

Overview, Features and Limitations

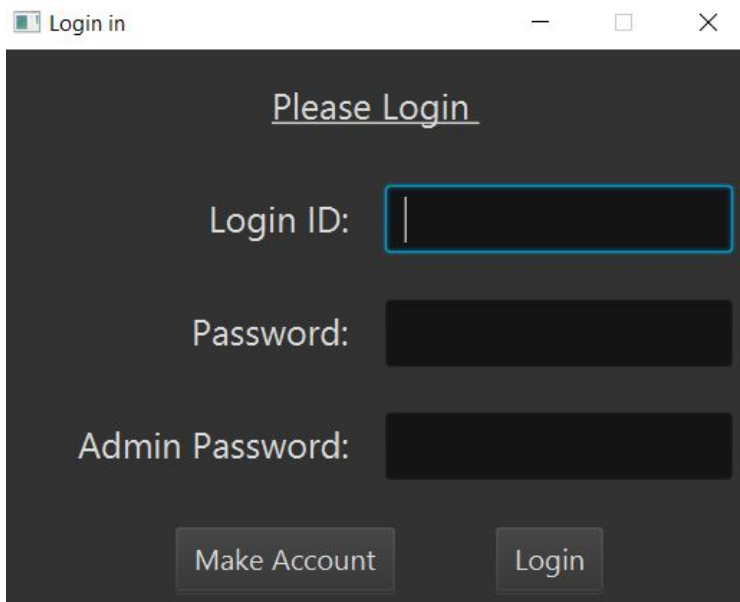
1. Splash Screen

The animated splash screen shows an animated box moving left and right whilst rotating. Application loading status and progress bar is shown. We note that there is an easter egg in this section.



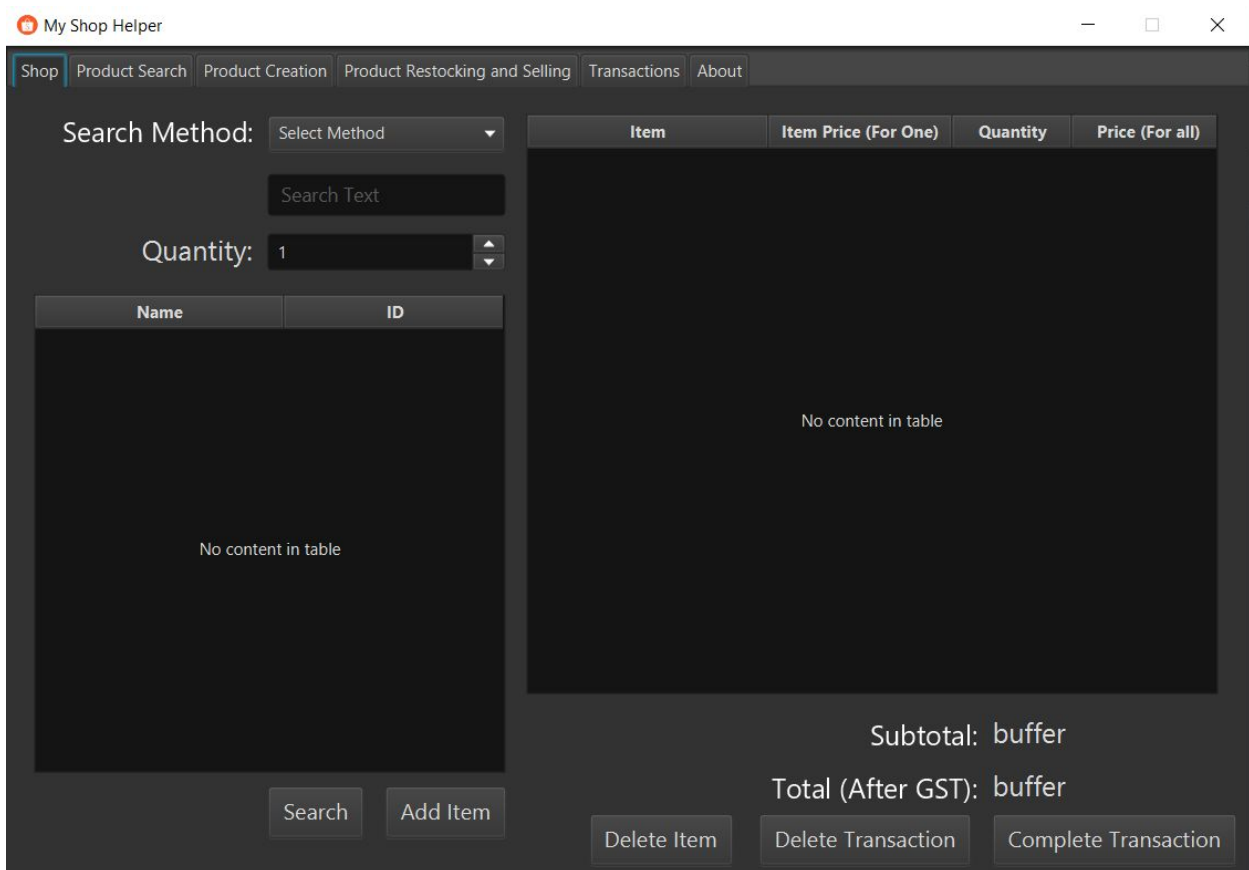
2. Login Tab

Immediately after this, the user is greeted with a login page. The user can either create an account with admin permission (admin needs to key in a special admin password to allow for the creation of an account), or log in to the program. As of now, the AES key is "floccinaucinihilipilification".



3. Tabs

From here, the user is greeted with multiple tabs. I will go through the tabs in an order that demonstrates the use of the program, instead of consecutively. I also note that the Products used in this example are more just to facilitate testing, and is not an example of the actual use of the program, where these makeshift “products” will be replaced with actual products (for example, Touhou games)



4. Product Creation Tab

The first tab that users will start on is the product creation tab. In this tab, as the name suggests, products can be created. Information about the product including (1) the product's name, (2) a description, (3) an image (a nolmage image is provided in the Files Tab of the Program Files), and (4) the list of prices need to be keyed in (Including a Price with a quantity of 1) can be filled in. An example of such a set of information is shown below.

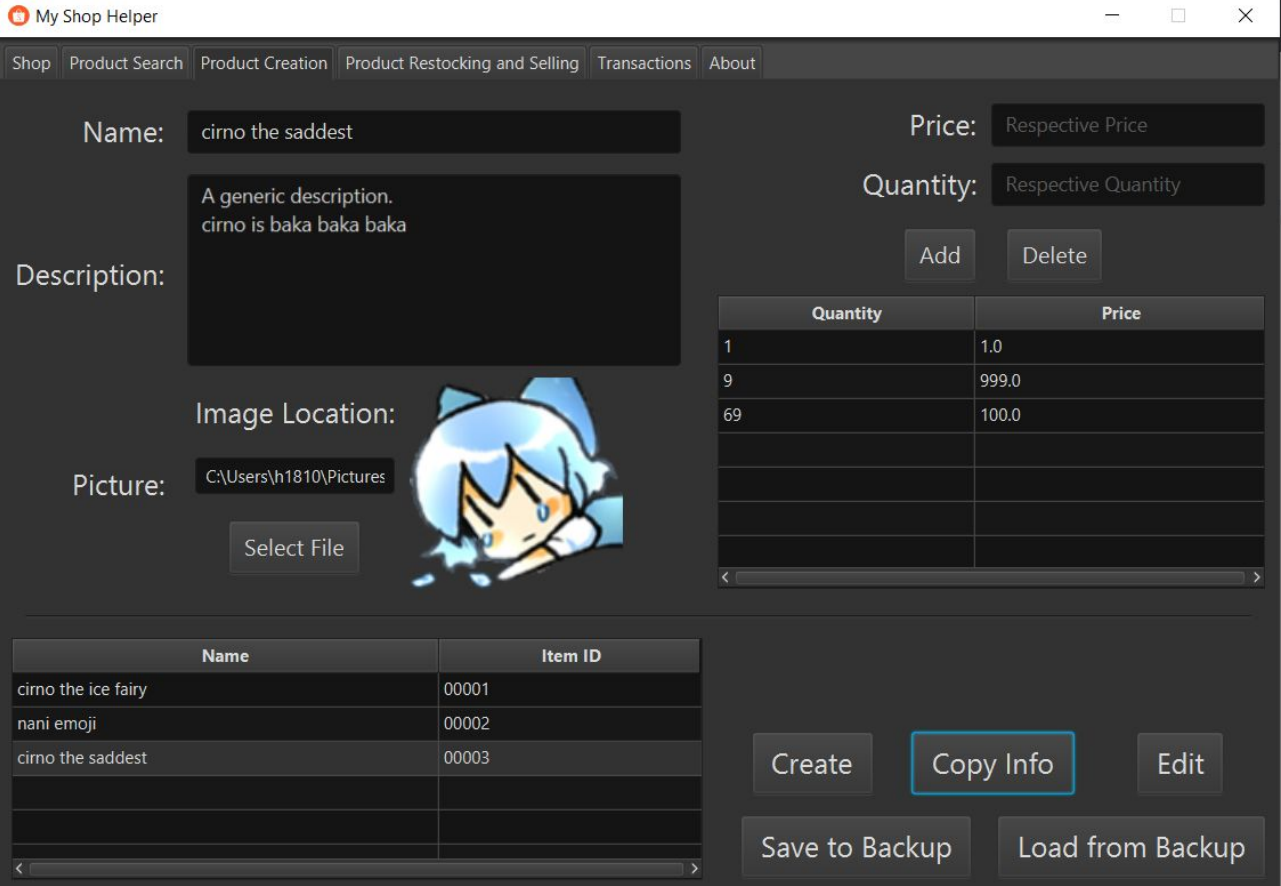
Then, with this information, a product may be created or edited. This is the basis of how the program works. Note that no two product's names may be the same, and all products are automatically assigned a unique product ID, with no repeats.

By selecting a product and clicking the copy info button, all the information about the product will be loaded into the text fields, textboxes, and tables. This facilitates the editing of information of a product, or making a product with similar images, descriptions, prices, etc. I note that after a product has been created or loaded, all the information from the program will be removed EXCEPT for the image and its details.

Finally, there is the option to save and load the information about the programs to/from a backup. This allows for users to test with the program without any implications. I note that by loading the information again, all the information from every tab will be reloaded, to revert the state of the program to its previous form.

I also note that when the program is closed, all information about products will be saved to a local backup. When the program is launched, all information about the products will be loaded into the program.

More information regarding how the information about the products was stored will be provided below.



My Shop Helper

Shop Product Search Product Creation Product Restocking and Selling Transactions About

Name: cirno the saddest

Price: Respective Price

Quantity: Respective Quantity

Add Delete

Description: A generic description.
cirno is baka baka baka

Image Location: C:\Users\h1810\Pictures

Picture: Select File

Quantity	Price
1	1.0
9	999.0
69	100.0

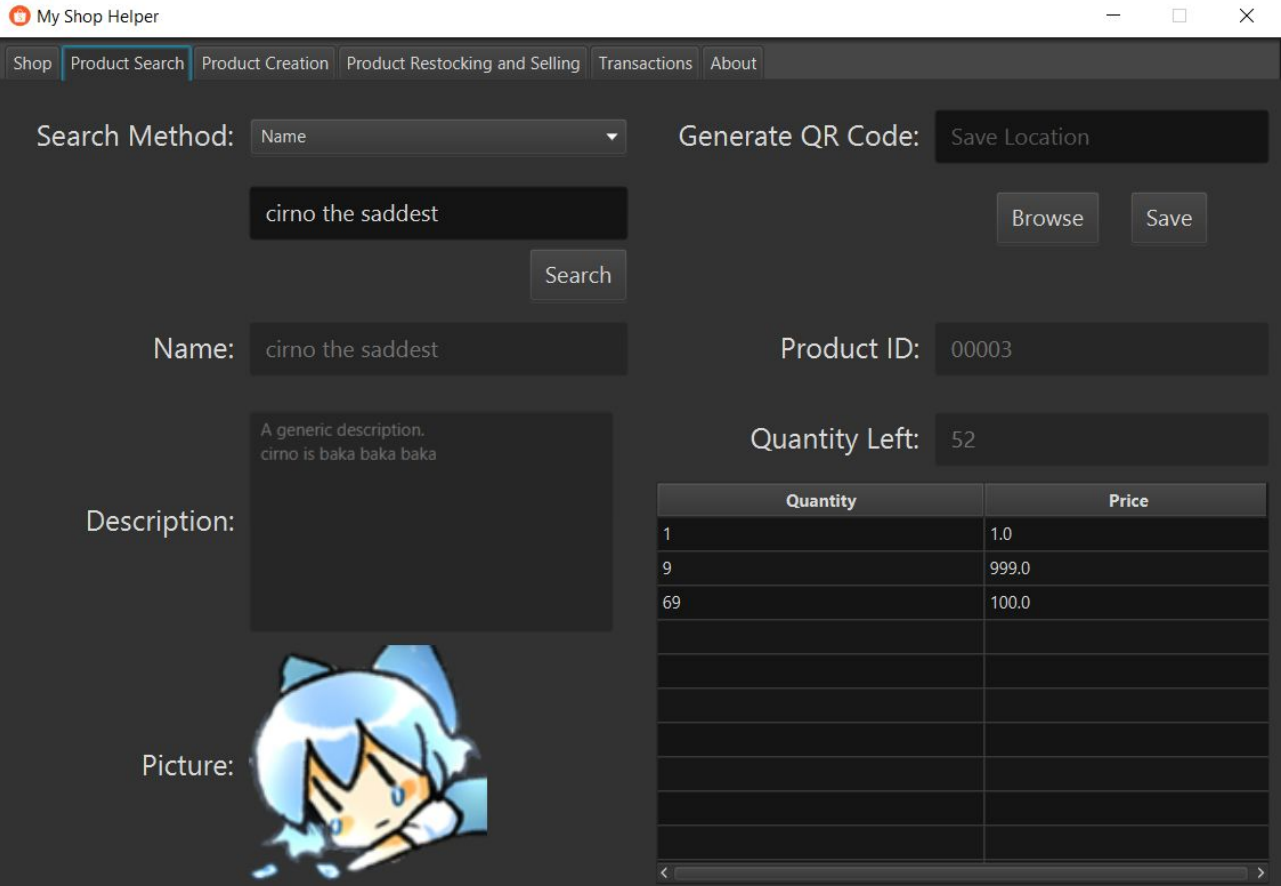
Name	Item ID
cirno the ice fairy	00001
nani emoji	00002
cirno the saddest	00003

Create Copy Info Edit

Save to Backup Load from Backup

5. Product Search Tab

In this tab, information about products may be searched about. By selecting a search method (Product name or ID), information about the product will be displayed on the tab. I note that the searched ID or name needs to be exact. If no product is found, then a dialog noting that no product was found would be presented. Furthermore, the details about the product shows the product's details during the time the product was searched. The product needs to be searched again for the details to be refreshed.

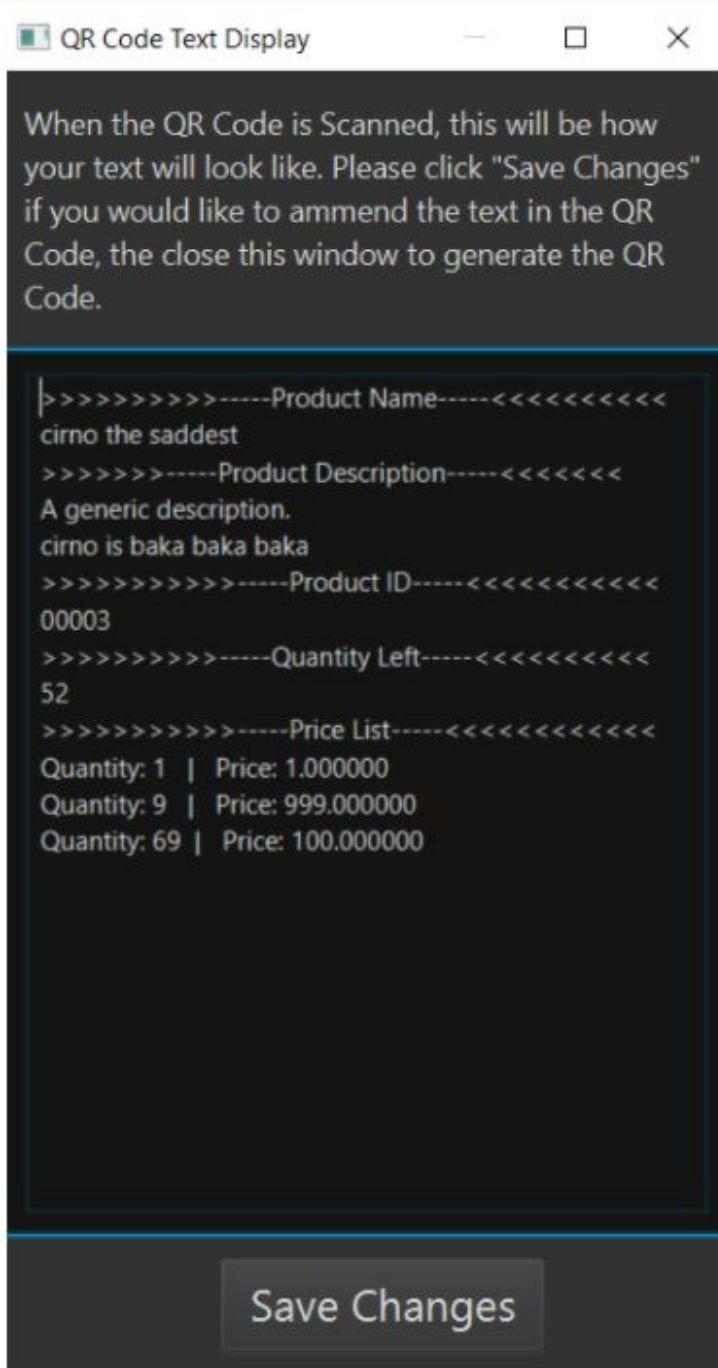


Quantity	Price
1	1.0
9	999.0
69	100.0

Alternatively, QR Codes may also be generated to show all the information by scanning the QR Code. Before the QR Code is saved to a file location for later use, the user is presented with a dialog asking the user to edit the text stored inside the QR Code. However, an automatically generated text is displayed on the screen that encapsulates the product's information. Upon saving changes, the new text will then be the text that will be stored in the QR Code.

The QR Code that stores this text is usually very large, as a compromise of the formatting of the text. Furthermore, other characters such as Chinese, Japanese, etc are also not supported. The QR Code is then saved as an image

to the specified file location. The file's name is the name and date that the QR Code was generated.



Yup. That QR Code is huge.

6. Product Restocking and Selling Tab

In this tab, users may “restock” on the amount of each item they have. It also allows for excessive amounts of one item to be mass sold. The total revenue from all sales will also be updated accordingly. Finally, the amount of each item left will also be displayed.

Note that the text boxes regarding the prices that the items are being purchased/sold at, and the quantity text boxes sync up with each other (ie when the price per item text box is edited, the total price text box will be edited accordingly. More details will be shown in the video.) However, in this part, the listeners don't always update. The amount used for purchase is updated from the "Price (In Total)" Textbox.

Furthermore, when any order is made, a transaction will be logged.

[illegible]

7. Shop Tab

This is the meat of the program. This is where items can be bought from the shop. In this case, the items displayed in the search will change whilst the search text is being keyed in. However, the search Button is still there in case the listener fails to function. Items can be added to the shopping cart.

deleted, or additional items may also be added. Note that the pricing of items is done collectively. For instance, if the prices of an item are:

Quantity	Price
1	1.0
9	999.0
69	100.0

then, 1 item will cost \$1,

9 items will cost \$999,

21 items will cost $999*2+3*1 = \$2001$, and finally

100 items will cost \$100 (for 69 items) + $999*3$ (for $9*3 = 27$ items) + $1*4$ (for 4 items)

Removal of a certain amount of an item may also be done, by inputting a negative quantity. However, this should never realistically happen. The program also allows for items to be removed, or for the whole transaction to be cancelled.

As before, after the transaction is complete, the transaction will be logged.

My Shop Helper

Shop Product Search Product Creation Product Restocking and Selling Transactions About

Search Method: Select Method

Search Text

Quantity: 1

Name	ID
No content in table	

Item	Item Price (For One)	Quantity	Price (For all)
No content in table			

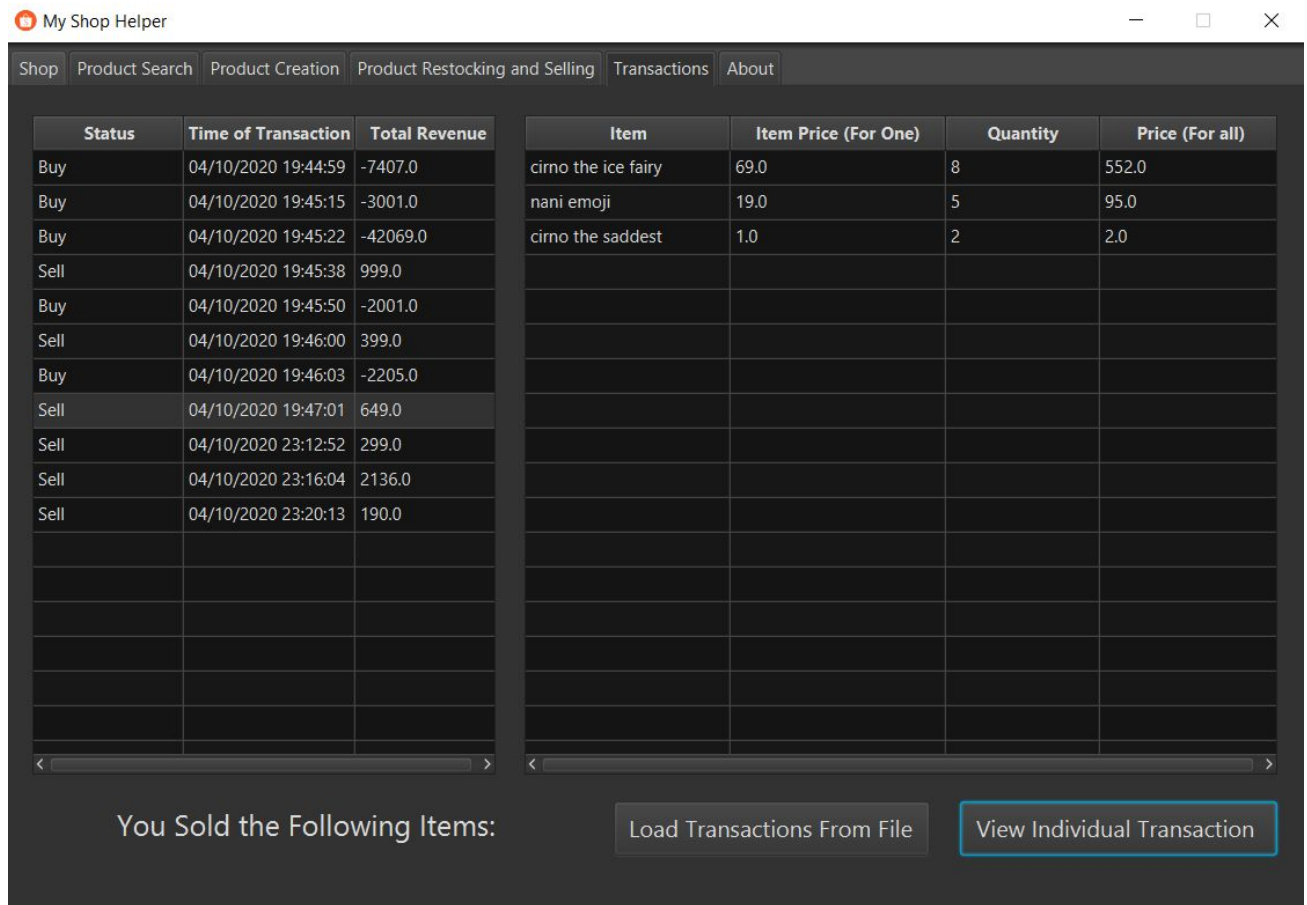
Subtotal: buffer

Total (After GST): buffer

Search Add Item Delete Item Delete Transaction Complete Transaction

8. Transactions Tab

In this tab, all transactions that have occurred will be displayed. Upon loading the transactions from a file, all transactions will be processed and formatted. Then, by selecting a transaction, and viewing an individual transaction, the details of the transaction will be displayed (ie all the items that were sold/purchased, and its quantity). Note that purchasing any item (or spending money) will be displayed as a negative revenue, to standardise things. Finally, the time at which the transaction occurred is displayed.



9. Internationalisation (WIP)

As of now, only the "About" page is internationalised. Translations for some languages are stored in .property files. I also note that some easter eggs regarding this are merely there as a personal joke, and nothing more. (For example, Greek having some very questionable translations)

Code Explanation of key algorithms and features

1. File Saving and Loading

In this section, I will detail the way which the files are stored as, and several setbacks regarding the way the file storage is done.

Login.txt

This file stores the information about the user login information. The information stored in this file encoded via AES.java. After decoding the text, the file takes the form of:

```
“userID_1,password_1,userID_2,password_2,...,userID_final,password_final”
```

ProductID.txt

This file stores the information about the next product ID available. This ensures that there are no duplicates in the productID. The file is stored in the format

```
“ABCDE”
```

,where the string ABCDE represents the 5 digit product ID code that will be used when the next product is created.

products.txt

This file stores the information about the products. We note that the delimiter used in this example is: ",/<>". As such, if this string is used anywhere in the product name, description, or file location, the reading of this file will break. However, I note that only if the user is trying to break the program in this way will the program break, and that under normal circumstances no user should key in such a substring in the product's name or description. Then, the file is stored in the format:

```
“name,/<>description,/<>filePath,/<>ID,/<>quantity_1,/<>price_1,/<>quantity_2,/<>price_2,/<>.....price_last,/<>-1,/<>-1,/<>”
```

for every product. The strings for every product are then appended together. For the sake of clarity, the code for this part is shown in the next page. The delimiter in this case is ",/<>".

```

public String toString(String delimiter){
    String out = this.name + delimiter + this.description + delimiter + this.filePath + delimiter + this.ID;
    for (Pair i:prices){
        out += delimiter + i.getQuantity() + delimiter + i.getPrice();
    }
    out += delimiter + "-1" + delimiter + "-1" + delimiter;
    return out;
}

```

itemAmounts.txt

This file stores the information about the amount of each product left in stock. The file is stored in the format

“quantity_1\nquantity_2....”

,where “\n” represents a new line.

totalEarnings.txt

This file stores the information about the total earnings. The file is stored in the form:

“Price”

itemTransactions.txt

This file stores the information about all the transactions that had occurred. The file is stored in the following format:

“Buy or Sell\nDD/MM/YY hh:mm:ss\nnumOfItemsOrdered\nProductID_1,
IndividualPrice_1,quantityOrdered1,totalPrice1.....\n”

for every product. Here, “\n” represents a new line. The strings for every transaction are then appended together. For the sake of clarity, the code for this part is shown below. Here, int sign represents whether the order is the user purchasing items, or selling items (in the case of the Shop Tab)

```

public void restockingWriteTransaction(Product p, int sign){
    Transaction t;
    double totalPrice = Double.parseDouble(restockPriceTotalTextField.getText());
    int quantity = Integer.parseInt(restockQuantityTextField.getText());
    if (sign == 1){
        t = new TransactionBuy(FXCollections.observableArrayList(new ItemOrder(p, individualPrice: totalPrice/quantity, quantity, totalPrice)));
    }
    else{
        t = new TransactionSell(FXCollections.observableArrayList(new ItemOrder(p, individualPrice: totalPrice/quantity, quantity, totalPrice)));
    }
    printTransaction(t);
}

```

A sample part of transactions.txt is shown below.

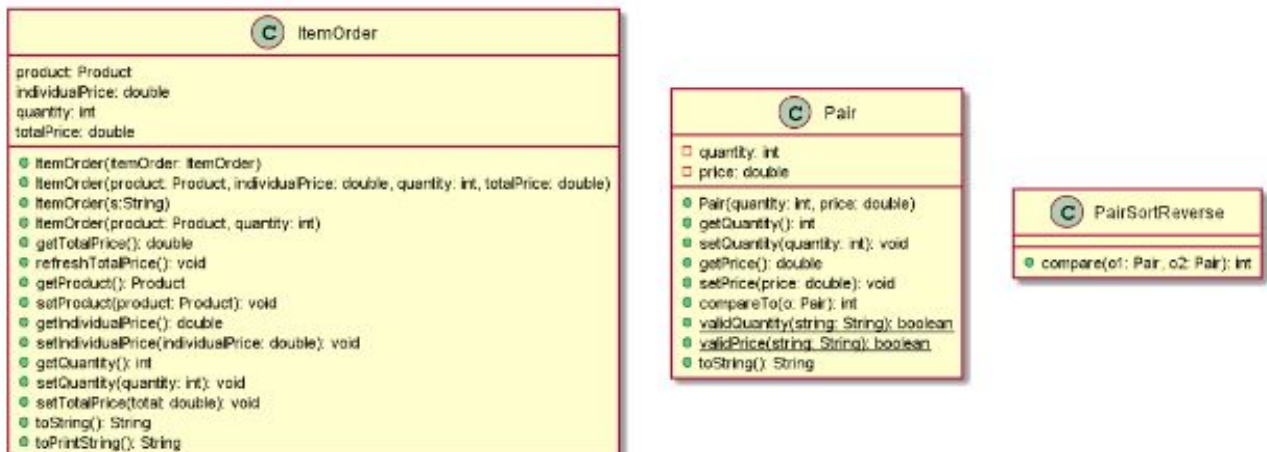
```
Buy
04/10/2020 19:46:03
1
00001,30.0,21,630.0

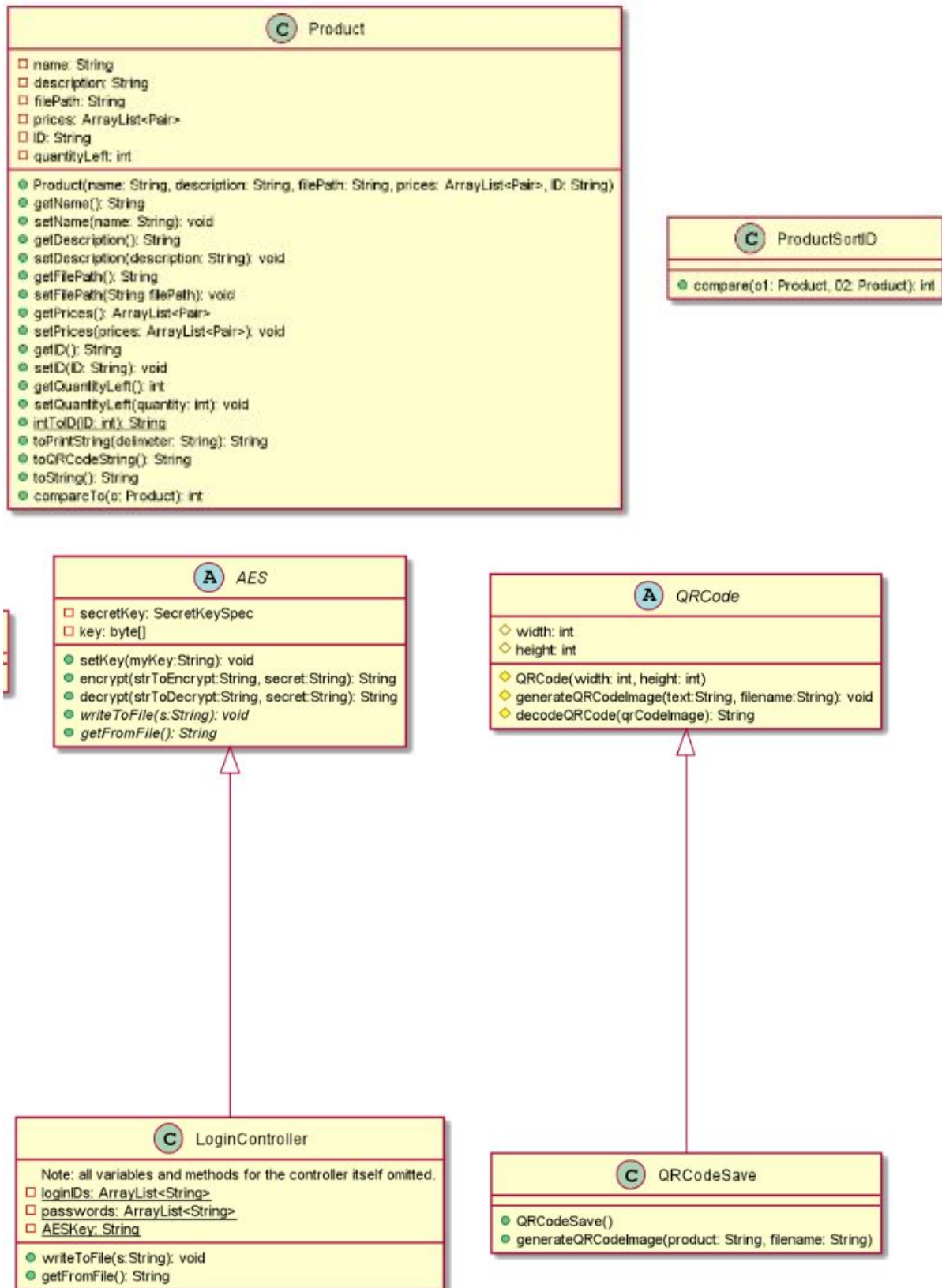
Sell
04/10/2020 19:47:01
3
00001,69.0,8,552.0
00002,19.0,5,95.0
00003,1.0,2,2.0

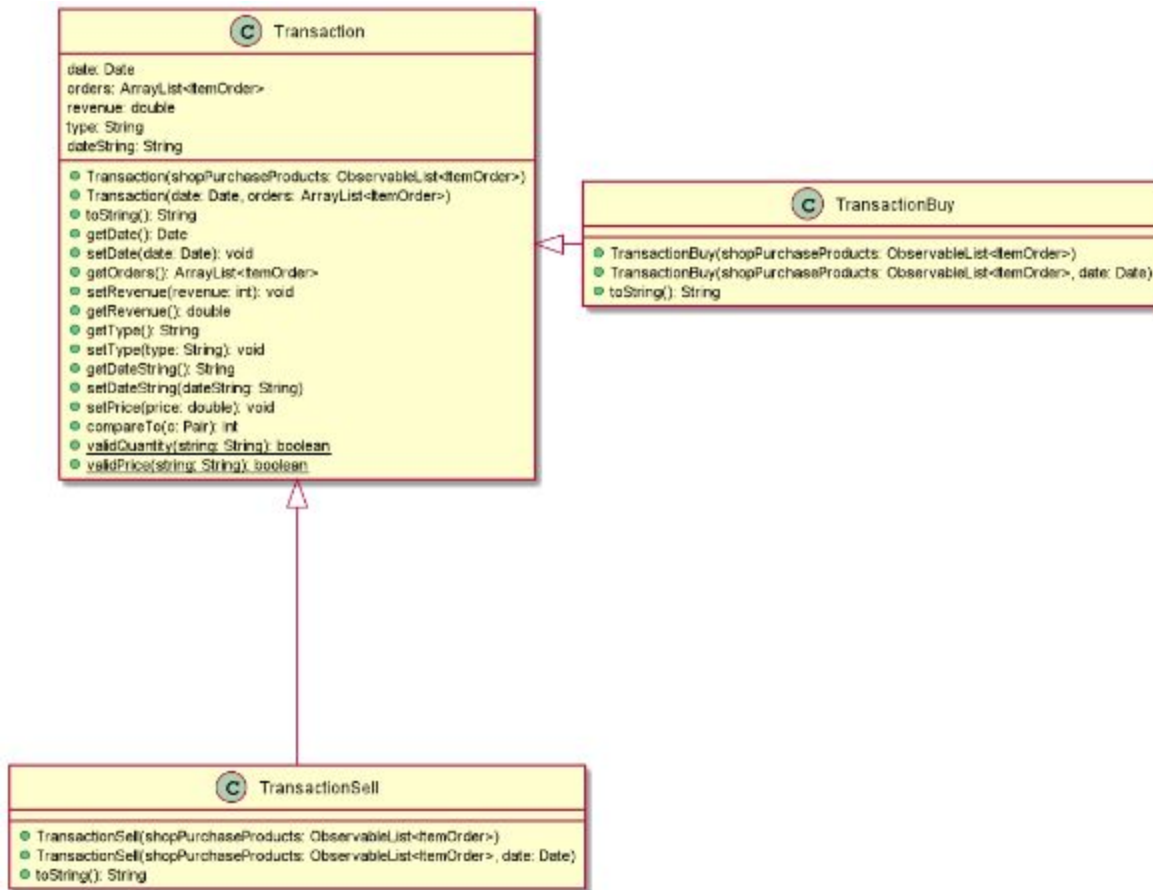
Sell
04/10/2020 23:12:52
3
00003,1.0,4,4.0
00001,69.0,4,276.0
00002,19.0,1,19.0
```

2. UML Diagram

Below is the UML diagram. The image was too wide, so I have split it into a few parts. The entire UML diagram is also attached in the files.

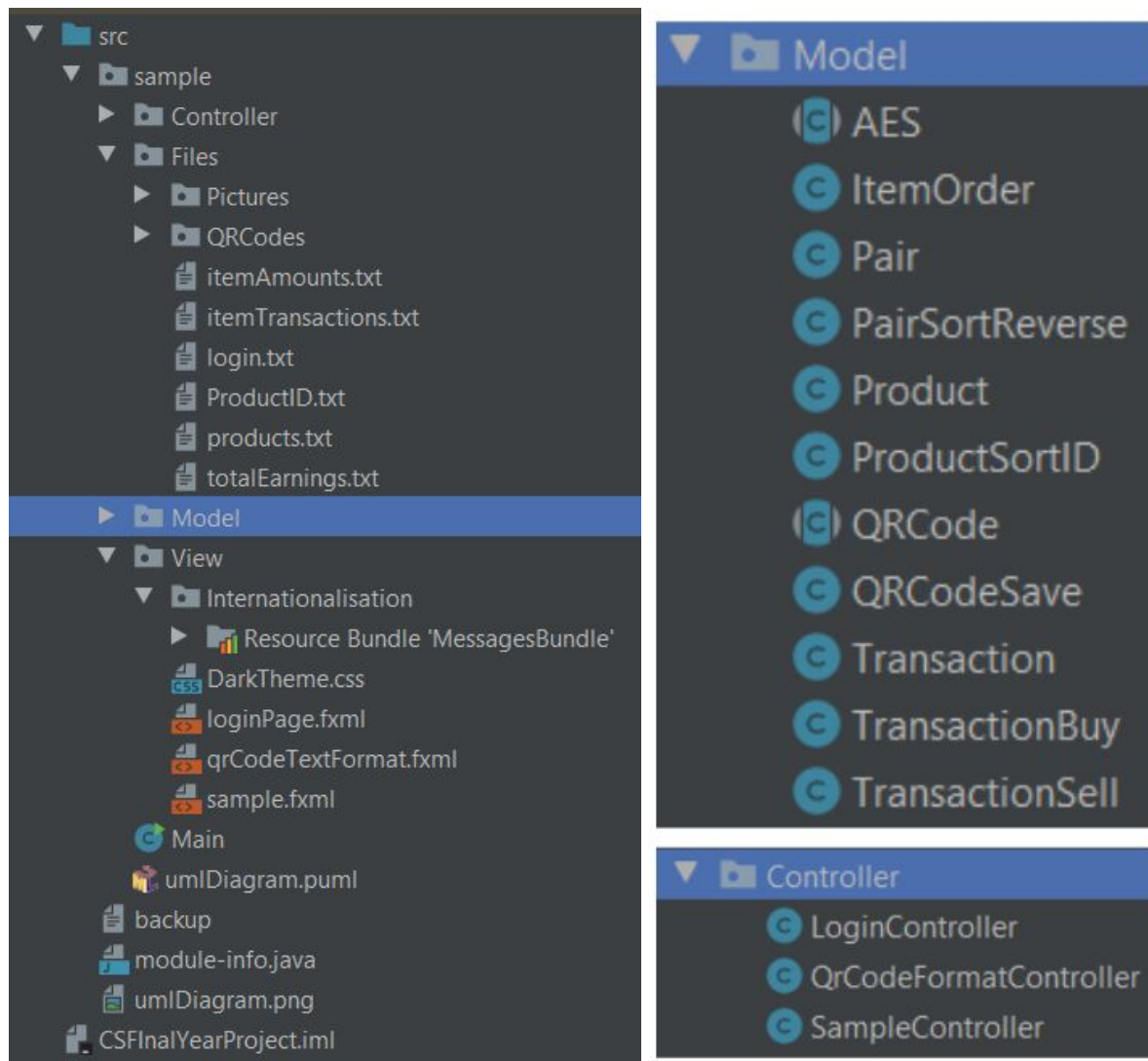






3. OOP Concepts and MVC structure

Below is the File structure of my program. The fxml files act as the view, the java controller files acts as the controllers, and the java files under Model act as the model.



The OOP concept of inheritance was used in the Transaction, TransactionBuy, and TransactionSell class, where a generic parent class Transaction was inherited by the child classes TransactionBuy and TransactionSell. In these classes, TransactionBuy and TransactionSell share the same properties, except for the computation of the total revenue, as well as the string that is printed out into transactions.txt.

The OOP concept of abstraction and encapsulation was done largely by giving meaningful variable names and function names that explains the method sufficiently. Functions such as the showAlert function generally also

showed the use of polymorphism, by allowing the user to have 2 different options of what type of alert to call: namely with a content text, or not.

Description of Testing Strategy and Scenarios

Largely, most of the testing strategy was done via enlisting the help of friends to test the program. Several bugs such as Tables not updating were fixed, while some bugs were inconsistent and hard to replicate. These included listeners sometimes faltering and updating incorrectly. To fix these, additional buttons were added as a failsafe in case these unpredictable scenarios occur.

To be able to handle all scenarios, many try catch statements were used, and many alerts were thrown to allow for all errors (with the exception of using the substring “;/<>” in a product name, description, or filename) to be handled. In this case, the whole main program (or all the parts except for the splash screen and the login page) will effectively be restarted upon the next run of the program.

Reflection

Q: What were some obstacles faced?

An obstacle that I faced was the lack of direction given. Examples of sample projects such as the ray simulator seem very daunting to students such as myself. Furthermore, despite the great assistance given by our teachers about possible proposals for the project, it is still difficult to understand that the size of the project isn't the project's emphasis, but more so showcasing to teachers what we understand and can apply through the module.

Using the TableView also presented several challenges, as the StackTrace provided was relatively arbitrary. For example, for not including getters in a class, a Null Pointer exception was thrown. This caused much of the debugging to be difficult, despite using Tables in the PA. The table would also not update if I used the .clear() function on an observable arraylist, even after calling a refresh for the table. Many such errors and bugs were arbitrary, and proved to be difficult to debug. This is noting the sheer size of the project.

Q: What have you learned through the project?

The main thing which I gained from this project is an appreciation for developers. Developing such a small application already proved to be a challenge for me, let

alone larger projects. I also learned the power of such powerful IDEs such as IntelliJ, where one can automatically generate functions. I also learned that Java is a very user friendly language, where all errors are explained relatively extensively. This is compared to other languages such as C++, where it is difficult to debug code. I also learned the use of markers such as `/** */` and `// todo`, that can help me to mark key locations in my code. Finally, I learned that having extra functions can really aid one in their projects, especially if they are called a lot (such as my `callAlert` function)

Q: What could you have done if more time was given?

Firstly, I would most likely invest in finding a way to store the data online, via google drive api. This can allow the app to be run by multiple users and not confined to one laptop. I can also implement a system where different users access different documents, allowing the program to support multiple users/organisations who wish to use the app. I can also consider encoding every single file in a more secure manner, allowing for no leakage of data. An automatically generated AES key system could also be created, where users may customise their AES key. Package discounts could also be added (eg: ordering 1 of order 1 and 2 of order 2) Finally, I would also be able to identify and fix more errors in my code.

Q: How could the task be improved?

In my opinion, some direction regarding the task and more check ins with the students could be provided. This would give the students more support for such a large project.