CS101- Algorithms and Programming I

Lab 08

Lab Objectives: Classes and Objects; Object vs. reference, comparing objects, copying objects.

- ❏ 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 Lab08 workspace (i.e. the folder H:\private\cs101\lab08).

---

a. Create a new project Lab08. Create a class, `LibraryBook`, that represents an individual book that can be borrowed from the library with the following properties and methods:

- **Instance Data Members:**
  - `title:` stores the title of the book as String
  - `author:` stores the author of the book as String
  - `dueDate:` stores the due date of the loaned book as String
  - `timesLoaned:` stores the number of times the book is loaned as integer

- **Methods:**
  - **Constructor**
    - Takes appropriate parameters. *Think* about the real life. (*Just an author and title would make sense, right?*)
    - Initializes the variables appropriately.

  - **Accessor & Mutator Methods**
    - getter and setter methods for the instance data members.

  - **Other Methods**
    - Add methods to loan the book and have the book returned. When loaning a book, you will need to specify the new `dueDate`, and increment the number of `timesLoaned`. When the book is returned you should simply reset the `dueDate` to an empty string. Provide two further methods `boolean onLoan()`, and `String toString()`.

b. Write another class, say `TestLibraryBook`, to demonstrate your `LibraryBook` class by creating a few instances of `LibraryBook` and displaying them on the console using the functions you have implemented. Next, without changing the

`LibraryBook` class, add code to the main method so as to compare two `LibraryBook` instances using == and .equals. Make sure your test data includes all three possible alternatives, that is (a) two references to a single object, (b) two references to two individual objects with different properties, (c) two references to two individual objects with identical relevant properties. Notice that your program compiles(!) and runs, and that both comparisons give the same results in all cases. Can you explain why?

c. Next add an `equals(LibraryBook other)` method to your `LibraryBook` class, so that books with the same title and author are considered the same (irrespective of their `dueDate` or `timesLoaned`). Run the program again, demonstrate & explain any changes in output from the previous run.

d. Finally, add a copy constructor and two other methods, `hasSameTitle()` and `hasSameAuthor()` to your `LibraryBook` class. Both methods should take a LibraryBook as a parameter and return a boolean value. Check they are all working correctly before moving on.

e. Let's level up. Create a new class called `Library` with the following properties and methods:

- **Instance Data Members:**
    - `b1, b2, b3,` and `b4,` that are `LibraryBook` instances. So, we have four books in our library when you think about it in terms of real life examples.

- **Methods:**
    - **Constructor**
        - It should create an empty library (by initialising the four properties to null)

    - **Accessor & Mutator Methods**
        - These types of methods are one of the most essential methods, but for this assignment and class we do not necessarily need them. Nevertheless, it is good practice to generate getter and setter methods after you define the properties of your classes.

    - **Other Methods**
        - `isEmpty():` returns true if there are no books in the library.
        - `toString():` returns a String representation of the library, one book per line (or a message if library is empty)
        - `add():` adds a new book --with the given title and author as parameters-- to the first available (null) property. Returns true if the operation is successful, false otherwise.

- `remove()`: removes the specified `LibraryBook` as parameters from the library by setting the property to null, returns false if book not in library
- `findByTitle()`: returns the first book with the given title as parameter or null if no such book is found.

f. Write a program, `LibraryTest`, that allows the user to manage a library by selecting options from a menu that includes: Show, Find, Add, and Exit. If the Find option locates a book with the requested title it should show it to the user and then allow them to Loan or Return it, or Remove it from the library, or simply return to the main menu. There are so many possibilities than we can show in sample runs. Try to cover them all in your test cases before presenting your work to your TAs. You can find one example in the `output.txt` file as an example.