Group members

Duy Bui (duyvu263@uw.edu)

Vincent Le (vle22@uw.edu)

Thong Ton (tonthong@uw.edu)

Overview

For this assignment design, we decided to have a store manager to manage the store's system. This is where the store takes in customers, movies, and customer actions. Each movie will have a specific genre and stock stored in the system to keep track of the inventory. Each customer will also have a specific ID. For each borrowed action or return action for the customers, the number of stock will increase or decrease depending on the number of orders and the type of operation. Each movie will have its information stored so the system manager can easily acquire its information or modify it. Most of the information will be stored inside a hashtable so it can be easily accessed.

Objects and classes

Medias Collection: Store all media objects.
Media Factory: Create a media object then return the object.
Media: Abstract parent class, use for specific media genre: movie, or further extend.
Movie Factory: Create a movie object then return the object.
Movie: Abstract parent class, use for specific movie genre: classic, comedy, drama, or further extend.
Classic: Have information of movie types of classical: stock, title, author, year.
Comedy: Have information on movie types of comedy: stock, title, author, year.
Drama: Have information of movie types of drama: stock, title, author, main actor, month, year.

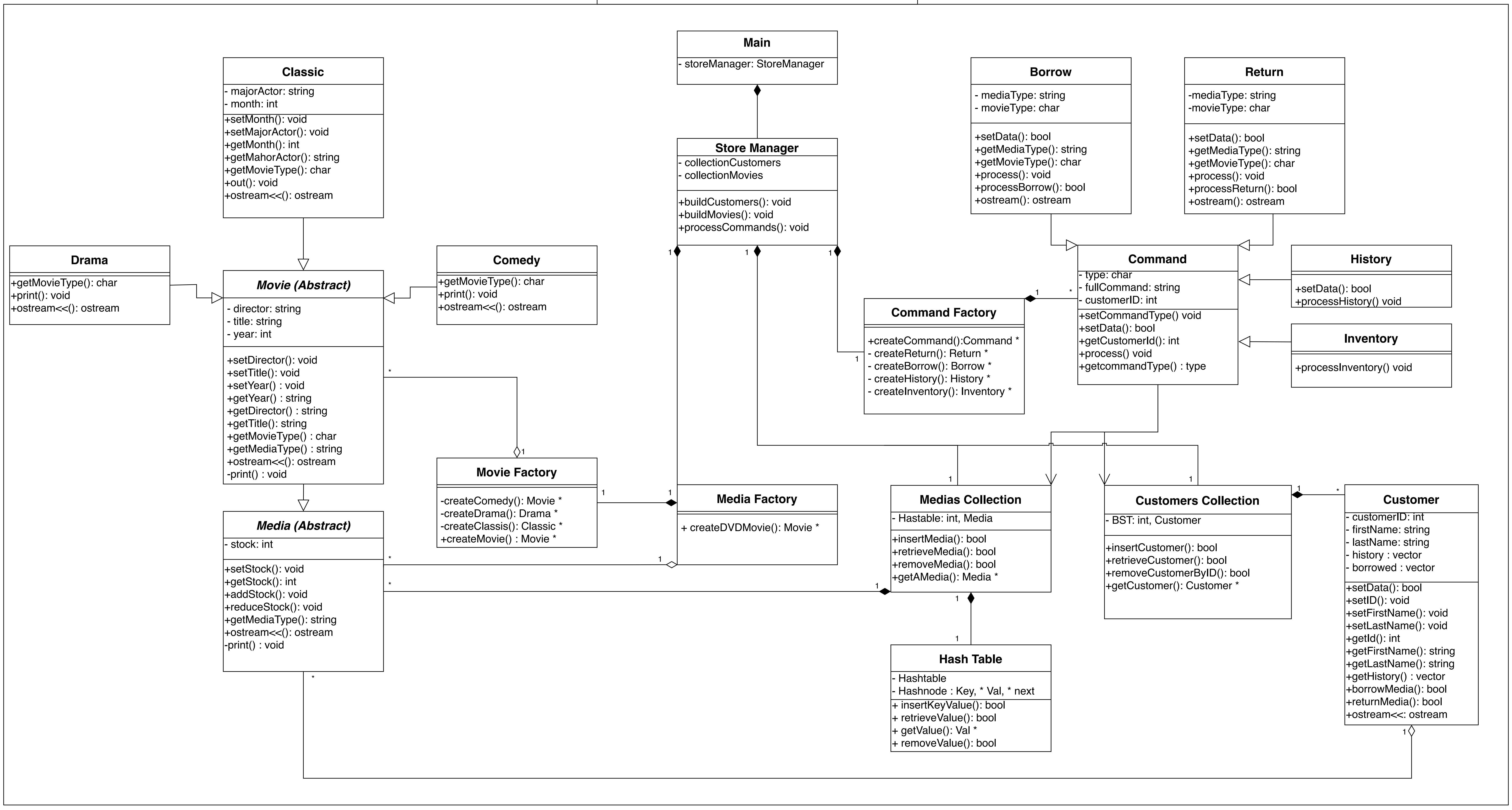
Customers Collection: Store all customer objects.
HashTable: Build a data structure to store customer objects.
Customer: Have a customer's information: id, current borrow media, history.

Command Factory: Create a command object then return the object.
Command: Parent class, use for the specific command type: borrow, return, history, inventory.
Borrow: Have information about the borrow command.
Return: Have information about the return command.
History: Have information about the history command.
Inventory: Have information about the Inventory command.

Store Manager: Read customers, movies from the given file then store them in the collection. Also, read files from the given file, process commands.

Main: Run the manager.

UML



Pseudocode

Manager

```
// instance customer collection
// instance media type movie collection

// function name: buildCustomers(file)
    // read until the end of file
        // create a customer object, set object data customer->setData(infile)
        // add each customer CollectionCustomers.insert(customer)
        // if CollectionCustomers.insert(customer) return false
            // handling error here

// function name: buildMovies
    // read until the end of file
        // create a movie object by MediaFactory::createDVDMovie(infile)
        // add movie CollectionMovies.insert(movie)
        // if CollectionMovies.insert(movie) return false
            // handling error here

// function name: processCommands
    // read until the end of commands file
        // create a command object by CommandFactory::createCommand(infile)
        // process commannd
            // command.processCommand(collectionMedias, collectionCustomers)
            // handling error here
```

Main

```
// start main
    // instantiate manager class: Store Manager
    // instantiate file stream for setting data:
        // read data4customers.txt
            // then build storeManager.buildCustomers(file)
        // read data4movies.txt
            // then build storeManager.buildMovies(file)
        // read data4commands.txt
            // then process commands by storeManager.processCommands(file)
// end of main
```

Media Factory

```
// start function createDVDMovie
    // return a pointer of movie  by MovieFactory::createMovie(infile)
// end function
```

Movie Factory

```
// start function createMovie
    // create pointer movie
    // making a switch or if-else
        // case D, create new Drama();
        // case C, create new Comedy();
        // case F, create new Classic();
        // default, if not D,C,F in valid command
    // if command exist
        // set data movie, movie->setData(infile)
    //return a pointer of command
// end function
```

Command Factory

```
// start function createCommand
    // create pointer command
    // making a switch or if-else
        // case B, create new Borrow();
        // case R, create new Return();
        // case H, create new History();
        // Case I, create new Inventory();
        // default, if not B,R, H, I in valid command
    // if command exist
        // set data movie, Command->setData(infile)
    //return command
// end function
```