CSE 1325-004 HW#4
Video Rental Store

Due September 28th, at 11:59PM

Remember the old video rental stores of the past? Well, time to make a user interface for one.

Requirements

A Video Rental store has a lot of different items for customers to rent out, but for this assignment, we'll focus on just types, movies and TV shows. These can come on different formats (DVD, Blu-Ray, and Streaming Rental, like Amazon). Each media, as what we'll call a generic term of any type of rental, falls into a particular genre (Action, Comedy, Drama, Fantasy, Documentary, Horror, Science Fiction, and Animated). Each media has an age rating. To keep things simple, let's use the MPAA ratings (the movie ratings) of (G, PG, PG-13, R, NC-17) for both Movies and TV. Each media will also have an ID number assigned by the store.

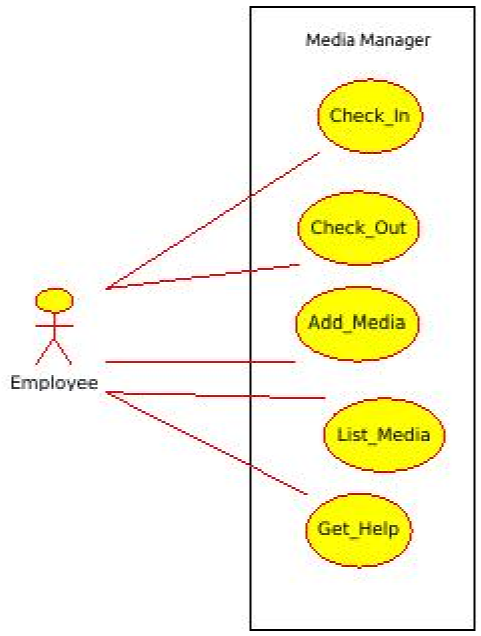
Design an object-oriented application that manages these media as objects.

We'll want to know the title, lead actor/actress, director, copyright year, genre, media, and age rating for each media. We'll also want to know if the publication is checked in or checked out by a customer, and if checked out, who is the customer (name and telephone number). The ID number is assigned manually when it is added to the system. It's ok at the full credit level to enter the customer information each time a media is checked out, but bonus levels will have to have a list of patron objects.

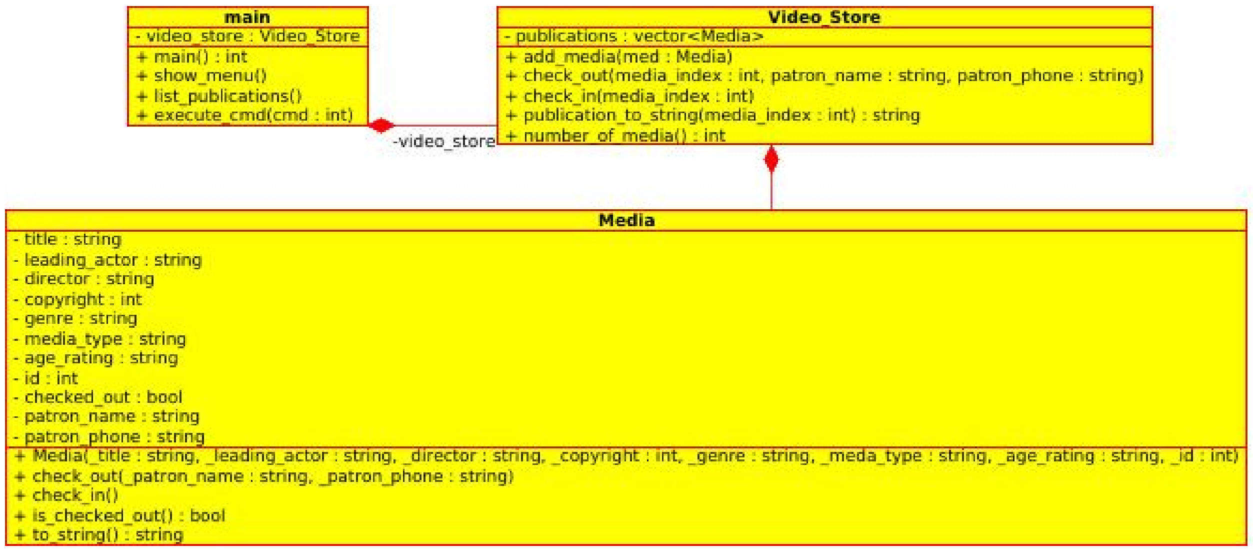
Each media object should be able to print its contents and its check out status in a format like this:

"Spaceballs the Movie" with Mel Brooks. Directed by Mel Brooks. 1987 (Comedy). PG. DVD. ID: 1001.
Checked out to Shawn Gieser (817-272-6738)

For the first sprint of this multi-week project, we'll target a simple console application with 5 operations shown in the diagram below: (1) Adding a new media to the system, (2) List all media in the system. (3) Check out a publication to a patron, recording their name and phone number, (4) Check in a publication that was previously checked out, and (5) some short basic documentation on how to use the system, like a help file.



One possible design is shown in the class diagram and suggested menu on the following page. This is a basic design that meets the requirements above, although, since you are more experienced at this point, you may implement a design of your own for this project. Implement the classes as specified, writing appropriate tests for each as you go, until your system works well.


main
- video_store : Video_Store
+ main() : int
+ show_menu()
+ list_publications()
+ execute_cmd(cmd : int)
-video_store
Video_Store
- publications : vector<Media>
+ add_media(med : Media)
+ check_out(media_index : int, patron_name : string, patron_phone : string)
+ check_in(media_index : int)
+ publication_to_string(media_index : int) : string
+ number_of_media() : int
Media
- title : string
- leading_actor : string
- director : string
- copyright : int
- genre : string
- media_type : string
- age_rating : string
- id : int
- checked_out : bool
- patron_name : string
- patron_phone : string
+ Media(_title : string, _leading_actor : string, _director : string, _copyright : int, _genre : string, _meda_type : string, _age_rating : string, _id : int)
+ check_out(_patron_name : string, _patron_phone : string)
+ check_in()
+ is_checked_out() : bool
+ to_string() : string

Below is a recommended menu for the Video Rental store to use with your software:

```
=======================================
CSE1325 Video Rental Management System
=======================================

Media
-----
(1) Add Media
(2) List All Media
(3) Check Out Media
(4) Check In Media

Utility
-------
(9) Help
(0) Exit
```

You will receive partial credit if you only implement a portion of the requirements. You will be graded on the extent to which you cover the requirements and the quality of your code. Be sure to use header files (.h) for including, and implementing files (.cpp) for implementing your algorithms. Use Scrum to manage this first sprint. If you don't understand the intent of a requirement, feel free to ask – although reasonable assumptions without asking are also fine if you have been to a video rental store.

Grading

You will deliver your .h and .cpp class implementations and a Makefile that is competent to rebuild only files that have been modified, screenshot(s) demonstrating each of the 5 operations above as images in PNG format, and your updated Scrum spreadsheet for the first sprint.

Bonus #1 (10 pts) – If you modify the design to add a Patron class, and modify the Video_Store class to maintain a vector of Patrons. Then modify Media to replace the patron_name and patron_phone fields with a reference to a Patron object. The menu system and some interfaces also need to be modified so that the Video_Store can create new patrons (menu item 5) and list all patrons (menu item 6). The dialogs to allow the Video_Store to select an existing Patron from the list of Video_Store patrons during checkout.

You will deliver a PNG image of your updated use case and class diagrams, your .h and .cpp class implementations in C++ and a Makefile capable of rebuilding only what has changed, screenshot(s) demonstrating each of the 6 operations (the new, 6th operation is "Create Patron", though that may be part of check_out in your design) in PNG format, and your updated Scrum spreadsheet for the first sprint showing the additional tasks

Bonus #2 (10 pts) – Rework the Media class to where it could be any type of media. Then, using inheritance, make new classes called Movie and TV_Show that are children of Media. These should have different variables, such as a movie could have a release date, while TV shows have season numbers (and even disc numbers within each season). There are many other different ways you can differentiate movies from TV shows, such as different rating systems. I will let you decide how to do that.

Define an output format for each subclass that adds its unique field, and update your user interface, methods, and Makefile to support them.

Deliver your full updated source code for the files you wrote, images of your UML-based design, images demonstrating the new-and-improved operations, and your updated Scrum spreadsheet for the first sprint showing the additional tasks.