Rohit Tandon
ICS372 Object Oriented Design and Implementation
Group Project One – Part One

Project Goals

- Form groups and plan work for successful project submissions
- Follow the specifications given and write compliant code.
- Go through system analysis and design for business scenario.
- Document and lay out your code properly as specified under ICS 372 Coding Standards.

Project Description

In this project, you will implement some of the functionality of a theater. The theater has a name and a flexible seating capacity.

The theater has a number of clients (the number varies with time). A client shows plays for a certain period of time (dates). For example, "Central Playhouse" may have the play "Mousetrap" which it shows for a two-week period starting June 11, 201*8*.

The theater sells tickets to registered customers; that is, to watch a show, a person must first become a member (or customer) (there is no fee to become a member), and have at least one valid credit card. A customer may have multiple credit cards. No two customers may have the same credit card.

A client has a name, address, and phone number. The theater assigns a unique id to each client and keeps track of the balance due to the client. In this iteration, you will not do anything with the balance.

The theater keeps track of the name, address, phone number, and all registered credit cards of a customer. Each credit card has a number and expiration date. The billing address of a credit card is assumed to be the same as that of the customer's address.

In this iteration, you will implement the following:

1. Exit the Application. Store the data on disk and quit the application.
2. Add Client. The system accepts the name, address, and phone number of the client. The system generates a unique id and sets the balance to 0. (The balance will remain 0 in this iteration.)
3. Remove Client. Remove a client with the given id. If a show is scheduled for the current or a future date for this client, the client cannot be removed.
4. List all Clients. Print information about every client.
5. Add Customer. The system accepts the name, address, phone number, and the number and expiry date of exactly one credit card. The system generates a unique id for the customer.
6. Remove Customer. Remove a customer with the given id. All credit cards related to the customer are also deleted.
7. Add a Credit Card. The system accepts the customer id, credit card number, and expiry date and remembers that the credit card belongs to this customer.
8. Remove a Credit Card. The system accepts the credit card number and removes the information related to the credit card. If this is the only credit card for the customer, it refuses to delete the credit card information.
9. List all Customers. Print information about every client, including credit card information.
10. Add a Show/Play. Add a new show for a client. The values accepted are the name of the show, the client id, and the period for which the client wants the theater for this play. The entire range of dates should be available, or the process fails.

11. List all Shows. List the names and dates of all shows.
12. Store Data. Store all data related to the theater (everything, including customers, shows, clients, etc.) on disk.
13. Retrieve Data. Retrieve all information related to the theater. This may be done at the start of any session. If stored data is found, the user has the option to use it. The user may also invoke a command to load data, provided he/she has not yet issued any data-related commands in that session.
14. Help. This should display all commands.

You must have a command line interface (not GUI) that uses numbers for issuing commands. It should be possible to invoke each functionality by typing the number associated with it as in the above list. For example, 0 must exit the application and 13 should display the help menu; 8 should list all customers.

Program Submission

1. A use case for every one of the 13 business processes written in the two-column format.
2. Conceptual class diagram.
3. Class diagram for every class and/or interface with their relationships.
4. Sequence diagram for every use case.
5. Java code for every class, including the user interface.

Zip the source files and submit in the D2L dropbox. *If the program has syntax errors, the grade will be 0: no exceptions.*