CSE 2221 - Project 10

Task
Familiarity with designing and coding a realistic component-based application program without being provided a skeleton solution.

Original Project Instructions
Project 10 Instructions from CSE2221 Project Site

Program Requirements

- You have a customer who wants an **easy to maintain** online glossary (essentially a dictionary)
- The output of your program should be a collection of HTML pages, similar to project 5:
  - Top-level HTML page
  - Separate HTML pages for each word
  - Clicking on a term in the top-level page should take you to the respect HTML page that gives definition, etc.
  - If a term in your glossary appears in the definition of a different word, clicking on that term in the definition should send you to the glossary page for that term
- Each term in the top-level HTML page should consist of a single word (i.e. no leading, trailing, or intermediate spaces / white-space)
- The terms in the top-level HTML page should be in alphabetical order
- In each term's glossary page, the term should appear in red boldface italics just before the definition (see an example of this on the original instructions)
- Your glossary should be generated in a batch-fashion (see my tips on this below)
- Your input file should be in the following format:

```
firstTerm
This is the definition for the first word, it can appear on a single line

secondTerm
Or the definition
can
occur on
multiple lines
,
 the definition ends when an empty line is found like the one below this one

lastTerm
The program should not check for valid inputs, the user
of the program is responsible for providing an input file
that meets the stated conditions
(this should be an empty line, just mentioning this since you can't really see it)
```

- The program shall ask a user for the name of the input file and for the name of a folder where all of the output files will be saved (data, a path to their Downloads folder, etc.)
- The output folder must already exist, your program is not responsible for ensuring it exists or creating it if it doesn't exist, that is the responsibility of the user

- The top-level HTML page should be named "index.html"
- The separate HTML pages for each term should be named in the format "term.html" where term is an actual term in your dictionary (e.g. house.html, book.html, etc.)
- The "name of the input file" and "name of the output folder" should be interpreted as follows:
  - Each item could be just a name or a path
  - If it is a path, it could be relative to the current folder or absolute from the top level of the file system
  - It is considered bad form to insert anything before or after what the user supplies as input (e.g. "/Users/user/Desktop/" + userFolderName)
  - Similarly, it is also considered bad form to insert a constant file extension to the end of whatever the user supplies as input (e.g. userFileName + ".html")
  - Therefore, the program shall respect the user's ability to input anything as a relative or absolute path as the name of the input file or the name of the output folder and will not augment the input in any way
  - Examples of valid user inputs for file name: "data/terms.txt", "terms.txt", "Users/user/Desktop/cse2221/example_inputs/terms.txt"
  - Examples of valid user inputs for output folder name: "data", "" (nothing, empty string), "Users/user/Desktop/cse2221/example_outputs/ex1/", "Users/user/Desktop/cse2221/example_outputs/ex1"
  - NOTE: All above extensions are for MacOS, it will be different for Windows and Linux users

You are completely on your own for how you choose to implement this. I recommend taking inspiration from some of the projects / labs you have done in the past and try separating your work into small, easy to manage methods. Do not make separate classes, just have everything in one file.

You will need to make method contracts for each method you create, so be wise.

If you have everything in your main, you are receiving a 0.

Tips, Rules, & Things to Note

- You can only use components from the OSU components package and components from the standard Java libraries that have been used before in lectures/labs/projects (if you are unsure, ask)
- You should not use other components from any other libraries that have not been used in CSE2221
- Keep everything to one file (i.e. don't make any additional classes)
- Make small, easy-to-manage methods (with JavaDocs and contracts)
- As for the batch fashion, my recommendation is to load everything from the file into some data structure, then process each item from the data structure one at a time (i.e. create the html page for the first time, fill in the page (with links to other words in the glossary), then move onto the next word)
- I recommend incorporating the nextWordOrSeparator and generateElements methods from lab, you will use these in SW 2 from what I have heard (therefore it is important they work correctly)
- We have sorted Queue objects in a previous lab
- Take some inspiration from project 4/5
- You should have method contracts for each of the methods you create

Data

Here is a sample input file (I suggest you create your own as well): terms.txt

Here is a sample output for the corresponding sample input file: index.html

When you click on a term in the sample output, notice the URL at the top of your browser (particularly what is at the end of the URL... the file name!)

This example input is the bare minimum, you can have it look like this, or do more

EXTRA CREDIT

There is extra credit for this project worth up to 20% of the project grade (i.e. you can get a 12/10 on the project)

Develop a JUnit test fixture that tests all of the methods that you designed in your program. We will only give credit for well-designed, well-thought-out, and well-documented test cases

Steps

1. Go read all of the feedback your graders gave you for projects 2-9
2. Copy and paste the *ProjectTemplate* project to create a new project folder for this project
3. Name the project *Glossary*
4. Open the *src* folder, then open *(default package)*
5. Rename any ONE file to *Glossary.java*
6. Delete the other files
7. Go reread all of the feedback your graders gave you for projects 2-9
8. Write your program to meet the program requirements mentioned earlier
9. Once finished implementing your solution, go reread your feedback from previous projects
10. Ask yourself: have you improved your code style & quality from previous projects? Is your current solution the most elegant or the most efficient? Is there any way you can make your code better? I guarantee the answer is yes, there is a way to make your code better
11. Create some test cases for each of your methods to get some sweet extra credit
12. Once completed and convinced your code looks good: Zip it up with the naming scheme I recommend and submit to Carmen