Christopher Friesen – CMF2536
Joseph Le – JSL2449

Assignment 3 Shopping Cart

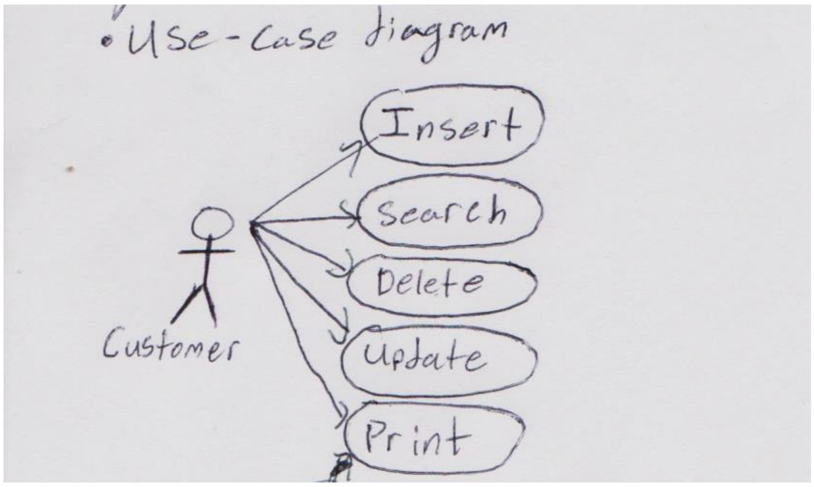
GitHub repository link: https://github.com/CFreezz/Assignment3.git

Analysis:

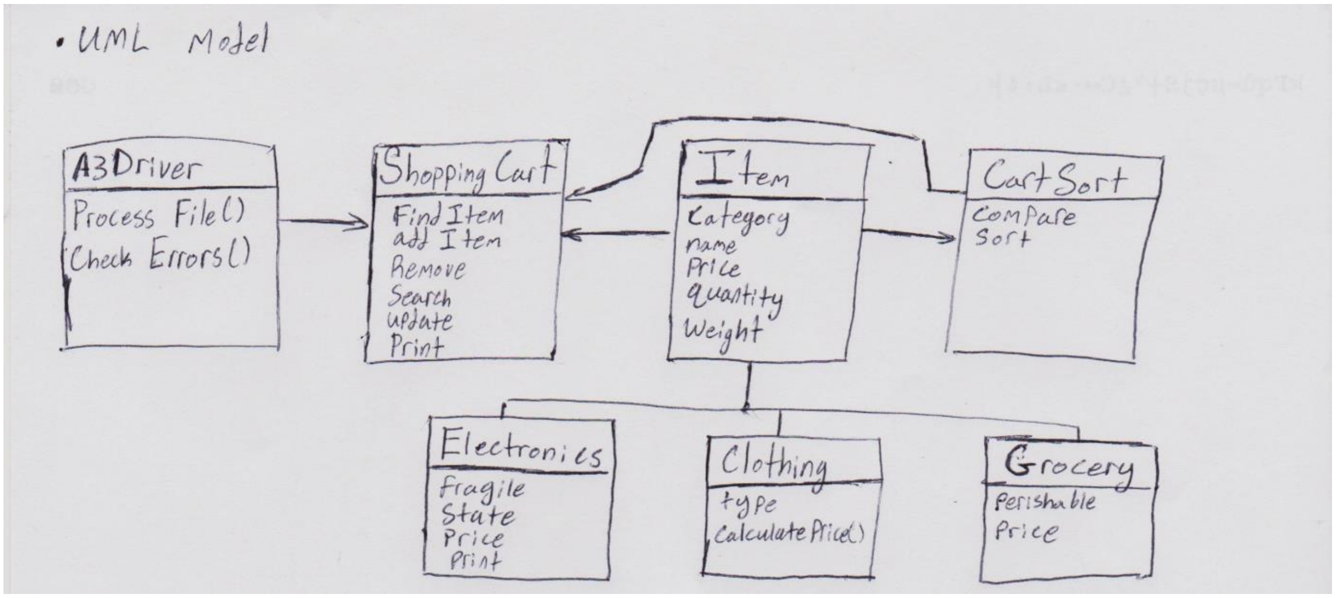
This project is to simulate an online shopping cart using OOP and an ArrayList. Our provides the bulk of the input validation and creates the shopping cart object that contains the database for the items. The Item class is the super class to the Clothing, Electronics, and Grocery classes.

How will we store the cart data?
  The ShoppingCart class contains an ArrayList that holds <Item> objects.
Does it need to be sorted? How?
  Yes, it needs to be sorted by ascending alphabetical name. We use a Comparator template to implement the sort.
Who handles faulty input?
  The driver holds the responsibility for validating input. Every other class assumes that input given to it is valid input.
Extending the existing driver…
  The driver is can be extended in the main method by adding more ShoppingCarts
Random Corner Cases…
  Inserting the same item twice will update item in shopping cart, adding the new quantity to the old
  We do not bother doing number formatting for cash values (example $57.00 is $57.0)
  We assume all numbers are between integer max and integer min
  If the shopping cart gets really big (like integer max big), then the program may slow down because our searches are linear
  If cash amounts are very large, we do not handle conversion from scientific notation

Use-case design:



UML Model:



ADT Description:

A3Driver: Contains the string arrays of valid operations, categories, and state names. This is the driver class. The main method takes in a file on the command line, validates its existence, then processes each transaction request in the file. It also validates all the transaction input and prints errors if they exist. It handles the front end operations of insert, update, delete, search, and print.

ShoppingCart: Has 2 attributes, an ID and an ArrayList<Items>. This class handles, the actual backend operations of insert, update, delete, search, and print. The ID of the ShoppingCart is used for if the manager of the program wants multiple shopping carts. It is intended to be unique, but that is up to the driver class that instantiates it

Item: Has 5 attributes, a name, basePrice, quantity, weight, and salesTax, which is hard-coded and constant This class has the getters and setters for these attributes as well as:

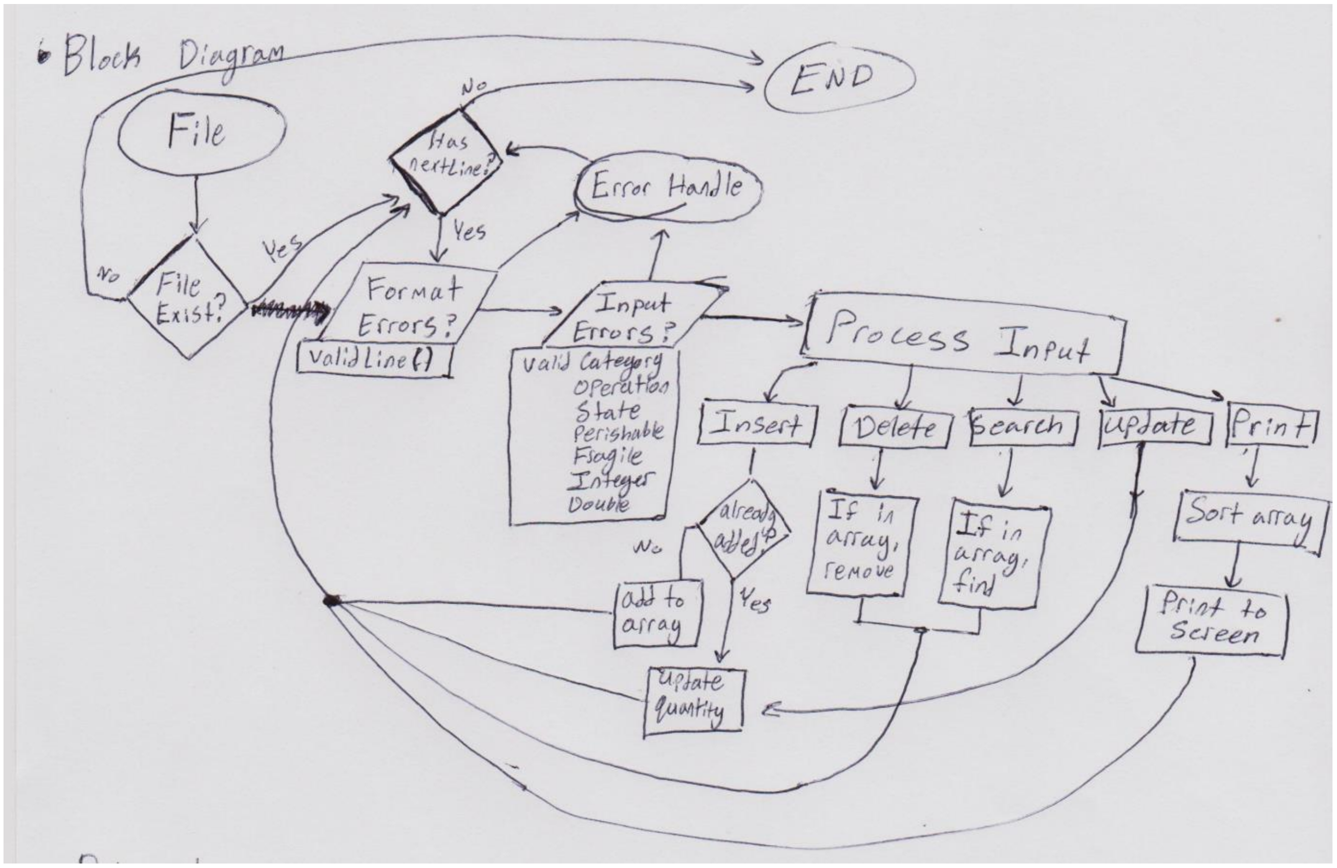
roundCents() method which that takes in a double primitive and truncates all but the last two decimals points.
StateTax() which returns false if the state has no sales tax, true otherwise
printItemAttributes() which prints the 4 initial item attributes

Electronics: extends Item, and contains 2 additional attributes, fragile and state (which is the state that the electronic is being bought from). CalculatePrice() and PrintAttributes() are adjusted here

Grocery: extends Item, and contains 1 additional attribute, perishable. CalculatePrice() and PrintAttributes() are adjusted here

Clothing: extends Item and has no additional attributes, just a CalculatePrice()

Block Diagram:



Algorithm for driver:

The driver as explained earlier handles file input and transaction input validation. This driver creates a ShoppingCart item with an id = getNewShoppingCartID() function. This ShoppingCart is modified through the driver life-span. The driver then goes on to reading the input file (printing an error if it cannot be found). If the file is found, the driver starts reading each line of the file as a potential transaction. Invalid transaction input is ignored and valid transactions are pushed to the single shopping cart using the process(String[], ShoppingCart). The process() method applies the transaction to the given ShoppingCart parameter. It takes as arguments, the (assumed to be valid) transaction line as an array of strings split on ' ' space character. When the file has ended, the program ends. It is up to the input_file to print out specifics of a ShoppingCart

As mentioned earlier, the driver can be extended to have multiple ShoppingCarts that could all be processed. It would be a quick modification to the main method.