Change request log

1 Team - bughunters

Nishant Kashiv – change request implementation (complete process)
Saurabh Deotale – documentation

Code repository - https://github.com/kashivns/cs515-001-s20-bughunters-jEdit
(For looking up changes, go to pull requests section for the repository and search with following query:
is:pr head:changeRequest#je3)

2 Change Request

Change Request Id – changerequest#je3

Description - In jEdit, the HyperSearch feature should list all occurrences of the search string (different Search>Find, which only locates the next match). However, the list of results of a HyperSearch only highlights the first occurrence of the search string in any given line, as shown in Figure 4 for the search string "the". Note that lines 522, 533, and 534, among others, contain more than one occurrence of this string, but only the first one is highlighted (in purple). The request is to fix the HyperSearch feature, so that its list of results highlights all occurrences of a search string.

3 Concept Location

Before performing the following steps, we changed the version of ivy from 2.2.0 to 2.5.0 in the build file to make sure the project builds, and the built application runs without any issues.

IDE used – IntelliJ IDEA

The following are the steps followed for concept location for change request #3:

Step #	Description	Rationale
1	● Run the installer version of the software to observe the behavior related to the change request	To get familiar with the system and its behavior.
2	● Search for a string in a text where it occurs multiple time in the line.	To get an idea of what the current behavior is and what is the expected.
3	● Observe the modules and their functions to reduce the scope of our search. ● We found a search package at org.gjt.sp.jedit.search which implements all the logic for Searching.	Try to identify modules involved by studying the software design.
4	● In the package “org.gjt.sp.jedit.search” we found the following classes which may contain the logic of highlighting the results: 1. HyperSearchFileNode 2. HyperSearchFolderNode 3. HyperSearchNode 4. HyperSearchOperationNode 5. HyperSearchRequest 6. HyperSearchResult 7. HyperSearchResults 8. HyperSearchTreeNodeCallback	Some code navigation as preparation for debugging the application and limiting the scope of the search.

5	We investigated the classes HyperSearchResult and HyperSearchResults as their names looked promising.	Code navigation for some files which look interesting and, they may be responsible for displaying of the search results.
6	Further investigate these classes to put breakpoints in their methods We found promising sub classes like TreeDisplayAction and HighlightingTree where we put the breakpoints.	Try to locate the code which highlights the search results. Add breakpoints to debug
7	We found that convertValueToText() method contains a while loop for traversing over all the matches. But while debugging the loop exited after the executing for just once, but the string contains multiple occurrences of the searched string.	
8	The above behavior was due to a finally block inside the while loop, where m was set to null. This broke the loop condition. Because of this the loop does highlight all the matches.	Try to check the output file size again and identify root-cause.
9	We commented the finally block to check our theory.	We confirmed this finally block should be removed.

Time spent (in minutes): 150

4 Impact Analysis

The impact analysis for this change was not very complicated as we are only changing the loop break condition variable in the catch block instead of the finally block.

Step #	Description	Rationale
1	The impact of the change which we identified is contained in that same method, as we are not changing any member variable or calling a method of any other class in the identified changes.	Track all the classes impacted.

Time spent (in minutes): 20

5 Prefactoring (optional)

The only change needed is to remove the finally block in the convertValueToText() method in the class HighlightingTree, which is a subclass in the HyperSearchResults.java. This change did not need any change in the existing structure of the code.

Time spent (in minutes): 0

6 Actualization

Step #	Description	Rationale
1	We removed the finally block in the convertValueToText() method of the class HighlightingTree and put the loop break condition in	The finally block is executed every time a try block is executed, regardless of any occurrence of any Exception. This caused the loop to exit after

	the above catch block.	executing once.

Time spent (in minutes): 20

7 Postfactoring (optional)

There is no need for postfactoring as the change is just re arranging the execution of a statement.

Time spent (in minutes): 0

8 Validation

We ran unit tests as well as manual tests for confirming correct behavior as well as output.

Step #	Description	Rationale
1	Manual Test Test case defined: Open a text file in jEdit, press Ctrl + F and enter a String in the text which occurs multiple times in a single line. Check the Hyper Search box and click on Find. A new window with all the matches will appear. Expected Output: All the occurrences of the searched string will be highlighted in the results.	This is the regular expected behavior. The test passed.
2	Manual Test Test case defined: Open a large text file of around 100MB. Search for a string using the HyperSearch feature. When the result window is displayed and before the results are displayed, close the results window. Expected Result: The result window should be closed without any unexpected behavior.	Testing the handling of InturruptedException.

Time spent (in minutes): 15

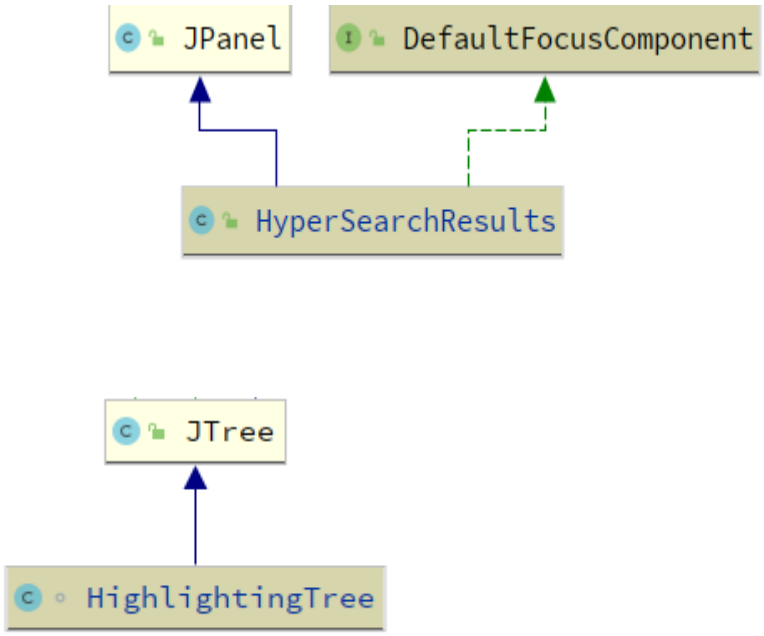
9 Timing

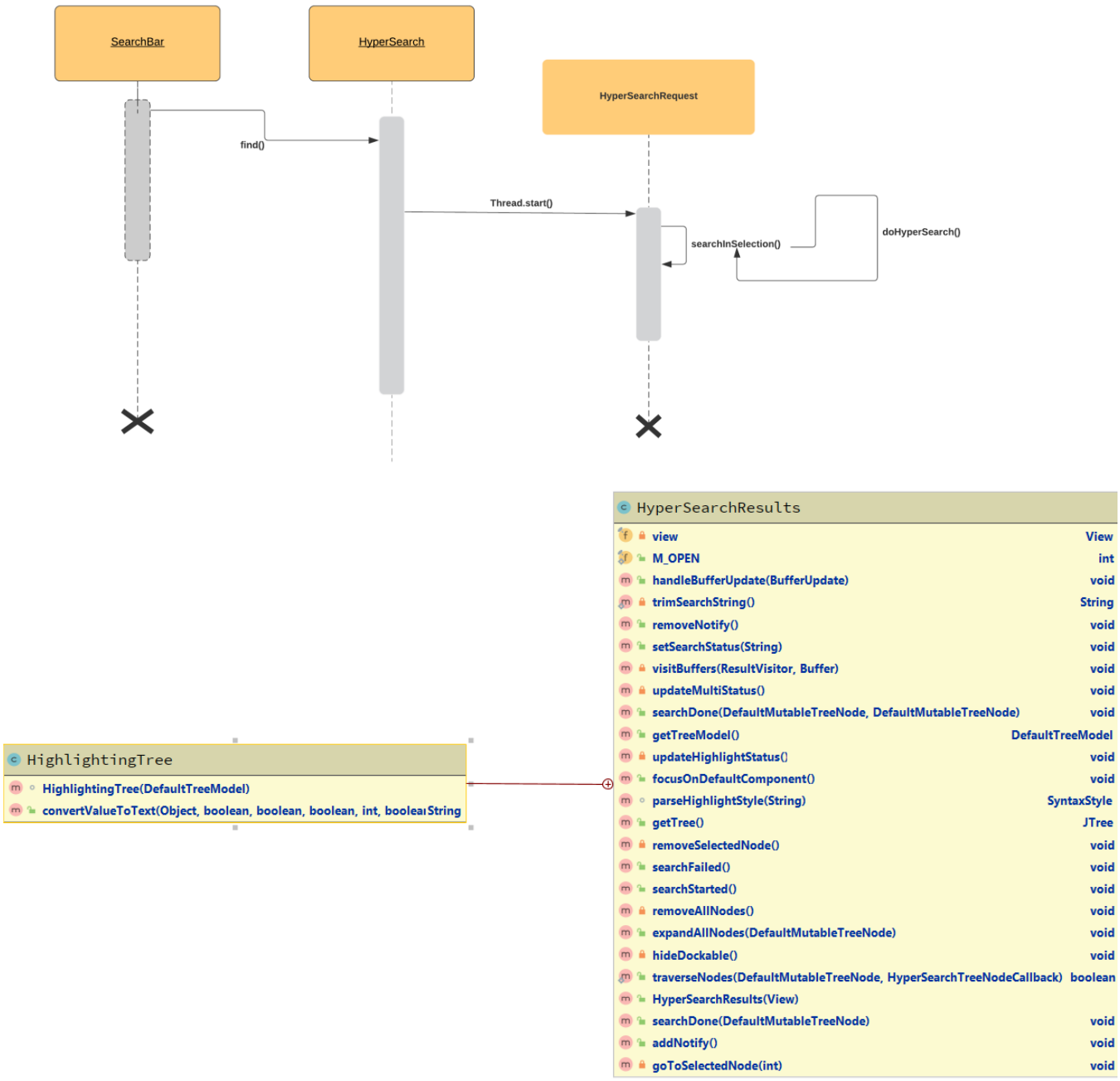
Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	150
Impact Analysis	20
Prefactoring	0
Actualization	20
Postfactoring	0

Verification	15
Total	205

10 Reverse engineering





11 Conclusions

For this change, concept location was again the most challenging step. Once concept location was completed rest of the process went by smoothly. The understanding of the software design helped immensely in locating the root-cause. We used IntelliJ IDEA IDE for all the steps performed (implementation and testing).

Classes and methods changed:

- /org/gjt/sp/jedit/search/HyperSearchResults.java
  - convertValueToText()