5
1212	pennynail	0.01	5000
1212	bolt	0.1	250

Item ID	3491
Item Name	Wrench
Price	24.56
Quantity	5

```
    [Serializable]
    public class Record
    {
        public Record();
        public Record(int id, string name, string store, double
              price, int quantity);

        public int ID { get; set; }
        public string Name { get; set; }
        public double Price { get; set; }
        public int Quantity { get; set; }
        public string Store { get; set; }
    }
```

HINTS:

To start a data file, you need to call a method you create in your main form load method that has the below code. ***Then after you can comment the call to the method.***

```
// object for deserializing Record in binary format
private BinaryFormatter reader = new BinaryFormatter();

// open file with write access
FileStream output = new FileStream( "Hardware.Inv",
     FileMode.OpenOrCreate, FileAccess.Write );
// Record containing TextBox values to serialize
Record record = new Record();

// store TextBox fields in Record
record.ID = 4568;
record.Name = "Hammer";
record.Price = 12.99;
record.Quantity = 5;
record.Store = "Hardware";

// write Record to FileStream (serialize object)
reader.Serialize( output, record );
```

YOUR RECORD CLASS MUST HAVE `[Serializable]` before the Record class declaration.

FORMS:
When you show your input form that gets the data of a new item to insert (or update) – don't use Show, use ShowDialog. ***This is a modal dialog that stops the execution in the main form until the form you are now showing is finished.***

For the main form to get the data from the insert(update) form (same form!!!) – call a method, or better yet a property, of the insertform. (You will have to add this method and write it – it is NOT automatically generated).

To send the data from the main form(parent) to a child form, you need to

1) Determine the active child object and I had to cast
   YourChildClass activeStore = (YourChildClass) this.ActiveMdiChild;
2) Call a method of the active child object (THERE IS ONLY ONE CHILD CLASS – but there are 2 or more instances of it) and pass the data. The child object puts the data in the list box.

From the last two paragraphs, you should deduce that YES, you can add methods to classes that you have added to your project. YOU HAVE TO! You can also add member data. THINK how you programmed in C++.

The input form and the login forms are NOT child forms. Do not use the instructions:

```
            inputForm.MdiParent = this;
            inputForm.Show();
```

These kind of C# instructions are only found in the new and open file event handlers.