Project Stage 9: Dealing with multiple versions

You should *work in self-selected pairs* on this stage. You must use pair programming—you may not work separately on different parts of the project. This stage involves updating the application so that it can display results from multiple versions of the Bible at the same time.

1. Update ***BibleReaderModel*** so that it properly deals with multiple versions of the Bible simultaneously. Read the documentation for all of the methods in ***MultiBibleModel*** (except for the three methods you haven't implemented yet) to ensure you have implemented everything in ***BibleReaderModel*** according to that documentation.
2. Obtain and run the Stage 9 tests. These will test that everything in the ***BibleReaderModel*** works properly with multiple versions. One of these tests depends on the Stage 7 tests, so make sure you still have those in your project.
3. Update the GUI code (i.e. ***ResultView*** and ***BibleReaderApp***) so that results from all currently loaded Bible versions (i.e., those that have been added to ***BibleReaderModel***) are displayed. Use an HTML table for the results, with one column for the reference and one column for the results from each version, with the appropriate column headers.
4. Change anything else in the GUI that is relevant to properly dealing with multiple versions.
5. Pick one of the following options for adding multiple Bibles to your model:
   a. For up to 90%, add the other two versions of the Bible (asv.xmv and esv.atv) to your model in the constructor of **BibleReaderApp**.
   b. For up to 100%, have the application still start by just reading the KJV automatically and add a **File→Open** menu item that will allow you to open additional Bibles.
      i. Use a **JFileChooser** to pick the file (see *ScatterPlot* for an example).
      ii. Read in the file, create a Bible object, and add it to the model.
      iii. For up to 100%, update the current results to take into account the additional version as soon as it is loaded. For up to 95%, make future searches include the newest version but don't update the current results.
6. Be sure to test your GUI very carefully. Make sure the text displayed is from the proper version(s). Also make sure it works for cases that have missing verses—I recommend doing a passage search for *Mark 9*, and word search for *worm dieth not*. In these cases the ESV should be missing a few verses.
7. Here is one way to test your application: Start your program (with just the KJV loaded). Search for **eaten**. Load the **ASV** and see if it loads the second column. The second result (Gen 3:17) should be slightly different in the ASV. Do a passage search for **John 3**. Now load the **ESV** and see if it updates properly. Now do passage search for **Mark 9** and verify that your program does not crash and that verses 44 and 46 are missing from the ESV (so there should be blanks in those spots). Finally, search for **worm dieth not** and verify that it displays 3 verses for ASV and KJV but only 1 for ESV (again, it should be leaving blank space where the missing verses are).
8. Create **MyGrade_P9.txt** and fill in your actual time spent and expected grade and include a brief justification of your expected grade.
9. Zip up your **src** directory and submit it along with **MyGrade_P9.txt** using Handin under assignment **235-P9**.
10. Grades will be based on passing the tests, the implementation of several of your classes, and whether or not your application runs as specified.

See next page for hints and tips.

Hints/Tips

- For a sample HTML table based on my solution, go to the ***Misc*** section of the ***Notes*** page (linked from the course website) and take a look at ***sampleHTMLTable2.html***. (Notice that this is similar to the one referenced in a previous stage, but this one has 4 columns.)
  - There are several ways to combine results from multiple versions. You can't do it by using ***addAll*** on an ***ArrayList<Reference>*** since this will just append the References from one list to the end of the other list. You can either do it by going through the lists and adding references at the proper location (since they need to be in order) or you can do it the easy way. The easy way involves using a better data structure. Since you want to combine lists and only keep one copy of each Reference, you need to use some sort of **Set**. The obvious choices in Java are **HashSet** and **TreeSet**. If you look at the APIs, one of these is the clear choice for our purposes (and the other one is clearly *not* the one to use).
- You can pass in collections to the constructors of other collections. For instance, if you want to create a ***TreeSet*** with elements from an ***ArrayList*** (or vice-versa), you can do that. In fact, the easiest (and best) way to implement most of the *find* methods is to create some sort of set to combine the results and then pass it into the constructor of an ***ArrayList<Reference>*** so you can return the correct type of object.
- As usual ***start early!*** Once you figure out what you need to do this stage isn't too difficult, but you might get stuck on either correctly combining the results or getting the GUI to work properly. You need to allow time to read some of the APIs, think about how to correctly code things, and experiment a little.
- If you run out of heap space when you run the tests, you can increase the heap size in Eclipse as follows:
  - Select ***Run->Run Configurations...***
  - In the left side of the window, find the ***JUnit*** heading.
  - Look under the JUnit heading for the test you are running and click on it (It might be the class name or the package name, depending on whether you run them one at a time or all together).
  - In the right side of the window, make sure the ***name*** field has the package/class you expect.
  - Click on the ***Arguments*** tab
  - In the box under ***VM Arguments*** (NOT Program arguments), put: **-Xmx256M**

  This will increase the maximum heap size to 256MB. You can change that to 512 or larger if you need to. If you run the tests separately, you will need to do this for each test.