|  |
| --- |
|  |
| CAB 302 Assignment 2 |
| Group 108 |

|  |
| --- |
| Timothy Melville (n8856508) & Dean McHugh (n9656286)  Semester 1, 2018 |

# Program Architecture

## Objects

The objects package contains classes related to objects used at a store level.

### Item

The item class is used to create Item objects. Items are used to depict each Item sold by the store, each having a name, price (Both for selling and buying), a re-order point, and a re-order amount. Optionally, an Item may have a maximum temperature it may be stored at. To accommodate for this, the Item class has an overloaded constructor that contains the additional temperature argument.   
The Item class also implements the Comparable class, allowing Items to be compared against each other so they may be sorted, by both name and temperature.  
The class otherwise contains a series of getter methods for Item properties.

### Stock

The Stock class depicts a Stock object used to contain a number of Items. This class extends the Iterable class, to be able to utilise for each loops for Items within a Stock object. Items are stored as keys within a TreeMap object, with quantities of Items stored as corresponding values. Using a TreeMap object allowed simple addition or removal of items and modification of quantities, as well as ensuring the Items contained would be consistently ordered. The class also contains getter methods for items, quantity, and the minimum temperature of contained Items.

### Store

The Store class implemented a Singleton pattern, allowing only one instance of the object. The Store contains two Stock objects, one representing inventory, and the other representing an order list to be used in creating an order manifest, as well as a double representing the Store’s capital, and a String representing its name. The class contains getter methods for Store properties, as well as methods for adding and selling items to/from the inventory. A final method, reduceCapital is used when a manifest is received by the Store, as the Store is charged for both Stock and delivery.

The Store’s capital is never rounded, but when accessed via getter method the returned value is rounded to two decimal places.

## Delivery

The Delivery class contains classes related to delivery of Stock and Items to the Store.

### Truck

The truck class is an abstract class that defines the basic structure of both OrdinaryTrucks and RefrigeratedTrucks. As it is extended by both OrdinaryTruck and RefrigeratedTruck to create different objects based on the same principles, it is an example of polymorphism. All methods besides the constructor and getter method for contents are abstract as they are implemented differently in each extension. Trucks as a whole have a cost, a maximum number of items permitted, and a type.

### OrdinaryTruck & RefrigeratedTruck

Both the OrdinaryTruck class extends the Truck class with different methods for determining cost and setting/getting temperatures. In particular, the ordinary truck throws an exception if the getter method for temperature is called.

### Manifest

The manifest object is used as a collection of Truck objects. It implements the Iterable class to allow for each loops of Trucks within a Manifest. An ArrayList is used to store Trucks, as it is re-sizable, elements inside can be accessed non-sequentially, and implementing the Iterable method ‘iterator’ is trivial. The manifest has two constructors, each for use in different scenarios. The first constructor with no arguments is for use by the CSVRead class, when Trucks are read and sequentially added to the manifest. The overloaded constructor with a Stock argument is used by the Store class, where an order list Stock object is used to create a Manifest, with the manifest automatically determining the best configuration of Trucks to use. The manifest object also contains a cost determined by the types of trucks and Items stored within each Truck.

## CSV

Both classes within the CSV package contain only static methods and variables, as neither class constructs an object. These classes are used for the reading and writing of CSV files at a pre-determined filepath.

### FileRead

The FileRead class is used for reading three separate files with different file structures, and as such, there are three public methods, each for a different file, and each returning a different object. The method used for reading sales logs increments a class-wide counter every time it is called in order to properly move through the sales log files.

### FileWrite

The FileWrite class contains methods only to be used for writing the manifest.csv file, and as such contains only one publicly-accessible method, using a manifest as an argument. The class, using a private helper method writes Trucks and the contents of each truck to file as necessary.

## Exceptions

Each exception class is functionally identical, each extending the Exception class, and overriding the default constructor, and constructor with a String message argument. The use of three exception classes with identical functionality means less generic exceptions are thrown, and the cause of the error can be found much more easily. The three exception classes used are:

### StockException

### DeliveryException

### CSVFormatException

## GUI

### EntryPoint

### GUI

# GUI Testing