COP3330 Object Oriented Programming
Dr. Andrew Steinberg
Spring 2022

Programming Assignment 3

My Book Collection

Max Points 100

Due: 3/6/2022 at 11:59pm

Background Story of this Assignment

You have been learning about Data Structures and their ADTs. We have learned the benefits of creating our own ADTs. For this assignment, you will create a LinkedList ADT that will manage a book collection.

Start Early and see the TAs and ULAs! They are here to help you! Don't procrastinate!

Assignment Details

For this assignment, you are going to implement your very own Linked List ADT that will manage a book collection. You will create specific generic methods along with methods that aren't typical in a built-in linked list implementation. Please read the following directions carefully for this assignment

The Classes Name (Yes you will have 2 classes in one file)

You are going to implement two classes in the same Java file.

1. The first class is the `Book` class. This private class acts like a Node in a typical linked list. It will have the following implementation.

   Book Class Attributes

   - A String reference called `title` that represents the name of the book.
   - A String reference called `author` that represents the author of the respective book.
   - A primitive double data value called `cost` that represents the price of the book.
   - A Book reference called `nextptr` that represents that next reference in the linked list. (Sounds familiar right?)

   Make all attributes except for nextptr private!

Book Class Constructor

The `Book` Class will have 2 overloaded constructors. No default constructor needed!

- The first overloaded constructor has three parameters. The first parameter is a String reference called `title`. The second parameter is another String reference called `author` and the third is a primitive double type called `cost`.
- The second overloaded constructor has only one parameter. The only parameter is a reference to a Book object called `old`. This constructor creates a "deep copy" (deep copy is when you have two references pointing to two different objects in the heap, but contain the same exact content).

Book Class Methods

The `Book` Class will have 3 user-defined methods. All 3 user-defined methods are accessor methods to the private attributes that were previously mentioned.

2. The second class is the `Library` Class. This class acts like the manager of the entire linked list. It will have the following implementation.

Library Class Attributes

- A Book reference called `first` that represents the first `Book` in the linked list.
- A primitive int data value called `total` that represents the number of `Book` objects in the list.

Library Class Constructor

The `Library` Class will have 2 constructors. One default and one overloaded.

- The first constructor is the default constructor. The default constructor sets the Library's attributes to default its respective values. The `first` reference is set to null and the `total` is set to 0.
- The second constructor is an overloaded constructor. This constructor takes one parameter which is another reference to some `Library` Object. This constructor will perform a deep copy.

Library Class Methods

The `Library` Class will have 9 user-defined methods. I have provided the method signatures for each one.

```
public void add(String title, String author, double cost)
```

This non-static method will add a new Book object to the end of the list. Make sure to increment your total attribute once added.

```
public Book search(String title)
```

This non-static method will perform some linear search in the list for a certain Book object that contains the matching the `title`.

```
public void reverse()
```

This non-static method will reverse the order of the list in which the Book Objects were inserted. That means the last node will become the new first reference and the old first reference will become the new tail.

```
public void remove(String title)
```

This non-static method will remove the Book reference from the list. You cannot assume that the Book reference containing the respective title always exist in the list. Make sure to decrement your total attribute once removed.

```
public void clearLibrary()
```

This non-static method will clear your entire Library list. In other words, all Book objects will be removed.

```
public boolean empty()
```

This non-static method will check to see if the Library has any books at all.

```
public boolean full()
```

This non-static method will check to see if the Library is full. Your library collection can only hold up to five books.

```
public String getTitle(Book temp)
```

This non-static method will access the Book `temp` reference's title and return it.

```
public void display()
```

This non-static method display the entire content of your Library list. All attributes except for the `nextptr` will be displayed.

The Provided Files

You were provided some files to assist you in this assignment.

1. A text file that shows a sample output of the program run in Eustis.
2. A python script that will test and verify that your code output is correct.
3. A runner class that contains the main method.

Requirements

Your program must follow these requirements.

- **DO NOT CREATE YOUR OWN MAIN METHOD! I have provided a runner file for you that you will use to test your code. The graders will also be using a similar runner file to test your code as well. If you create your own main method and submit it, you will lose points in the respective categories of the rubric it affects.**
- **Name your java file Library.java.**
- Do not add any additional attributes to the classes that were describe to you. This will result in point deductions.
- The output must match exactly (this includes case sensitivity, white space, and even new lines). Any differences in the output will cause the grader script to say the output is not correct. Test with the script provided in order to receive potential full credit. Points will be deducted! Check out the sample text file.
- Do not change the method signatures. Any changes to the method signatures will result in points being deducted.
- You are welcome to create additional helper methods as long as you do not remove the required methods that have been asked in this assignment.
- Do not make any changes to the Runner file that was provided for you. Any changes will result in points being deducted.
- Your code must work on Eustis. If it does not work on Eustis, points will be deducted and not changed as mentioned previously.
- Make sure you include a comment header. Check the assignment page of how it should. **It should be exactly the first line of your Java source file. See the assignment page for more info.**

The Rubric

Please see the assignment page in Webcourses for the Rubric of how the assignment will be evaluated.

Testing the Solution with the Python Script

Once you have completed the assignment, you will need to test it to make sure it matches Dr. Steinberg's sample output. In Eustis, make sure to upload the Python script, your Java Solution, the Runner file, and sample text output. I would highly recommend that you have a folder with those files only.

Once all of those files are uploaded into a directory in Eustis, run the command "`python3 p3testscript.py`". The script will compile your Java source and execute it. You will then see the result in the form of a happy face or sad face. The happy face means your output was correct. The sad face means something was off with the output. Remember, the script is very picky with white space and new lines. Make sure you do not add any extra trailing white space or new lines. Look at the sample text output.

Tips in Being Successful

Here are some tips and tricks that will help you with this assignment and make the experience enjoyable.

- **DRAW! We are dealing with ADTs!! Drawing is going to help you in preventing Null Pointer Exceptions!!**
- Do not try to write out all the code and build it at the end to find syntax errors. For each new line of code written (my rule of thumb is 2-3 lines), build it to see if it compiles successfully. **It will go a long way!**
- After any successful build, run the code to see what happens and what current state you are at with the program writing so you know what to do next! If the program performs what you expected, you can then move onto the next step of the code writing. If you try to write everything at once and build it successfully to find out it doesn't work properly, you will get frustrated trying find out the logical error in your code! **Remember, logical errors are the hardest to fix and identify in a program!**
- Start the assignment early! Do not wait last minute (the day of) to begin the assignment.
- Ask questions! It's ok to ask questions. If there are any clarifications needed, please ask TAs/ULAs and the Instructor! We are here to help!!! You can also utilize the discussion board on Webcourses to share a general question about the program as long as it doesn't violate the academic dishonesty policy.