COMP 2404 -- Assignment #1

Due: Friday, January 31, 2014 at 11:00 AM

Goal

You will design and implement an object-oriented program in C++ to manage a movie database. You can start from an existing program originally designed for a procedural language (C). The procedural program can be found here: a1Posted.tar

Learning Objectives

- design an object-oriented program with entity, user interface (UI) and control objects
- write a small program in C++ that is modular, correctly designed and documented

Instructions:

1. Design:

- Once you have read and thoroughly understood the existing (procedural) movie database program, identify all the classes and objects that will be needed to implement this program in an object-oriented language. Your design must include, at minimum:
  - one control object that encapsulates all the control flow for the program
  - one UI (view) object that is responsible for all user I/O
  - entity objects that represent real-life objects or concepts
  - utility objects (for example, collections)

  Notes:
  - Remember! `main` is a function, not a class!
  - all control flow must be contained within your control object; your `main` function must have **no code** other than creating a control object and invoking its launch operation
- Using a drawing package of your choice, draw a UML class diagram to represent the objects you identified. You must show all classes, their attributes and operations, the associations between the classes (inheritance or composition), and the multiplicity and directionality of all associations.

2. Implementation

You will implement, in C++ using the Ubuntu Linux environment provided in the course virtual machine (VM), the movie database program. Your program must match your design, and it must abide by the constraints set out below. You can find a C++ coding example with useful features here.

Constraints

- your program must be written in C++, and **not** in C
- do not use any functions from the C standard library (e.g. printf, scanf, fgets, sscanf, malloc, free, string functions)
- do not use any classes or containers from the C++ standard template library (STL)
- do not use low level data types when there is a better one available (e.g. do not use character arrays, use string objects instead)
- do **not** use any global variables or any global functions other than `main`
- do **not** use structs, use classes instead

Submission

You will submit in *cuLearn*, before the due date and time, the following:

- a UML class diagram (as a PDF file) that corresponds to your program design
- one **`tar`** file that includes:
  - all source and header files for your movie database program
  - a Makefile
  - a readme file that includes:
    - a preamble (program author, purpose, list of source/header/data files)
    - compilation instructions
    - launching and operating instructions

Grading

- **Marking breakdown:**

Component	Marks
UML class diagram	20
Implementation of control class	25
Implementation of UI (view) class	20
Implementation of utility class	20
Implementation of entity class	15

- **Deductions:**
  - Up to 50 marks for any of the following:
    - the code does not compile using g++ in the VM provided for the course
    - the code cannot be tested because the program consistently crashes
    - the design does not generally follow correct design principles (e.g. data abstraction)
    - prohibited library classes or functions are used
  - Up to 20 marks for any of the following:
    - the Makefile or readme file is missing
    - global variables, global functions, or structs are used
    - the code is not correctly separated into header and source files
  - Up to 10 marks for missing comments or other bad style (non-standard indentation, etc.)
- **Bonus marks:**
  - Up to 5 extra marks are available for fun and creative additional features