Program 4

Worth: 100 points

Due: Tuesday, December 3 (by 11:59 PM)

Purpose: This assignment explores the creation of a reusable class and separate Console application that creates a list objects.

Create a Console application with two classes in separate files. One class will be the test application where the *Main* method is located. The other **public** class will be named **LibraryBook** and will be our initial attempt at representing the books held by a library. At the moment, our book class will be very simple. Each **LibaryBook** object will keep track of some basic information: the book's title (a **String**), author (a **String**), publisher (a **String**), copyright year (an **int**), call number (a **String**), and checked out status (a **bool**). Later, we may add more capabilities (such as who checked out the book and when it is due back at the library) but this is a good start.

The specific **public** requirements for the **LibraryBook** class are listed below. You may not change or add to the **public** interface described here.

- A 5-parameter constructor that accepts the book's title (a **String**), author (a **String**), publisher (a **String**), copyright year (an **int**), and call number (a **String**), in this order. When a book is created, it will not be checked out yet. Use the set properties for all relevant fields (except the checked out status which does not have a property) to establish their initial values (instead of directly changing instance variables). No other constructor exists for this class (there is NO parameterless constructor for this class).
- A **String** property named *Title* with a get and set. No validation need be done on the title at this time. You may use an auto-implemented property.
- A **String** property named *Author* with a get and set. No validation need be done on the author's name at this time. You may use an auto-implemented property.
- A **String** property named *Publisher* with a get and set. No validation need be done on the publisher at this time. You may use an auto-implemented property.
- An **int** property named *CopyrightYear* with a get and set. To practice validation, you **must provide validation** in the set accessor. It must ensure that the copyright year value is non-negative when attempting to set; otherwise, set the copyright year to a default value of 2019.
- A **String** property named *CallNumber* with a get and set. The library uses the call number to categorize the book (like the Dewey decimal system). No validation need be done on the call number at this time. You may use an auto-implemented property.
- A method named *CheckOut()* that returns no data (**void**) and accepts no parameters. This method will change the book's checked out status to reflect that the book has been checked out by a patron by setting the checked out status instance variable to **true**.
- A method named *ReturnToShelf()* that returns no data (**void**) and accepts no parameters. This method will change the book's checked out status to reflect that the book has been returned by a patron and is no longer checked out by setting the checked out status instance variable to **false**.
- A method named *IsCheckedOut()* that returns a **bool** and accepts no parameters. This method will return a **true** value when the book is currently checked out and a **false** value when the book is not out. In other words, return the value of the **bool** checked out status instance variable.
- A method named *ToString()* that returns a **String** and accepts no parameters. Remember, you must also use keyword **override** when defining a *ToString* method. This method will create a formatted string that has the book's title, author, publisher, copyright year, call number, and checked out status each on a separate line. Precede each item with an identifying label. You may use string interpolation to create the formatted text that the method will return. Instead of concatenating the string literal "\n" to add a newline to the string, use the string constant **Environment.NewLine** instead. Note well, the *ToString* method just builds and returns a string. It does no output of its own. That is up to client classes to perform, as the output may be directed to the console or a GUI or a web page. See the *PayrollSystem* or *Time* examples from class for an example of how *ToString()*

should be written.

In addition to the LibraryBook class, you will need to write a simple Console application to test your books. In your other class file (where the **Main** method is located), your simple test program must do the following:

- Create at least 5 **LibraryBook** objects (you may hard code your test data) and store them in an array.
- Using a method, print out all the books' original data to the Console. Your method should be **static**, **void** and will accept the array of **book** objects as its parameter.
- Next, for each book, use the appropriate properties and methods to change either the book's publisher or call number or check out the book - you must check out at least 2 books. These changes can be hard coded (not in a loop).
- Again, using the method you wrote previously, print out all the books' new data to the console by passing the array of **book** objects again to the method.
- Call the appropriate **LibraryBook** method to return each book that was checked out.
- Finally, print all books' data to the console (using the method you wrote previously) one last time.

Since you are printing each book's data to the console several times, you must write a method to accomplish this task in your test application program. As noted earlier, the method should be **static**, **void** and will accept the array of **book** objects as its parameter.

Be sure to add appropriate comments in your code, including your **Grading ID** (not name nor student ID), program number, due date, and course section. Each variable used in your program needs a comment describing its purpose. **Preconditions and postconditions are now required**, as well. Remember, each method, constructor, property get accessor and property set accessor needs their own precondition/postcondition comments. Review the Time example from class for how the comments should appear in business logic class.

As with our labs, I'm asking you to upload a compressed ZIP archive of the entire project. Rather than giving me floppy disks or printouts, you will upload **all your files** to Blackboard using the *Assignments* tool. I'm asking you to upload a compressed ZIP archive of the entire project, just as with our labs. The steps for doing this will vary somewhat based on the ZIP utility being used. Before you upload this .ZIP file, it's a good idea to make sure that everything was properly zipped. Make sure your code is present and you can run your file. Once you have verified everything, return to the *Assignments, Programs* area of Blackboard. Click on "Program 4" and the *Upload Assignment* page will appear. Add any comments you like in *Comments* field. Click *Browse* next to *File to Attach* to browse the system for your file. Browse to the location of your .ZIP file and select it. Note, multiple files may be attached using the *Add Another File* option. For this assignment, we just need the "Prog4.zip" file. Make sure everything is correct in the form and then click *Submit* to complete the assignment and upload your file to be graded.

Remember, this is an **individual** assignment. Please be mindful of the syllabus' statement on academic dishonesty. If you are unsure about what constitutes academic dishonesty, **ASK!**