Lab 11 – Object-Oriented Thinking Lab

A class definition provides the blueprint that is used to create objects of that data type. The design of a class supports encapsulation by declaring the instance variables **private** and providing **public** getters and setters (aka accessor and mutator methods), as well as **public** constructors. Private instance variables of a class are visible by all the methods of the class, but are not visible or modifiable outside of the class. Please use these guidelines when building your classes.

1) Navigate to the **Labs** package in Intelji and create another package named **Lab11** and create a Java class that is public, without a main method, called **`Book`** to model a book on a bookshelf. This book class will have two instance variables: a title (String) and an author (String). Write a default constructor which sets title to "Test" and author to null. Write a parameterized constructor that will receiving a title and author for these fields. Lastly, implement getter and setter methods for your instance variables.

2) In **`Book`**, write a method, called toString(), which returns a string with the book's information (The values stored in the Book objects instance variables). Please have the toString() method use the format below. Use \" to have a quote character in a string. Please note that you will be overriding a built in toString method, so add the annotation if suggested by IntelliJ.

    "The Lord of the Rings" by J.R.R. Tolkien
    "Nixonland" by Rick Perlstein

3) Download `BookTest.java` and test your **`Book`** class. Make sure all tests pass before moving onto the **`Bookshelf`** class, as it requires a fully functional **`Book`** class to work.

4) In the **Lab11** package, create a class called **`Bookshelf`**, again without a main method, to model a bookshelf. This bookshelf object will have two instance variables: size (int), which specifies how many books can be stored on the bookshelf, and books (ArrayList of type Book), which contains the Book objects on the bookshelf. Write a default constructor which sets the size to 2 and initializes the ArrayList. Write a parameterized constructor which receives a single parameter for size and initializes the ArrayList. Write a getter method for the size instance variable. (Don't write a setter method, you can't change the size of a bookshelf after it's been created!) Write a getter method for the books instance variable.

5) Rather than a single setter method, write three specific methods to modify the contents of the bookshelf: a public void method **addBook**, which takes a single book parameter and adds it to the books ArrayList if there is room on the bookshelf; a public method **removeBook**, which takes no parameters, removes the first book on the bookshelf (the book at position 0 in the books ArrayList), and returns it (if there are no books in the bookshelf, then return null); and a public void method **emptyBookshelf**, which takes no parameters and removes all of the books from the shelf (empties the books ArrayList).

6) Add Javadoc comments to your code to fully document what you have created. Below is an example:

```
/**
 *This method returns the value of the variable x
 */
Public int getValue(){      }
```

7) Test your classes by creating another class called **`Lab11`** within the **Lab11** package. This class will contain a main method. Within the main method create three objects of the class **`Book`**. You can use any data you would like to populate the objects. Use the toString() method to output the data from each **`Book`** object. Then create one **`Bookshelf`** object where you add the three book objects to it. Display the contents of the bookshelf by writing an enhanced for loop to iterate over the **`Book`** objects in the books ArrayList, calling the toString() method of each **`Book`** object. Now empty the bookshelf and again display the contents of the bookshelf, which should be a blank output.

8) Take a screenshot of the running of your `Lab11.java` class.

9) Test your code by running the Junit tests: `BookTest.java` and `BookshelfTest.java`. Make sure all tests pass.

10) Take a screenshot of the running of `BookTest.java` and `BookshelfTest.java` with all test cases passing.

11) Upload `Bookshelf.java` and `Book.java` to Gradescope and ensure that all tests pass.

12) Upload your `Lab11.java, Bookshelf.java,` and `Book.java` along with the <u>three</u> screenshots to the submission area within Canvas.