Change request log

1. Team

Veshwaj – Driver

Joshua – Navigator

2. Change Request

Change request # 1: adding word count to status bar

3. Concept Location

Use the table below to describe each step you follow when performing concept location for this change request. In your description, include the following information when appropriate:

- IDE Features used (e.g., searching tool, dependency navigator, debugging, etc.)
- Queries used when searching
- System executions and input to the system
- Interactions with the system (e.g., pages visited)
- Classes visited
- The first class found to be changed (this is when concept location ends)

When there is a major decision/step in the process, include its rationale, i.e., why that decision/step was taken.

Make sure you time yourselves when going through this process and provide the total time spent below.

The following is an example of a concept location process for the change request "Color student schedule":

Step #	Description	Rationale
1	We ran the system	
2	We interacted with the system: we looked for where in the UI the change would be made. After hovering the mouse over that portion of the status bar, a tool tip popped up that read “line, column[-virtual] (offset/length)”	We figured that this string is somehow associate with where the change needed to be made, so this tool tip gave us a target to search for
3	We searched in all project files for "line, column[-virtual] (offset/length)” in the project files and found a string in jedit_en.props at line 537 that was named “view.status.caret-tooltip”	This string must be referenced somewhere in order for the tooltip to display. So now we have a variable name to search for to find where this is being used
4	Searching for “view.status.caret-tooltip” in all project files lead us to StatusBar.java where this String variable is being utilized in line 82 and was being used to set a property of the object named “caretStatus”	If the string variable is being used in this class, this is likely the class that we need to make the change in. Likewise, this object is likely associated with what needs to be changed.
5	We searched the StatusBar.java class for “caretStatus” to find where this object is being used and found a method called updateCaretStatus	The comments for this method read “Updates the status bar with information about the caret position, line number, etc” so this is a method we will need to edit for this change request

Time spent (in minutes): 90

4. Impact Analysis

Use the table below to describe each step you follow when performing impact analysis for this change request. Include as many details as possible, including why classes are visited or why they are discarded from the estimated impact set.

Do not take the impact analysis of your changes lightly. Remember that any small change in the code could lead to large changes in the behavior of the system. Follow the impact analysis process covered in the class. Describe in details how you followed this process in the change request log. Provide details on how and why you finished the impact analysis process.

Step #	Description	Rationale
1	We looked at where updateCaretStatus is used by right clicking it and selecting "Find Usages"	To track the classes that could be impacted by the change.
2	We found that this method is called in several other classes that track events such as when the caret is moved.	Ultimately we concluded that changing this method would have no impact on these other classes, assuming the changes we made did not cause any compilation errors.

Time spent (in minutes): 20

5. Actualization

Use the table below to describe each step you followed when changing the code. Include as many details as possible, including why classes/methods were modified, added, removed, renamed, etc.

Step #	Description	Rationale
1	We noticed that updateCaretStatus was determining the number of characters in the file by calling buffer.getLength().	We knew we'd need a different method to find the word count, but we figured that we could follow a similar logic that the program used to find character count
2	Before writing a new method to find word count, we used intelliJ's Navigate feature to search for word count and found doWordCount() in jeditTextArea.java	Because this method was set to protected, we determined we could use the algorithm in this method in another method to carry out the word count, wordCount().
3	We determined that we would need to write a new method that would determine the current word position of the caret, so in jeditTextArea.java we wrote wordOffset()	wordOffset counts the number of spaces and new line characters in the document prior to the caret's position
4	Calling both wordCount() and wordOffset() in StatusBar.java, we put the returned strings into 2 variables we created, wordCount and wordOffset	We were able to use these variables alongside reusing the formatting used for displaying the caret character position and character count to complete the change

Time spent (in minutes): 60

6. Validation

Use the table below to describe any validation activity (e.g., testing, code inspections, etc.) you performed for this change request. Include the description of each test case, the result (pass/fail) and its rationale.

Step #	Description	Rationale
1	We ran JEdit and opened a text file with a relatively small number of word that we could easily count manually.	We expected that the word count would match. and when moving the caret from word to word, we verified that the caret position word count also updated The test passed.

Time spent (in minutes): 10

7. Timing

Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	90
Impact Analysis	20
Actualization	60
Verification	10
Total	180

8. Reverse engineering

Create a UML sequence diagram (or more if needed) corresponding to the main object interactions affected by your change.

Create a partial UML class diagram of the classes visited while navigating through the code. Include the associations between classes (e.g., inheritance, aggregations, compositions, etc.), as well as the important fields and methods of each class that you learn about. The diagram may have disconnected components. Use the UML tool of your preference. When a significant fact about a class or method is learned, indicate it via annotations on the diagram. **For each change request, start with the diagram produced in the previous change request. For the first, you will start from scratch.**

9. Conclusions

For this change, locating where the change needed to be made was probably the most challenging part of the code. However, the implementation of the change was not too difficult and did not take too long. The only methods and classes that were changed were the StatusBar.java and JEditTextArea,java class.Testing was completed by running the code, and checking to see if the status bar changed as I typed words into JEdit.

Classes and methods changed:

- *org/gjt/sp/jedit/StatusBar.java/UpdateCaretStatus()*
    - *New variable created to show word count and offset*
- *org/gjt/sp/jedit/JEditTextArea.java/WordCount()*
- *org/gjt/sp/jedit/JEditTextArea.java/WordOffset()*
    - *New methods created to return word count and offset*