Change request log

# Team

Specify the team members working on this change request.

Aditya Mantri and Meah Tahmeed Ahmed

# Change Request

Provide the id and description of the change request.

Change request #5

When searching, it is often helpful to highlight all search hits of entered string. Currently, clicking Find will only highlight the next (or previous, if searching backward) occurrence of the search string after the caret position (see Figure 5). You are requested to implement search highlighting without moving to highlight all occurrences.

# 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 | Searched the text of the source for strings “search”, “find”, and “replace” | These terms seemed most relevant to modifying a searching behavior |
| 2 | Used jEdit’s various search functions to see their names and purposes | This would allow us to match source code to behavior, as there are multiple classes related to various searches |
| 3 | Narrowed the searching behavior down to the find static method of SearchAndReplace |  |

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

# 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 | Examined related classes to confirm the isolation of the find method |  |

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

# 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 | Attempted to convert find from returning a single selection to an array of selections | This seemed to be the simplest way to convert the behavior to highlighting all search terms |
| 2 | The difficulty of this grew the further we got in, and once we got what we expected to work it froze jEdit. We were not able to solve this freeze. |  |

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

# 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 | N/A because actualization failed |  |

**Time spent (in minutes):** N/A

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 90 |
| Impact Analysis | 30 |
| Actualization | 40 |
| Verification | N/A |
| Total | 160 |

# 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.**

Diagram

Description automatically generated

# 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 example:

Deciding where the concept was located was rather difficult, and even now we cannot know if we were correct. Terms such as “highlight” proved fruitless, as the code never described itself as highlighting search terms. Future impact of this code may increase, as it is possible to use find in far more locations than initially intended. Once we found what we expected was the proper code to change, it was not structured in a way that would make changing its behavior feasible in the time remaining.

Classes and methods changed (attempted):

* org/gjt/sp/jedit/search/SearchAndReplace.java/SearchAndReplace
  + boolean find(View view, Buffer buffer, int start, boolean firstTime, boolean reverse)