**Program 1A**

**Due**: September 26 (by class)

**Worth**: 60 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 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 use without change from your instructors solution to Program 0), in the same way as the **LibaryBook** did. 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. 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. **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** add 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. 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 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 this PDF file:[Prog1ClassDiagram.pdf](https://blackboard.louisville.edu/bbcswebdav/pid-6459799-dt-content-rid-2054185_2/xid-2054185_2" \t "_blank) .

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

**LibraryItem** - public abstract class

* A constructor that accepts item's title (a **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days) and call number (a **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 data element named as follows: *Title*, *Publisher*, *CopyrightYear*, *LoanPeriod*, *CallNumber*. *CopyrightYear* and *LoadPeriod*must be non-negative. Illegal values should throw an exception (as shown in the text).
* Methods *CheckOut*, *ReturnToShelf*, *IsCheckedOut*, and *GetPatron* should be implemented as they were in Program 0's **LibraryBook** class. No changes need to be made to these methods. You can copy and paste them.
* Abstract method *CalcLateFee*. This method will return a **decimal** and accepts a single **int** parameter, the number of days late. 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 **String**), author (a **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days) and call number (a **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*.
* Method *CalcLateFee*. This method will calculate the late fee as $0.25/day late with no limit.
* 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 **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days), call number (a **String**), and duration (a **double**).
* 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).
* Abstract property named *Medium* which will get/set a *MediaType* **enum**.
* 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 **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days), call number (a **String**), duration (a **double**), director (a **String**), medium (a *MediaType* **enum**), and rating (a *MPAARatings* **enum**).
* An **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).
* 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).
* 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 **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days), call number (a **String**), duration (a **double**), artist (a **String**), medium (a *MediaType* **enum**), and number of tracks (an **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 non-negative. Illegal values should throw an exception (as shown in the text).
* Method *CalcLateFee*. This method will calculate the late fee as $0.50/day late with a $20.00 limit.
* 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 **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days), call number (a **String**), volume (an **int**), and number (an **int**).
* Properties (get and set) for each new data element named as follows: *Volume*, *Number*. *Volume* and *Number* must be non-negative. Illegal values should throw an exception (as shown in the text).
* 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 **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days), call number (a **String**), volume (an **int**), number (an **int**), discipline (a **String**), and editor (a **String**).
* Properties (get and set) for each new data element named as follows: *Discipline*, *Editor*.
* Method *CalcLateFee*. This method will calculate the late fee as $0.75/day late with no limit.
* 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 **String**), publisher (a **String**), copyright year (an **int**), loan period (an **int**, in days), call number (a **String**), volume (an **int**), and number (an **int**).
* Method *CalcLateFee*. This method will calculate the late fee as $0.25/day late with a $20.00 limit.
* 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.

Please review the documentation requirements specified in the syllabus that are expected for each file. These include precondition and postcondition comments for each method, property, and constructor. Also, each variable needs a comment specifying its purpose.

Rather than giving me floppy or Zip disks, you will upload **all your files** to using the *Assignments* tool. 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!**