Change request log

1. Team

Derik Flores and Joe Allam

2. Change Request

Change request #2: Change the highlighting method of the recent files search function to highlight all files containing the search query, rather than only files that start with the query.

3. Concept Location

Step #	Description	Rationale
1	Ran the system	
2	Looked in the browser folder	At a glance, the browser folder seemed to hold a few functions pertaining to the UI, and a few methods for opening files, so it might be a good place to look
3	Looked in BrowserCommandsMenu.java	Looked for browser commands such as “Recent Files”
4	Looked in GUI folder’s ActionBar class	The browser folder didn’t yield anything. Since the menu is a gui element, we decided to look in there instead.
5	Performed a search with keyword “recent” in main directory	Initial manual scanning wasn’t working, so we did a search with “recent” and found a file called RecentFilesProvider.java, which may play into the search function
6	We inspected the RecentFilesProvider.java	
7	We looked at update() in RecentFilesProvider.java	We saw that the update() method had a keyReleased listener, as well as a comment that mentioned “old style” searching that matched the start of the file name, as well as a pattern matcher that used the regex of the typed text.
8	Disabled the recent.setEnabled function	To test if we were in the right spot, and since the search function didn’t work after disabling the pattern matching, we identified this as the right spot to be modified.
9	We marked the class RecentFilesProvider and update() as "located".	We confirmed this class had to be modified.

Time spent (in minutes): 30

4. Impact Analysis

Step #	Description	Rationale
1	RecentFilesProvider.java, and the update() method had internal variables that did not affect other classes.	Since the variables like “pattern” are local to the update() function, and update handles UI

		output, changes to update() do not impact other parts of the system greatly.
2	update() is the only function that we need to change.	We needed to update the highlight portion of the class.
3		
4		

Time spent (in minutes): 20

5. Actualization

Step #	Description	Rationale
1	Commented out the previous regex and pattern matching system	Previously, the system handled the recent searches with a pattern matcher that looked at the beginning of the file. Due to this system being obsolete for the change we want to make, the code has been commented out in favor of a "contains" system
2	Added a line of code that highlighted file names based on if the inputted string exists in a given file name.	Since we only care to highlight files that contain the typed file, we can simplify the pattern matching system to use Java's String.contains() method to match searches.
3	We created unit tests for the new class and also performed functional testing. We also ran the existing test cases.	To make sure everything works.
4		

Time spent (in minutes): 10

6. Validation

Step #	Description	Rationale
1	Test case defined: search ".txt" when all recent files have .txt in the name, i.e. end of string searches Inputs: .txt Expected output: All files highlighted	This is the regular expected behavior. The test passed, highlighting all file names
2	Test case defined: search middle of file names Inputs: il Expected output: only output file.txt	The test passed, only enabling file.txt since it was the only file that contained "il" in its name.
3		
4		

Time spent (in minutes): 20

7. Timing

Summarize the time spent on each phase.

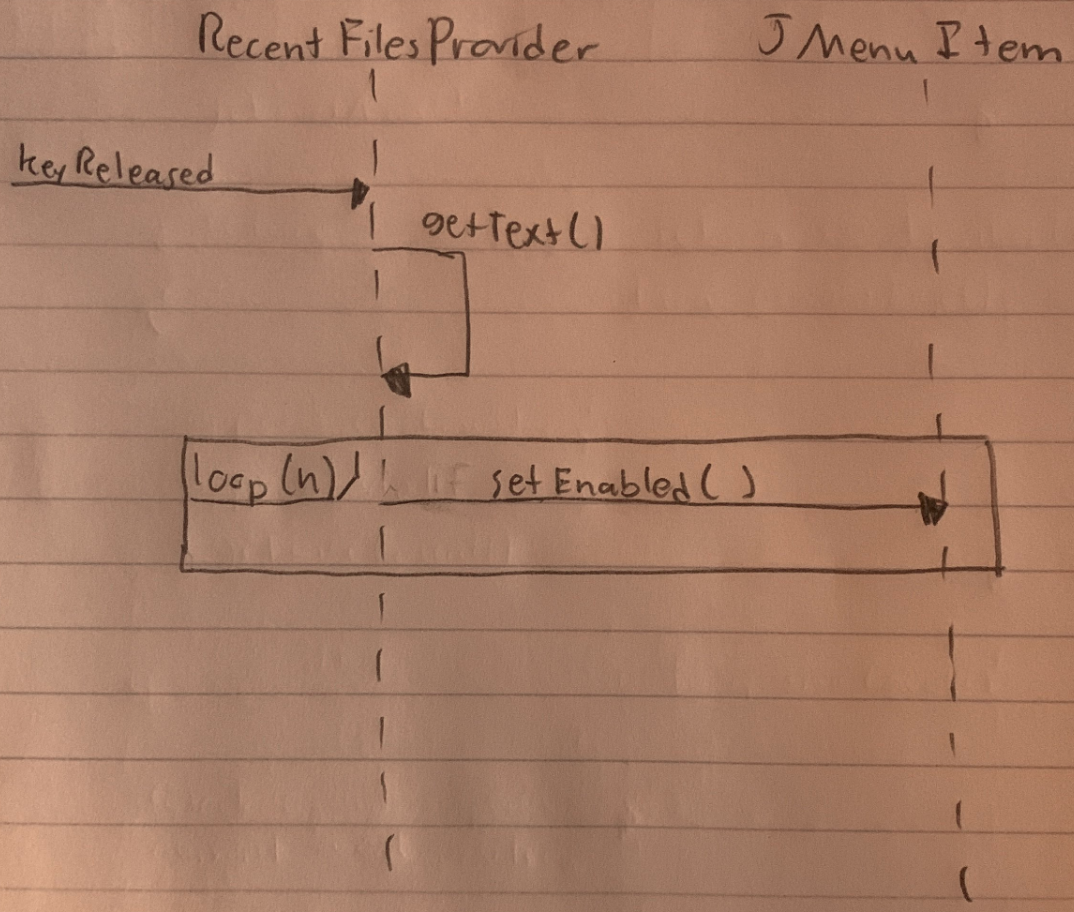
Phase Name	Time (in minutes)

Concept location	30
Impact Analysis	20
Actualization	10
Verification	20
Total	80

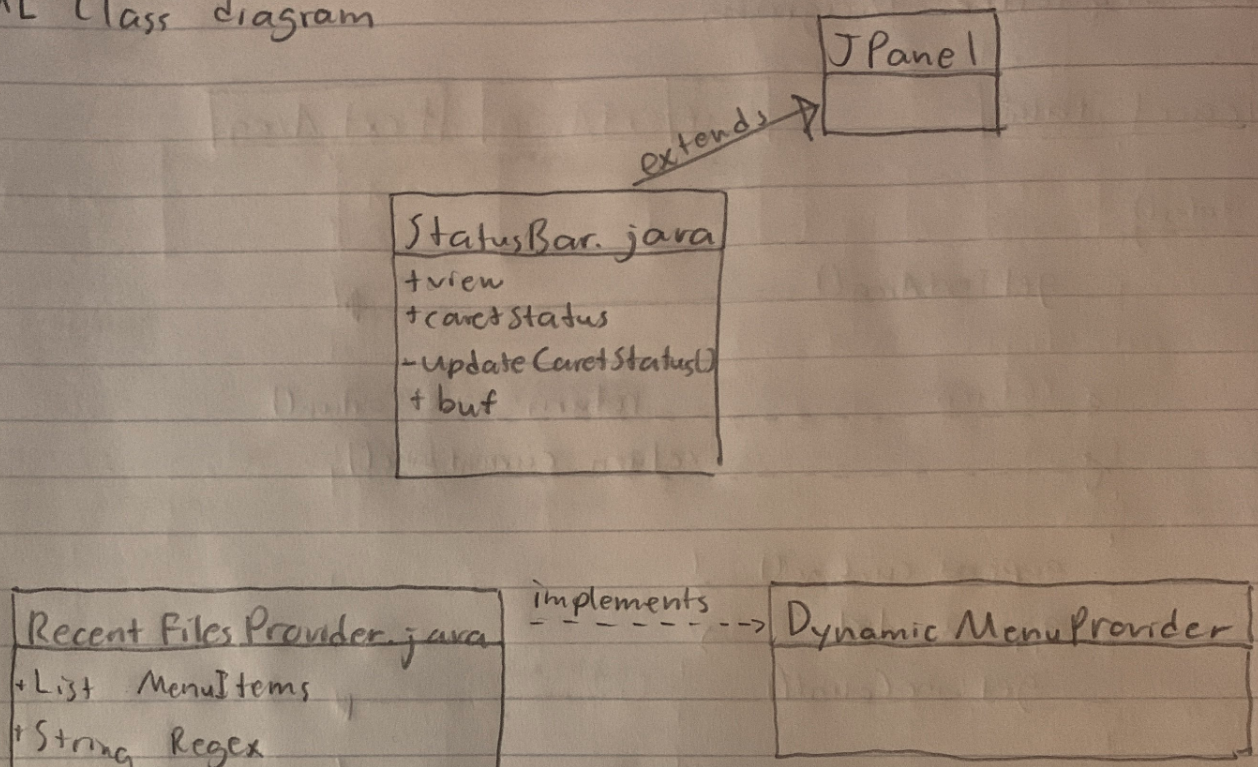
8. Reverse engineering

Create a UML sequence diagram (or more if needed) corresponding to the main object interactions affected by your change.

UML Sequence Diagram



UML Class diagram



9. Conclusions

For this change, though concept location was the hardest part due to red herrings, it was still relatively doable because of directory search functions that allowed for quicker identification. Impact analysis was also simple because it mostly managed output rather than affecting other systems directly. Testing took a little longer as we had to set up open files in order to test the "recent" function, but it was not tedious

Classes and methods changed:

- org/gjt/sp/jedit/menu/RecentFilesProvider.java
    - void update()