Getting Organized – Library Model

CSE 1325 – Fall 2016 – Homework #4
Due Thursday, September 22 at 8:00 am

Organizing information is usually left to databases, but object-oriented design principles fit the problem space quite well. In this homework, you'll analyze requirements, then design and implement a portion of a simple library management system.

Requirements

The library consists of a lot of publications in several types of media – books, periodicals (also called magazines), newspapers, audio, and video. Each publication falls into a particular genre – fiction, non-fiction, self-help, or performance – and target age – children, teen, adult, or restricted (which means adult only). Each publication also includes a unique ISBN identifier, which is just text.

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

We'll want to know the title, author, copyright year, genre, media, target age, and ISBN for each of our publications. We'll also want to know if the publication is checked in or checked out to a customer, and if checked out to a customer, who that customer is (their name and telephone number). It's OK to enter the customer information each time a publication is checked out.

Each publication object should be able to print its contents and its check out status something like this:

> "The Firm" by John Grisham, 1991 (adult fiction book) ISBN: 0440245923
> Checked out to Professor Rice (817-272-3785)

We'll need a simple console application with 5 operations: (1) Create a new publication, (2) List all publications created 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. (Persistence is NOT required. Each time your program is run, it may start without publications.)

You should design this system in UML first, creating (at least) a basic Use Case diagram, an Activity diagram for at least action (3) above, and a class diagram. A good top level diagram to help understand your design will help your grade if provided, but is NOT required. Then implement the classes, writing appropriate tests for each as you go, until your system works well.

(This homework is much less prescriptive. **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, the quality of your object-oriented design, and the quality of your code – all are important. Be sure to use header files (.h) for including, and implementation files (.cpp) for implementing your algorithms. If you don't understand a *intent* of a requirement, feel free to ask – although reasonable assumptions without asking are also fine (you've used a library?) – but we won't do your design for you!)

Deliverables

You will deliver each of your UML diagrams as well as screenshot(s) demonstrating each of the 5 operations above as images in PNG, JPG, or GIF format, along with all of your source code in .cpp and .h formats. A correct Makefile is highly recommended to help the grader build your code.

Grading

- **Full Credit** – If you deliver the full source code *for the files you wrote*, images of your UML-based design, and images demonstrating the 5 operations.

- **Bonus** – If you add one or more of the following C++ operators to the class(es) that manage publications:
  - An “==” operator that returns true if the ISBN match;
  - An “!=” operator that returns true if the ISBNs don't match;
  - An “<<” operator that adds the text defined above that represents the publication to the stream.

  Replace calls to the methods used to print publication information in the Full Credit version with the “<<” operator where appropriate.

  Deliver your full updated source code *for the files you wrote*, test code verifying that these work, and images demonstrating tests.

- **Extreme Bonus** – If you modify your design to use inheritance, such that each different type of media has a unique field.
  - Books – Identify format as hardback or paperback.
  - Magazines – Identify publication frequency as weekly, monthly, or quarterly.
  - Newspapers – Identify the city of publication.
  - Audio – Identify the format as CD, cassette tape, or MP3.
  - Video – Identify the format as Blu-ray, DVD, VHS, or MP4.

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

  Deliver your full updated source code *for the files you wrote*, images of your UML-based design, and images demonstrating the new-and-improved 5 operations.