CAB302 – Software Design

Assignment 2

Kok Fuh Teo (09830642)

Zhi Choon Ng (09056572)

# Technical Description

## START UP:

Upon starting up the program, the GUI will load with a screen and an empty table accompanying it. The capital will be shown on the screen along with the instructions.

## GUI BUTTON 1: Load properties Doc

Upon pressing the ‘Load properties Doc’ button, the program will access the *readFile* function in the *GUI.java* file and load up the *item\_properties.csv.* The method will access each line in the *item\_properties.doc* until a null is reached, indicating an empty line and the end of the document. The data gathered will be placed in a new instance of *Item* and added to a List variable. This is done using polymorphism via *Item.java* and is stored in *Store.java* as well. We will need to Export the Manifest then, which is covered the ‘Export Manifest’ button below (GUI BUTTON 4: Export Manifest).

## GUI BUTTON 2: Load Manifest

Once exporting is done, the user is required to load the manifest which they exported. Pressing the button will load up *manifest.csv.* This will load up a *readFile* method that is specialised for this task. The loading will skip the line that starts with ‘>’ as it’s merely for readability that tells the reader whether the goods are dry goods or those that requires refrigeration. Using polymorphism, the data will be stored in *Manifest.java.* As the program will realise the stock available in under the reorder amount, it will then purchase a static amount of stock, known as the reorder amount. This will cause our inventory to increase and our capital to decrease. When this happens, the screen will print the updated capital. Should there be not enough capital, a *StockException* will be thrown.

## GUI BUTTON 3: Load Sales Logs

Upon pressing, the user will be allowed to choose one of the five sales logs through the *SalesLogBox.java.* When the user loads up the *sales\_log.csv,* the program will store the document using a similar method with the *readFile* function except it is customized for the sales csv file instead with only two variables instead of the usual five or six. Should the *.csv* document be tampered with, resulting in a format that is not the type of a csv, the program will throw a *CSVFormatException* error. Using polymorphism, the data is stored in the *Sales.java* file. Our capital will increase as we turn a profit by selling the stock at a higher price (Indicated by [price] in the *item\_properties.csv*) when it costs us less (Indicated by [cost] in the *item\_properties.csv)*

### Assumptions:

### It is assumed that the user doesn’t change the name of the *sales\_logs\_0.csv* and beyond. If it is changed, the program will return an exception indicating the file does not exist.

## GUI BUTTON 4: Export Manifest

Upon pressing the button, the program will detect whether the file *manifest.csv* exist. Should it not, an exception warning will appear. If it is successful in loading, the program will use the *exportManifest* method within *GUI.java* and will start appending the name and quantity of the items, saperated by whether they require an ordinary or refrigerated truck.