Project Stage 1: Tests for the basic classes

You should *work in pairs* on this stage.

1. Fork the repository at https://bitbucket.org/ferzle/biblereaderstartstage1 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 some time to look at all of the classes to figure out what they are supposed to do. Several are helper classes that have already been implemented for you. Others are classes that you will implement very soon (but not quite yet).
3. In particular, take a look at ***Stage02BibleTest*** as an example of JUnit testing.
4. Write complete tests for the **Reference** and **Verse** classes.
    a. The classes **Stage01StudentReferenceTest** and **Stage01StudentVerseTest** contain skeleton code you will need to test the *Reference* and *Verse* classes. Begin with the tests for *Reference* (that class has a few more comments and sample code in it).
    b. For both classes:
        i. Implement the required tests by adding methods like *testEquals*, *testGetters*, etc.
        ii. Construct sample references and verses based on whatever values you want. Except for the books (which need to correspond to actual books from BookOfBible), the contents don't need to correspond to actual verses.
        iii. Test *equals*, *hashCode* (mostly that it is consistent with equals), and *compareTo*.
        iv. A few tests of the getters and *toString* is sufficient.
        v. You can test the constructors along with *equals* by constructing several objects that are equal by using different constructors.
    c. *NOTE: You haven't implemented the classes yet so your tests will fail if you try to run them. In fact, if they pass then they aren't very good tests.*
5. Use Handin under assignment **235-P1** to hand in just the two java files for your tests: **Stage01StudentReferenceTest.java** and **Stage01StudentVerseTest.java**. (Make sure you submit the ***.java*** files, NOT the .class files!) Only one person in the pair should submit the files, but make sure both names are in both files.
6. Grades will be based on things like:
    a. Did you test all of the methods?
    b. Are your tests correct?
    c. Did you include all cases? (e.g. For methods that return a boolean, test cases that return both true and false. For **compareTo**, test cases where it should return a negative number, zero, and a positive number, etc.)
    d. Can I be pretty certain that if my code passes your tests that it is implemented correctly (i.e. free of errors)?
    e. Did you test that the "contract" related to **equals** and **hashCode** is maintained?

Hints: When writing tests, never test what things shouldn't be—always test what they *should* be. For instance, do *not* write a test that asserts that `getVerse()` should not return 3 for Ruth 2:1. Instead, test that it *does* return 1. It is also a good idea to put only a few (or one, according to some people) tests per method. This is because once a test fails in a method, the rest of the tests in that method are not run. Thus, you can only see the results of those tests after you fix the first bug and rerun the tests. There may be another bug that you didn't identify because the next test in the method that identifies the bug didn't run.

Project Stage 2: Basic classes

You must *work by yourself* on this stage.

1. Finish the implementation of the classes **Reference** and **Verse** by implementing all of the methods that are not implemented (They will have a "TODO" comment in them).
2. Add complete Javadoc comments to **Reference**. Notice that **Verse** has them as an example to follow.
3. Add your name as an author to the relevant classes.
4. Implement the first version of the **SimpleBible** class by implementing all of the methods with the comment *// TODO Implement me: Stage 2*.
5. Obtain the Stage 2 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.
6. Run the tests on your classes and make sure you pass all of the tests.
7. Once you can pass the official tests, run your own tests. This will help you to evaluate your tests. Specifically, if any of your tests fail, you will probably discover that the tests were incorrect and not the code.
8. Make sure you remove extraneous code and comments from all of your files and format your code nicely. Use CTRL-SHIFT-F (to format your code) and CTRL-SHIFT-O (to organize import statements).
9. Fill out the **MyGrade_P2.txt**. Replace the underscores with 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.
10. Use Handin under assignment **235-P2** to hand ***Reference.java***, ***Verse.java***, ***SimpleBible.java***, and ***MyGrade_P2.txt***.
11. Grades will be based on whether or not you passed all of the tests (it is expected that you will), on the implementation details (i.e. did you make good choices), and documentation.

Hints:

- As you are thinking about implementing **equals** and **compareTo**, look at some examples of implementations of these methods on other classes to get the idea of how to do it. But also keep reading for some more tips about implementing them.
- **equals** needs to determine whether or not two objects are equal to each other based on the values of the fields using equals and/or == (on the fields), and if-else statements as appropriate. In our case, we need all of the fields to be equal in order for the objects to be equal. (For some classes only some of the fields are used.).
- For **equals**, don't forget that the parameter is an *Object*. Thus you will need a call to instanceof (in an if statement), perform a cast, and usually use a local variable so you don't have to cast multiple times.
- **compareTo** needs to determine which one is less by using the values of the fields, utilizing *compareTo*, <, and > (on the fields), and if-else statements as appropriate.
- Enums (like **BookOfBible**) are treated like integers, so you can use == on them. You can also use the **compareTo** method on two enums, as long as they are of the same type.
- Additional hints are provided as comments in some of the methods.