Project Stage 7: Passage lookup

You must work in the *self-selected pairs*. Implement *passage searching* in your Bible and model classes, add passages searching to the GUI, and add some menu items in your GUI. ***There are a lot of nit-picky details so read the description carefully***.

1. Implement the remaining methods of **ArrayListBible**.
2. Add Javadoc comments to the 4 methods in ***MultiBibleModel*** interface that have TODO comments. These methods should do exactly what their names and arguments suggests. Do this *before* you implement them so you are clear what they are supposed to do.
3. Implement the methods marked for Stage 7 from **BibleReaderModel**. Think about all of the methods you have to implement before you jump in. If you plan ahead, you will be able to reuse code, both in the sense of calling other methods and cut-and-pasting similar code.
4. Obtain and run the Stage 7 tests. They will be available on https://cusack.hope.edu/Notes/?Instructor=BibleReader. Copy them into the ***bibleReader.tests*** package of your project—copy them in the file system and refresh the project in Eclipse and they will appear.
5. Add a second button for passage lookup to your GUI code. The search button should remain.
6. The "search stats" should now output a reasonable message depending on whether the user performed a word search or passage lookup, and it should display a helpful message if the user provides an invalid passage (being as specific as possible). It should at the very least give the number of search results and remind them of what they searched for.
7. Add a **File→Exit** and **Help→About** menu items. The **About** item should pop up a window with information about the application, including your name(s) as the author(s).
8. As usual, create **MyGrade_P7.txt** (copy it from a previous one) 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.
9. One person should zip up ***src*** folder and submit it and **MyGrade_P7.txt** using Handin under assignment **235-P7**.
10. Grades will be based on passing the tests, the documentation, correctness of the GUI, how informative/correct your search stats are, and on the details of your implementation.

Hints

- Look at the test cases to determine the various ways a passage might be input by the user.
- The hardest method to implement is `getReferencesForPassage(String reference)` from ***BibleReaderModel***. It will parse the input and call one of the other find methods. You can use *regular expressions* to help you implement this method.
- Once you implement one of the "get" methods, the others should be pretty easy. After implementing the first one, the rest took me 5 minutes to implement. Think before you code!
- Some of the methods in the Bible class may be tricky, but take advantage of code reuse. That is one of the reasons I included methods like ***getVersesExclusive*** and ***getVersesInclusive***. For example, if you want *John 2-3*, you can use the ***getVerseExclusive*** with references *John 2:1* and *John 4:1* so you don't have to know how many verses are in chapter 3. This should even work if the next chapter doesn't even exist. (If you intend to do this, you should add *Dummy 1:1* to the end of your Bible and then make sure to remember that it is there.)

- Do not have ***getVersesInclusive*** call ***getReferencesInclusive*** or vice-versa. It will likely result in really poor performance for one of these. The same is true of the “exclusive” versions of these methods. This is a case where code reuse is not worth it.
- Do not use the method ***getVerses(ArrayList<Reference> references)*** as you implement the other methods for this stage. If you do an analysis of your algorithms you will realize that they become unacceptably slow when you use this method.
- See the ***Regular Expression Example*** on the ***Links*** page on the course website for an example of one way that you can parse the input. It is an incomplete example of exactly what you need to do. You can start with it and fill in the remaining details if you wish. See Appendix F of *JSS* or the ***Java Regular Expressions Tutorial*** on the ***Links*** page for a more on regular expressions.