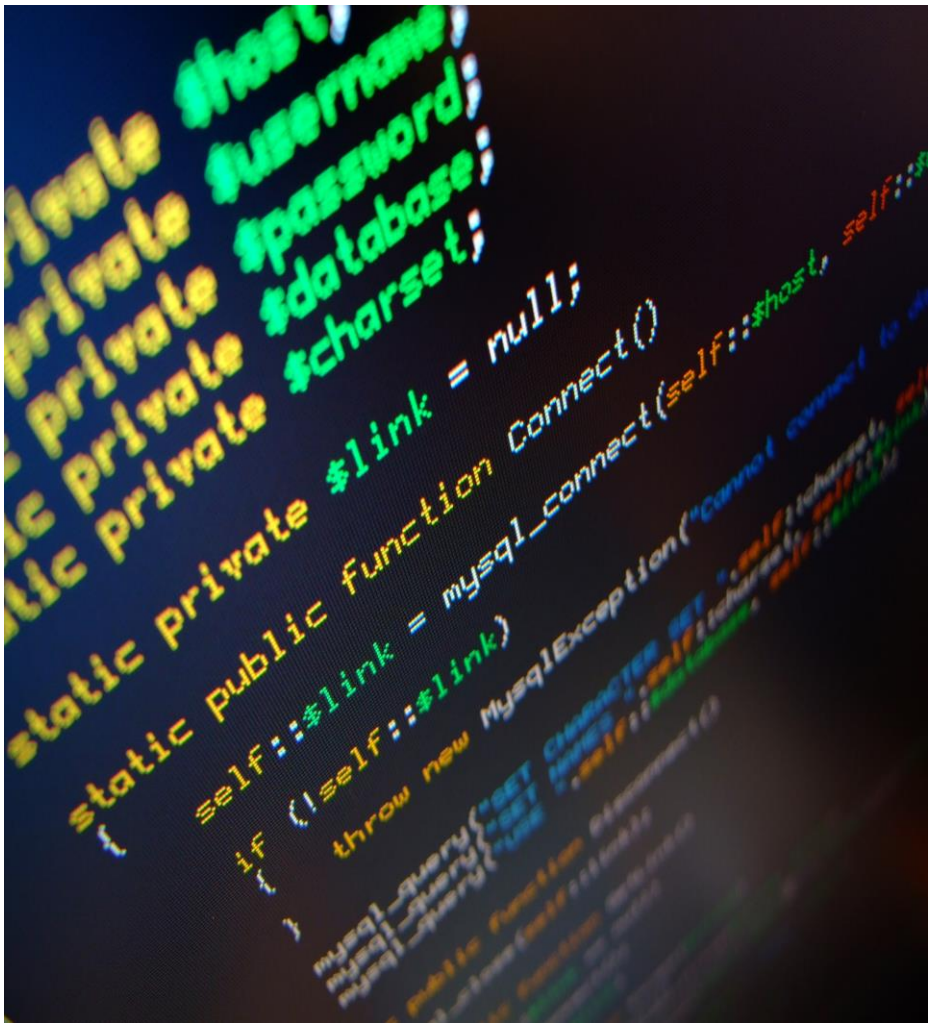


CAB302 Assignment 2

ASHLEY COTTRELL
RADHIMAS DJAN

n9721436 | n9549714 | 27/05/2018

Technical Description

The program uses the four fundamental object-oriented programming concepts encapsulation, inheritance, polymorphism and abstraction to hide complexity from the user. Encapsulation is the mechanism of wrapping variables and methods that act upon the data together into a single component this is known in Java as a class. Abstraction is where you only show the relevant data and hide unnecessary details from the user. Inheritance is the process where one class acquires the methods and fields of another class this is known in Java as a subclass. Polymorphism is the ability for an object to have many forms giving the ability to assign a different meaning or usage to something in different contexts.

STORE

The store class calls upon the stock, manifest and IOCSV class to perform functionality. The store class uses a singleton pattern to prevent the program from creating multiple stores. The store class uses abstraction to be able to perform most of its functionality, for example importing a CSV involves the store class calling the readCSVFile() function from IOCSV class and passing it a file location. All the logic and complexity of importing and reading the data is handled by this class which is useful for hiding the complexity of the program. The use of abstraction applies to all methods within the store class including importing CSV files, creating inventory, exporting manifests, dealing with sales logs and manifests files.

STOCK

The Stock class calls upon the item class for creating an item and gets called upon by store and truck classes. The stock class uses a constructor to create new Item classes, this allows the program to be able to handle any number of items and without needing to hardcode the items into the program. The stock class calls upon the item class to perform functionality.

ITEM

The Item class does not call upon any other classes within the program to perform functionality but is called upon by the stock class. Item uses encapsulation to prevent the rest of the program from accessing and editing variables that should not be changed. This is important because if a variable such as a sale price is changed and set to a negative, this means that the store will not only be giving the item away but also giving the customer money. The Item class uses overloaded constructor where it comes with two constructors to create refrigerated items and unrefrigerated items.

MANIFEST

Manifest is a collection of a truck where the constructor act as making a new ordinary and refrigerated truck, the process of adding more truck happen in here. Manifest act as a quite important class where it connected to all other classes and it is where the logic of import and export manifest happen which are the main feature of the program. The manifest

classes are connected to a supermart to get its instance so that it can get data needed from supermart, it also connected to a stock where it is a collection of the item, the stock is used as an inventory in the manifest.

TRUCK

Truck is an abstract class where it acts as the superclass of the refrigerated truck and ordinary truck. Since the Truck act as a superclass for the refrigerated and ordinary truck, inheritance will happen where the subclass which in this case refrigerated and ordinary truck will have the same method as its parent.

REFRIGERATED TRUCK

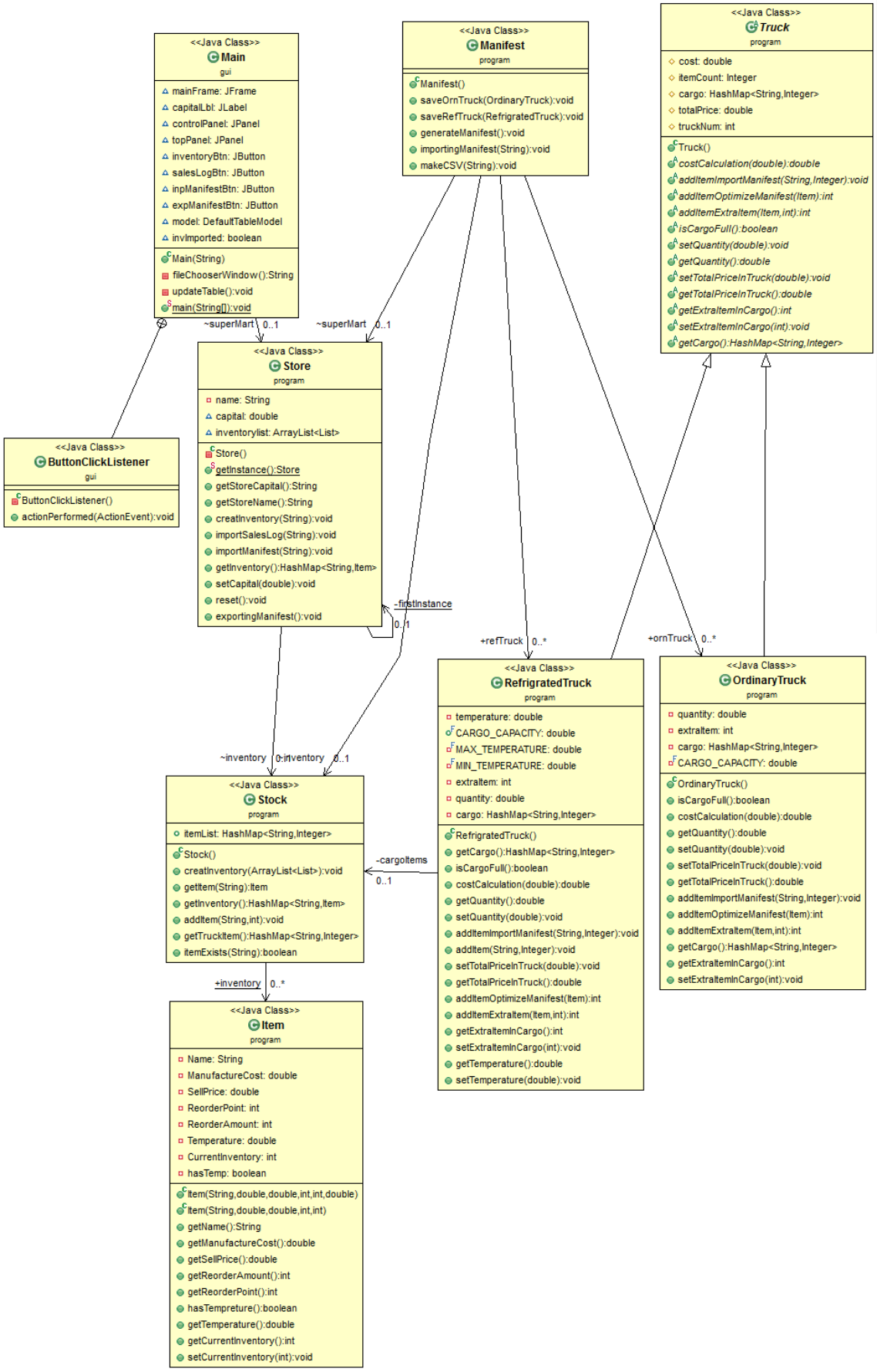
Refrigerated truck is the subclass of the Truck wherein the refrigerated truck is a truck that can hold a refrigerated item but also dry goods if needed, this class uses an encapsulation as well to prevent the rests of the program to change its variable. This class are connected to the manifest where it is used as a cargo for refrigerated and non-refrigerated items if needed.

ORDINARY TRUCK

Ordinary truck is the subclass of the Truck wherein the ordinary truck is a truck that can only hold a non-refrigerated item or dry goods, this class uses an encapsulation as well to prevent the rests of the program to change its variable. This class are connected to the manifest where it is used as a cargo for a non-refrigerated item

EXCEPTION CLASSES

CSV Format Exception, Delivery Exception and Stock Exception classes all inherit the Exception class by use of an extension. Polymorphism is in use within these classes because they have been set up to be able to create multiple types of exception such as plain exceptions and exceptions with messages. The exception classes are called by IOCSV, Store and Stock classes respectively for handling various errors associated with program functionality.


<<Java Class>>
Main
gui
mainFrame: JFrame
capitalLbl: JLabel
controlPanel: JPanel
topPanel: JPanel
inventoryBtn: JButton
salesLogBtn: JButton
inpManifestBtn: JButton
expManifestBtn: JButton
model: DefaultTableModel
invImported: boolean
Main(String)
fileChooserWindow():String
updateTable():void
main(String[]):void
<<Java Class>>
Manifest
program
Manifest()
saveOrnTruck(OrdinaryTruck):void
saveRefTruck(RefrigratedTruck):void
generateManifest():void
importingManifest(String):void
makeCSV(String):void
<<Java Class>>
Truck
program
cost: double
itemCount: Integer
cargo: HashMap<String,Integer>
totalPrice: double
truckNum: int
Truck()
costCalculation(double):double
addItemImportManifest(String,Integer):void
addItemOptimizeManifest(Item):int
addItemExtraItem(Item,int):int
isCargoFull():boolean
setQuantity(double):void
getQuantity():double
setTotalPriceInTruck(double):void
getTotalPriceInTruck():double
getExtraItemInCargo():int
setExtraItemInCargo(int):void
getCargo():HashMap<String,Integer>
~superMart
0..1
~superMart
0..1
<<Java Class>>
Store
program
name: String
capital: double
inventorylist: ArrayList<List>
Store()
getInstance():Store
getStoreCapital():String
getStoreName():String
creatInventory(String):void
importSalesLog(String):void
importManifest(String):void
getInventory():HashMap<String,Item>
setCapital(double):void
reset():void
exportingManifest():void
<<Java Class>>
ButtonClickListener
gui
ButtonClickListener()
actionPerformed(ActionEvent):void
-firstInstance
0..1
+refTruck
0..*
+ornTruck
0..*
<<Java Class>>
RefrigratedTruck
program
temperature: double
CARGO_CAPACITY: double
MAX_TEMPERATURE: double
MIN_TEMPERATURE: double
extraItem: int
quantity: double
cargo: HashMap<String,Integer>
RefrigratedTruck()
getCargo():HashMap<String,Integer>
isCargoFull():boolean
costCalculation(double):double
getQuantity():double
setQuantity(double):void
addItemImportManifest(String,Integer):void
addItem(String,Integer):void
setTotalPriceInTruck(double):void
getTotalPriceInTruck():double
addItemOptimizeManifest(Item):int
addItemExtraItem(Item,int):int
getExtraItemInCargo():int
setExtraItemInCargo(int):void
getTemperature():double
setTemperature(double):void
<<Java Class>>
OrdinaryTruck
program
quantity: double
extraItem: int
cargo: HashMap<String,Integer>
CARGO_CAPACITY: double
OrdinaryTruck()
isCargoFull():boolean
costCalculation(double):double
getQuantity():double
setQuantity(double):void
setTotalPriceInTruck(double):void
getTotalPriceInTruck():double
addItemImportManifest(String,Integer):void
addItemOptimizeManifest(Item):int
addItemExtraItem(Item,int):int
getCargo():HashMap<String,Integer>
getExtraItemInCargo():int
setExtraItemInCargo(int):void
~inventory
0..1
0..1
<<Java Class>>
Stock
program
itemList: HashMap<String,Integer>
Stock()
creatInventory(ArrayList<List>):void
getItem(String):Item
getInventory():HashMap<String,Item>
addItem(String,int):void
getTruckItem():HashMap<String,Integer>
itemExists(String):boolean
-cargoItems
0..1
+inventory
0..*
<<Java Class>>
Item
program
Name: String
ManufactureCost: double
SellPrice: double
ReorderPoint: int
ReorderAmount: int
Temperature: double
CurrentInventory: int
hasTemp: boolean
Item(String,double,double,int,int,double)
Item(String,double,double,int,int)
getName():String
getManufactureCost():double
getSellPrice():double
getReorderAmount():int
getReorderPoint():int
hasTempreture():boolean
getTemperature():double
getCurrentInventory():int
setCurrentInventory(int):void

GUI Test Report

Note: Due to my PC having a high DPI display and Java 8 not having support for high dpi scaling the pictures below will appear to have small buttons, slight blurriness in text and other issues such as the window title being big compared to the rest of the program.

When the program is first to run export manifest, import manifest and import sales log button are disabled (figure 1)

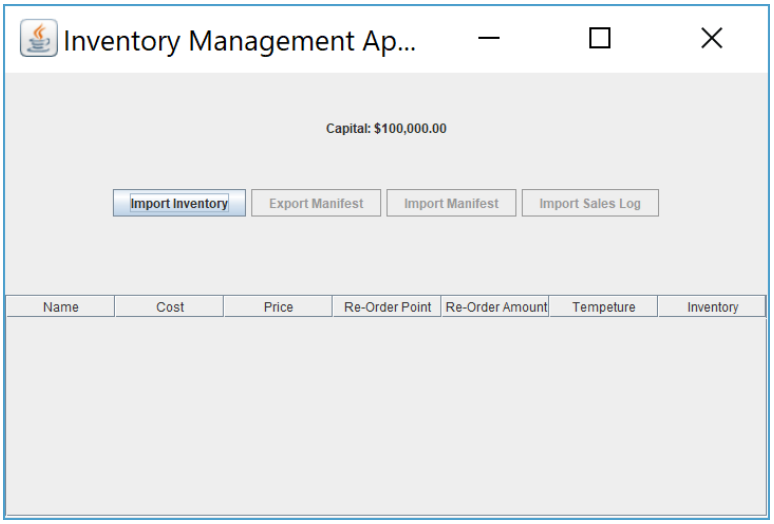


Figure 1

Upon Clicking the import inventory button, a file chooser window will upon Which will allow the user to select a CSV file. The file chooser window has been set up to only show CSV files unless the user changes the file type to all files. (figure 2)

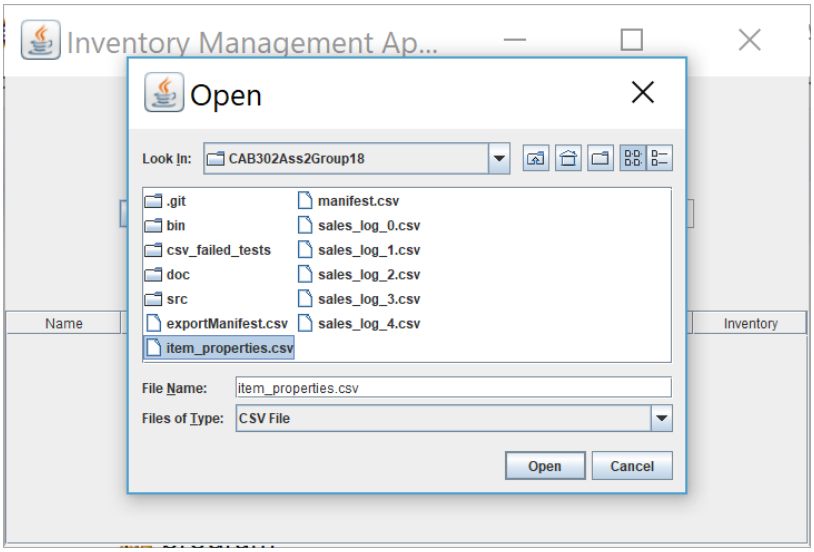
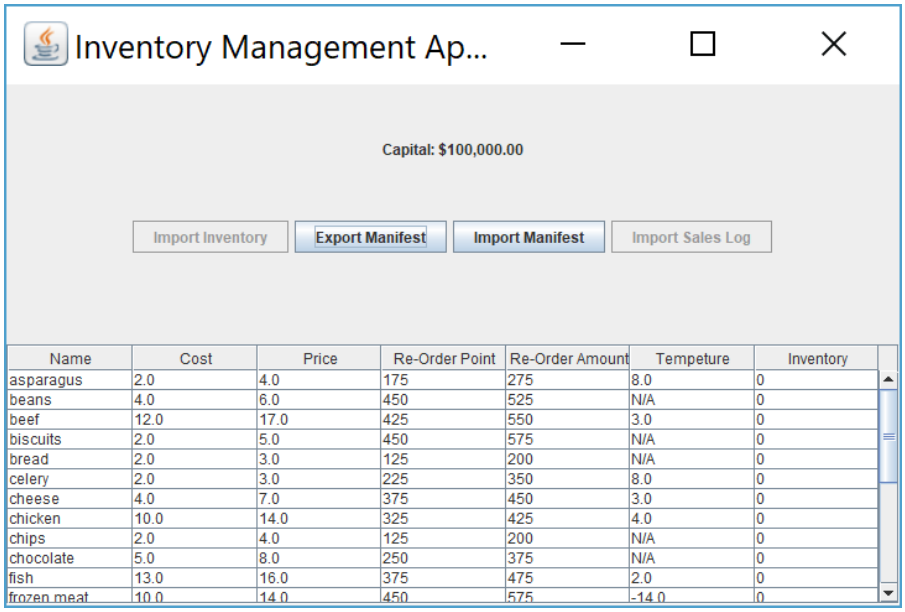


Figure 2

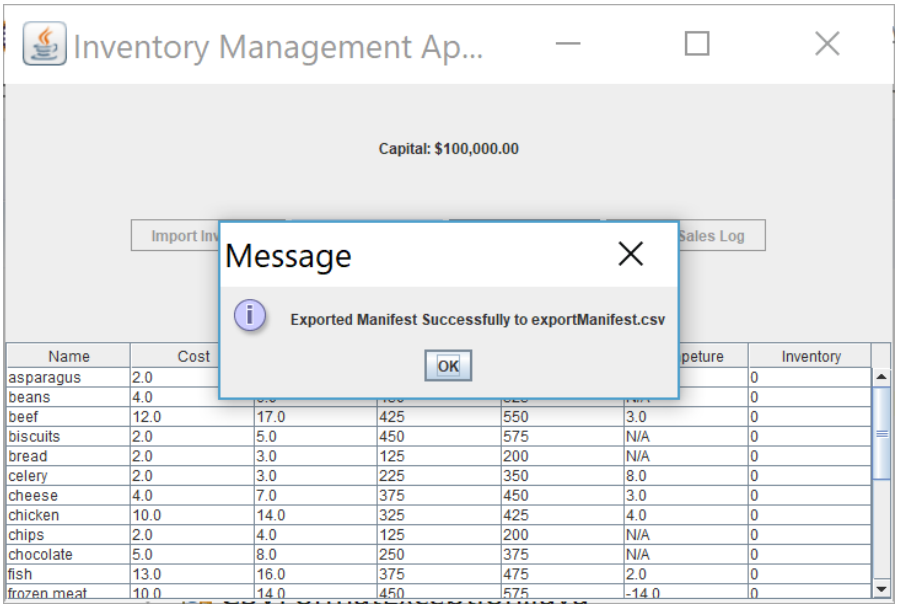
Once the program imports and item properties file the import inventory button will be disabled and the export manifest button will be enabled. The inventory will also be sorted and displayed in a table, allowing for easy access to the items information. (figure 3)



Name	Cost	Price	Re-Order Point	Re-Order Amount	Tempeture	Inventory
asparagus	2.0	4.0	175	275	8.0	0
beans	4.0	6.0	450	525	N/A	0
beef	12.0	17.0	425	550	3.0	0
biscuits	2.0	5.0	450	575	N/A	0
bread	2.0	3.0	125	200	N/A	0
celery	2.0	3.0	225	350	8.0	0
cheese	4.0	7.0	375	450	3.0	0
chicken	10.0	14.0	325	425	4.0	0
chips	2.0	4.0	125	200	N/A	0
chocolate	5.0	8.0	250	375	N/A	0
fish	13.0	16.0	375	475	2.0	0
frozen meat	10.0	14.0	450	575	-14.0	0

Figure 3

Upon clicking the export manifest button, the program will export the manifest to a CSV file called exportManifest.csv in the program directory and will display a message informing the user that the file has been exported. (figure 4)



Name	Cost				peture	Inventory
asparagus	2.0					0
beans	4.0					0
beef	12.0	17.0	425	550	3.0	0
biscuits	2.0	5.0	450	575	N/A	0
bread	2.0	3.0	125	200	N/A	0
celery	2.0	3.0	225	350	8.0	0
cheese	4.0	7.0	375	450	3.0	0
chicken	10.0	14.0	325	425	4.0	0
chips	2.0	4.0	125	200	N/A	0
chocolate	5.0	8.0	250	375	N/A	0
fish	13.0	16.0	375	475	2.0	0
frozen meat	10.0	14.0	450	575	-14.0	0

Figure 4

Upon successful export, the user will click the import manifest button and upon clicking the button a file chooser window will open and the user will select the exported manifest file. (figure 5)

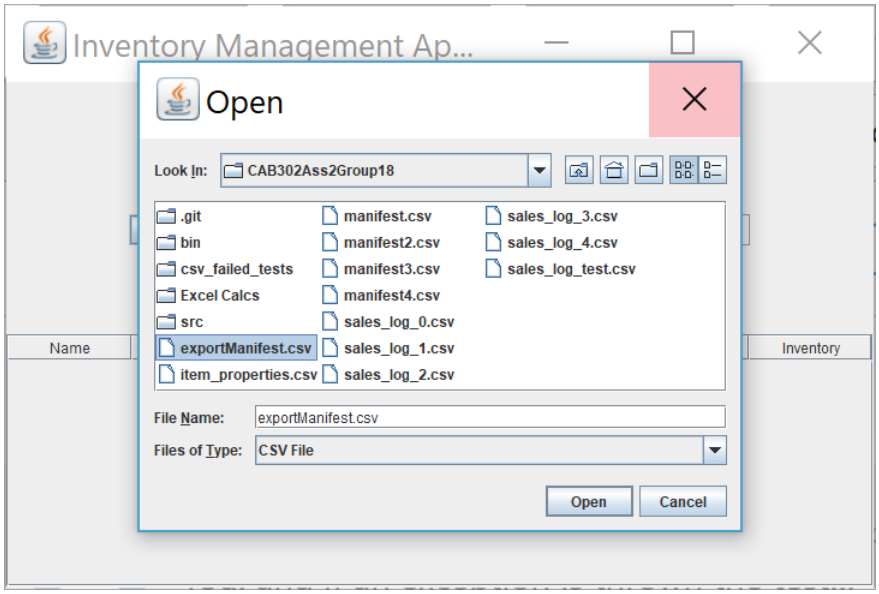
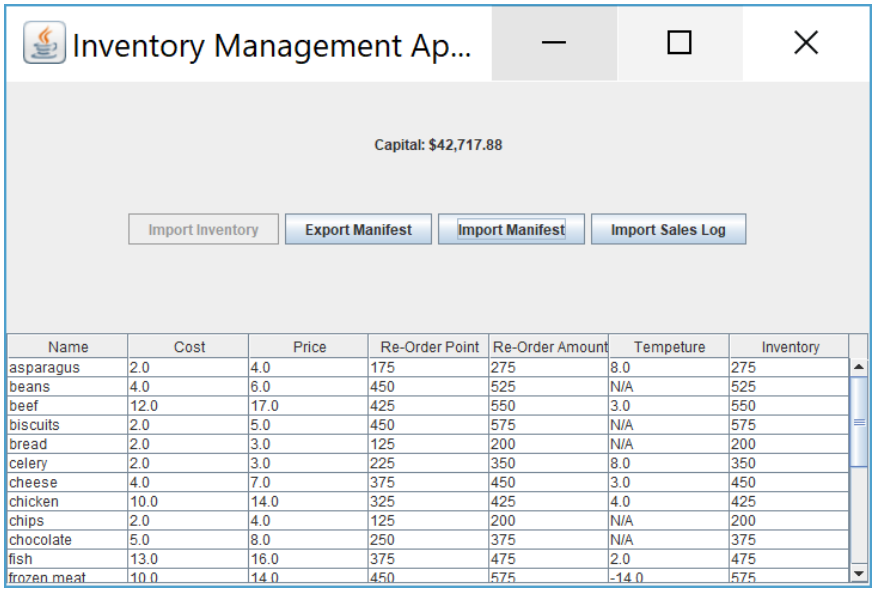


Figure 5

Upon success full import the table and capital will be updated accordingly and the sales log button is enabled and the program is fully set up and ready to be used. (figure 6) The user can now import sales logs which involve selecting a sales log using the file chooser window. (figure 7)



Name	Cost	Price	Re-Order Point	Re-Order Amount	Tempeture	Inventory
asparagus	2.0	4.0	175	275	8.0	275
beans	4.0	6.0	450	525	N/A	525
beef	12.0	17.0	425	550	3.0	550
biscuits	2.0	5.0	450	575	N/A	575
bread	2.0	3.0	125	200	N/A	200
celery	2.0	3.0	225	350	8.0	350
cheese	4.0	7.0	375	450	3.0	450
chicken	10.0	14.0	325	425	4.0	425
chips	2.0	4.0	125	200	N/A	200
chocolate	5.0	8.0	250	375	N/A	375
fish	13.0	16.0	375	475	2.0	475
frozen meat	10.0	14.0	450	575	-14.0	575

Figure 6

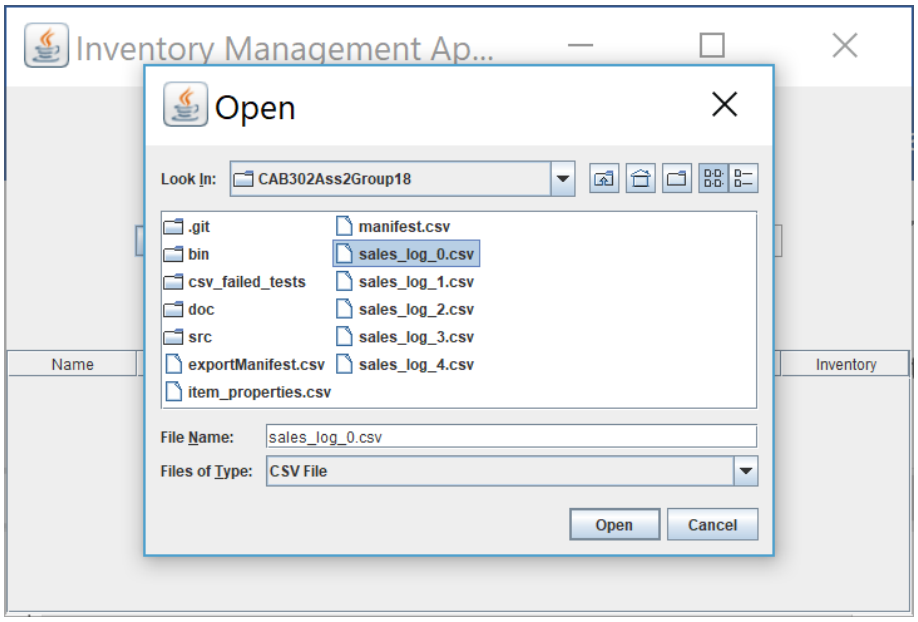
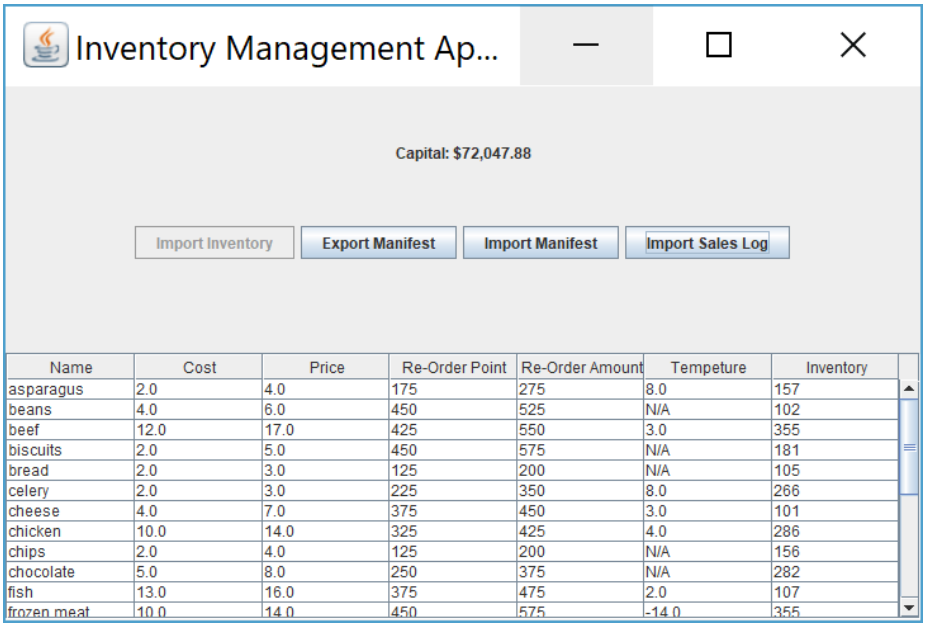


Figure 7

The program upon successful import will update capital and the table to show the updated data. (figure 8)



Name	Cost	Price	Re-Order Point	Re-Order Amount	Tempeture	Inventory
asparagus	2.0	4.0	175	275	8.0	157
beans	4.0	6.0	450	525	N/A	102
beef	12.0	17.0	425	550	3.0	355
biscuits	2.0	5.0	450	575	N/A	181
bread	2.0	3.0	125	200	N/A	105
celery	2.0	3.0	225	350	8.0	266
cheese	4.0	7.0	375	450	3.0	101
chicken	10.0	14.0	325	425	4.0	286
chips	2.0	4.0	125	200	N/A	156
chocolate	5.0	8.0	250	375	N/A	282
fish	13.0	16.0	375	475	2.0	107
frozen meat	10.0	14.0	450	575	-14.0	355

Figure 8

EXCEPTIONS HANDLING

When the program encounters an error, an exception will be thrown and displayed is a message box alerting the user to the problem so they can fix it. (figure 9)

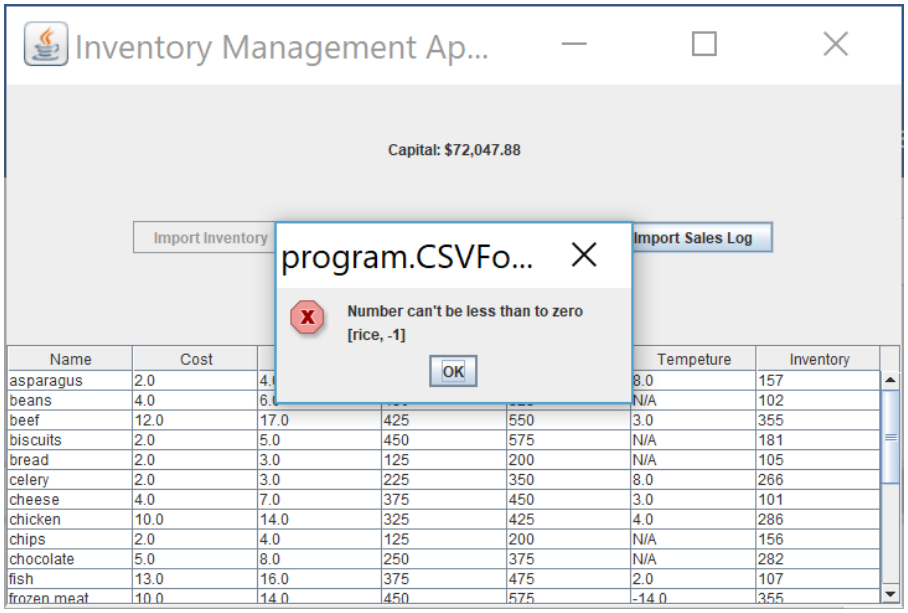


Figure 9

List of exception thrown by the program

CSV Format Exception

- Item_Properties.csv
    - Item contains incorrect number of columns
    - Manufacture cost is not a number
    - Sell price is not a number
    - Re-order point is not a number
    - Re-order Amount is not a number
    - Temperature is not a number
    - Number can't be less than or equal to zero
    - Temperature is greater than 10 degrees
    - Temperature is less than -20 degrees
- Manifest.csv
    - Item contains incorrect number of columns
    - Truck type can only be >Refrigerated or >Ordinary
    - Amount Purchased is not a number
    - Number can't be less than or equal to zero

- Sales_Log.csv
    - Item contains incorrect number of columns
    - The number sold is not a number
    - Number can't be less than to zero

Stock Exception

- Item doesn’t exist in inventory
- Cannot sell more items than there are in the inventory

Delivery Exception

- Item doesn’t exist in inventory
- No Items Need to be ordered