Project Stage 3: Tests for searching and passage lookup

Work in groups of 2 or 3 for this assignment. Submit the test cases as specified below in paper form at the beginning of class. You should have at least a two dozen test cases for each of the following two categories. Try to cover the full range of relevant inputs, including both valid and invalid inputs. *Do not reuse any of the examples given below in your solution!*

1. **Searching:** The first version of the Bible Reader Application will perform searches by using the *contains* method on *String*. Thus by default it will do exact phrase matching. You need to design a set of test cases to verify that it works as expected. Make sure to cover all of the types of searches a user might try, especially those that will not work as they expect. Eventually we will need to implement these tests, so we will need to know the expected results. But for now list your test cases by specifying the search word(s) and as much as you can say about the expected results. Group your tests, providing an explanation for each group. Your test suite should look something like this:

```
 These searches should return no results since none of these words
appear in the Bible:
           ferzle
           computer
These searches won't return any results because we are doing
exact matching:
           God world
           man woman
 These should return exactly one verse:
   For God so loved the world that
```

2. **Passage lookup:** In the second version of the software you will add the capability of looking up passages. You need to provide a set of test cases that verify that that functionality is working correctly. Consider things like: different abbreviations users might enter for the name of a book, different ways they might specify passages, incorrect books/chapters/verse numbers, etc. Group the cases according to how you expect the code will parse the input and expected results. In this case you only need to specify whether the passage is valid or invalid. Here is a list to help you think about some (but not all) of the possibilities:

```
 These should be valid:
       Gen (or Genesis or Ge)
       Ruth 2 (or Rth 2)
       1 Cor 4:3-7 (or ICor 4:3-7 or 1Cor 4:3-7)
       Lev 11:7
       1 Peter 2-3
       John 2-3:4
 These should be invalid:
       1 Hesitations 2 (invalid book)
       John 2:12-26 (invalid verse number)
Ps 149-160 (invalid chapters)
```

Project Stage 4: Basic searching and File I/O

You must *work by yourself* on this stage.

In this stage you will implement several methods related to searching a Bible and implement a method to read in the verses from a file. You will start with a new project this time. The **Reference** and **Verse** classes are implemented for you already. The only code you might need from your Stage 2 solution is from the ***SimpleBible*** class.

1. Fork the repository at https://bitbucket.org/ferzle/biblereaderstartstage4 and use *SourceTree* to checkout your forked copy of the project. Then import it into Eclipse. See the video at *http://www.screencast.com/t/xoqGWn9hbp* for a reminder of how to do this (using the above URL instead of the one in the video).
2. Take a look at the **VerseList** class. It is just a wrapper class for an *ArrayList* with two fields added for convenience. You need to understand this class to implement searching and file I/O.
3. Next look at the **Bible** interface and the two classes that implement it (well, they will when you are done), **TreeMapBible** and **ArrayListBible**.
4. Notice that there are Javadoc comments for the methods in the **Bible** interface, but not in the classes that implement it. You should read the Javadocs for the methods in the **Bible** interface to make sure you are properly implementing each method in your class.
5. For this stage, implement **ArrayListBible**. You will implement **TreeMapBible** at a later stage. Implement all of the methods that are labeled **Stage 2** or **Stage 4**. As mentioned above, you can copy methods from your Stage 2 solution, but do so method-by-method, being careful to make sure you don't delete anything important.
6. Implement the **readATV** method of **BibleIO**. See the comments in that file for hints.
7. Do not use a try-with-resource statement when you implement **readATV**. Although I am really glad Java added this feature to Java 7, it will complicate grading if you use Java 7 features. (In general, do not use *any* Java 7 features for any stage of the project.)
8. Run the tests for Stage 4, starting with **Stage04BibleIOTest** since some of the other tests depend on that class working correctly. Fix all of your bugs until you pass all of the tests.
9. As usual, add your names and other documentation in the appropriate places, remove any extraneous comments and code, and format your code before submitting it.
10. Make a copy of **MyGrade_P2.txt,** naming it **MyGrade_P4.txt,** and fill in your actual time spent and expected grade (keep these on the same line as the headings!) and include a brief justification of your expected grade.
11. Use **Handin** under assignment **235-P4** to hand in ***ArrayListBible.java***, ***BibleIO.java***, and **MyGrade_P4.txt**.
12. Grades will be based on: correctness (mostly passing the tests), the sanity and efficiency of your code, formatting/organization of your code, and proper documentation.

Hints/Comments:

- You should not change the signature of any of the methods that are already in the code.
- When searching, you should call **toLower** on both the search string and the text of the verse you are searching.