Change request log

# Team

Specify the team members working on this change request.

Jacob Wiedemeier, Travis Dula

# Change Request

Provide the id and description of the change request.

Change request #1

The left side of the status bar of jEdit reports: the line number containing the caret, the column position of the caret, the character offset of the caret from the beginning of the file, and the number of characters in the file (see Figure 1). You are requested to modify the status bar to show: the word offset of the caret from the beginning of the file and the number of words in the file. As a result of this change, the status bar for the example of Figure 1 should report: “2,6 (76/138886)(12/23731)”.

# 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 | *Used ctrl+t in Visual Studio to find the StatusBar.java file* | *The place we’re editing was called a status bar in the change request, and there’s already an existing way to display information there – just need to find where to piggyback off.* |
| 2 | *Locked in on the updateCaretStatus() method as where our changes will be made* | *Description of the method matched what we were looking for just about perfectly.* |
| 3 | *Confirmed that contents of updateCaretStatus() support appending additional information* | *Verifying that step 2 was correct* |

**Time spent (in minutes):** 10

# 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 | *Noted that updateCaretStats() is a void method and the only variable outside of the scope of the method accessed is caretStatus* | *Minimize the search space for impact* |
| 2 | *Ensured that no part of the code is dependent on the contents of caretStatus.getText* | *Verified that there is little to no chance of side-effects. Any that slip past here can be caught in postfactoring.* |

**Time spent (in minutes):** 15

# 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 | *Noted that many parts of the caretStatus can be toggled. Also noted that making this aspect of caretStatus toggleable was outside the scope of this change request.* | *Minimized impacted code by reducing our changes to only affecting a single method, rather than adding new toggles elsewhere in the code.* |
| 2 | *Investigated the Buffer and JEditTextArea classes to search for existing ways to get the values we are displaying* | *Ideally we only have to make changes in updateCaretStatus(), so being able to reuse code* |
| 3 | *Copied logic from JEditTextArea.showWordCountDialog() to find words until position and words in file* | *Assuming that existing code works, this is the easiest way to implement the change.* |
| 4 | *Changed logic to use caretPosition as the end of the first code segment inspired by doWordCount() call to only count words up to caretPosition* | *Changing logic slightly to fit the needs of this change request* |

**Time spent (in minutes):** 40

# 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 | *Tested with empty file.* | *This is the regular expected behavior.*  *Works as expected.* |
| 2 | *Tested with file of words where caret was between words.* | *This is the regular expected behavior.*  *Works as expected.* |
| 3 | *Tested with file of words where caret was inside of a word.* | *This is the regular expected behavior.*  *Works as expected.* |
| 4 | *Tested with file of words where caret was at start of file.* | *This is the regular expected behavior.*  *Works as expected.* |
| 5 | *Tested with file of words where caret was at end of file.* | *This is the regular expected behavior.*  *Works as expected.* |

**Time spent (in minutes):** 15

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 10 |
| Impact Analysis | 15 |
| Actualization | 40 |
| Verification | 15 |
| Total | 80 |

# 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.**

Graphical user interface

Description automatically generated with medium confidence

# Conclusions

Perform and analysis of the change requests and the change process. List the major challenges this change request posed.

List all the classes and methods you have changed.

*For this change, most of the issues came from trying to figure out how to run the program. There was an additional hiccup when we realized that doWordCount() was a protected method, so we had to copy the logic instead of just running it. Testing was trivial once we managed to run the program.*

*Classes and methods changed:*

* *Org/gjt/sp/jedit/gui/StatusBar.java/updateCaretStatus*