Change request log

# Team

Samuel Armstrong

Ashwin Somasundaram

# Change Request

6.1 : Implement an option in the View menu that allows to hide the scrollbars.

# Concept Location

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We went into the file list and started looking for anything having to do with the status bar. | We can go through all of these and determine where we need to make our changes, |
| 2 | We messed around in the StatusBarOptionPane. | To see if we could find what we might need to change in this class. |
| 3 | We inspected the StatusBar class and found the section of code we will need to change. | We figured this is where the code that needed to be changed would be. |

**Time spent (in minutes):** 65

# Impact Analysis

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We went through all the files that we thought might need to be changed and checked what we needed to change in them | To help make sure we didn’t miss any file changes we wanted to be as thorough as possible. |
| 2 | We inspected the class StatusBar.java | We determined this class would need to be changed, this is where the majority of the logic for the changes would happen. |
| 3 | We inspected the class StatusBarOptionPane.java | We determined that no changes needed to be made in this file. |
| 4 | We inspected the class JEditTextArea.java | We determined this would be the best place to count the words, as there was no method for that in the code already. |

**Time spent (in minutes):** 20

# Actualization

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We changed part of StatusBar.java. | We retrieved the total number of words and the number of words at the caret position and displayed them as instructed, using the wordCount method we made in JEditTextArea.java to count the words to be displayed. |
| 2 | We added the wordCount method to JEditTextArea.java | We created a wordCount method that takes in text and counts the number of words in it, we would use this in the StatusBar.java class |

**Time spent (in minutes):** 75

# Validation

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Test Case:  Input: (Caret at the beginning)  These are a bunch, of words! It works!!  Here are more words!  Output (in Status bar): 0/12 | This is the regular expected behavior.  The test passed. |
| 2 | Test case defined:  Inputs: (Caret at the end of “bunch”)  These are a bunch, of words! It works!!  Here are more words!  Output (in Status bar): 4/12 | This is the regular expected behavior.  The test passed. |
| 3 | Test Case:  Input: (Caret at the end of the input)  These are a bunch, of words! It works!!  Here are more words!  Output (in Status bar): 12/12 | This is the regular expected behavior.  The test passed. |

**Time spent (in minutes):** 60

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 65 |
| Impact Analysis | 20 |
| Actualization | 75 |
| Verification | 60 |
| Total | 210 (3 ½ hrs) |

# 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.**

# Conclusions

For this change, concept location was pretty easy as we only had to make a few changes, so we didn’t have to look around too much to find the places in the code where we would need to implement changes, most of the time spent in this section was spent familiarizing ourselves with the project. Impact Analysis was also relatively fast and easy as we only had to change a few files for the project to work properly.

Classes and methods changed:

* org/gjt.sp/jedit/gui/StatusBar.java
  + changed updateCaretStatus method
* org/gjt.sp/jedit/textarea/JEditTextArea.java
  + added method wordCount

Testing was also very easy and quick, as all we had to do was test a few different caret positions on the page, and add some commas and such to test the edge cases.