PHASE 2: FRONT END RAPID PROTOTYPE & DESIGN DOCUMENTATION

By: Group007

Abdullah Alakeel

Saurabh Mishra

Priya Mohan

Prashakar Prabagaran

**Implementation Notes/Code Structure**

Our implementation for the front-end part of the ticket system is split into 3 sections.

1. Prompter: this class interacts with the user directly and only takes input. Apart from basic text validation there is no logic in this class.

when given valid input, it calls

1. Transactions: this class contains the six main methods that make up the front-end of the ticket system. These methods take care of logging each change into a transaction file that is printed out at the end of each session (once a user has been logged out) and also takes care of the logic behind each transaction.
2. TransactionHelper: this class holds all helper methods for the methods in the Transactions class that were removed to minimise code redundancy as much as possible. This is also where any file access occurs.

**Input/Output Files**

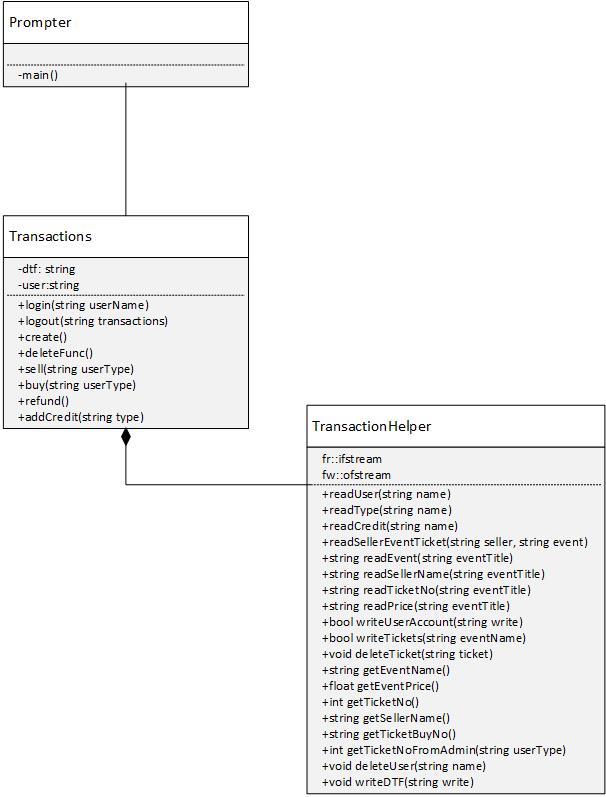
This system uses 3 files currently, userAccounts.txt, availableTickets.txt and the dtf file. UserAccounts holds all user info including user name, privilege type and credit. AvailableTickets holds all ticket info including event name, ticket prices, ticket quantity and seller name. Based on the transaction selected, the program can read/write to the user and ticket files. The dtf file is never read from; once user logs out, the transactions that occurred during the session are all printed to the dtf file (the file is overwritten at the end of every session).

**Class/Method Descriptions**

|  |  |
| --- | --- |
| Class/Method | Description |
| **Prompter** | **Class name; provides basic user-end prompts.** |
| **Transactions** | **Class name; all of the 6 required methods for the ticket system.** |
| *login* | checks user exists and returns the user type |
| *logout* | logs user out and prints dtf file |
| *create* | creates users and |
| *deleteFunc* | deletes events and users |
| *sell* | takes the event title, sale price and amount of tickets and sells ticket(s) for an event |
| *buy* | takes the event title, sale price and seller and prints out prompts that allow the user to buy tickets |
| *refund* | returns credit to seller’s accounts. |
| *addCredit* |  |
| **TransactionHelper** | **Class name; holds all helper methods, including methods accessing extra files** |
| *readUser* | checks if user exists and returns user name if it does and error string if not. |
| *readType* | if user exists, returns user type and error string if not. |
| *readCredit* | if user exists, returns user credit amount and error string if not. |
| *readSellerEventTicket* | checks if event exists and returns seller if so and error if not |
| *readSellerName* | finds and returns seller name from the ticket file |
| *readTicketNo* | finds and returns number of available tickets from the ticket file |
| *readPrice* | finds and returns the ticket price from the ticket file |
| *writeUserAccount* | writes info to user account file |
| *writeTickets* | write info to tickets file |
| *deleteTickets* | deletes tickets/events from tickets file |
| *getEventName* | asks user for event name and validates input |
| *getTicketNo* | asks user for ticket amount and validates input |
| *getSellerName* | asks user for seller name and validates input |
| *getTicketBuyNo* | asks BS/FS user for ticket amount and validates input |
| *getTicketNoFromAdmin* | asks admin for ticket amount and validates input |
| *deleteUser* | deletes user from user accounts file. Main helper method for deleteFunc |
| *writeDTF* | prints daily transaction file. |

This was implemented in Visual Studio 2013.

**UML Class Diagram**

****