CS101- Algorithms and Programming I

Lab 09

Lab Objectives: Collections using Java ArrayLists

- ❏ For all labs in CS 101, your solutions must conform to these CS101 style guidelines (rules!)
- ❏ Remember to include javadoc comments for each class and method.
- ❏ Create a Lab09 workspace (i.e. the folder H:\private\cs101\lab09).

---

a. In the previous assignment you wrote a program that allowed the user to create and manage a library with up to four books in it. Obviously in real life, libraries have many more books, but replicating the code for 10, 100 or even more books would be tiresome. This is where Java's ArrayList class comes to the rescue.

Create a new project Lab09a. Copy your `LibraryBook`, `Library` and `LibraryTest` files from Lab08 into the new Lab09a project folder.

The `LibraryBook` class and the `LibraryTest` class should not need any changes made to them (assuming you did them properly last time!) Only the `Library` class needs changing.

Redesign your existing `Library` class so that it should have the following properties and methods:

- **Instance Data Members**
  - `bookList`: an ArrayList of LibraryBook
- **Methods**
  - **Constructor**
    - It does not take any parameters.
    - Initializes the property
  - **Accessor & Mutator Methods**
    - Do we need getter and setter methods for this class?
  - **Other Methods**
    - `isEmpty()`: returns `true` if there are no books in the library.
    - `add()`: takes String author and String title as parameters and adds a new book to the appropriate place so that `bookList` is *sorted* alphabetically with respect to books' author. If the authors of the books are the same, use the title to sort the books. Do you think there is a possibility that the library is full?
    - `remove()`: takes a `LibraryBook` instance as a parameter and removes the book from the library.

Returns true if the operation is successful, false otherwise, such as in a case that there is no such book in the library.

- `findByTitle()`: takes a String title as a parameter and returns the first book with the given title or null if there is no such book is available in the library.
- `findByAuthor()`: takes a String author as a parameter and prints the String representations of all the books that belong to the given author. If the given author does not have any book in the library, print an error message.
- `toString()`: returns a String representation of the library, one book per line (or a message if library is empty)

So basically, what you need to do is to replace the four `LibraryBook` properties with a single property, an ArrayList of LibraryBook, then modify the constructor and instance methods so they work with this new property. *Note:* you should not change any of the method signatures –only their implementations– that way you will not need to modify the other classes at all. Learn what JavaDoc comments are and write it for your class. I can hear you say “Why do we need such things anyway?” Try to answer it yourselves. *Note:* the ArrayList "manages" the collection of LibraryBook objects so you no longer need to check for or set the references to null as you were supposed to in Lab08.

Make necessary changes so that we will be able to use `findByAuthor()` method in our menu driven application `LibraryTest`. See a very simple run in `output.txt`.

To test your methods easily, I recommend you not to use the menu application first. After you are sure everything is working as expected, you can use the menu app. Also you may initially add some books manually to the library to ease the reviewing process by TAs.

b. For this problem, you are given some code in the Lab09b.java file. Download it because you are expected to complete it according to the problem definition. *After you read the problem definition, please study the code given to you to fully understand what you are given, what you are supposed to do, how to use methods etc.*.

An ArrayList is *monotonic* if it is either monotone increasing or monotone decreasing.

An ArrayList A is monotone increasing if for all $i <= j, A(i) <= A(j)$.
An ArrayList A is monotone decreasing if for all $i <= j, A(i) >= A(j)$.
where $A(x)$ is equal to the value at index *x* of ArrayList A.

In the file, you are given an ArrayList `testList`, which consists of String variables for test purposes. You are also given an ArrayList `correctResultList`, which consists of the correct results of the test cases. You are not supposed to change these variables. These are for easing the test process of your code.

You are supposed to do *TODO* tasks in the file. There are two static methods you are expected to implement:

- `parseString()`: takes a String str as a parameter. These strings are the final test strings defined at the top of the file. Please observe them. They have a common structure. While parsing the string, fill an ArrayList of integers and at the end return the list.
- `isMonotonic()`: takes an ArrayList of integers as a parameter. Returns true if the list is monotonic, `false` otherwise.

After you implement the methods, use it properly in the main method to compare the results you have found with the correct results (`correctResultList`). Print it in a nice format as shown in the sample run.

Sample Run:

	Run	Expected
Test 1	true	true
Test 2	true	true
Test 3	false	false
Test 4	true	true
Test 5	true	true
Test 6	false	false
Test 7	true	true

The column *Expected* shows the correct results whereas the column *Run* shows the results you have found. If the *Run* and *Expected* values in all rows are the same, you have implemented correctly.

c. Create a new project Lab09c. For this problem, you should input integers from the user. You will construct an ArrayList of integers that contains those inputs and pass to a static method `solution1()` as a parameter. That method should return a new ArrayList such that each element at index i of the new ArrayList is the product of all the numbers in the original ArrayList except the one at index i. See the sample runs.

For these types of problems, after figuring out how to solve it, you should consider all the corner cases. For example, what happens at least one of the elements in the list is 0? What if the product of the values exceeds int and long capacity? For the sake of simplicity in the problem, we will ignore these cases and the list will not contain 0 because you are supposed to stop taking input if you encounter a value 0.

The solution for this assignment is quite trivial. So, what if you are <u>not</u> allowed to use division or logarithm in your solution. Can you come up with a solution then? Design your algorithm using a static method `solution2()` same as before.

<u>Sample Runs:</u>

```
Please enter the values: 2 4 6 8 0
Output list: { 192, 96, 64, 48 }
```

```
Please enter the values: 1 2 3 4 5 0
Output list: { 120, 60, 40, 30, 24 }
```

```
Please enter the values: 1 2 3 0 4 5
Output list: { 6, 3, 2 }
```