**CPSC 6119**

**Objected-Oriented Development with Components**

**Small Project (MVC)**

**Due: 11:59 PM Friday April 21**

**This small project needs to be completed individually**. The objective of this project is to develop a movie rental management system using the **MVC compound pattern**. The project can be implemented in either Java, or C++ or C#. The system is made of the following entities:

1. Clients: They rent movies in the system. Clients are stored in a file called clients.txt.
2. Movies: These are the objects to be rented. Movies are stored in a file called movies.txt.
3. Rental\_Info: These are the information about who rented which movie, the date of rental, and date of return. Rental\_Info are stored in a file called rental\_info.txt.

**File Structure**

**clients.txt**

client\_id; name; deleted

if deleted=0 then client is active in the system, if deleted=1 then the client is logically deleted.

Example:

3; Thomas Jefferson; 0

**movies.txt**

movie\_id; movie\_title; rented

if rented=0 then is not rented currently, if rented=1 is rented.

Example:

1; Altitude; 0

2; Alien vs Ninja; 0

**rental\_info.txt**

rental\_id; client\_id; movie\_id; date\_out; date\_in

date\_out and date\_in should be stored in format mm/dd/yyyy

if the movie has not been returned, then date\_in is empty.

A record should be placed in this file every time a movie is rented, and a record is updated every time a movie is returned.

Example:

1; 3; 1; 4/18/2016;

2; 3; 2; 4/18/2015; 4/19/2015

In the first case shown above, the client called Thomas Jefferson rented the movie Altitude on 4/18/2016

In the second case shown above, the client called Thomas Jefferson rented the movie Alien Vs. Ninja on 4/18/2015 and returned it on 4/19/2015

**Requirements**

The system should be able to handle the following transactions **with GUIs**:

**For Clients:**

1. Create a client.
2. Delete a client. A client does not get physically deleted from the system. A flag in the record in file just states if the client is “logically” deleted.
3. Search a client by last name, first name and show its information ( **5% Extra credit** )
4. Show all clients

**For Movies**

1. Create a movie.
2. Rent a movie.
3. Return a movie.
4. Show all movies, showing current rented movies first.
5. Show current rented movies.
6. Show who has historically rented a particular movie, sorting in descending order by date of rental.

**Restrictions**

1. **The interface for the system must be graphically**, however your design must follow the dynamics of the Model Viewer Controller design.
2. Every time the app starts, it should allow the system to find/choose the location of the files.
3. A client can rent at most 3 movies at a time. If a client tries to rent a 4th movie, the system must tell the user that the client is unable to rent the movie.
4. The current rental business only keeps one copy of a movie. If the user tries to rent a movie that is currently rented, the system must tell that the movie is not available.

**What to submit?**

1. If you use Java, submit your NetBeans project in a ZIP file. This project should be developed using Java SE only (not intended to be a web-based project).
2. A PDF document with the information about your classes as well as the class diagram describing your components of the system. Your written document must explicitly describe which components are the Model, which are the Controller(s), and which are Viewers.
3. If you used any other patterns, please write down where you used them.