

# COMP 2404 -- Assignment #1

Due: Thursday, October 2, 2014 at 9:00 PM

## Goal

You will design and implement an object-oriented program in C++ to manage a library that contains books and the patrons (i.e. the "borrowers") who borrow those books. The Pandora Library System (PLS) program allows the user to access the patron menu, the administrator menu, or to view the collection of books in the library. The patron menu allows an existing patron to check out a book, check in a book, or view the list of books that they have currently borrowed. The administrator menu allows the user to add or delete a patron. You can start from an existing program originally designed for a procedural language (C). The procedural program can be found here: [a1Posted.tar](#)

You can also view a demo of the program [here](#).

## 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:

- Read and make sure that you thoroughly understand the existing (procedural) library management 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

#### Notes:

- For this assignment, you can use C-style arrays for all the collections in the program (books, patrons, etc.).
- 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.

#### Note:

- You do not need to show the getter and setter member functions in the UML.

### 2. Implementation

You will implement, in C++ using the Ubuntu Linux environment provided in the course virtual machine (VM), the PLS library management program. Your program must match your design, and it must abide by the constraints set out below. You can find a C++ coding example that uses "safe" I/O functions [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 any global variables or any global functions other than **main**
- do **not** use structs, use classes instead
- objects should always be passed by reference, not by value

## 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 library management program
  - a Makefile
  - a readme file that includes:
    - a preamble (program author, purpose, list of source/header/data files)
    - compilation, launching and operating instructions

## Grading

- **Marking breakdown:** The grading scheme is posted in [cuLearn](#).
- **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 it doesn't run
    - unauthorized changes have been made to the code provided for you, where applicable
    - prohibited library classes or functions are used
  - Up to 20 marks for any of the following:
    - the design does not generally follow correct design principles (e.g. data abstraction, encapsulation)
    - your program uses global variables, global functions, or structs
    - your program passes objects by value
    - the Makefile or readme file is missing or incomplete
    - 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