Change request log

1 Team

Glade Snyder

2 Change Request

Change request #je-3 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

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	*Did a Find in Path Search in Intellij for the string HyperSearch*	*Wanted to see what files and paths that involved the Hypersearch functionality*
2	*Used Find in Path Search again but instead used a Regex search and used a file mask of *.java so that only Java files were pulled up in the results*	*Wanted to narrow the search and its results more to pinpoint where the main functionality of the Hypersearch was.*
3	*Did a class search Ctrl-N for HyperSearch. I was wanting to know if Intellij had one of these searches and found that it did online.*	*Wanted to find the class Hypersearch if it existed. Turns out it did. I saw that there was a package that pertained to Hypersearch so I thought it would be worthwhile to see if there was a class.*

4	*Was in wrong Class HyperSearchFileNode so I decided to find all classes that referenced it. I did this by click on variable and show references.*	*I found the class HyperSearchFileNode.java and looked through the class. It didn't have anything to do with the actual Hypersearch function but rather was a support class. So, I wanted to see what classes incased this class.*
5	*Decided to click on HyperSearchRequest.java file that referenced the HyperSearchFileNode Class.*	*I saw that one of the files HyperSearchRequest.java referenced the aforementioned class. It seemed like the most likely file that dealt with performing the actual search.*
6	*Looked at all of the class functions inside of Class HyperSearchRequest.*	*New that if this file did the HyperSearch that it would be in the class function definitions.*
7	*Found Function doHyperSearch(buffer buffer, int start, int end, DefaultMutableTreeNode, bufferNode) and decided that this is the function I wanted based on the name.*	*Function seemed to be named appropriately for what it was trying to do. I also looked at the guts of the function and it had comments and code that alluded to it being the Hypersearch Functionality.*
8	*Put a breakpoint in the above function and ran Jedit in debug mode and performed a Hypersearch*	*I wanted to confirm that I was in the right class and function.*
9	*Marked HyperSearchRequest and doHyperSearch as the class and function that I will start my modification.*	*The breakpoint stopped right where I was expecting, and I stepped through the function to guarenteed it was the function I wanted.*
10	*I then used the above class to determine which class contained the search result strings and where they get modified. I decided to go into the HyperSearchResults class and look.*	*Came to realize quickly that I wasn't in the correct subclass. I need to go down deaper because the text was getting highlighted at a lower level. I also was able to determine that the doHyperSearch Request was correctly finding all of searches even when two were in the same line. I knew that I needed to go to a different class that changed the results to print on the screen. This led me to HyperSearchResults Class.*
11	*Looked for hints of where strings were being highlighted. I did a search throughout the HyperSearchResults.java and ended up going down a false path where there were matches that would be set and if there was another pattern match it created a child on that same line. I believed this to be the spot but it turned out to be unused code and found out quickly that it wasn't by setting a breakpoint and it never hitting.*	*I decided to go through this class because it was the most obviously named for what I was trying to find and edit. By having results in the class name I concurred that it must contain the results that were returned in the results.*

12	*Found a sub class that is inside of the HyperSearchResults.java called Highlighting tree class which had a function inside called convertValueToText. Set a breakpoint inside of it to see if was highlighting the matches on the line*	*Really looked in there because I found a function in the class that was called convertValueToText which inside of the function it had references to highlighting text.*

Time spent (in minutes): 240

4 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.

Make sure you time yourselves when going through this process and provide the total time spent below.

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 detail 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	*I made a list of classes and Methods that would be were touching and impacted by a change in the HyperSearch doHyperSearch function*	*To track the classes that were directly using and used by the Function. Specifically, ones that were part of the search.*
2	*I inspected SearchMatcher and BoyerMooreSearchMatcher*	*These were the Classes with the methods that had the algorithms for finding string matches.*
3	*I removed SearchMatcher and BoyerMooreSearchMatcher from files/classes that would need to be modified.*	*I was able to determine that the functions were working appropriately and the api's were returning the right results. Whether it was the first match or second match of line.*
4	*I inspected HyperSearchOperationNode*	*I thought this Node would contain possible information of the lines that have matches and thus would need to be modified*
5	*I removed HyperSearchOperationNode from the list of files/classes that would possibly need to be modified*	*Was quickly able to see that this class had nothing to do with the lines that had match results by looking at the Class API's*

6	*I inspected HyperSearchResults*	*This Class seemed like the most obvious class that would contain the results of a Hyper Search due to its name.*
7	*I determined that HyperSearchResults was the Class that needed to be modified. I pinpointed the function convertValueToText as the one that needed to change.*	*I was able to determine that was the desired function by seeing local variables that were setting the Font, Highlight of the text. Especially the function call highlightString gave me all that I needed to know. I then used a breakpoint there to confirm.*
8	*I removed all other Classes and Files from lists of ones that need to be changed except for the HyperSearchResults Class.*	*I was able to limit the scope of change to the function convertValueToText inside of the HyperSearchResults.hava file.*
9	*I added HighlightingTree as a class that I was going to change.*	*Inside of the HyperSearchResults Class it turned out that convertValueToText was a public function of a sub class of HyperSearchResults. The subClass being HighlightingTree*

Time spent (in minutes): 60

5 Prefactoring (optional)

Using the table below, describe each step you follow to prefactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that were modified, added, removed, renamed, etc.

Make sure you time yourselves when going through this process and provide the total time spent below.

Step #	Description	Rationale
1		
2		
3		
4		

Time spent (in minutes): 0

6 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.

Make sure you time yourselves when going through this process and provide the total time spent below.

Step #	Description	Rationale
1	*I deleted the finally branch that set m to null after the try called function to find next match Inside of the HighlightingTree Class*	*After spending a good amount of time in the debugger stepping through the code I was able to see that the while loop was dependent upon m != null. Since finally branch would always set m to null even when a new match was found in the new line the while loop would break because m was set to null. I removed the finally branch so that all other matches were found and added to the matches variable. The reason it isn't needed is because the function sets m to null already if a match isn't found.*
2		
3		
4		

Time spent (in minutes): 30

7 Postfactoring (optional)

Use the table below to describe each step you followed to postfactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that were modified, added, removed, renamed, etc.

Make sure you time yourselves when going through this process and provide the total time spent below.

Step #	Description	Rationale
1		
2	.	
3		
4		

Time spent (in minutes): 0

8 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.

Make sure you time yourselves when going through this process and provide the total time spent below.

Step #	Description	Rationale
1	*Test case defined:* *Did a hyper search for a very common string "a"* *Expected output:* *All letter "a"'s will be highlighted*	*Wanted to see if I fixed it to where it finds all matches in a line rather than just the first several.* *The test passed.*
2	*Test case defined:* *Did a hyper search for a common string "as"* *Expected output:* *All letter combinations as are found*	*Wanted to see if one that has just a few duplicates would work appropriately.* *The test passed.*
3	*Test case defined:* *Did a hyper search for a not common string "letter"* *Expected output:* *Wherever "letter" is found as word by itself or inside of a word will be highlighted*	*Wanted to test a rarer word and make sure it comes up well and finds it once with no exceptions* *The test passed.*
4	*Test case defined:* *Did a hyper search a word found once in text "currently"* *Expected output:* *Currently is highlighted once.*	*Since I am removing the finally I wanted to make sure I didn't break a corner case where it will grab garbage on the next if there is only one found.* *The test passed.*
4	*Test case defined:* *Did a hyper search for a string that isn't in file "Glade"* *Expected output:* *Should have no results*	*Wanted to make sure I didn't break a failing case where the search doesn't bring up anything* *The test passed.*

Time spent (in minutes): 30

9 Timing

Summarize the time spent on each phase.

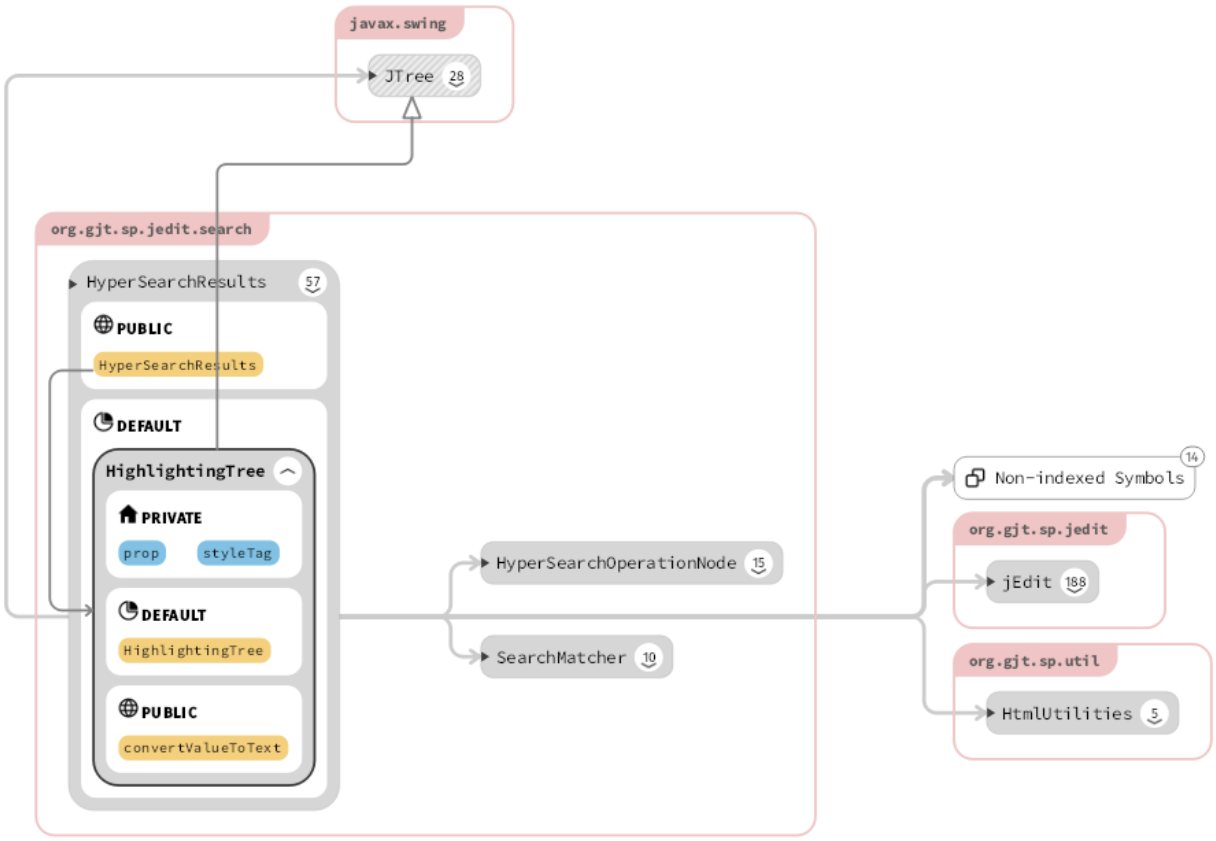
Phase Name	Time (in minutes)

Concept location	240
Impact Analysis	60
Prefactoring	0
Actualization	30
Postfactoring	0
Verification	30
Total	360

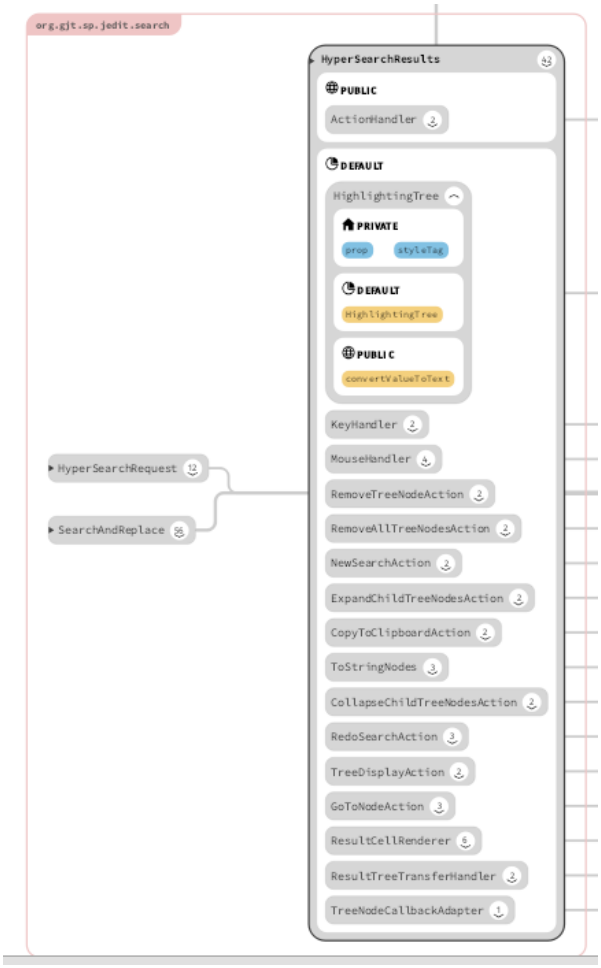
10 Reverse engineering

Here is the package of all the search classes and modules. The HyperSearchResult is part of the package and you can see all the HyperSearch related classes in the package. The ones that were touched in the assignment was HyperSearchRequest.

This is the Diagram of the HighlightingTree class which is inside of the HyperSearchResults class that I edited and made changes to. Inside of the class I made changes to the convertValueToText function. As yiu can see the Highlighting tree is a Inherits from JTree. You also can see that the HyperSearchResults class uses HyperSearchOperationNode and SearchMatcher classes.



This is the UML of the classes that use the HyperSearchResults Class and thus correlate with the Highlighting Tree. None of these I had to change or were impacted. The HyperSearchRequest and the SearchAndReplace classes used the HyperSearchResults Class



11 Conclusions

This change request wasn't too difficult. Most of the time that I spent on it was simply trying to get familiar with the code and understanding where I needed to make the changes to the code. I wanted to make sure that I really understood the project and what was going on before I made any changes. Almost always if one tries to make a change without fully understanding the problem it causes more problems and makes the problem cascade into many. Since I spent adequate time to understand the exact problem, I was able to scaffold in a fix that didn't require too many changes.

Classes and methods changed:

- Org/gjt/sp/jedit/search/HyperSearchResults.java/HighlightingTree
    - `public String convertValueToText(Object value, boolean selected, boolean expanded, boolean leaf, int row, boolean hasFocus)`