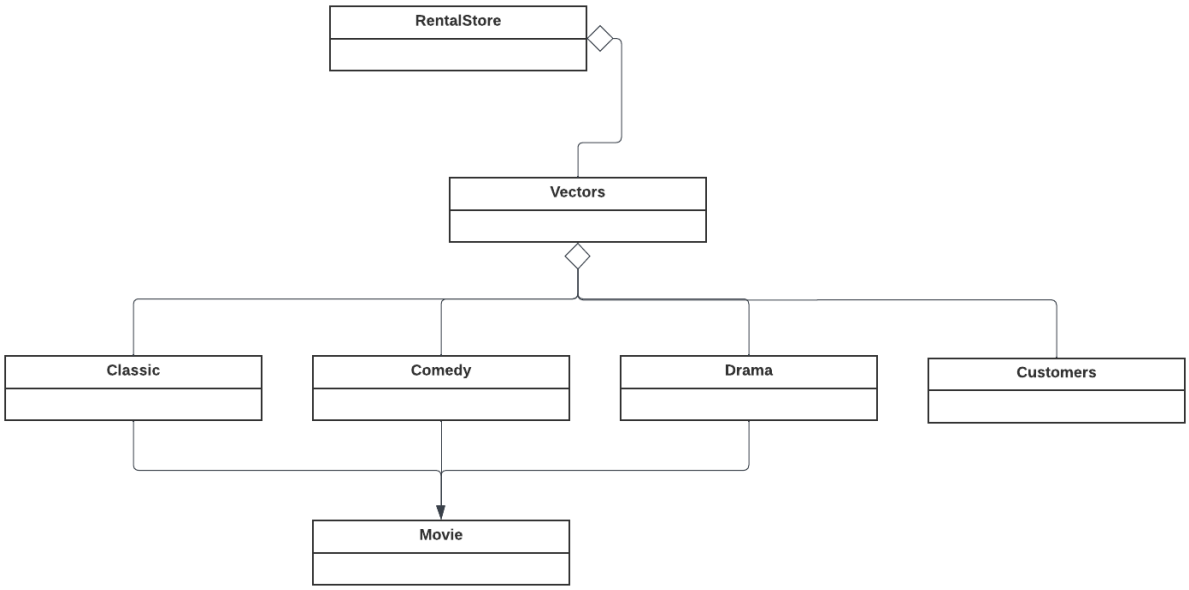
Movies Project Design

By: Nicolas Gioanni, Trevor Nguyen, Gerel Pasag, and Sandesh Rai

UML Diagram



Class Interactions

RetailStore will interact with all of the classes (classic, movie, drama, and customers). Retail will handle inserting the info of each movie and customer into itself for storage.

RetailStore -> RetailStore read file functions -> RetailStore create objects -> Store objects in vectors

In terms of manipulating data, RetailStore will call methods on objects that exist. This includes: borrow, return, and getInv. The flow will go like this:

RetailStore -> RetailStore manipulation function -> Vector chosen genre -> chosen genre manipulation function -> chosen genre variables

Main

All main will be doing is initializing a RetailStore variable, and using the methods to read the files. Read file methods will handle creating objects, and running the commands. A bool value will return to show if the code works or not.

.h files

```h
#ifndef CLASSIC.H
#define CLASSIC.H

#include "movie.h"

using namespace std;

class Classic : public Movie {
    public:
        // Constructor
        Classic(int kStock, string kDirector, string kTitle, string kFirst, string kLast, int kMonth, int kYear);

        // Comparator. First by release date, then major actor.
        bool compare(Classic classic);

        // Prints out classic movie info. Includes variables from movie.
        friend ostream& operator<<(ostream& os, const Classic classic);

    private:

        // Name of major actor
        string firstName;
        string lastName;

        // Release date of movie
        int month;
        int year;
};

#endif
```

```
#ifndef COMEDY.H
#define COMEDY.H

#include "movie.h"

using namespace std;

class Comedy : public Movie{
    public:
        // Constructor
        Comedy(int kStock, string kDirector, string kTitle, int kYear);

        // Comparator. First by title, then by year
        bool compare(Comedy comedy);

        // Prints out comedy info. Includes variables from comedy
        friend ostream& operator<<(ostream& os, const Comedy comedy);

    private:

        // Release year of the movie
        int year;
};

#endif
```

```
#ifndef CUSTOMER.H
#define CUSTOMER.H

#include <string>
#include <vector>
#include "movie.h"

using namespace std;

class Customer {
    public:

        // Constructor
        Customer(int kId, string kFirst, string kLast);

        bool borrowMovie(Movie movie);

        bool returnMovie(Movie movie);

        bool hasMovie(string title);

        vector<string> getTransactions();

        void printHistory();

    private:

        // Customer ID
        int id;
```

```
        // Name of customer
        string firstName;
        string lastName;

        // Movies held by the customer
        vector<Movie> heldMovies;

        //All transactions performed by the customer
        vector<string> transactions;
};

#endif
```

```
#ifndef DRAMA.H
#define DRAMA.H

#include "movie.h"

using namespace std;

class Drama : public Movie{
    public:

        // Constructor
        Drama(int kStock, string kDirector, string kTitle, int kYear);

        // Comparator. First by director, then title
        bool compare(Drama drama);

        // Prints out drama info. Includes variables from movie
        friend ostream& operator<<(ostream& os, const Drama drama);

    private:

        // Release year of the movie
        int year;
};

#endif
```

```
#include <string>
#include <ostream>

using namespace std;

class Movie {
    public:

        // Default constructor
        Movie();

        // Constructor
        Movie(int kStock, string kDirector, string kTitle);

        // Takes a movie from the stock
        bool borrowMovie();

        // Returns a movie to the stock
        bool returnMovie();

        // Returns the stock
        int getInv();

        // Interface for comparing between movies. Child class defines this.
        //virtual int compare(Movie movie);

        // How many movies are in inventory
        int stock;

        // The max amount stock can be. Impossible for stock to go above.
        int maxStock;

        // Director name
        string director;

        // Title of movie
        string title;
};

#endif
```

```
class RentalStore {
    public:
        // Reads file that contains all the movies and adds them to their respective genre vector
        bool readMovieList(string fileName);

        // Reads file that contains all customers and adds them to the customer vector
        bool readCustomerList(string fileName);

        // Reads commands and performs their action depending on what they do
        bool readCommands(string fileName);

    private:

        // Stores all of the different genres of movies
        vector<Comedy> comedyList;
        vector<Drama> dramaList;
        vector<Classic> classicList;
        vector<Movie> shelf;

        // List of customers
        vector<Customer> customers;
};

#endif
```