**Design Team 7**

Kien Chin

Juliano Nguyen

Shanelee Tran

William Verthein

**Overview**

In the Business Logic class, it will contain methods to build the movies, build the customers, and process the commands. Main will use these functions from the Business Logic class. Main will have a Customer object and a Movie object. The HashTable class will store all the customers from the Customer class, and the Movies will be stored in a vector, which will be from the STL. The Transaction class will interact with the Inventory, History, Borrow, and Return classes. The Transaction Factory will be a composition to the Transaction class, which will create a Transaction depending on the type of those four commands. The Transaction Factory will also be a composition to the four command classes as well. The Movie class will have a Comedy, Drama, and Classics class, where they will inherit the data and functions from the Movie class.

**CLASSES**

**Rental Store**

This base class will serve as main and it will run the Business Logic methods.

**Business Logic**

This class will contain methods to build the movies, build the customers, and process the transactions. Each method will pass in a text file to read and build/process the data.

**Transaction**

This class is the base class for the different types of transactions a user can do. It will have Borrow, Return, Inventory, and History for its sub-classes. This contains a doTrans() method that will call an overwritten doTrans() in its sub-classes depending on the transaction code called. Transaction is also responcible for converting the string commands to managable variable that it then send to the transaction factory.

**Transaction Factory**

This class will contain a createTrans() method that will create an instantiated object depending on the transaction code from the Transaction class. It will use a switch case to determine what type of object will be created and returned according to the read transaction code.

**History**

This will be a sub-class that will inherit from the Transaction class. It will output all the transactions of a customer in chronological order, from the latest to the earliest transaction. It will also show whether a movie was borrowed or returned in that transaction. This transaction type will be labeled as “H’.

**Inventory**

This will be a sub-class that will inherit from the Transaction class. It will output the inventory of all the items in the store in a sorted order. It will output Comedy, Drama, then Classics movies. Each movie will be sorted by their order rule from the document. This transaction type will be labeled as “I’.

**Borrow**

This will be a sub-class that will inherit from the Transaction class. It will subtract 1 from the stock when a movie has been borrowed. It will read in a text file with a customer ID, a media type, movie type, and movie data per line. This will output an error message to the user if there is incorrect data or incorrect command. This transaction type will be labeled as “B”.

**Return**

This will be a sub-class that will inherit from the Transaction class. It will add 1 to the stock when a movie has been returned. It will read in a text file with a customer ID, a media type, movie type, and movie data per line. This will output an error message to the user if there is incorrect data or incorrect command. This transaction type will be labeled as “R”.

**HashTable (Open)**

This class will store all the customers from the text file. It will have a hash table from index 0-9 and the hash function will be h(x) = x % 10. It will read in a text file of customers and this hash function will store the customer ID at the index from the last digit in their ID.

**Customers**

This class represents the Customers from the text file. It will contain the customer ID, last name, and first name. Customers will also contain a struct class for the user's history which has two data types, the movie that was borrowed and whether or not it was returned.

**Vector for Media Type**

This class will store the different types of media that a rental store can have, such as Movies, Music, etc.

**Vector**

This class will create a new vector for each unique media type. For example, a vector will store the different categories of movies, such as Comedy, Drama, Classics.

**Movie**

This class is a base class for the different types of movies. It will have Comedy, Drama, and Classics for its sub-classes. This contains the stock, director, title, and year released for the movie. This class will be used to store the movies in a HashTable class.

(vector of linked lists)

**Comedy**

This sub-class will inherit the data members and methods from the Movie class. The movies will be labeled as “F”.

**Drama**

This sub-class will inherit the data members and methods from the Movie class. The movies will be labeled as “D”.

**Classics**

This sub-class will inherit the data members and methods from the Movie class. This will also contain additional data members, such as major actor and day of release. The movies will be labeled as “C”.

**FUNCTIONS**

**doTransaction (Inventory)**

This method will be overwritten to satisfy the Inventory functionality.

Outputs all Comedy, Drama, and then Classics movies.

Each movie category is sorted according to its own criteria.

Output on one line per item.

**doTransaction (History)**

This method will be overwritten to satisfy the History functionality.

Takes in a customer ID.

Outputs a list of transactions of a customer from latest to earliest.

Outputs whether a movie has been borrowed or returned.

**doTransaction (Return)**

This method will be overwritten to satisfy the Return functionality.

Adds one to the current stock of a movie item.

**doTransaction (Borrow)**

This method will be overwritten to satisfy the Borrow functionality.

Subtracts one to the current stock of a movie item.

**TransactionFactory (constructor)**

This will receive the Transaction’s data (customer ID, media type, movie type, movie data) and determine what type of transaction it would do in a switch/case statement. Then, it will create a new object of that transaction type.

**TransactionFactory (destructor)**

This will delete the object in respect to its transaction type. This will be used in an if statement to determine the transaction type for the object that will be deleted.

**Movie (constructor)**

This will create a movie object that contains the data members according to its movie type (Comedy, Drama, Classics).

**Movie (destructor)**

This will go through each index, redirect the pointer to the next movie object, and delete the current head until all movies in the vector are deleted.

**Customer (constructor)**

This will create a customer object that contains the customer ID, last name, and first name.

**Customer (destructor)**

This will go through each index, redirect the pointer to the next customer object, and delete the current head until all customers in the hash table are deleted.

**getStock**

Returns the total stock value of a movie item as an int.

**insertMovie** (Comedy)

This method will be overwritten to satisfy the Comedy sorting criteria. It will insert the comedy movie object in the vector in sorted order by Title, then Year released.

**insertMovie** (Drama)

This method will be overwritten to satisfy the Drama sorting criteria. It will insert the drama movie object in the vector in sorted order by Director, then Title.

**insertMovie** (Classics)

This method will be overwritten to satisfy the Classics sorting criteria. It will insert the classic movie object in the vector in sorted order by Release date, then Major actor.

**buildMovieInventory**

It will read in a text file of different types of movies and add them to a vector in its respective slot (Comedy, Drama, Classics). This will print an error message if a line contains a negative stock number and/or invalid movie type.

**buildCustomers**

It will read in a text file of customers containing their ID, first name, and last name per line. This will print an error message if their ID is below 0000 or over 9999. It will store the customers in a hash table according to their last digit.

**processCommands**

It will read in a text file of transactions per line. This will print an error message if a line contains an invalid transaction type, incorrect customer ID, invalid media type, invalid movie type, and/or invalid movie data.

**getDirector**

Returns the director’s name of the movie item as a string.

**getTitle**

Returns the title of the movie item as a string.

**getYear**

Returns the year released of the movie item as an int.

**getDayRelease**

Returns the day released of the movie item as an int.

**getMajorActor**

Returns the name of the major actor of the movie item as a string.

**getName**

Returns the customer’s first and last name as a string.

**getID**

Returns the customer’s ID as an int.

**hash**

It will build a hash table to store all the customers by their ID. The hash table will be from 0 to 9, and each customer will be stored by their last digit into the table.

**findCustomer**

It will find the customer in the hash table according to their ID to find their transaction history and whether a movie in each transaction has been borrowed/returned.