Assignment 4: Design

Elias Alabssie, Antong Cheng, Lloyd Deng

Overview

The Inventory class stores Customer, Movies, and Commands data. Customers can be identified by their ID number, thus will be stored in a Hash map. Movies within a genre must be sorted by their attributes, thus will be stored in a set. A vector will store the sets of movies, allowing easy addition of genres. Commands will be read and stored in a Vector. If instructed to do so by main, Inventory may execute each of the four available commands, detailed at the bottom of this page.

Customer data type contains data of the customer's ID number and their name.

Command data type contains all information listed in each command, a getter for the ID, and a method to return all data as a formatted string. Data values in a Command object may vary depending on the movie genre.

Movie contains the genre of movie, the amount of copies of the movie in stock, the title of the movie, a getter and setter for the stock to handle adding duplicate movies, and a getter for the title of the movie. It also contains one of the three child classes, each representing a movie fitting in one of the three genres.

A genre class contains data values unique to that genre, and comparison operators which sort these values against movies of the same genre. Additional child classes may be easily added with introduction of new genres.

The main class will read the data4commands.txt, data4customers.txt, and data4movies.txt files. It will store data into the Hash map, vector of sets, and vector, respectively. Once finished, it will iterate through the vector of commands, making calls to the printInventory();, printHistory(int ID);, borrow(string title); and return(string title); functions.

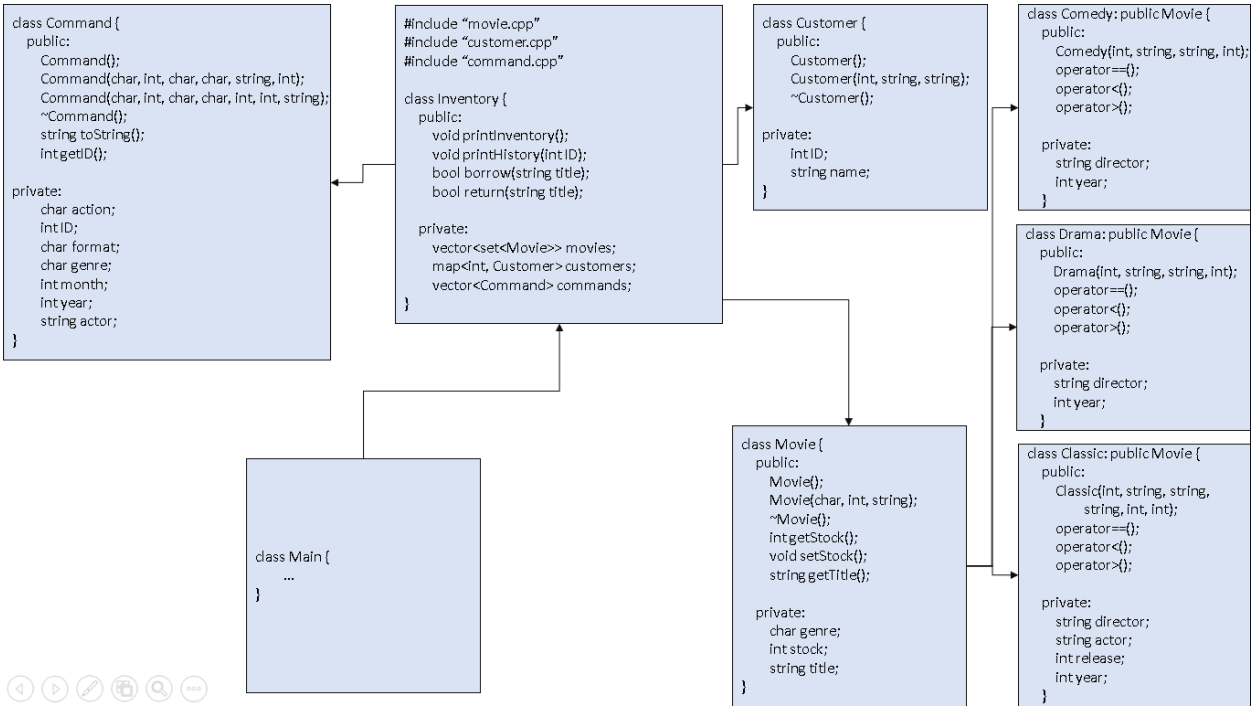
printInventory(); prompts the program to iterate through and cout << getStock(); for each vector in the array of Movies.

printHistory(int ID); prompts the program to iterate through and cout << toString(); for each element in the commands vector.

borrow(string title); prompts the program to iterate through each vector in the array of Movies, until reaching one whose getTitle(); matches string title. Then, it will setStock() = getstock() - 1;

return(string title); prompts the program to iterate through each vector in the array of Movies, until reaching one whose getTitle(); matches string title. Then, it will setStock() = getstock() + 1;

Class Diagram



Class Descriptions

Global Headers:

Assignment: Lab 4
Group Members: Lloyd Deng, Antong Cheng, Elias Alabssie
Date: March. 3, 2019

local headers:

(Note: all the local headers are for the .cpp files. For .h files we will change the ".cpp" to ".h" and "implementation" to "interface".)

/*class Inventory

File Name: Inventory.cpp

Parent: None

Child: None

Implementation file for the inventory of the movie archive. There are 4 major actions: Borrow, Return, PrintInventory, and History; 2 major data members: customers and movies, both hash tables, each containing a list of pointers to "Customer" and "Movie" objects, respectively.

*/

/*class Commands

File Name: Commands.cpp

Parent: None

Child: None

Implementation file for the input commands (as in the provided txt file). Each line of command is created as a command object before being processed by the Inventory class. Contains 2 major functions, “toString” for printing out the command lines and “getID” for returning the ID of the customer performing that action. Contains 7 private instances for identifying the command. For different genres of movies, the command lines are different, so extraneous instances are left as “null” and whenever we perform an action with a command, the genre is checked so we won’t try to access invalid data.

*/

/*class Customers

File Name: Customers.cpp

Parent: None

Child: None

Implementation file for the individual customers of the movie archive. Contains 3 major data members, “serial”, “firstName”, and “lastName” to identify the customers identity. There are 3 operator overloads, “>”, “<”, and “==” to assist sorting the customers in the hash table. (details for overloads are not implemented until we reach the hash table chapter in class)

*/

/*class Movies

File Name: Movies.cpp

Parent: None

Child: Classic, Drama, and Comedy

Implementation file for the generic movie type. Contains only one major data member: a character “Genre” indicating the genre of the movie. There are 3 operator overloads, but they are going to be further overloaded in the child classes to apply individual sorting principles.

/* class Classic

File Name: Classic.cpp

Parent: Movies

Child: None

Implementation file for the classic movie type. Contains 6 major data members to indicate the identity of the movie: stock number, director, title, actor, release month, and release year. Contains 3 operator overloads that sort the classical movies based on release date (year then month), then major actor (details for overloads are not implemented until we reach the hash table chapter in class).

*/

/* class Drama

File Name: Drama.cpp

Parent: Movies

Child: None

Implementation file for the drama movie type. Contains 4 major data members to indicate the identity of the movie: stock number, director, title, and release year. Contains 3 operator overloads that sort the classical movies based on director, then title (details for overloads are not implemented until we reach the hash table chapter in class).

*/

/* class Comedy

File Name: Comedy.cpp

Parent: Movies

Child: None

Implementation file for the comedy movie type. Contains 4 major data members to indicate the identity of the movie: stock number, director, title, and release year. Contains 3 operator overloads that sort the classical movies based on title, the release year (details for overloads are not implemented until we reach the hash table chapter in class).

*/