Group Members: Connor Underwood, Neel Acharya, Suhani Yadav, Zeyad Adham

Part One

The marketplace that our team has decided to design is for shoes. We have created five classes in the implementation, each performing a specific role of a party responsible for the effective functioning of a shoe market: the sellers, the stores, the customers, the shoes, and the marketplace itself.

The central coordinating class is the *Marketplace* class. The user will interact with this class throughout his visit to the shoe market. A user begins his interaction with the *Marketplace* class by either logging in or creating a new account, a functionality we achieved by creating an *Accounts* file. Once the login process is complete, we check if the user profile is a seller or a customer. Once we know that, we create an object for the user associated with the type of class: *Seller* or *Buyer*. Thereon, *Marketplace* will be the centre of interaction between *Buyers* and *Sellers* with *Stores* and *Products*. Each *Seller* has multiple *Stores*, and each *Store* has multiple *Shoes*. The same has been implemented by creating an ArrayList of *Stores* for each *Seller*, and an ArrayList of *Shoes* for each *Store*.

Once the *Seller* logs in, he is presented with multiple options on a switch case menu. Methods implemented in different classes, according to the menu options, are called from the *Marketplace,* and the desired results are achieved. The menu options for the *Seller* include:

1. To edit a *Shoe* in one of his *Stores:* which includes a menu of which part of the product they wished to edit - name, price, description, quantity, or shifting the product to a different store.
2. To edit the *Store:* which includes adding and removing *shoes* from a *store*.
3. To view all the *stores* owned by the *seller*, accompanied by information on each purchase made by *customers* in those *stores*.
4. To add more *Stores*.

Similarly, a menu for *Customers* is also displayed. They can:

1. View the market product-wise.
2. Search for the product they want by the name of the *shoe*, the description of the *shoe*, the name of the *store*, and the maximum price for the *shoe*, or filter out the out-of-stock *shoes*.
3. Customers can also view their purchase history with the *shoe* data, all of which will be stored in a file.

The selective features we implemented are Files and Shopping Cart. For Files: all the input and output data persist in the form of CSV files for all the *sellers* in the *Sellers*' common file. Correspondingly, the same has been implemented for *customers* and their files.

In addition to that, we have implemented the Shopping Cart feature. As the *customers* browse through the market, the *shoes* would be added to their chosen list with the status set to "in the shopping cart." Once they are satisfied and decide to check out, *shoes* would be accordingly removed from the chosen list or added to the purchase history. *Sellers* would also have access to view any *customer'*s shopping cart at any point in time.

At last, our group decided to implement the 'Customer Reviews' optional feature. To implement that, we receive *customer* reviews for the *shoes* they have purchased and present those before the corresponding *sellers.*

Part Two

Part Three