**Assignment 4 (Group 2)**

**Group members:** Rui Qi Huang, Brittney Oeur

**What each member did:**

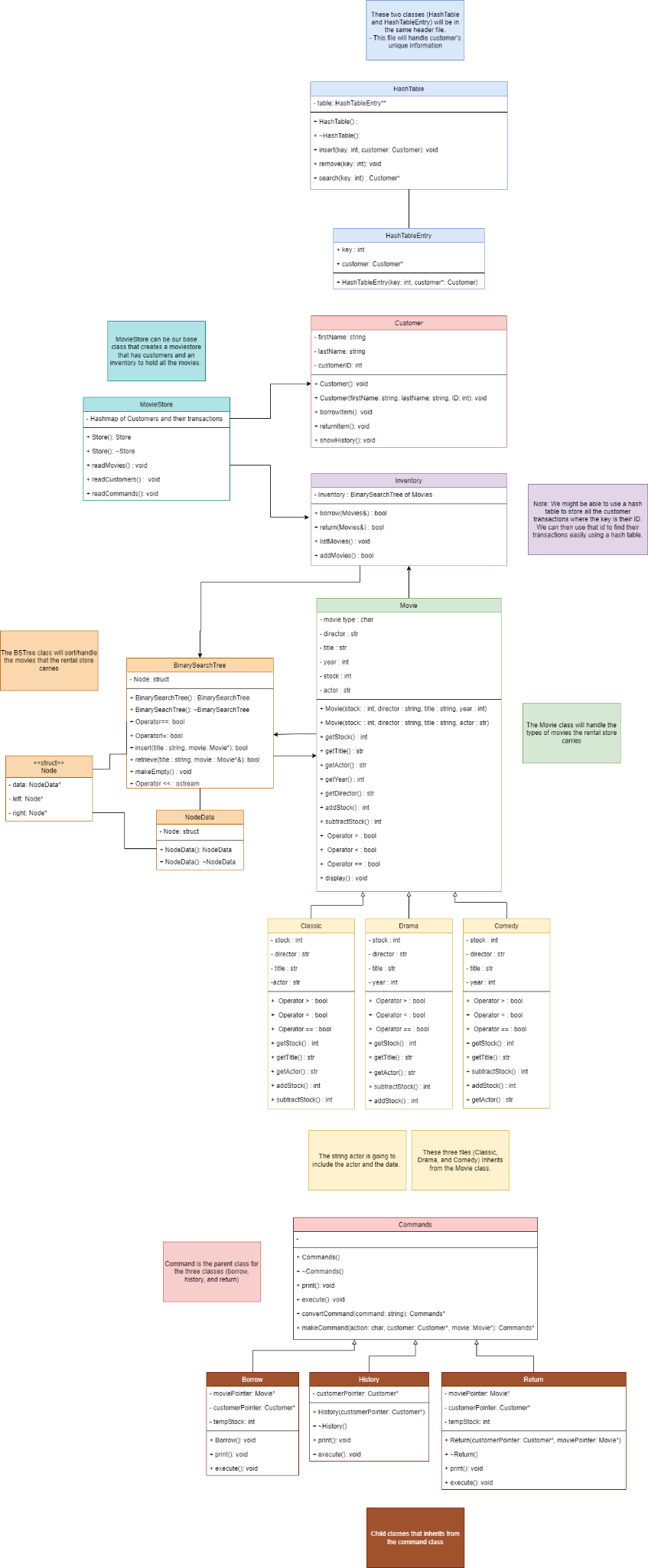
**Rui Qi:**

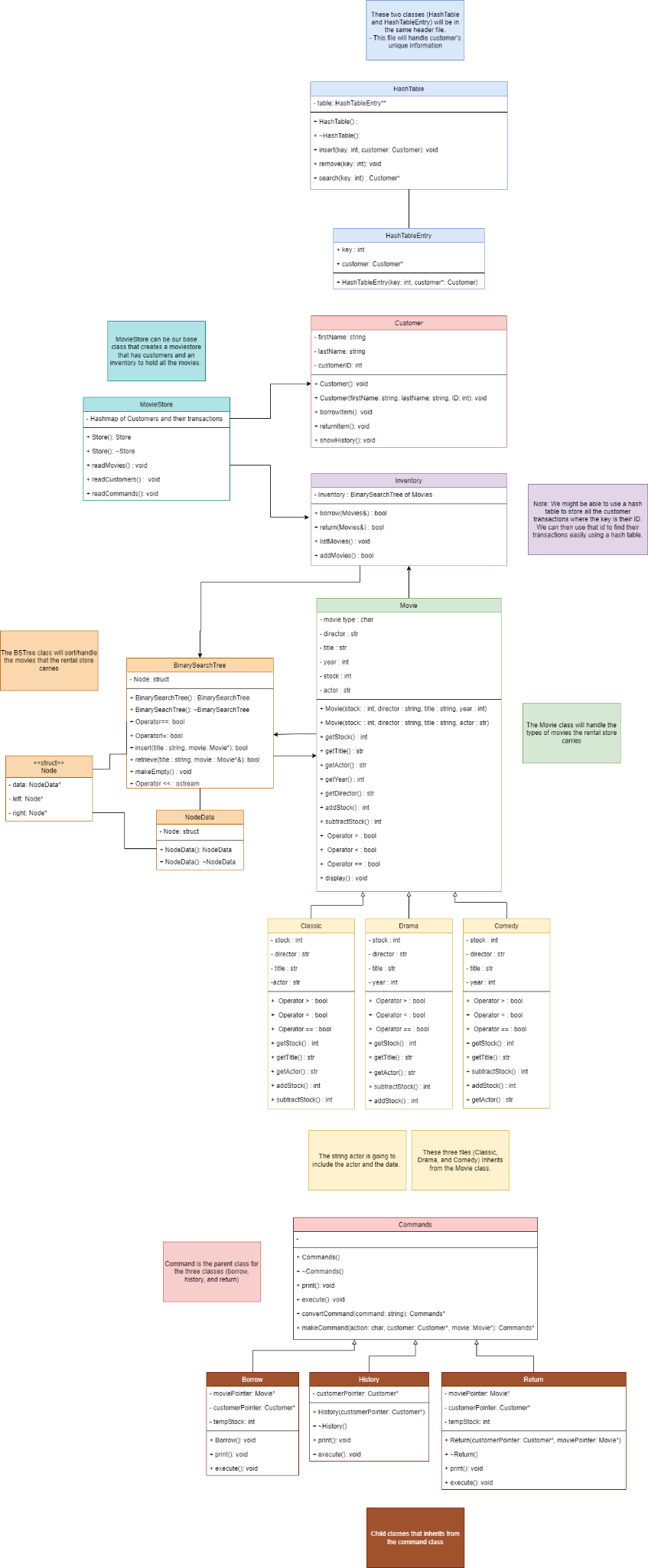
* Worked on the UML design
* Designed the BSTree implementation
* Designed the Movies implementation
* Designed the three genres of movies implementations (EX: Comedy, Classic, and Drama)
* Designed the MoveStore implementation
* Designed the Inventory implementation

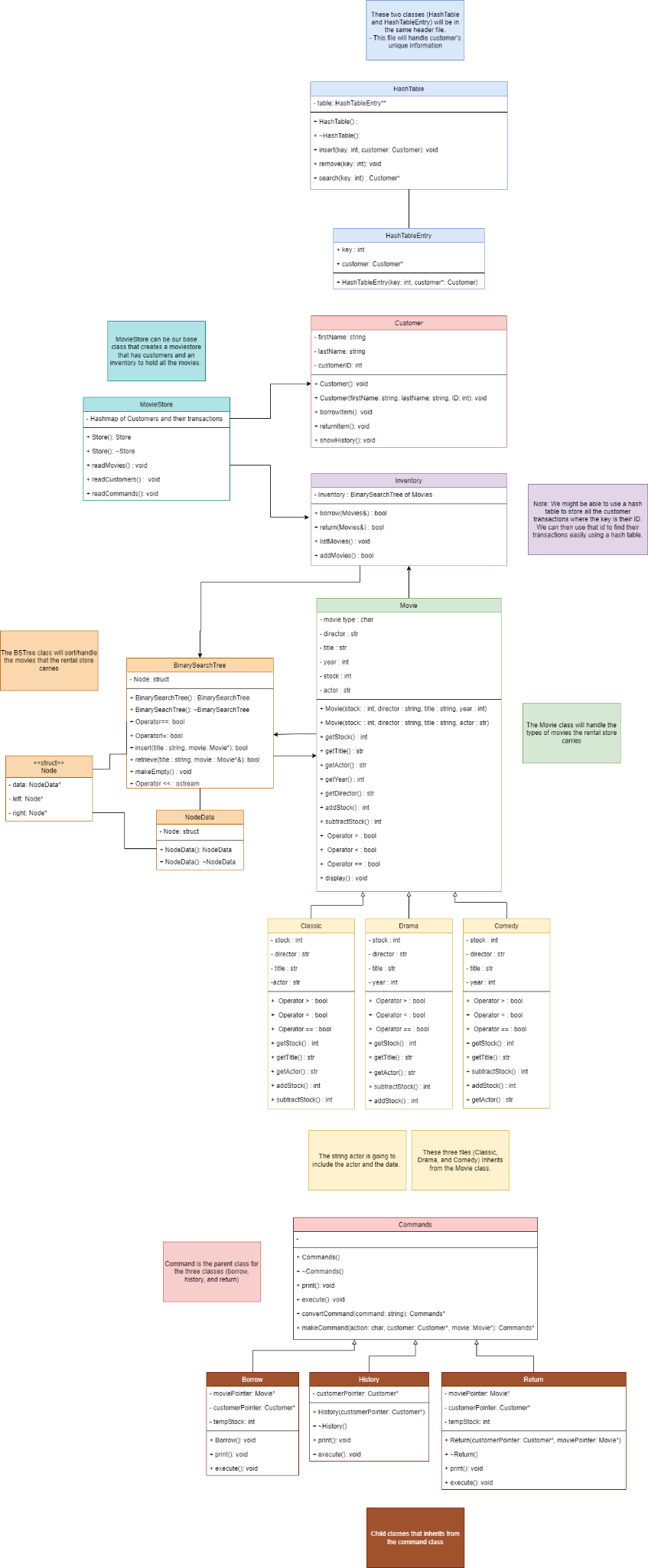
**Brittney Oeur:**

* Created and organized the Word document
* Worked on the UML design
* Designed the HashTable implementation
* Designed the Commands implementation
* Designed the three commands implementation (EX: Borrow, History, and Return)
* Designed the Customer implementation

**UML DESIGN**



****



**Explanation**

**Customer.h**

* This will store the customer’s information, such as their digit ID number that uniquely identifies them, followed by their last name, first name
  + Has different methods for each actions the customer wish to do

**Movie.h**

* This will handle the types of movies the rental store carries
* Some output method
  + (ostream& out, Movie& movie)
    - Out << movieType << “, “ << stock << “, “ << director << “, “ << title;
    - Return out;
      * Does not return all the different types of information from a movie, as each movie has its own special data that must be printed in its own class. Such as classics, you must include major actors and release date in a unique format

**Comedy.h**

* Child class from the Movie class
  + Constructor class to set the default director, title, year
    - Default and parameterized constructors
* Overload operator ==
  + Checks whether the two comedy movies are equal to each other
  + Returns bool
* Overload operator < & overload operator >
  + Checks whether one comedy object is less or greater than the other
    - This will help sort out how the comedy movies will be listed
    - Possibly something like, bool Comedy::operator<(const Movie& movie) const
* Some output, that utilizes the Movie class
  + ostream& Comedy::(ostream& out, Movie& movie) const
    - Out << “, “ << release year
    - Return out;
      * Written this way because you must include the released year for comedies

**Classic.h**

* Child class from the Movie class, will hold the classic movie type’s attributes
* Overload operator ==
  + Checks whether the two comedy movies are equal to each other
  + Returns bool
* Overload operator < & overload operator >
  + Checks whether one comedy object is less or greater than the other
    - This will help sort out how the comedy movies will be listed
    - Possibly something like, bool Comedy::operator<(const Movie& movie) const
* Some output, that utilizes the Movie class

**Drama.h**

* Child class from the Movie class, will hold the drama movie type’s attributes
* Overload operator ==
  + Checks whether the two comedy movies are equal to each other
  + Returns bool
* Overload operator < & overload operator >
  + Checks whether one comedy object is less or greater than the other
    - This will help sort out how the comedy movies will be listed
    - Possibly something like, bool Comedy::operator<(const Movie& movie) const
* Some output, that utilizes the Movie class

**HashTable.h:**

* This will handle/organize the unique customer information
  + By using the hash table, it will be easy to find their transactions (based on the customer’s ID)

**BinarySearchTree.h**

* This will help handle organizing/handling the movies that the rental store carries
* It includes the ‘NodeData.h’ file in this implementation
* Includes a ‘struct’ called ‘Node,’ that will handle the pointer to the data object, left and right subtree pointers, as well as the root pointer

**NodeData.h**

* This helps organize the nodes in the binary search tree

**Inventory.h**

* It is in charge of the inventory of the rental store, this includes borrowing, listing the remaining movies, adding movies, returning the movies, and lastly a binary search tree to organize the movies

**MovieStore.h:**

* It is the base class, where it creates the movie store, which includes customers, the inventory, and the transaction

**Commands.h**

* It will handle the given commands that were written from the text file
* Execute method will handle executing the commands
* convertCommands method will handle converting the char type from the text file to the command that fits with it
* makeCommands method will create the different types of commands (borrow, return, and inventory)

**Borrow.h**

* Child class from the Commands class, this class borrows movies from the rental store
  + Holds pointers of Customers and Movies
  + tempStock, stores the stock after it has been borrowed
  + Prints out what was borrowed

**Return.h**

* Child class from the Commands class, returns the movies from the rental store
  + Holds a pointer of Customer
  + Prints out what was returned

**History.h**

* Child class from the Commands class, will print out the history, output all the transactions for each customer
  + Holds pointers of Customers and Movies
  + Prints out what is the history of transactions of each customer