1. Class Diagram for **Movie Rental Inventory Tracking System**

Graphical user interface

Description automatically generated

1. Classes
2. business.h: Business is the controller, take charge of processing input files and printing the right output

#include <fstream>

#include <iostream>

#include <string>

#include "customer.h"

#include "movie.h"

using namespace std;

class Business

{

public:

Business();

~Business();

//deal with movie file and create 3 type movie data into its corresponding linkedList

void processMovies(ifstream&); //process movie file

Movie\* movieFactory(ifstream&, string);

//deal with customer file and add customer data into hashTable[ ]

void processCustomers(ifstream&);//process customer file

void addHash(int, string, string); //add new trans into HashTable

//deal with command file and its actions

void processCommands(ifstream&); //process commands file

void displayInventory(); //print all the movies

Customer\* findHash(int); //if customer existing

void processHelper(ifstream&, string, int, string); //help processCommands to implement fuctions

// getter and setter

int gethashSize();

void setHashSize(hashTable\*, int); //change hashtable size

//Borrow helpers

bool borrowComedy(string); // movie title

bool borrowDrama(string); // movie title

bool borrowClassic(string, string); // first name, last name

//Return helpers

bool returnComedy(string); // after ID is verified

bool returnDrama(string); // after ID is verified

bool returnClassic(string, string); // first name, last name

private:

//Hash table for customers, open hashTable

int hashSize=0;

Customer\* hashTable[hashSize];

Movie\* comedyHead;//Head pointer to comedy linked list

Movie\* dramaHead;//Head pointer to drama linked list

Movie\* classicHead;//Head pointer to classics linked list

};

1. customer.h: define customer private data members and provide help for Business Class to implement its functions

#include <string>

#include <iostream>

#include "transaction.h"

using namespace std;

class Customer

{

public:

Customer(); // id = 0, firstName = "", lastName = ""

Customer(int, string, string); //id, firstName, lastName

~Customer(); //clears out transaction history

Transaction\* getTransHead(); //Return head of transaction history list

void setNextCustomer(Customer\*); //Set pointer pointing next customer in list

Customer\* getNextCustomer(); //get pointer pointing next Customer

void printHistory(); //prints transaction history

void addTrans(Transaction\*); //add transtion to history

//Getter functions

int getID(); //getID

string getFirst(); //get cutomer first name

string getLast(); //get cutomer last name

private:

int id; //customer ID

string firstName; //customer first name

string lastName; //customer last name

Transaction\* TransHead; //Pointer for head of trans history LinkedList

Customer\* nextCustomer; //Pointer for next customer

};

1. movie.h : define movie private data members and provide sortInsert() to insert new item into different types’ LinkedList.

#include <string>

#include <iostream>

using namespace std;

class Movie

{

public:

Movie();

virtual ~Movie();

virtual void printMovie() = 0; // pure virtual function

virtual void sortInsert() = 0; // pure virtual function

//set and get for private data members

char getFormat();

void setStock(int stockCount);

int getStock();

string getDirector();

string getMovieTitle();

int getReleaseYear();

void setNextMovie(Movie\* nextMovie);

Movie\* getNextMovie();

private:

char format; // format of the Movie

int stock; // copies of the movies

string director; // movie's director

string title; // movie's title

int releaseYear; // movie releasing year

Movie\* nextMovie; // Pointer for next movie stored by linked list

};

class Comedy : public Movie

{

public:

Comedy(char, int, string, string, int); //format, stock, director name, movie title, release year

~Comedy(); //empty Comedy movie linked List

void printMovie();// print current Comedy movie

virtual void sortInsert(Comedy\*, string&); //insert Comedy item into Comedy LinkedList

};

class Drama : public Movie

{

public:

Drama(char, int, string, string, int); //format, stock, director name, movie title, release year

~Drama(); //empty Drama movie linked List

void printMovie(); // print current Drama movie

virtual void sortInsert(Drama\*, string&); // insert Drama item into Drama LinkedList

};

class Classic : public Movie

{

public:

Classic(char, int, string, string, string, string, int, int); //format, stock, director name, movie title, actor first name, actor last name, release month, release year

~Classic(); //empty Classic movie linked List

void printMovie(); // print current Classic movie

virtual void sortInsert(Classic\*, string&); //insert Classic item into Classic LinkedList

//getters for Classic data members

int getReleaseMonth();

string getActorFirstName();

string getActorLastName();

private:

int releaseMonth;

string actorFirstName; //major actor's firstName

string actorLastName; //major actor's lastName

};

1. Transaction .h: A function Class, just to help Business to do the movie search when implement “B/R” action based on different condition formats, besides help to print history when implement “H” .

#include <string>

#include <iostream>

using namespace std;

class Transaction

{

public:

Transaction();

virtual ~Transaction();

virtual void printSelf() = 0;

char getTransType();

char getTransFormat();

void setNextTrans(Transaction\*);

Transaction\* getNextTrans(); //returns next transaction

protected:

char transType; //trans (B, R)

char transFormat; // just DVD

Transaction\* NextTransaction; //pointer pointing next trans history

};

class ComedyTrans : public Transaction

{

public:

ComedyTrans(char, char, string, int); //trans code (B,R), format(D), movie title, release year

~ComedyTrans();

virtual void printSelf(); //print current trans

string getMovieTitle();

int getReleaseYear();

private:

string movieTitle;

int releaseYear;

};

class DramaTrans : public Transaction

{

public:

DramaTrans(string, string, string, string);//Trans (B,R), format(D), director name, movie title

~DramaTrans();

virtual void printSelf(); //print current trans

string getDirectorName();

string getMovieTitle();

private:

string directorName;

string movieTitle;

};

class ClassicTrans : public Transaction

{

public:

ClassicTrans(string, string, int, int, string, string); //action code(B,R), format, release month, release year, actor firstName, actor lastName

~ClassicTrans();

virtual void printSelf();//print current trans

int getReleaseMonth();

int getReleaseYear();

string getActorFirst();

string getActorLast();

private:

int releaseMonth;

int releaseYear;

string actorFirst;

string actorLast;

};