Change request log

# Team 4

* Cade Bradshaw (Pilot)
* Dax Collison (Navigator)

# Change Request #2

* Full String Recent Files Modification: Adds the ability for the recent files search to include substrings found within the titles of files instead of just the first word.

# Concept Location

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Ran jEdit from IntelliJ |  |
| 2 | Interacted with jEdit. Inspecting different parts of the software. We noticed that to access Recent Files you must use the options at the top. | This helped us identify what to search for. |
| 3 | We searched for “Recent Files” using the IntelliJ’s search everywhere feature. | We identified that “Recent Files” will need to be labeled in the code. |
| 4 | From 2 results, we clicked on the class RecentFilesProvider and GeneralOptionPane. | We selected these classes because we needed to investigate |
| 5 | We inspected both classes and were not satisfied with the direction it was taking us. | We noticed that we were deeper into the code and did not know where it started |
| 6 | We searched for a basic Javax Gui on the internet | To find out an example of a simple version of a GUI |
| 7 | From the knowledge we gained we searched for the term JFrame and found the View class. This class acts as the view of a MVC architecture and anything relating to the GUI is linked there. | Based on what we found online JFrame is one of the building blocks of the GUI so if we can find where a lot connects we can have a better way of knowing the structure of the code. |
| 8 | Searched for the term Menu | We realized that the File button is located within the menu so we searched for it. |
| 9 | We found the View class creates a menubar object. Investigating. | We want to understand where the menubar is controlled in the GUI to be able to adjust the code |
| 10 | We then went to the internet again to look at the docs for creating a menubar with javax. | This helped us to understand the basic structure of the menubar so that we could know a little bit more about what we should be looking for. |
| 11 | Started searching for MenuItem in the search bar | This is what a menu contains so we looked for this. |
| 12 | We found GUIUtilities class and clicked and read through it. This led us back to the jEdit.java class were there is a variable called a property manager. This then led us to the PropertyManager class. | This trail led us to somewhat of a dead end and we needed to find out where things like “New in mode” strings were located as this would point us to the menu code. |
| 13 | We searched for “New in mode” and found a jedit\_gui.props file. This included all of the text used in the menu bar. We then searched for the file itself and found that it is used in line 3562 of jEdit.java. | The jedit\_gui.props file is used by jEdit class to populate the gui with text and helps define the actions available to the program. |
| 14 | We searched for “recent files” in the jedit\_gui.props file and found that the text presented in the menu after pressing “File” is shown in this file. Below that starting on line 185 it shows that  recent-files.code=new RecentFilesProvider() is called after the pictures are loaded into the menu. So this takes us back to the RecentFilesProvider class as the one that needs to be changed. | This confirmed our original hunches that this was a class that needed to be modified, however because we went to understand the building blocks of the main library used we had a better understanding of the structure of the software. We also are now aware that the jedit\_gui.props file needs to be modified in conjunction with the RecentFilesProvider. |

**Time spent (in minutes):** 155

# Impact Analysis

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | *We searched for RecentFilesProvider and found that the only instances of the string are in the class itself and in jedit\_gui.props.* | *This was done to see where the class is used and therefore could have an impact.* |
| 2 | *We made a list of all the classes that were called by the methods used by RecentFilesProvider.* | *These are classes that might be needing modification as well.* |
| 3 | *Lines 90-120 of RecentFilesProvider needs changing and it does not use other classes that need modification.* | *Only class that needs changing is the RecentFilesProvider class.* |

**Time spent (in minutes):** 12

# Actualization

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | *We modified RecentFilesProvider. Line 104/105 and Line 107/108.* | *We realized that the only thing that needed changing was how the regex was setup. Its current state was only using “\*”.* |
| 2 | *Updated lines 104/105 from  regex += “\*” to regex = “.\*” + typedText + “.\*”* | *This allowed for the word typed in to be found anywhere in the string of the file name. The “.” Character signifies any character while the “\*” character specifies that there can be 0 or more of that character. We used the internet to find some basic regex guides.* |
| 3 | *Updated lines 107/108 from pattern = Pattern.compile(StandardUtilities.globToRE(regex), Pattern.CASE\_INSENSITIVE); to pattern = Pattern.compile(regex, Pattern.CASE\_INSENSITIVE);* | *This had to be done because StandardUtilities.globToRE() will convert a Unix style glob into a regular expression when the string is already in a regular expression form. This allowed for a more common regex style to be used. The jEdit documentation was used to find this out.* |

**Time spent (in minutes):** 32

# Validation

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | *Test case defined:*  *Inputs: words that are not at beginning of file name*  *Expected output: file will be enabled in recent files menu if the word is contained any where in the string* | *This is the regular expected behavior.*  *The test passed.* |
| 2 | *Test case defined:*  *Inputs: words at beginning and end of file*  *Expected output: file will be enabled if word is at beginning and/or end* | *This is an regular expected behavior.*  *The test passed.* |

**Time spent (in minutes):** 5

# Timing

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 155 |
| Impact Analysis | 12 |
| Actualization | 32 |
| Verification | 5 |
| Total | 204 |

# Reverse engineering

Diagram

Description automatically generated

# Conclusions

For this change the concept location was time consuming. It was confusing figuring out how the menu bar is created and therefore edited. Once it was established that there are two main properties files that control the contents of the menu bar it was a straightforward process. The concept location was done using Visual Studio Code, IntelliJ, and javax documentation. Impact analysis was straightforward as only a small section of independent code needed to be changed. The actualization was also easy as it was tested just by typing in strings that were in various positions of previously opened files in the recent files menu. We used the jEdit documentation and the regex documentation for the actualization as well.

Classes and/or methods changed:

* org/gjt/sp/jedit/menu/RecentFilesProvider.java/update/anonymous KeyAdapter/keyReleased
  + public void update(JMenu menu)

Links used:

* <https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html>
* <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html#package_description>
* <https://regexr.com/>
* <http://www.jedit.org/api/org/gjt/sp/util/StandardUtilities.html#globToRE-java.lang.String->
* <https://docs.oracle.com/javase/7/docs/api/javax/swing/JMenuBar.html>