Class Design Part 1

The next two projects will focus on developing the skills to design functions and classes in C++. The second project will build on the result of the first project.

In this project you will design three different sets of classes according to the constraints described below. For this project focus on *encapsulation* and maintaining *class invariance*.

An invariant of a class is a single valid state for that class to be in. For example, if you have a class representing a fraction, it would be invalid (mathematically) for the numerator or denominator to not be initialized. It would be ok to have a default value for the numerator and denominator because then an instance of the class still represents a valid fraction. You would want to design the fraction class so that the numerator and denominator are initialized at the time of construction, and never un-initialized.

For this project, the constraints below will describe the invariants you should maintain with your class design.

Encapsulation is related to invariance. The private data of your class should remain in a valid state, and therefore you should control access to the private member data such that it is not possible for a user of your class to put it into an invalid state.

For this project, the constraints and invariants described below should be maintained using encapsulation.

Project Requirements

You should submit a set of header (.h) and source (.cpp) files for each of the below sections. It is ok for everything to be submitted at once (one GitHub repo), just make sure each of the following sections are covered.

Section A

Design a class for representing a web URL. To keep this simple, consider a URL only to have a *scheme*, *authority*, and *path* (you can use the same code you wrote in a previous project). Your class should have a constructor that takes a `std::string` as a parameter. You should not be able to edit (modify) the URL once it is created. You should have four *getter* functions which return the full URL as a string, and each of the three components individually also as strings.

Section B

Design a class for managing the metadata for an image taken by a camera. The metadata should include at least the following properties:

- File Name (string)
- Image Type (must be one of the following only "PNG", "GIF", or "JPEG")
- Date Created (you can choose how you want to represent a date)
- Size (in MB, stored as a double)
- Author Name (string)

- Image Dimensions (will have width and height)
- Aperture Size (represented as f/# where # is an integer like f/8 or f/22)
- Exposure time (shutter speed, is a fraction of a second like 1/30, or 1/1000)
- ISO value (another integer like 600, or 3000)
- Flash enabled (boolean value)

The image metadata must contain all of these values, and there should never be an uninitialized value. Be sure to provide getters and setters so the values can be read and modified. Prevent invalid values - particularly with the image type, you should not be able to change the image type to an invalid type.

Provide an external function (not a member function) that takes an image metadata object, and prints out all of the above information. The format of how this is printed is up to you, just make sure all the data is present.

Section C

Design a set of classes and functions to manage an online store.

The first class should represent an *item* that can be bought in the store. An item should have a user-friendly name (i.e. "Book") represented as a `std::string` as well as an *id* which should be represented as a `long` value. Every item should have a *price* represented as a `double` as well as an `int` indicating how many are in stock.

The second class should represent a *store*, which contains a list of all the items currently available in the store. Provide a function to return the total number of items in stock for the entire store, as well as a function that prints out the full contents of the store with each item and the number of that item in stock:

example

```
Store:
Book x 12
Colored Pencils x 15
Coloring Paper x 20
Markers x 50
Crayons x 3
Staples x 7
```

NOTE: Keep in mind that you are only writing the classes and functions. You can provide example code that actually creates an instance of this class and calls the function to print the result as in the above example, but I will not be checking for this. I am only going to look at the class and function definitions so ensure that they would behave according to the above example, given the right setup and input.

The third class should represent an *order*. An order should consist of a list of items. Provide a member function that allows you to add items to the order, as well as a getter function that returns the total price of all the items in the current order.