Program 1A

Due: Thursday, February 13 (by 11:59 PM) - **EXTENSION**

Worth: 75 pts.

Purpose: This program explores the creation of a simple class hierarchy including (limited) use of polymorphism.

In this assignment you will use what you have learned through Chapter 12 to design and implement a simple class hierarchy for a library that is based on the ideas explored in your instructor's solution to Program 0. The hierarchy will consist of several abstract and concrete classes. At the top of the hierarchy is the abstract base class **LibraryItem**. It will contain all the common data that items held by a library would have, including Title, CallNumber, Publisher, LoanPeriod, CopyrightYear, etc. We will assume (for the sake of simplicity) that all library items may be checked out by a patron. This will be modeled using a conditional *HAS-A* relationship with class **LibraryPatron** (which you may incorporate from your instructors solution to Program 0), in the same way as the **LibaryBook** did there. It will support methods for checking out the item and returning it, etc. just as the **LibraryBook** did. In addition, all library items will be able to calculate the late fee for a given number of days late (using method **CalcLateFee**). This method will be abstract for class **LibraryItem** and then made concrete by later, derived classes. It will serve to address what-if questions about how much MIGHT be charged if the item is late. The Library itself will be responsible for making these charges to the patron, so the item itself just offers the amount for a given number of days late. Derived from **LibraryItem** are three classes, **LibraryBook**, **LibraryMediaItem**, and **LibraryPeriodical**. **LibraryBook** is concrete and will charge $.25/day late (no limit). It will also keep track of the book's Author (which you can keep from your instructor's Program 0). **LibraryMediaItem** is still abstract. It will have two derived classes that will be concrete, **LibraryMovie** and **LibraryMusic**. Media items will add a Duration (in minutes) and, also, keep track of the item's Medium (such CD, DVD, etc.). **LibraryMovie** adds the film's Director and Rating (G, PG, etc.). Movies are charged $1.00/day late for DVD and VHS media, and $1.50/day late for BluRay media (both with a $25.00 limit) . **LibraryMusic** will keep track of the item's Artist and the number of tracks contained (*NumTracks*). Music items are charged $.50/day late ($20.00 limit). The final branch of the hierarchy will be for periodicals. **LibraryPeriodical** will remain abstract and will keep track of the periodical's publication Volume and Number. Each issue of the periodical will have its own object. Derived from **LibraryPeriodical** are two classes, **LibraryJournal** and **LibraryMagazine**. Journals will have an Editor and a Discipline and are charged $.75/day late (no limit). Magazines are charged $.25/day late ($20.00 limit). The hierarchy is represented visually in the attached PDF file.

The detailed **public** requirements for the classes in this assignment appear below. You may not add to or change these **public** requirements but you may include **private** or **protected** helpers.

LibraryItem - public abstract class

- A constructor that accepts item's title (a required **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days - must be non-negative) and call number (a required **String**). When an item is created, it will not be checked out yet. Since the item is not checked out, it does not have a patron relationship yet. The special value **null** may be stored instead of an object reference to indicate this. Use the code from your instructor's Program 0 to implement most of these requirements.
- Properties (get and set) for each data element named as follows: *Title*, *Publisher*, *CopyrightYear*, *LoanPeriod*, *CallNumber*. *CopyrightYear* and *LoanPeriod* must be non-negative. Illegal values should throw an exception (as shown in the text and your instructor's solution to Program 0). Read-only property *Patron* will be used to retrieve the associated **LibraryPatron** object when the item is checked out (null otherwise). Many of these can be implemented much as they were in Program 0's **LibraryBook** class. No changes need to be made to many of these properties and methods. You should be able to

just copy and paste them.
- Methods *CheckOut*, *ReturnToShelf*, and *IsCheckedOut*, should be implemented as they were in your instructor's Program 0's **LibraryBook** class. No changes need to be made to these methods. You should be able to just copy and paste them.
- Abstract method *CalcLateFee*. This method will return a **decimal** and accepts a single **int** parameter, the number of days late (which must be non-negative). The concrete derived classes will provide a body for this method.
- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

LibraryBook - public class, *IS-A* **LibraryItem**

- A constructor that accepts item's title (a required **String**), author (a **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days, - must be non-negative) and call number (a required **String**). When an item is created, it will not be checked out yet. Since the item is not checked out, it does not have a patron relationship yet. The special value **null** may be stored instead of an object reference to indicate this.
- Properties (get and set) for each new data element named as follows: *Author*. This property can be implemented much as it was in Program 0's **LibraryBook** class. No changes need to be made, you should be able to just copy and paste it.
- Method *CalcLateFee*. This method will calculate the late fee as $0.25/day late with no limit. Magic numbers are not permitted.
- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

LibraryMediaItem - public abstract class, *IS-A* **LibraryItem**

- A constructor that accepts item's title (a required **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days - must be non-negative), call number (a required **String**), and duration (a **double** - must be non-negative).
- An **enum** named *MediaType* that consists of the following values: DVD, BLURAY, VHS, CD, SACD, VINYL.
- Properties (get and set) for each new data element named as follows: *Duration*. *Duration* must be non-negative. Illegal values should throw an exception (as shown in the text and your instructor's solution to Program 0).
- Abstract property named *Medium* which will get/set a *MediaType* **enum**. Abstract properties use a syntax identical to auto-implemented properties except keyword **abstract** is added to the header. This does NOT auto-implement anything, nor does it create backing fields at this time. It simply requires that derived, concrete classes have a fully-bodied version of the property. As with **abstract** methods, the derived, concrete classes must use keyword **override** when implementing their complete version of the property.
- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

LibraryMovie - public class, *IS-A* **LibraryMediaItem**

- A constructor that accepts item's title (a required **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days - must be non-negative), call number (a required **String**), duration (a **double** - must be non-negative), director (a required **String**), medium (a *MediaType* **enum**), and rating (a *MPAARatings* **enum**).
- A public **enum** named *MPAARatings* that consists of the following values: G, PG, PG13, R, NC17, U.
- Properties (get and set) for each new data element named as follows: *Director*, *Medium*, *Rating*. *Medium* must one of the following: DVD, BLURAY, VHS (a subset of the *MediaType* **enum**). Illegal values should throw an exception (as shown in the text and your instructor's solution to Program 0).

- Method *CalcLateFee*. This method will calculate the late fee as $1.00/day late for DVD and VHS media, and $1.50/day late for BluRay media (both with a $25.00 limit). Magic numbers are not permitted.
- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

LibraryMusic - public class, *IS-A* **LibraryMediaItem**

- A constructor that accepts item's title (a required **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days - must be non-negative), call number (a required **String**), duration (a **double** - must be non-negative), artist (a required **String**), medium (a *MediaType* **enum**), and number of tracks (a positive **int**).
- Properties (get and set) for each new data element named as follows: *Artist*, *Medium*, *NumTracks*. *Medium* must one of the following: CD, SACD, VINYL (a subset of the *MediaType* **enum**). *NumTracks* must be positive (>= 1). Illegal values should throw an exception (as shown in the text and your instructor's solution to Program 0).
- Method *CalcLateFee*. This method will calculate the late fee as $0.50/day late with a $20.00 limit. Magic numbers are not permitted.
- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

LibraryPeriodical - public abstract class, *IS-A* **LibraryItem**

- A constructor that accepts item's title (a required **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days - must be non-negative), call number (a required **String**), volume (a positive **int**), and number (a positive **int**).
- Properties (get and set) for each new data element named as follows: *Volume*, *Number*. *Volume* and *Number* must be positive (>= 1). Illegal values should throw an exception (as shown in the text and your instructor's solution to Program 0).
- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

LibraryJournal - public class, *IS-A* **LibraryPeriodical**

- A constructor that accepts item's title (a required **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days - must be non-negative), call number (a required **String**), volume (a positive **int**), and number (a positive **int**), discipline (a required **String**), and editor (a required **String**).
- Properties (get and set) for each new data element named as follows: *Discipline* , *Editor*. Illegal values should throw an exception (as shown in the text and your instructor's solution to Program 0).
- Method *CalcLateFee*. This method will calculate the late fee as $0.75/day late with no limit. Magic numbers are not permitted.
- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

LibraryMagazine - public class, *IS-A* **LibraryPeriodical**

- A constructor that accepts item's title (a required **String**), publisher (a **String**), copyright year (an **int** - must be non-negative), loan period (an **int**, in days - must be non-negative), call number (a required **String**), volume (a positive **int**), and number (a positive **int**).
- Method *CalcLateFee*. This method will calculate the late fee as $0.25/day late with a $20.00 limit. Magic numbers are not permitted.

- A method named *ToString* that returns a **String** and accepts no parameters.This method will create a formatted string that has the item's data on separate lines.

This part of the assignment will only focus on the hierarchy. While you will need to write a test program of some sort to verify that your classes work, it will not be graded. Only the **LibraryItem** hierarchy classes will be evaluated.

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 1A" 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 "Prog1A.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!**