UFCFW4-30-2 Design and analysis of data structures and algorithms

Coursework submission

Justification

During the design phase of the project, some modifications were made to the format of the provided data to better fit the theme of a software application designed to deal with user, inventory and transaction management. Despite the fact that the assignment specification does not call for some of the features implemented, the additional modifications on top of the original spec help to realise the theme of the project and better flesh out the system from a top down perspective.

All of the original requirements have been met and the project can be considered functionally complete despite attempts to further progress the software beyond the original design requirements.

Early on in the design phase, the decision was made to utilise JSF (Java Server Faces) as a framework for the system. The component driven UI model of JSF allowed for more complicated UI elements to be quickly put together, prototyping and iteration became much more streamlined as the design evolved.

As mentioned previously, another alteration to the original spec removes the CSV file and replaces it with a JPA driven Hibernate database. Allowing for the creation and use of data objects for interacting with the persistence entities for Members, Items and Transactions. This way, the application is configured to drop and create the database from a template every time the application is redeployed, preventing the need to create multiple copes of the provided CSV files.

The use of Hibernate was inspired by the binary tree data structure used by SQL based databases, binary tree provides search and insertion in logarithmic time across a sorted set. As Transactions are added sequentially, this was an ideal fit for the problem space as the data set is naturally sorted by the manner in which it is inserted.

Additionally, Array Lists are used heavily by the application as it was decided that almost all of the functionality required by the design could be achieved using object Array Lists of type Item, Transaction and Member respectively.

These are iterated over using a search method of linear time (O(n)) when performing comparison and whilst it is understood that this is not the most efficient solution, it is acceptable for the size of the problem space. Particularly as most of the comparison operations performed by the software do not require iteration over the entire data set, rather a subset of that data relevant to the last 7 days worth of transactions.

An additional class called SimpleTransaction was created to allow comparison of transaction elements without worrying about the unique identifiers using just the item and member numbers of each transaction.

The only other data structure in use by the system is a queue linked list used for storing transactions for items that are currently unavailable, due to the specification not strictly requiring that the queue list process queued transactions, the logic to process the queue is not implemented. However, the logic to add, pop and peek elements from the queue is in place and demonstrable.

When voiding old transactions, a flag was added to indicate that a voided transaction had been processed to prevent it from being processed again the next time the transaction retrieving daemon was invoked.

A full suite of CRUD (create, read, update and destroy) operations are in place for the Member and Item classes, CRD is in place for Transaction as it was decided that transactions would not need to

be edited if they can be deleted and issued again, preventing the need for more complex validation code of live transactions to ensure that Item availability counts are accurate

The system checks all transactions older than 7 days by running a retrieval daemon to analyse and transactions with an expiration flag of 0, 1 indicated that the transaction is either within the 7 day window or has not been voided and will be collected next time the daemon runs.

There are numerous validation processes to check that one user can not book more than 5 items (queued or active) in the same week or book the same item twice.

Additionally, there are methods in place to search Items on an Item type, this was not included in the original data format guidelines and was added as additional ‘nice to have’ functionality.

The spec was interpreted as master application with only one primary user, as a result there is no log in system and the app runs on the JBoss application server for java web apps.

What follows are dependency requirements for building the project on a local machine:

- JDK 1.8 + for the project Java version
- JDK 1.7 for JBoss AS.
- Project specific dependencies are managed through Maven and can be examined in the projects .pom file.