CAB302 Assessment 2 Report

Kwun Hyo Lee (N9748482)

Jonathan Wai (N7570080)

Unit: CAB302 Software Development

Git: https://lkwunhyo@bitbucket.org/lkwunhyo/cab302assign2.git

Program Architecture and Object-Oriented Programming Design (OOP) Concepts

To achieve robust software architecture, the SuperMart Project was designed with consideration for the user, the system, and potential changes in business goals. Defining a structured solution which met all technical and operational needs, optimised the software for functionality, ease of use, flexibility, and scalability.

Implementation of the four main pillars of OOP, including abstraction, inheritance, encapsulation, and polymorphism, helped achieved such goals. The use of OOP provided the application a clear modular structure, defining a clear interface while keeping its implementation details hidden. Following the OOP structure also eliminated the need for duplicate code and improved code readability and maintainability.

The structure and class hierarchy of the application is as illustrated below,

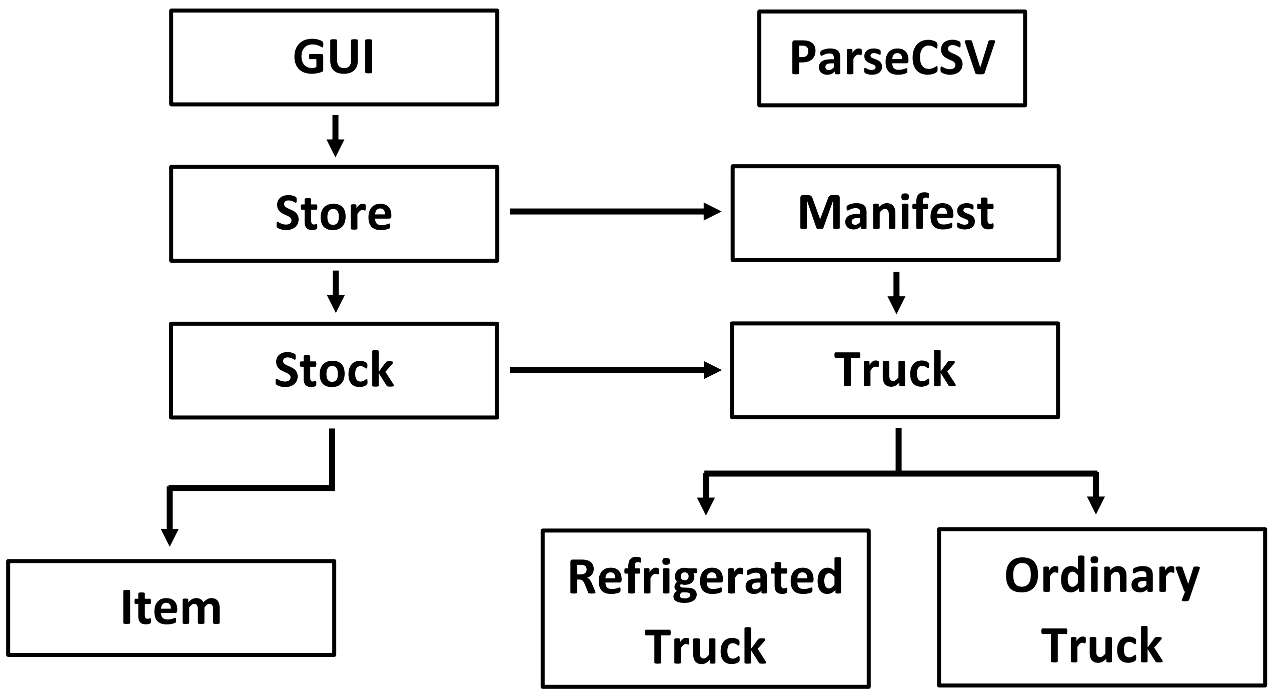


Figure 1: Class Hierarchy of the SuperMart Project

Item

As represented in Figure 1, the Item class forms a solid foundation for the entire application. The Item class represents a store item, which contains several properties, including the item name, cost, selling price, reordering point, reordering amount, and temperature. Encapsulation was used within the Item class getters and setters, to hide internal data and restrict access from other modules.

Stock

The Stock class represents a collection of items, stored within a LinkedHashMap of Item keys and an Integer quantity. Store inventory, stock orders, sales logs, and truck cargos can therefore take form of a Stock. The Stock class also utilises encapsulation to hide implementation details and mechanisms, and to restrict access from outside modules.

Truck, RefrigeratedTruck, and OrdinaryTruck

The RefrigeratedTruck and OrdinaryTruck classes hold specific constants and inherit from the abstract Truck class. The Truck class contains a Stock named cargo, and other necessary functions to calculate costs and other properties, according to the given cargo. Abstraction was used within the Truck class to expose essential features while hiding implementation mechanisms. Doing so improved code readability and flexibility as shown in appendix 1.

The Truck class also demonstrates polymorphism, as a Truck object can take form of either a RefrigeratedTruck or OrdinaryTruck object.

Manifest

The Manifest class organises Truck type objects into an ArrayList, and provides functionalities to handle writing and exporting manifests.

Store, and ParseCSV

The Store class adopts the Stock and Manifest class objects, providing new functionalities. These include loading an items property list, processing manifest and sales logs, and managing an inventory. ParseCSV is another class which is utilised by the Store class, solely for CSV formatting functionalities, and can be excluded from the OOP design. The Store class also adopts the Singleton Design Pattern, to eliminate potential multiple instantiation of the Store object. This was necessary as only a single store should exist.

GUI

The GUI class accepts a store object as a parameter, designing a user-friendly graphical interface to help perform functionalities of the given Store object.

StockException, DeliveryException, and CSVFormatException

The SuperMart Project also includes custom Exception classes, including StockException, DeliveryException, and the CSVFormatException classes. As the custom Exception classes are a type of Exception, they all use inheritance to utilise functionalities of their super class, Exception. The GUI class also uses inheritance by extending the JFrame class, as the GUI object is an instantiation of a JFrame.

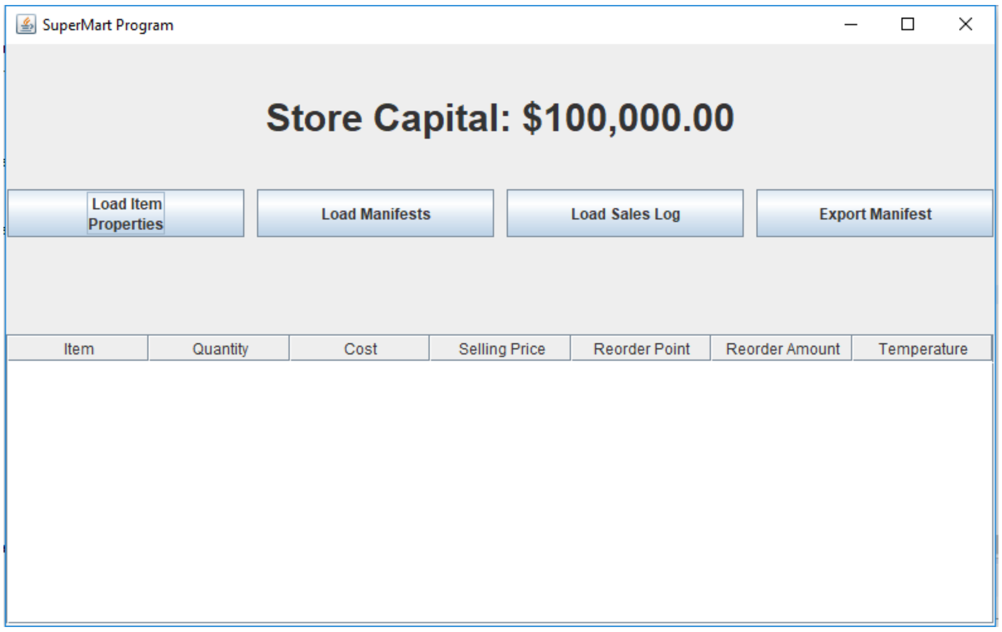
Conclusions

As previously discussed, the back-end implementation of the SuperMart Project is clearly optimised for functionality, flexibility, and scalability, by the use of OOP principles, and the consideration of the users, and the system.

SuperMart Graphical User Interface (GUI)

Main Frame

The main frame of the SuperMart Project is as shown below,


SuperMart Program

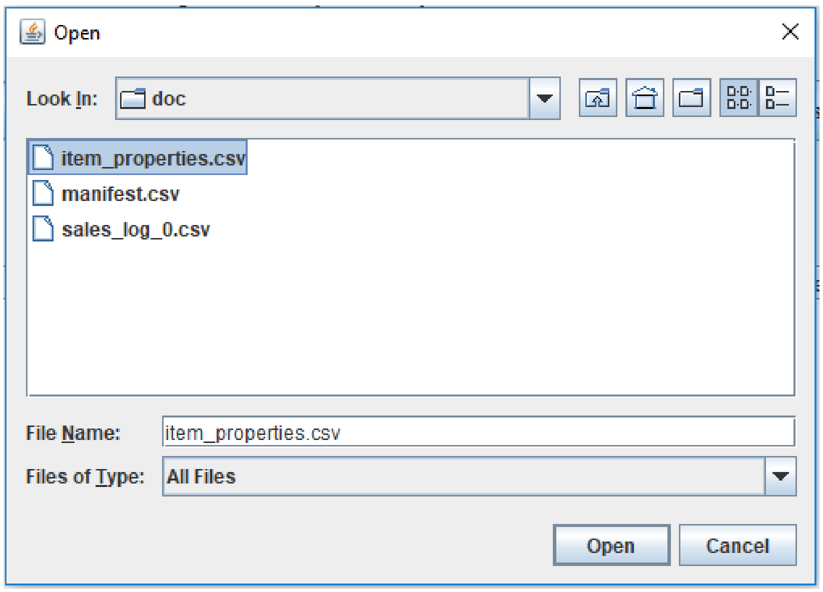
Store Capital: $100,000.00

Item	Quantity	Cost	Selling Price	Reorder Point	Reorder Amount	Temperature

The screenshot above shows the initial store capital set to $100,000.00, and displays various buttons and a table. The initial table is blank as shown, however, is altered by the functionalities of the buttons. The buttons include a load items property, load sales log, load manifest, and an export manifest function.

Load Properties

When the user clicks on the 'Load Item Properties' button, they are met with a JFileChooser window as shown below.


Open

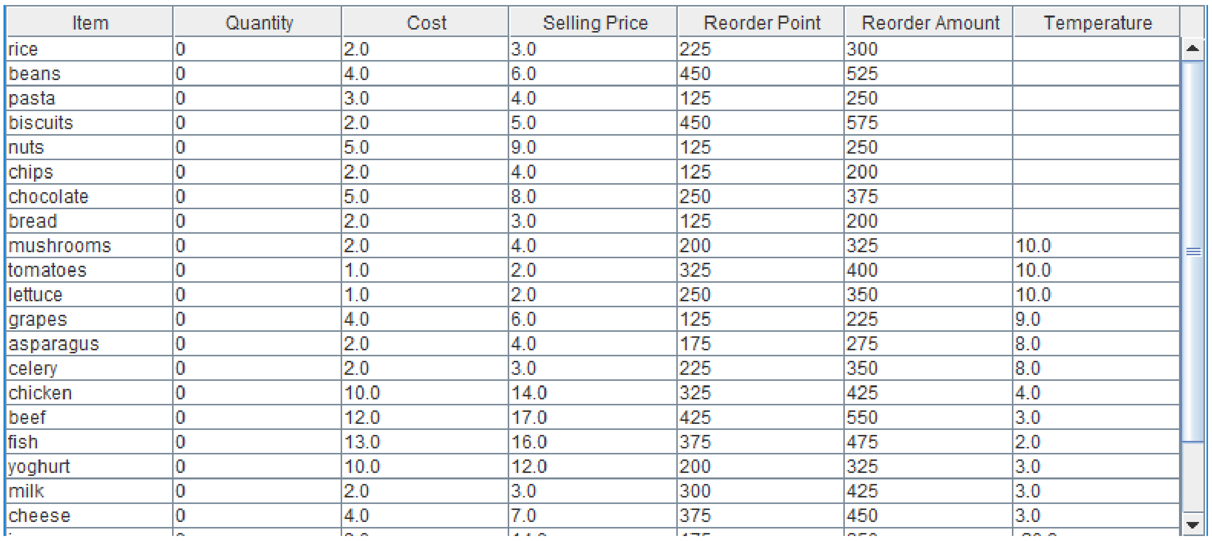
Look In: doc

item_properties.csv
manifest.csv
sales_log_0.csv

File Name: item_properties.csv

Files of Type: All Files

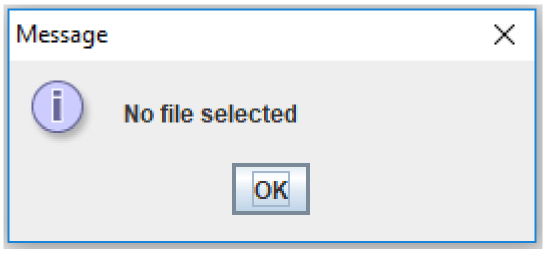
The user is required to navigate through their drive to search for a CSV file of item properties. Once selected, the JFileChooser window is closed and the table in the main window is updated with the items, their quantity, and their properties.



Item	Quantity	Cost	Selling Price	Reorder Point	Reorder Amount	Temperature
rice	0	2.0	3.0	225	300	
beans	0	4.0	6.0	450	525	
pasta	0	3.0	4.0	125	250	
biscuits	0	2.0	5.0	450	575	
nuts	0	5.0	9.0	125	250	
chips	0	2.0	4.0	125	200	
chocolate	0	5.0	8.0	250	375	
bread	0	2.0	3.0	125	200	
mushrooms	0	2.0	4.0	200	325	10.0
tomatoes	0	1.0	2.0	325	400	10.0
lettuce	0	1.0	2.0	250	350	10.0
grapes	0	4.0	6.0	125	225	9.0
asparagus	0	2.0	4.0	175	275	8.0
celery	0	2.0	3.0	225	350	8.0
chicken	0	10.0	14.0	325	425	4.0
beef	0	12.0	17.0	425	550	3.0
fish	0	13.0	16.0	375	475	2.0
yoghurt	0	10.0	12.0	200	325	3.0
milk	0	2.0	3.0	300	425	3.0
cheese	0	4.0	7.0	375	450	3.0

As the initial store does not have any quantity of items, the table will show 0 items at initiation.

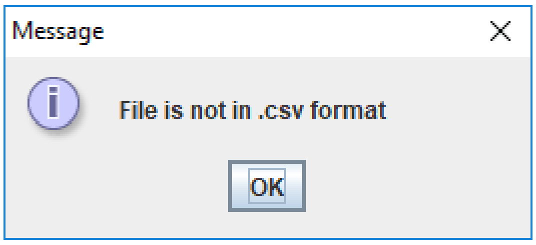
If the user fails to select a file, the user is notified by a dialog box that no file was selected.



Message

No file selected

If the user selects an invalid file, the user is notified that a file did not meet the specifications of the CSV format.

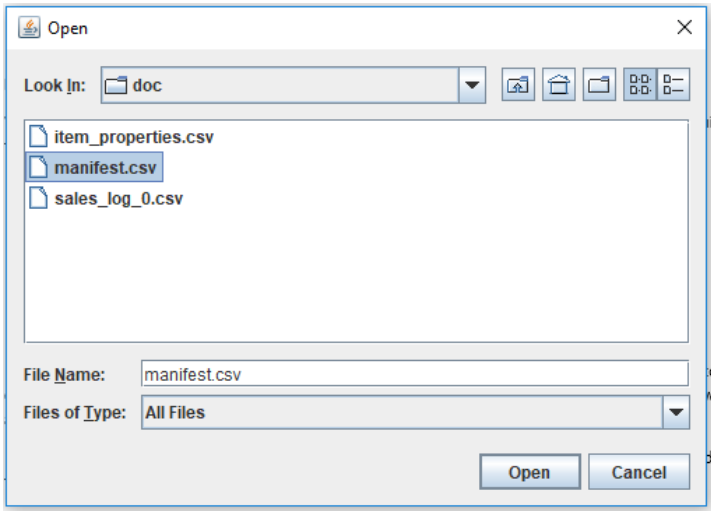


Message

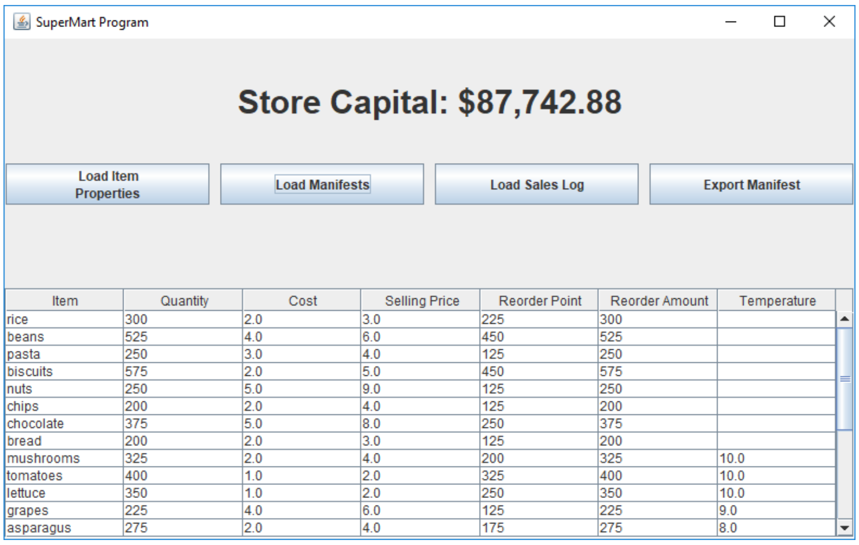
File is not in .csv format

Load Manifest

When the ‘Load Manifest’ button is clicked, the user is faced with a JFileChooser window, requiring the user to search for a valid manifest CSV.



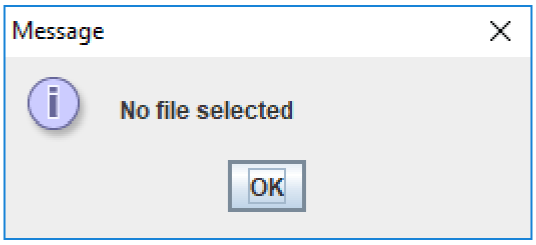
If a valid file is selected, the window is closed, and the quantity of items is increased and the store capital is decreased accordingly. The table in the main window is also updated to show the newly adjusted values.



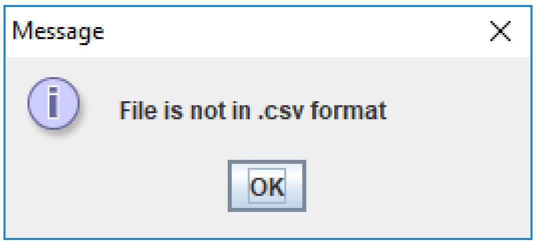
Store Capital: $87,742.88

Item	Quantity	Cost	Selling Price	Reorder Point	Reorder Amount	Temperature
rice	300	2.0	3.0	225	300	
beans	525	4.0	6.0	450	525	
pasta	250	3.0	4.0	125	250	
biscuits	575	2.0	5.0	450	575	
nuts	250	5.0	9.0	125	250	
chips	200	2.0	4.0	125	200	
chocolate	375	5.0	8.0	250	375	
bread	200	2.0	3.0	125	200	
mushrooms	325	2.0	4.0	200	325	10.0
tomatoes	400	1.0	2.0	325	400	10.0
lettuce	350	1.0	2.0	250	350	10.0
grapes	225	4.0	6.0	125	225	9.0
asparagus	275	2.0	4.0	175	275	8.0

If a file is not selected, the user will be notified via a dialog box and if the user selects an invalid file, the user will be presented with a dialog box notifying that the file is invalid for use.



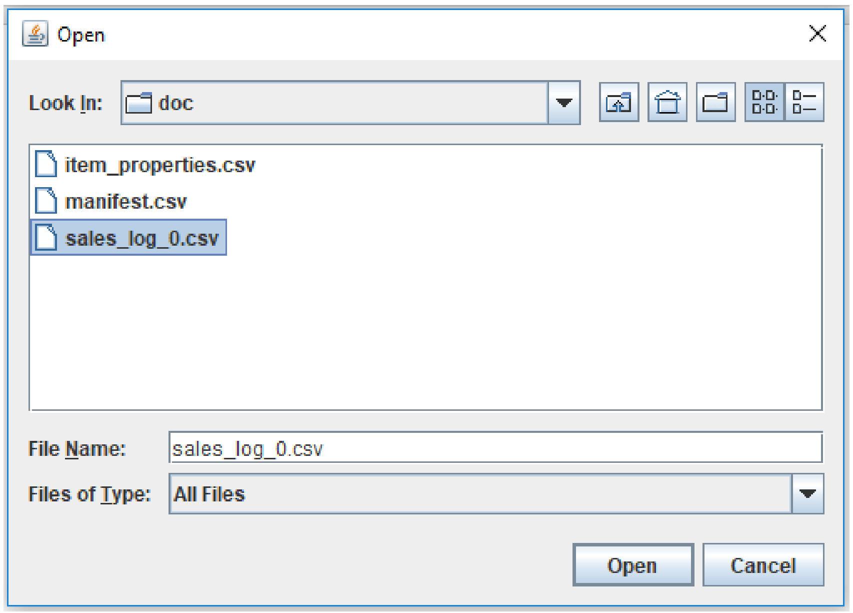
No file selected



File is not in .csv format

Load Sales Log

When the user clicks on the ‘Load Sales Log’ button, a JFileChooser window is opened. The user is required to navigate their drive to search for a valid CSV file of their sales log. Once a valid file is selected, the store capital is automatically increased and the item quantity is decreased accordingly.

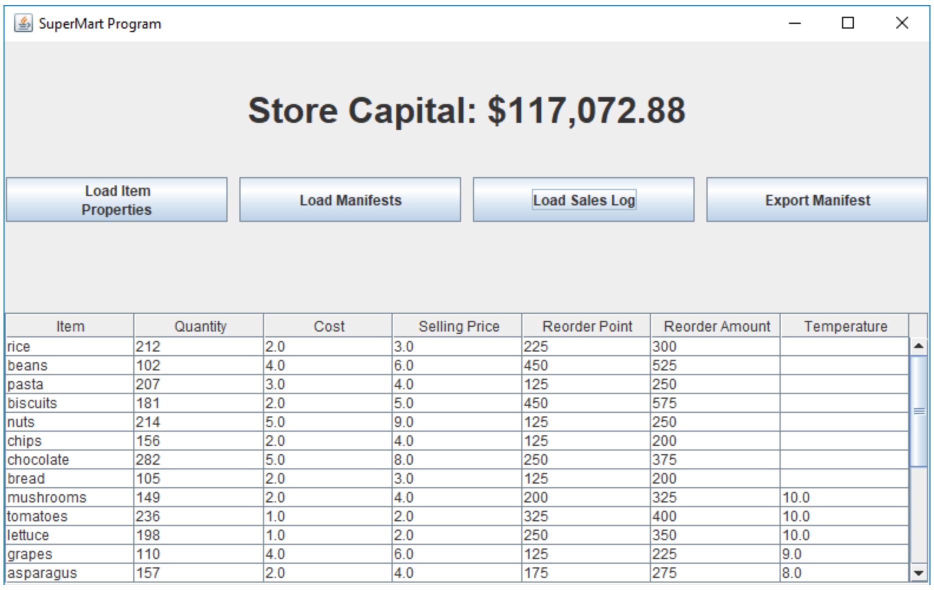


Open

Look In: doc

item_properties.csv
manifest.csv
sales_log_0.csv

File Name: sales_log_0.csv
Files of Type: All Files



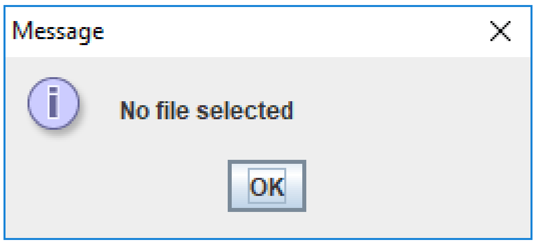
SuperMart Program

Store Capital: $117,072.88

Item	Quantity	Cost	Selling Price	Reorder Point	Reorder Amount	Temperature
rice	212	2.0	3.0	225	300	
beans	102	4.0	6.0	450	525	
pasta	207	3.0	4.0	125	250	
biscuits	181	2.0	5.0	450	575	
nuts	214	5.0	9.0	125	250	
chips	156	2.0	4.0	125	200	
chocolate	282	5.0	8.0	250	375	
bread	105	2.0	3.0	125	200	
mushrooms	149	2.0	4.0	200	325	10.0
tomatoes	236	1.0	2.0	325	400	10.0
lettuce	198	1.0	2.0	250	350	10.0
grapes	110	4.0	6.0	125	225	9.0
asparagus	157	2.0	4.0	175	275	8.0

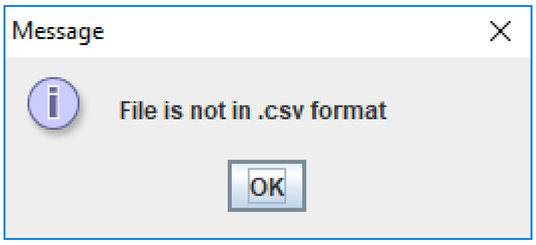
As shown above, the store capital increases from $87,742.88 to $117,072.88 after loading the sales_log0.csv file.

If the user fails to select a file, the user is met with a dialog box, notifying that a file was not selected, and if the user selects an invalid file, a dialog box will notify the user that the file is invalid.



Message

No file selected

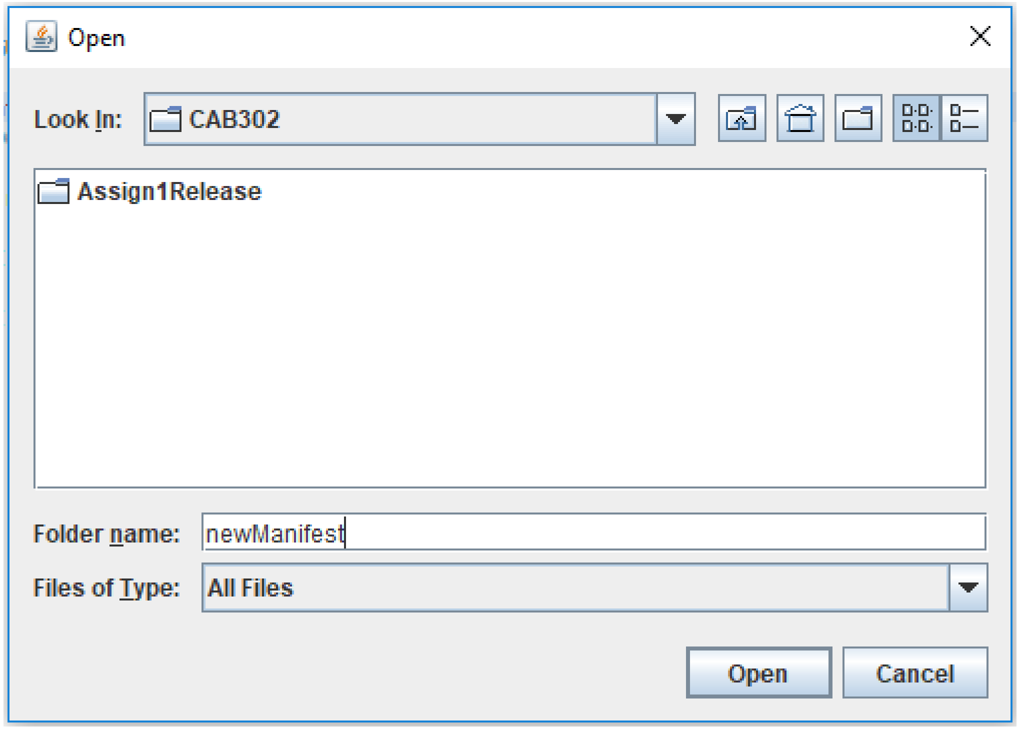


Message

File is not in .csv format

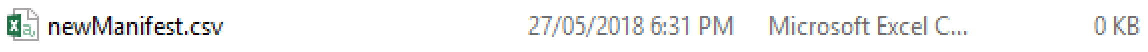
Export Manifest

When the ‘Export Manifest’ button is selected, the current store inventory will be scanned and a manifest will be generated, according to the items’ reorder point and reorder amount. The user will be prompted to save the manifest in a directory with a JFileChooser window as show below. The user will be required to input a name into the text box and press open to save a new CSV file.



Folder name: newManifest
Files of Type: All Files

Once a file path is selected, a manifest is generated and saved at the chosen file path.



newManifest.csv 27/05/2018 6:31 PM Microsoft Excel C... 0 KB

Appendices

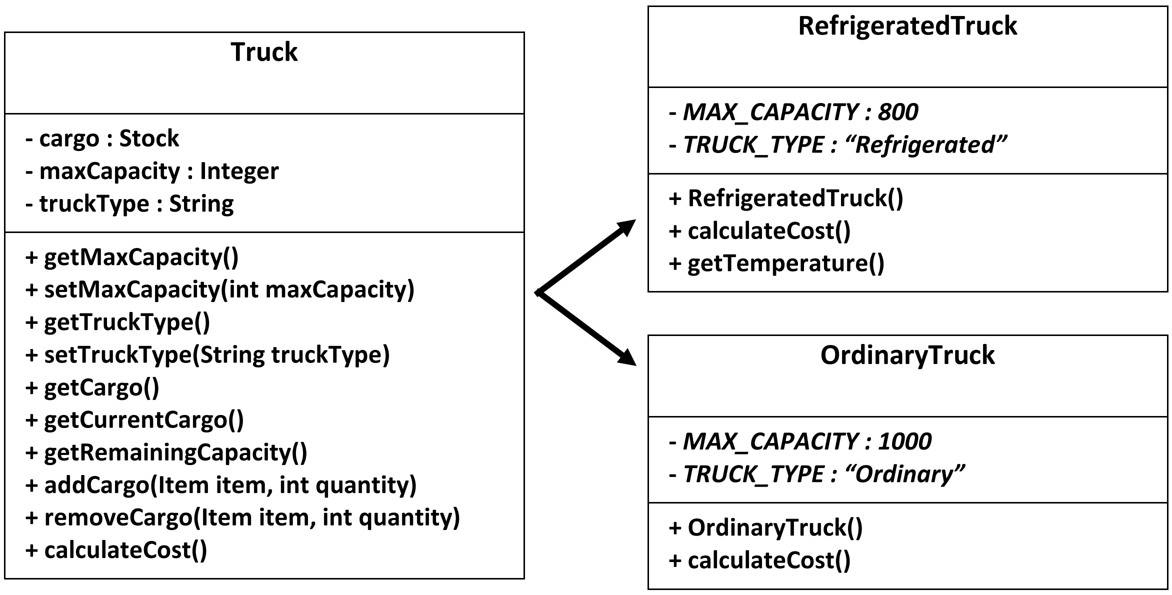


Figure 2: Abstraction of the Truck Class shows Essential Functionalities of a Truck Object