AuctionBay

Project Phase Four

CSCI3060U, Software Quality Assurance

Miljanovic, M.
Ontario Tech University, Faculty of Science
March 17, 2023

Tuesday Lab (CRN: 72674):

Rija Baig (100746674)
Preet Panchal (100707094)
Eihab Syed (100707448)
Nathaniel Tai (100662284)

AUCTIONBAY

TABLE OF CONTENTS

AUCTIONBAY

1.0. INTRODUCTION

AuctionBay is an Auction-style Sales Service. The system consists of two parts: the Front End, a point of purchase terminal for bidding and advertising items and the Back End, an overnight batch processor to maintain and update a master auctions file intended to run each day at midnight. This is Phase 4 of the project; it contains the design document for the overall structure of the backend for AuctionBay. This design document will include a UML class diagram with class and method descriptions to explain each functionality. Phase 4 also represents the first version of our design implementation written in Java source code.

1.1. PLAN FOR PHASE 5

- **Rapid Prototype Highlights:**
  - For Phase 4, we have built a complete rapid prototype for our Backend for the AuctionBay program. We have a simple working backend application that completes the basic intended functionality, however it is not ready for testing as only a few constraints were solved.
- **Missing Components:**
  - In this phase, as this is a rapid prototype for our backend, there will be a few errors that are not handled properly and some of the console messages and output format needs to be corrected. Furthermore, we plan on working on the two main constraints required for the backend.
- **Next Steps:**
  - For Phase 5, our main goal is to have all constraints for each operation solved and be able to handle all errors that we will create for our JUnit test suite using path coverage testing. We also need to make sure that all formats for the backend console application matches with what we have in these test cases as well as match the client requirements for recording transactions. Lastly, everything must be organized a little better, so we can ensure a smooth testing experience in future Phases.

2.0. UML DIAGRAM

Please kindly review the AuctionBay Phase 4 UML diagram here: https://github.com/preet-panchal/AuctionBay/blob/main/PHASE-4/Phase%20Four%20UML%20Diagram.pdf

3.0. UML METHOD DESCRIPTIONS

3.1. User

The User Class is used to initialize and set user attributes based on the transactions read from the merged daily_transaction file and current_user_accounts file.

File input: daily_transaction.txt, current_user_accounts.txt
File output: new_current_user_accounts.txt

- **+ User(_username : String, _accountType : String, _creditAmount : double, _password : String)**
  - This function is the constructor for the User class which initializes the username, user account type, credit balance, and user password.
- **+ getUsername() : String**
  - This function returns the current user's username.
- **+ setUsername(username : String) : void**
  - This function sets the username to the current user.
- **+ updateCreditAmount(credit : double) : void**
  - This function adds to the current user's credit balance when it reads an 'addcredit' operation on the merged daily_transaction file.
- **+ getAccountType() : String**
  - This function returns the account type for the current user.
- **+ setAccountType(accountType : String) : void**
  - This function sets the account type to the current user.
- **+ getCreditAmount() : double**
  - This function sets the credit amount to the current user.
- **+ setCreditAmount(creditAmount : double) : void**
  - This function sets the credit amount to the current user.
- **+ getPassword() : String**
  - This function returns the password for the current user.
- **+ setPassword(password : String) : void**
  - This function sets the password to the current user.
- **+ formatOutput() : String**

- This function sets the format for the string to be written to the new_current_user_accounts file.

3.2. UserManager

The UserManager Class is used for handling all the user related transactions (ie. create, delete, refund, addcredit, resetpassword) and update each user attribute accordingly before writing to the new_current_user_accounts file.

File input: daily_transaction.txt, current_user_accounts.txt
File output: new_current_user_accounts.txt

- **+ UserManager()**
  - This function is the constructor for the UserManager class which initializes a vector of users.
- **+ validateDouble(doubleStr : String) : double**
  - This function validates whether the credit amount read from the current_user_accounts file is a double and returns the casted double value if not.
- **+ displayUsers() : void**
  - This function prints all the users and their information from the current_user_accounts file to the console.
- **+ load(filename : String) : void**
  - This function reads the current_user_accounts file line by line and splits each attribute into its designated variables (ie. username, user type, credit balance, and encrypted password).
- **+ createUser(transaction : String) : void**
  - This function creates a new user to append to the new_current_user_accounts file if there is a ‘create’ transaction in the merged daily_transaction file.
- **+ deleteUser(transaction : String, auction : Auction) : Auction**
  - This function deletes a user and updates the user deletion to the new_current_user_accounts file if there is a ‘delete’ transaction in the merged daily_transaction file.
- **+ refund(transaction : String) : void**
  - This function processes the ‘refund’ transaction and returns credit from a seller’s account to a buyers account and this is recorded in the new_current_user_accounts file.
- **+ addCredit(transaction : String) : void**

  - This function processes the ‘addcredit’ transaction and adds the credit to the user’s account and this is recorded in the new_current_user_accounts file.
- **+ resetPassword(transaction : String) : void**
  - This function processes the ‘resetpassword’ transaction and resets the user’s password and stores the encrypted value in the new_current_user_accounts file.
- **+ closeAuction(buyer : String, seller : String, amount : double) : void**
  - This function processes the event of an auction completed. This means that an item has reached 0 days left, the buyer with the highest bid loses its credit amount to the seller and this is recorded in the new_current_user_accounts file.
- **+ write(filename : String) : void**
  - This function is used write all of the updated user information to the new_current_user_accounts file.

3.3. Item

The Item Class is used to initialize and set item attributes based on the transactions read from the merged daily_transaction file and available_items file.

File input: daily_transaction.txt, available_items.txt
File output: new_available_items.txt

- **+ Item(_itemName : String, _sellerName : String, _buyerName : String, _daysRemaining : int, _highestBid : double)**
  - This function is the constructor for the Item class which initializes a new Item for the auction system.
- **+ getItemName() : String**
  - This function returns the current item's name.
- **+ getSellerName() : String**
  - This function returns the seller of the current item.
- **+ getBuyerName() : String**
  - This function returns the current potential buyer of the current item.
- **+ getHighestBid() : double**
  - This function returns the current highest bid of the current item.
- **+ getDaysRemaining() : int**
  - This function returns the amount of days left for the current item to be held in the auction.
- **+ setDaysRemaining(days : int) : void**
  - This function sets the amount of days for the current item to be held in the auction.
- **+ newBid(buyer : String, bid : double) : void**
  - This function updates the potential buyer if a new valid bid is made on the item.
- **+ formatOutput() : String**
  - This function sets the format for the string to be written to the available_items file.

3.4. Auction

The Auction Class is used for handling all the item related transactions (ie. advertise, bid) and update each item attribute accordingly before writing to the new_available_items file.

File input: daily_transaction.txt, available_items.txt
File output: new_available_items.txt

- **+ Auction()**
  - This function is the constructor for the UserManager class which initializes a vector of items.
- **+ displayAuctions() : void**
  - This function prints all the items and their information from the available_items file to the console.
- **+ deleteAllAuctions(username : String) : void**
  - This function deletes all items that are sold by the username that is provided from the *new_available_items* file if there is a 'delete' transaction in the merged daily_transaction file.
- **+ load(filename : String) : void**
  - This function reads the avialble_items file line by line and splits each attribute into its designated variables (ie. seller, buyer, highest bid, and duration).
- **+ advertise(transaction : String) : void**
  - This function creates a new item to append to the *new_available_items* file if there is a 'advertise' transaction in the merged daily_transaction file.
- **+ bid(transaction : String) : void**
  - This function processes the 'bid' transaction and creates a new bid to an item and this is recorded in the *new_available_items* file.
- **+ endDay(userManager : UserManager) : UserManager**
  - This function processes the end of day and subtracts the remaining days for each item at the end of the day by 1 and to be updated in the *new_available_items* file.
- **+ write(filename : String) : void**
  - This function is used write all of the updated item information to the *new_available_items* file.

AUCTIONBAY

4.0. INSTRUCTIONS TO RUN PROGRAM

Application Build: Java version 17.0.2

1) Clone repo or download ZIP folder
2) Open project via IntelliJ IDEA (https://www.jetbrains.com/idea/)
3) Within project structure (on the left hand side) and expand `src/main/`:
    a) Right-click on **Main.java** and select Run `Main.main()`