**Description of each Class and the function of their most significant Methods**

**Overview:**

This project is designed to track the inventory for a local movie rental store. Customers can borrow and return movies from the stores and this program helps keep track of the activity of each customer and the movie availability in the store. The program saves all activities of a customer to a History object so that Customer can access their transaction history. The Program prevents incorrect transaction and if valid, the customer transaction is processed and returns a confirmation notice of the successful transaction.

**BinarySearchTree Class**

The Binary Search Tree Class will act as our data structure to store the assortment of movie objects. This containment system will act as a templated class to handle the different functionalities of the the different sorts for categorization in each movie genre(Drama, Classic,Comedy). It will function to insert, retrieve, and display the available to movies

**NodeData Class**

The Node Data Class serves as the logic behind the manipulation between the comparison operations and the properties of how the different elements should work together.

**Movie Class**

The Movie Class is an abstract class that has the attributes of a movie’s stock title, director, year, and movie genre. These variables will This class functions to hold the abstract methods of the comparison operator overloads that will function to help determine the sorting interactions within the Binary Search Tree action methods. Because the design documents require us to sort differently for each type of genre, it will work best to extend the methods for comparison. There are also assumptions being made about the valid syntax for the functions, with the genre matching the available classes.

**Classics Class**

The Classics Class is the derived class of the base class Movie Class. It will be the process for inventory storage of the Classics genre. Unlike the Comedy and Drama Class, which is listed by Stock, Director, Title, and Year it released, The Classic Class is sorted by Stock, Director, Title, Major actor Release datewhich differs slightly in properties for data collection and sorting. The data will also be stored privately as the variables actor, month, and movieType(genre). The Classic movies will be denoted by C and will be sorted according to the Release date, then Major actor.

**Drama Class**

The Drama Class is the derived class of the base class Movie Class. It will be the process for inventory storage of the Drama genre. Similar to the Comedy Class, it will be listed by Stock, Director, Title, Year it released. The Drama movies will be denoted by D and will be sorted according to the Director, then Title.

**Comedy Class**

The Comedy Class is the derived class of the base class Movie Class. It will be the process for inventory storage of the Comedy genre. Similar to the Drama Class, it will be listed by Stock, Director, Title, Year it released. The Drama movies will be denoted by C and will be sorted according to the Title, then Year it released.

**Inventory Class**

The Inventory Class is the management class for how the inventory will be controlled and validated through the changes of transactions in the program. The Inventory class will manipulate the stock of each movie and update valid actions of returns and borrows for each transaction made.

**Factory Class**

The Factory Class will be the central process for accessing and processing data that is inputted from the customers, movies, and commands files. It will build the inventory based on the valid imputed movies, which will be stored in the BinarySearchTree, read in the customers, which will be stored in the hashTable, and process the Transactions in the transaction class for each type of command. If the transaction type is **B** which is “borrow”, a private method will check if Inventory Class has enough stock. If so, it will give authorization to process the transaction. If the transaction type is **R** which is “return”, a private method will access the Transaction class to check if the return transaction is processable. If it is valid, the Factory class will access the Inventory class and add stock numbers.

**HashTable Class**

The HashTable Class will act as a data structure to store customer’s data including customer ID, its first name and last name. HashTable will be implemented as a pointer array which can store STL lists. It will have N number of buckets which will be defined through its constructor. To insert customer data into the hash table, hashFunction will be called to find the hash index for the given key which will be customer ID. It will be calculated as “index = customerID % (# of buckets)”. Then the customer data will be stored at the index of the hash table.

**Customer Class**

The Customer Class will have customer’s ID, first name, and last name as variables. It contains getter and setter functions for all variables. A function will read and set customer’s data to the HashTable class which will be inherited by Customer class.

**Transaction**

The Transaction Class is an abstract class that has the attributes of a Transaction type and media type to differentiate and keep track of the kind of movie the customer wants to borrow, return or get history from. This class contains a function setData(isStream& ) that gets commands from an input file and then computes the transaction based on the type of transaction that the customer ordered. Processes its transaction based on the customer preferences readed from the input file, and then records the transaction to its History derived class of the customer.

**Borrow**

The barrow class is derived from the transaction class. Denoted as ‘**B**’, it returns the number of movie transactions that have been borrowed by customers. Every time a movie a customer effects a barrow transaction, this Barrow class will keep track of each item borrowed by decreasing the stock by one (stock – 1).

**Return**

Derived from the Transaction class, this class denoted **“R”** keep track of movie that has been returned by customers. A function setData() will keep track and increment the stock by one every time the customer will effectuate a return transaction.

**History**

The History class derived from the Transaction class and it is denoted **“H”.** This class contains a display() function that will help the Transaction class generate a call to display the transaction History of a specific customer. The build() methods help the generate a transaction history for a customer that would be stored in the History to be accessible for the customer record.

**Description of Main**

The main function will be used to read in the file from the provided methods of

+ buildInventory(istream&): void

+ ReadCustomer(istream& ): bool

+ processTransaction(istream&): void

+ movieInventory[]: Inventory

+ transactionInventory[]: Transaction

- validReturnTransaction(istream& ): bool

- validBorrowTransaction(istream& ):bool

+changeStock(int numOfStock): bool

These methods will act as a central process for accessing and processing data that is inputted from the customers, movies, and commands files.

It will build the inventory based on the valid imputed movies, which will be stored in the BinarySearchTree, read in the customers, which will be stored in the hashTable, and process the Transactions in the transaction class for each type of command.

The building of the inventory will take in the movie type and insert the movie with the type of genre that matches the read in line into the BST that has the correct type of sort.

The transaction process will determine the read in commands that will be valid for the manipulation of the inventory stock.