COMSC-205PY FALL 2020 ASSIGNMENT 2

Due November 5th at 9:00am Eastern Time

Submission checklist

Your final submission should have the following files:

- Book.py
- CD.py
- DVD.py
- additionalFunctions.py
- Journal.py
- Resource.py

At the top of each file, clearly state:

- Your name and the name of the assignment
- Attribution for any sources used, including people (other than instructors) who you asked for help
- Description of the file

Read all the way through this assignment twice before you start working on it. The second time through, take notes, plan out how you will work on this over the next 6 days, and work on a design for the program.

Background

Congratulations! You've been hired on a temporary basis by LITS, the Library, Information, and Technical Services department here at Mount Holyoke. LITS runs the library system, using databases to organize all the different items that can be borrowed.

For your first job, they are asking that you write a program to help track four types of lendable objects: Books, Journals, CDs, and DVDs. Each has a number of different pieces of information related to it. The database needs to be able to hold all those different pieces of information. Some information is the same; for example, every item has a title, and every item should be able to be checked out. Some of the information is different: Books have an *author*, CDs have an *artist*, DVDs have *performers*, and Journals have a *range of years published*.

The Assignment

This assignment is to write several classes and supporting code to represent a basic library that holds these four kinds of materials. We will make use of inheritance to model the real-world relationship between the objects and their properties, which also will minimize repeated code.

Resource

Resource.py will be the parent class for the other four kinds of materials. It should have instance variables for:

- Title
- Publisher
- Type (what kind of resource it is)
- Is it electronic or not
- Is it checked-out or not
- Who it is checked-out to

Resource should also have the following methods:

- a constructor that requires the following information be passed in when a new *Resource* is created: title, publisher, type, if it's electronic or not. All Resources start as not checked out; who it's checked out to can be initialized to an empty string.
- four getters: for Title, Type, Is it checked-out or not, and Who it is checked-out to
- a __str__ method that returns a string containing the variable information in an easy-to-read way.
- a checkIn method to check in a resource. Checking in a resource should set the check-out variable to False, and who it's checked out to is an empty string; it doesn't need any parameters.
- a checkOut method to check out a resource. Checking out a resource requires as an argument a string of who it's going to (the borrower). It sets the check-out variable to True, and sets who it's checked out to to the appropriate name.

Use your judgement about parameters, return values, and similar.

Book, Journal, CD, DVD

Each of these classes "is a" Resource and should inherit from the Resource class. Their constructors should call their parent constructor, and set their respective Types correctly: Book's Type should be "Book", Journal's Type should be "Journal", etc. Each class should provide its own _str__ method which also uses the __str__ method of Resource.

Each class has different instance variables.

The *Book* specific variables are:

- Author

- Number of pages
- Publication year
- Genre

The *Journal* specific variables are:

- Range of years published
- How often they are published (publishing frequency -- could be monthly, quarterly, etc.)
- Is it electronic or not: This is not a new variable, but all Journals ARE electronic

The *CD* specific variables are:

- Artist
- Year
- Length
- Is it electronic or not: This is not a new variable, but all CDs are NOT electronic

The *DVD* specific variables are:

- Year
- Length
- Performers
- Genre
- Is it electronic or not: This is not a new variable, but all DVDs are NOT electronic

additionalFunctions.py

This file should include several additional functions that perform different tasks on a list of Books, DVDs, CDs, and Journals (the list of Resources represents an entire library's collection). Keep in mind that a Book ***is a*** Resource, a DVD ***is a*** Resource, and so on. Write the following functions:

- *getAllCheckedOut*, which takes a list of *Resource* objects, and returns another list containing the subset of them that are checked out.
- *getAllUserHasCheckedOut*, which takes a list of *Resource* objects and a String--the name of the borrower, and returns the list of *Resource* objects that the user has borrowed.
- *getAllOfType*, which takes a list of *Resource* objects and a String--a type, and returns another list containing the subset of *Resource* objects of the specified type. You can check what type an object is by using the getType method you wrote.

main.py

A main function is not required for submission; however, we strongly recommend you write one so that you can easily test your code. Your code will be tested with a program file similar to the main.py uploaded to Moodle; make sure your code can run with it!

Comments and Coding Style

Make sure every class has a docstring comment, every method has comments, and that your code is well-organized.

How to Approach the Assignment

It's a bad idea to try writing every class at once and then test the entire program. A suggested order of operations:

1. Write Resource.py. Start with the variables, the constructor, and __str__
2. Create main.py. Start the main function, create a list of two or three Resources. Make sure all the methods in Resource work as expected.
3. Keep adding and testing methods in Resource until you have them all.
4. Write one of the child classes, such as Book.py. Start with the constructor, and __str__.
5. In main.py, replace the Resources with Books. Make sure everything still works as expected.
6. Write another child class, such as DVD.
7. In main.py, add some DVDs to the list. Again, make sure things still work. Your list can contain both DVDs and Books!
8. Repeat the above two steps until all child classes are complete.
9. Create all the additional functions, testing each after you write it.
10. Test your code with the test file provided. Make any adjustments necessary so they run well together.

An example of printing some resources out of a library:

```
Title: Ancillary Justice
Type: Book
Physical
Available
Author: Ann Leckie
Page Count: 409
Year Published: 2013
Genre: Science Fiction

Title: Inks : the journal of the Comics Studies Society
Type: Journal
```

```
Electronic
Checked out to Valerie Barr
Year Range: 2017-
Frequency: Three times a year

Title: Sweeney Todd: the demon barber of Fleet Street
Type: DVD
Physical
Checked out to Tayloe
Performers: Angela Lansbury, George Hearn, and the original Broadway cast
Year Published: 2004
Length (min): 139
Genre: Musicals
```

Rubric

Each section will be given a letter grade. The average of those letters will generate your final score. Note: Moodle does not allow letters, so they will be translated into a percentage there.

	F	D	C	B	A
Resource	Not present	Has less than half the required variables and methods	Has at least half of the required variables and methods, but half are missing or do not work correctly	Has all required variables and methods, but two or more methods work incorrectly	Has all required variables and methods, works correctly
Book, Journal, CD, DVD	Not present	Has less than half the required classes; or classes exist but do not add any functionality	Two out of four classes exist; or all classes exist, but don't inherit	Three out of four classes exist; or all classes exist but are missing two or more variables or methods; or all classes exist but unnecessarily overwrite parent methods or variables	All classes exist, with the correct variables and methods
additionalFunctions	Not present	One function is present and works correctly	Two functions are present and work correctly	All functions are present, with one or more errors in functionality	All functions are present and work without error

Comments, naming, style	No comments present, no name in file, method names incorrect	Name in file, some methods named correctly, no comments	Name in file, description of file in comment, methods named correctly (typos ok), more than one error that prevents running	Name in file, description of file and most methods, methods named correctly, some variables named meaningfully, significant over- or under-use of comments or one error that prevents running	Name in file, description of file and all methods, methods named correctly, all variables named meaningfully, other comments as needed, no errors that prevent running

Have you read all the way through this assignment twice? If not, read it again!

Do you have questions? Are there things you don't understand? Talk to classmates and ask questions on Piazza.