Program 1B

Due: Thursday, February 20 (by 11:59 PM) - **EXTENSION**

Worth: 50 pts.

Purpose: This program explores the creation of a simple class hierarchy including (limited) use of polymorphism and LINQ to produce simple reports.

In this assignment, you will use the Library class hierarchy developed for Program 1A (either your solution or your instructor's) in a simple test program. Instead of using arrays of library items and patrons, your program will use the **List** collection described in Chapter 9. You must create a **List** of **LibraryItem** objects filled with a least two instances of every concrete class from the hierarchy. You will need to add at least two **LibraryMagazine** objects with the same title (but different volume/number). You must also create a **List** of **LibraryPatron** objects with a least 5 patrons.

The detailed **public** requirements for the classes in this assignment appear below.

- Display the list of items to the console immediately after construction (none should be checked out yet) and then pause and wait for the user to enter a key.
- Check out at least 5 items and display the list of items again. Pause and wait for the user to enter a key.
- Using LINQ, select all the items from the list that are checked out and display the resulting items and the count of checked out items (hint: use the Count extension method on the LINQ result variable). Pause and wait for the user to enter a key.
- Using LINQ, filter the previous result set to select only checked out **LibraryMediaItem**s. Display the results to the console and then pause and wait for the user to enter a key.
- Using LINQ, select and then display the *unique* titles of the **LibraryMagazine** objects from the list to the console. Pause and wait for the user to enter a key.
- For each item in the list, calculate what the late fee would be if it were returned 14 days late. Display the item's title, call number, and the late fee to the console. Pause and wait for the user to enter a key. This is just a loop task, no LINQ is required.
- Return all the checked out items. Show that the count of checked out items is now zero using the earlier LINQ result variable. This will demonstrate that LINQ uses *deferred execution*. Pause and wait for the user to enter a key.
- For each **LibraryBook** in the list, display its current loan period and then modify the book's loan period to add 7 more days and display the new loan period for each book. Pause and wait for the user to enter a key. This is just a loop task, no LINQ is required.
- Finally, display the entire list of items to the console once more.

This assignment will only focus on the test program and not the hierarchy classes. If your hierarchy classes don't work, you will need to use your instructor's solution to Program 1A as a starting point.

Be sure to add appropriate comments in your code for each file, including your **Grading ID** (not name nor student ID), program number, due date, course section, and description of each file's class. Each variable used in your program needs a comment describing its purpose. These requirements are expected for every program and are listed in the syllabus. Preconditions and postconditions are required, as well. So, for each constructor, method, get property, and set property a pair of comments describing the precondition and postcondition must be provided. Please review the PowerPoint presentation (under Course Documents) for further details about preconditions and postconditions.

As with our labs, I'm asking you to upload a compressed ZIP archive of the entire project. 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* area of Blackboard. Click on "Program 1B" 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 "Prog1B.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!**