

Inventory Management Application

By

Jacob Lyons (n9507175) & Mitchell Willemse (n9470620)

For

**CAB302 – Software Development
Queensland University of Technology**

25th May 2018

Technical Description

The inventory manager program was written in Java using an object-orientated design approach and compiled using Eclipse. A test-driven development approach was adopted from start to finish, with each member working on the class that the other created the unit tests for.

The program is divided into several packages, each containing classes and/or Junit tests. Each class contributes different functionality to the overall program and are grouped as such. This is used to create abstraction between the classes and make the code easier to understand. For example, the method to export a manifest does not need to access the implantation details of the Manifest class or its private variables. It only needs the methods to work as intended, and the implementation details are abstracted away.

Overall, 12 packages are used:

- **exceptions** – The three exception classes specified by the criteria; CSVFormatException, DeliveryException, StockException.
- **item** – The Item class and the corresponding Junit test.
- **main** – The Main class that creates the GUI of the program and contains the main method that is the entry point of the program. This package also includes the Listener classes for each button and a class that updates the GUI table and label.
- **manifest** – The Manifest class and it's Junit test.
- **manifestCreator** – The ManifestCreator class that creates Manifests.
- **ordinaryTruck** – The OrdinaryTruck class, a child of Truck, and Junit test.
- **readInCSV** – The CSVFormatCheck, ReadItemsCSV, ReadManifestCSV and ReadSalesLogCSV classes that correspond to the programs input functionality.
- **refrigeratedTruck** – The RefrigeratedTruck class, a child of Truck, and Junit test.
- **stock** – The Stock class and Junit test.
- **store** – The Store class and Junit test.
- **truck** – The abstract Truck type.
- **writeOutCSV** – The WriteOutCSV class that corresponds to the programs output functionality.

The listener classes are an example of object-orientated design as they implement the interface class of ActionListener. Other examples of this include the Stock and Manifest class, which both implement the Iterable interface class, with Item and Truck type respectively.

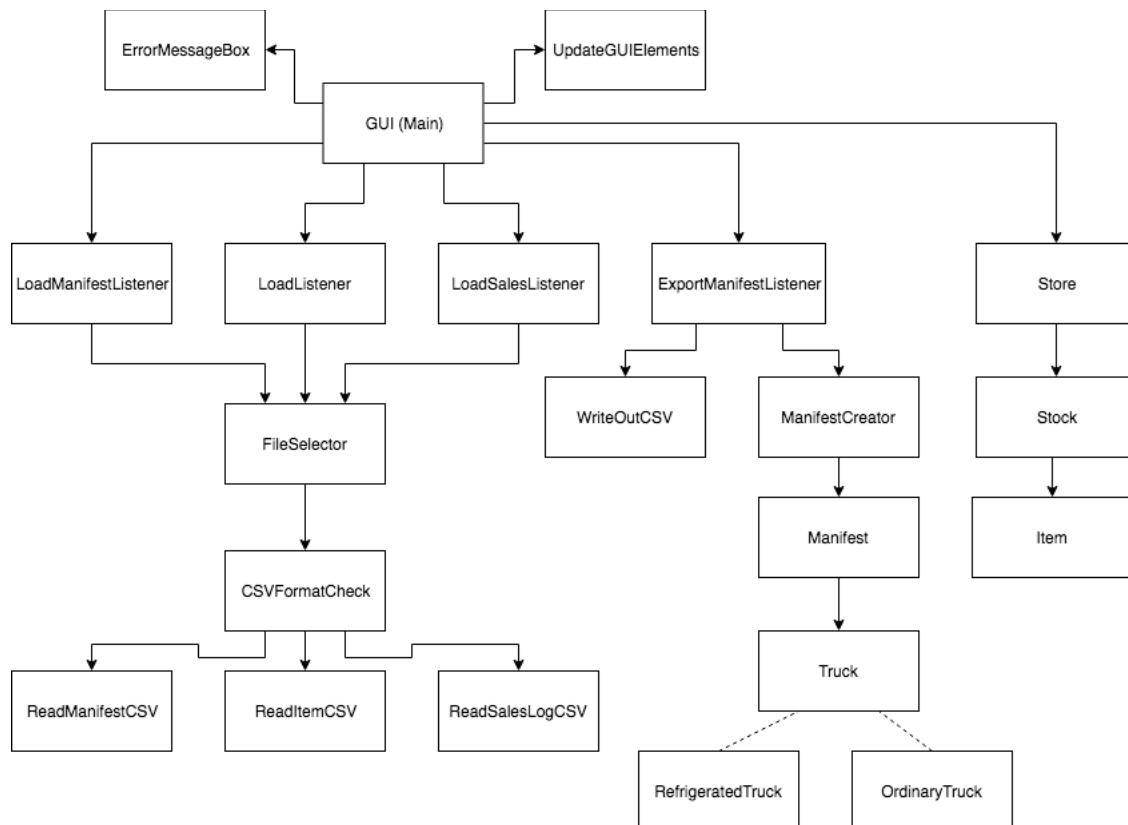
The Truck class is an abstract class that holds a series of abstract methods that are required for any child of truck. RefrigeratedTruck and OrdinaryTruck both inherit from the Truck class and extend its abstract methods with slight variations, such as differences in getCost() and getMaxCargo() that are specific to the truck type. Some methods specified in the Truck abstract class are added as necessary. For example, RefrigeratedTruck adds a method to get the temperature that is not applicable to OrdinaryTruck (getTemperature()).

Polymorphism is used in the ManifestCreator class, where the abstract Truck class is used when creating the interface to access methods such as getMaxCargo() or addItem() without

having to specify if it as a RefrigeratedTruck or OrdinaryTruck. This is an example where the parent class (Truck) is used to refer to the child class (OrdinaryTruck or RefrigeratedTruck).

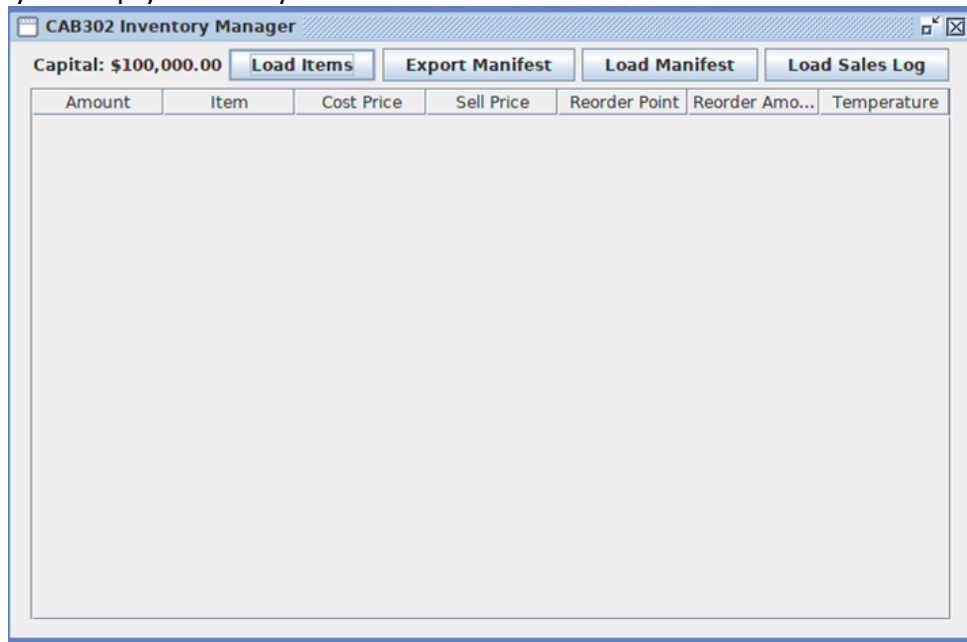
Standard software design patterns are also used extensively in this application. Namely, the Store class is an implementation of the singleton design pattern. Only one instance of Store can exist, with its constructor made protected so that it cannot be called directly. The UpdateGUIElements class also implements this same design pattern, so that the table (displaying the inventory) and label (displaying the store capital) are both singletons and only have one instance.

The following diagram demonstrates a simplified interactions between the classes.

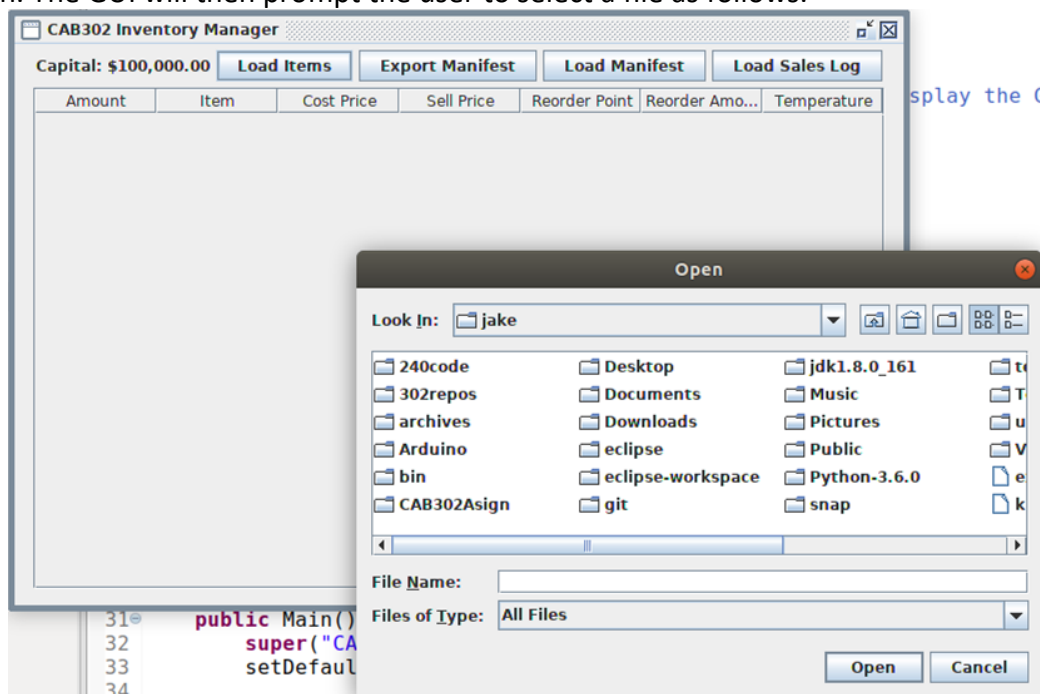


GUI Test

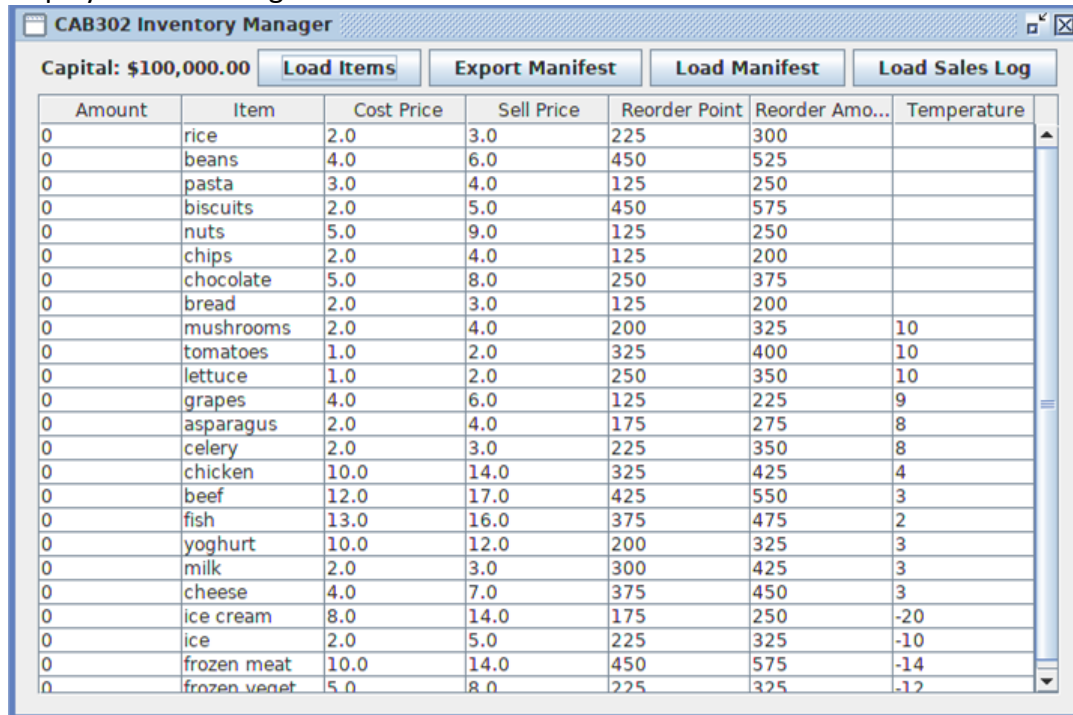
This section of the report offers the client a description of the programs Graphical User Interface (GUI) and the functionality it offers. When the program is first opened it will display an empty inventory as follows.



The user will have to import the store's inventory from a CSV file by clicking the Load items button. The GUI will then prompt the user to select a file as follows.



The GUI will then fill its table with the store's inventory and each item's property, as displayed in the image below.



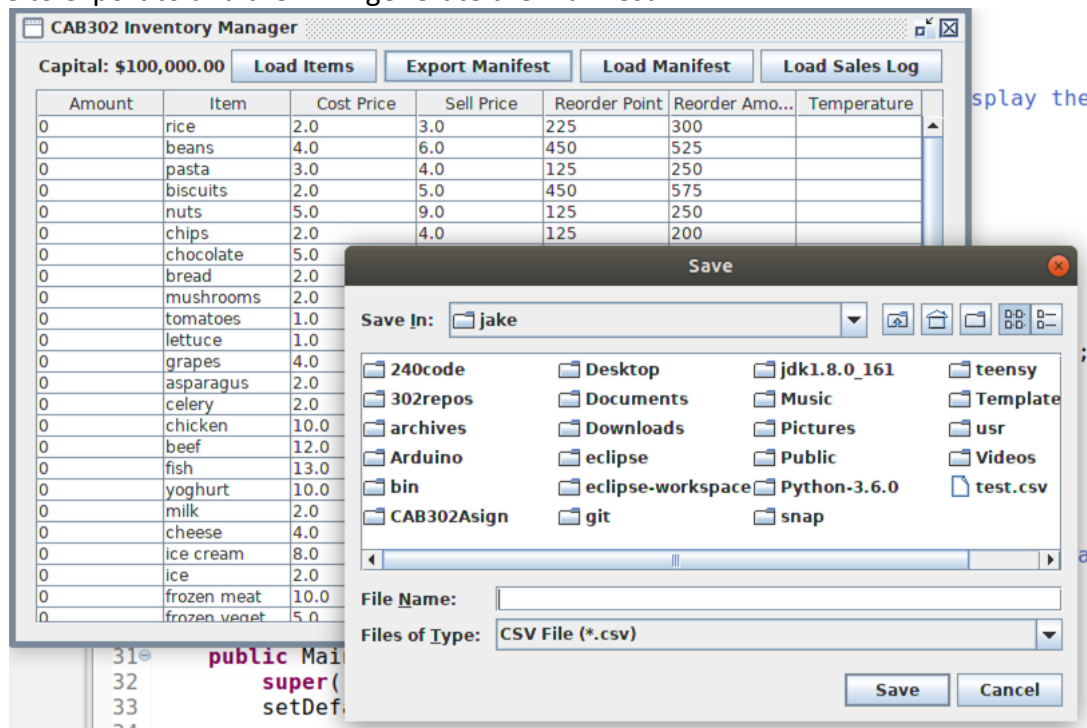
CAB302 Inventory Manager

Capital: \$100,000.00 Load Items Export Manifest Load Manifest Load Sales Log

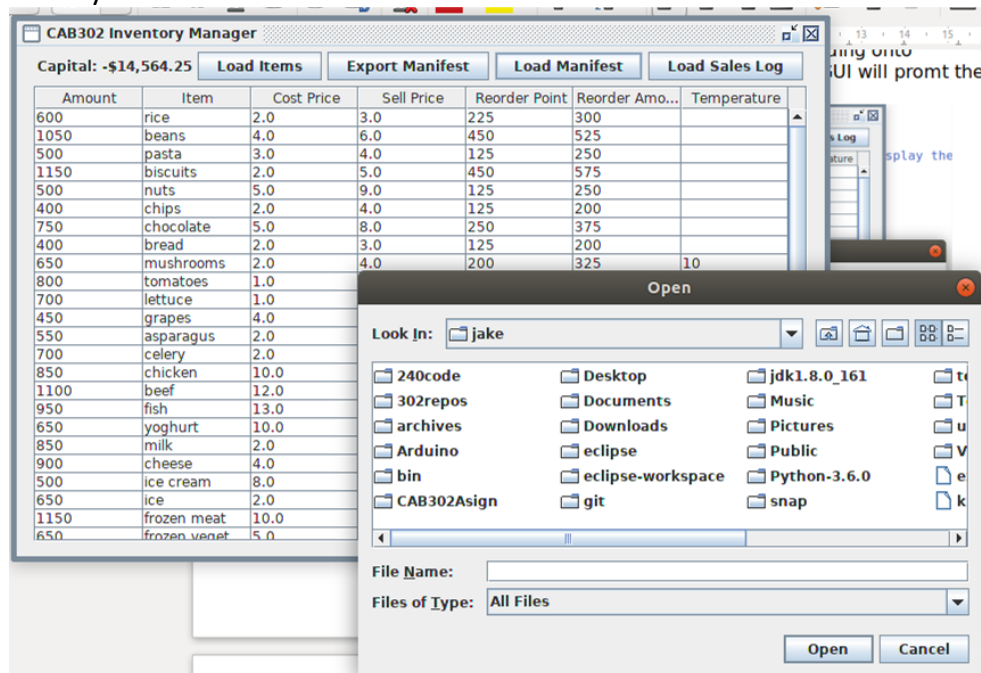
Amount	Item	Cost Price	Sell Price	Reorder Point	Reorder Amo...	Temperature
0	rice	2.0	3.0	225	300	
0	beans	4.0	6.0	450	525	
0	pasta	3.0	4.0	125	250	
0	biscuits	2.0	5.0	450	575	
0	nuts	5.0	9.0	125	250	
0	chips	2.0	4.0	125	200	
0	chocolate	5.0	8.0	250	375	
0	bread	2.0	3.0	125	200	
0	mushrooms	2.0	4.0	200	325	10
0	tomatoes	1.0	2.0	325	400	10
0	lettuce	1.0	2.0	250	350	10
0	grapes	4.0	6.0	125	225	9
0	asparagus	2.0	4.0	175	275	8
0	celery	2.0	3.0	225	350	8
0	chicken	10.0	14.0	325	425	4
0	beef	12.0	17.0	425	550	3
0	fish	13.0	16.0	375	475	2
0	yoghurt	10.0	12.0	200	325	3
0	milk	2.0	3.0	300	425	3
0	cheese	4.0	7.0	375	450	3
0	ice cream	8.0	14.0	175	250	-20
0	ice	2.0	5.0	225	325	-10
0	frozen meat	10.0	14.0	450	575	-14
0	frozen veget	5.0	8.0	225	325	-12

Once the store's inventory is loaded three major elements of functionality are available, Export Manifest, Load Manifest and Load Sales Log.

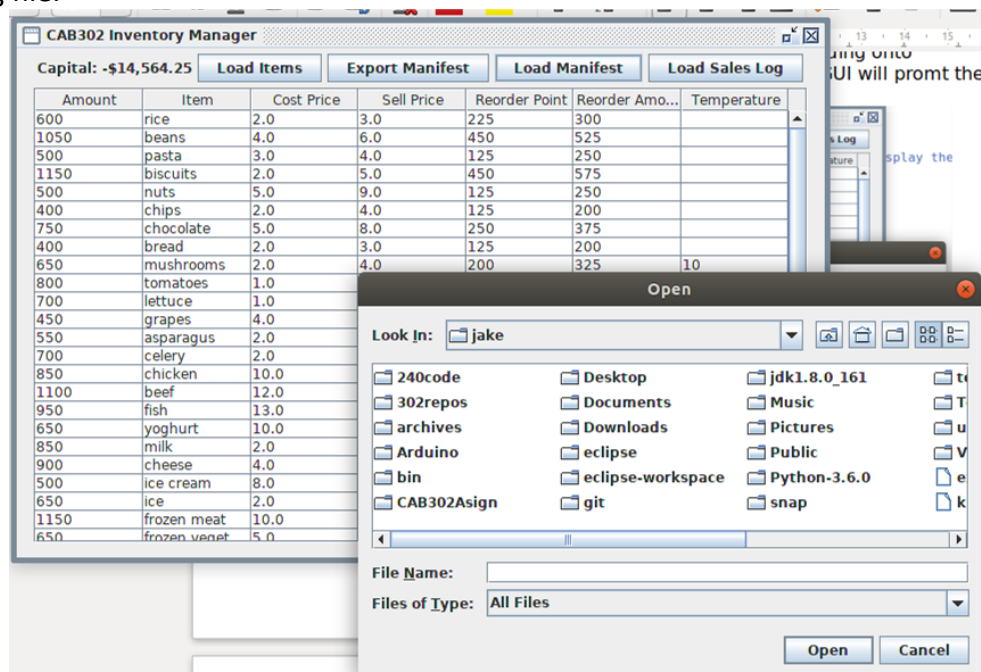
Export Manifest generates a purchase order for the store's supplier finding the most cost-efficient way of loading the store's required goods onto delivery trucks. Upon selecting the "Export Manifest" button the GUI will prompt the user to select a directory and file name to export to and then will generate the manifest.



Load Manifest updates the store's inventory and capital when the goods are delivered from the store's supplier. When the user selects the load manifest the GUI will prompt the user to select a delivery manifest.



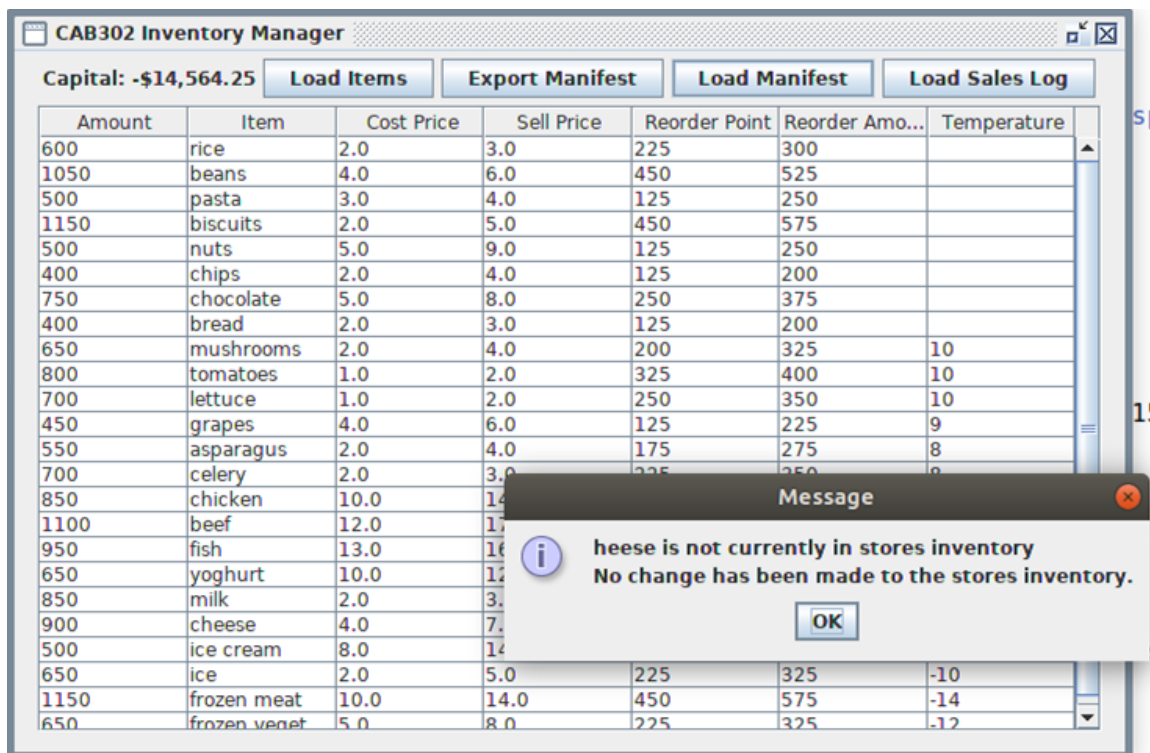
Load sales log updates the store's inventory and capital when goods have been sold in the store. When the "Load Sales log" button is pressed the GUI prompts the user to select a sales log file.



The program also offers a sweet of safe guard's which protect the store's inventory from corruption. If the user try's to;

- Import the wrong file type.
- Import a file with the incorrect format
- Import a file which tries to update the properties of an item which doesn't exist in the store's inventory
- tries to add a negative quantity of an item
- tries to remove more items from the store than exists

The GUI will not update the stores item properties and the user will be informed of the error with a pop-up window describing the exact cause of the problem, so the user can fix the error. An example will can be seen below.



Finally, the program only expects CSV's for the sales Log, manifest and item properties formatted in a specific way the exact format can be seen below.

Item properties

[item],[cost],[price],[reorder point],[reorder amount],[temperature]
[item],[cost],[price],[reorder point],[reorder amount],[temperature]
[item],[cost],[price],[reorder point],[reorder amount],[temperature]

Items that are not temperature controlled do not have the [temperature] value.

These items instead have the following grammar:

[item],[cost],[price],[reorder point],[reorder amount]

Manifest

>[truck type]
[item],[quantity]
[item],[quantity]
[item],[quantity]
>[truck type]
[item],[quantity]
[item],[quantity]
[item],[quantity]

Sales log

[item],[quantity]
[item],[quantity]
[item],[quantity]