**Change request log**

* **Team**

Specify the team members working on this change request.

Documentation Youssef Donia

Coding Ryan Lea

* **Change Request**

Provide the id and description of the change request.

commit 9272e90d92dc028649fb4c24d42d6cefaadc1f6d

Added variables that count how many words there are total and how many are to the left of the caret.

Also added appends to ensure that the data would be shown to the user,

* **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* | *To get familiar with some of the features of the system, and identify the screens or graphical elements we had to change.* |
| 3 | *We looked for the character count and how it worked* | *We found out the characters are updated on the statusbar* |
| 4 | *We used the github repository to search for any files with the name StatusBar and found one named statusbar.java* | *We selected this class because it was a part of the gui and have the logic for displaying the charactercount* |
| 5 | *We inspected the class statusbar.java and found that it used jedittextArea to display and store data about the text.* | *We noticed the method used jedittextArea to store the information about the text.* |
| 6 | *Because the jedittextArea has all of the information we need in order to calculate the amount of words in the area we figured this would probably be the place to change* | Need to test. |
| 7 | *We decided to place a break point at line 338 to see if*  *the character count changed.* | *When the caret status changes updateCarotStatus is called and this changes the statusbar.* |
| 8 | *We marked this Class as located!* | *We confirmed that this class needs to be modified.* |

**Time spent (in minutes):** 120

* **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 made changes to the StatusBar class to add a way to count words* | *The operations were done in the statusbar class in order to display the word count next to the pervious data* |
| 2 | *This change had little more impact besides possible increases in processing.* | The string is converted into a String[] and then the length is used for the word count |
| 3 | *Jpanel will display different text* | Now with this change the jpanel will have additional information to display. |

**Time spent (in minutes):** 30

* **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 | *The only class that was changed was StatusBar* | *We had the reasoning that all of the operations should be handled in the same place that character count is.* |
| 2 | *Made sure to understand where to insert the word count, and made the change* | To find the place where we should add the code we looked for the character count and caretposition in the code. |
| 3 | We compiled and ran jEdit and tested the wordcount and total words feature, we used spaces new lines and tabs. | *To make sure everything works.* |

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

* **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 | *Figuring out what kind of test cases we will need to do.*  *Typing one word*  *inputs: a*  *output: (1/1)* | *This is the regular expected behavior.*  *The test passed.* |
| 2 | *Test case defined:*  *Inputs: a a a... a*  *Expected output: 200/200* | *This shows that it can be many words on one line and spaces are delimiters for words* |
| 3 | Test case:  input: A  A  OUTPUT:  2/2 | This shows that words can be delimited by return characters |

**Time spent (in minutes):** 30

* **Timing**

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 120 |
|  |  |
| Impact Analysis | 30 |
| Actualization | 60 |
| Verification | 30 |
| Total | 240 |

* **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**

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

The major challenges of this project was getting the compiler to work with ant. After that I believe the assignment was not that difficult. I used tools in intellij to help me find the impact analysis and testing was done by hand for a while. This assignment was helpful to get me to actually do some software engineering.

I changed:

org/gjt/jedit/gui/StatusBar.java/updateCaretStatus()

List all the classes and methods you have changed.

For example:

*For this change, concept location was relatively easy because the system is small and its architecture and code are not complicated. Concept location, impact analysis, actualization (and change propagation) was done using JRipples, which was very useful. Testing was performed using JUnit and Abbot. It took me long time to make Abbot work, mainly because I had to learn how to use the tool.*

*Classes and methods changed:*

* *org/gjt/sp/jedit/search/HyperSearchResults.java/HyperSearchResults*
* *void foo(p:integer)*
* *org/gjt/sp/jedit/search/HyperSearchResults.java/HyperSearchFind*
* *etc.*