Cart Validation application
Name:Geetika kapil
SJSU ID: 015233348
Email: geetikakapil123@gmail.com

● **Problem Statement:**

- The application should maintain an internal, static database (inventory of stock) (this may be developed using HashMaps and/or other built-in Java Data structures). This means once we re-run the program, the changes to the data would not persist.

  Input file should be processed as follows:
  - Validate if the requested quantity for each item is permissible. For example, if the request is to order 3 soaps, check the database if we have at least 3 soaps in our inventory.
  - There will be a cap on the quantity of each category that can be ordered in one single order. For example, restrict Essentials to a maximum of 3, Luxury to 4, and Misc to 6. (This will be configured beforehand)
  - In case it is an incorrect request, generate and output TXT file with message "Please correct quantities." and include the items with incorrect quantities
  - After this validation, if the cart is valid, calculate prices for the cart.
  - Take the card number of the user and if it is not present in DB add it.
  - Output the CSV list with the total amount paid.

Pattern used

Singleton: Here the instance of staticDB is used as the singleton. It is global throughout. So we have used it to insert the data in the database and access the database when needed throughout the application. We have also used to store the credit card numbers and access those credit cards when needed. This pattern involves a single class which is responsible for creating an object while making sure that only a single object gets created.

private StaticDB db = StaticDB.getInstance(); line number 7 in InputDatabaseMaker.

Factory :

- Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

- Here I have used FileFactory.java as the factory method which return the object of errorfile and outputfile. So if anything changes in the main application we dont have much to change in main application. Here Errorhandler and finalfile implements FilewriteInt and filefactory solves the calling of the two classes.

Iterator

This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation. Here in my code I have used an iterator to go through the credit card details. In generateNewOrder file i have used iterator at line no 104 which is using the iterator object from ItemsIterator.java . ItemsIterator implements interface IteratorInt. Thus it helps to supports variations in the traversal of a collection. It simplifies the interface to the collection.