Change request log

1. Team

Team member: Antu & Johnny

Coder: Antu

Tracker: Johnny

2. Change Request

ID: jedit-cr-2

Description: In the File » Recent Files main menu of jEdit, the text box on top of the recent files list allows highlighting recent files names that match with a given string (see Figure 7). The string in the text box should match all the files that contain it anywhere in their name. However, the highlight works only when the string matches the beginning of a file name. You are requested to modify this feature so that the highlight occurs for the cases when the string is contained anywhere in the file name.

3. Concept Location

The table below describes each step we followed when performing concept location for this change request.

Step #	Description	Rationale
1	*We ran the system*	*to gain understanding of the system*
2	*open up a file, then open recent files and perform a search*	*to see how the feature works*
3	*global search for 'recent files'*	*see where this concept is in the code base*
4	*we find 'recentFilesProvider.java' file that contains mention of recent search as well as a textfield*	*to locate where in the file that mentions searching a text field*
5	*we learn that the code is using regular expressions to search for whether a phrase is contained within the list of recent file*	*observation*
6	*we locate the area in our concept location's file that is responsible for finding the regular expression within a search term (starting with)*	*to see how regex is working and hypothesize what change we need to make*
7	*we notice a line "pattern.matcher" that seems to be responsible for matching regex pattern to the entered search text*	*observation*
8	*we find a Java class named "PatternSearchMatcher"*	

Time spent (in minutes): 120

4. Impact Analysis

The table below describes each step we follow when performing an impact analysis for this change request.

Step #	Description	Rationale
1	*For the first solution, we tried different assignments for the regex variable.*	*To understand the impact of our changes.*
2	*We notice that for our first regex attempt, there is no pattern match found. (no highlighting occurred)*	*Observation.*
3	*For the second solution, we do not change the regex variable.* *Instead, we change* *pattern.matcher(recent.getText()).matches()* *to* *pattern.matcher(recent.getText()).find()* *This change has no effect on the overall application outside the desired outcome.*	*Observation.*
4	*For this case, there is no impact on other parts of the program.*	*Conclusion.*

Time spent (in minutes): 60

5. Prefactoring **(optional)**

The table below describes each step we follow to prefactor the code.

Step #	Description	Rationale
1	*We tried multiple different regex patterns within recentFilesProvider.java*	*To understand and refamiliarize ourselves with the behavior of regex pattern matching.*
2	*We first tried:* *regex = "*" + regex;*	*To observe the behavior of the variable regex*
3	*We then tried:* *regex = regex + "*" + "*";*	*To observe the behavior of the variable regex*
4	*We then tried:* *regex = "*" + regex + "*";*	*To observe the behavior of the variable regex*

Time spent (in minutes): 30

6. Actualization

The table below describes each step we followed when changing the code.

Step #	Description	Rationale
1	*We visit recentFilesProvider.java class.*	*This is where we first found a mention to our concept.*
2	*We looked at line 113, and observe the functionality* *regex = regex + "*";* *We see that this is building a string to find, in the recent files list. The pattern starts building from left to right*	*observation*
3	*We change this line to* *regex = "*" + regex + "*";*	*This allows the pattern to be contained anywhere in the name of the desired PDF to find.*
4	*The change request is now implemented. We can search for a pattern that is contained anywhere in the PDF title.*	*observation*
5	*We then notice a second way to make this change request in JEdit.*	*observation*
6	*We can keep (regex = regex + "*";) in place. We can then change line 126 pattern.matcher(recent.getText()).matches()* *to* *pattern.matcher(recent.getText()).find()*	*This allows the pattern to be contained anywhere in the name of the desired PDF to find.* *This is an alternative method to do the same thing as described in our previous solution above. We are including steps for both for completeness.*

Time spent (in minutes): 90

7. Postfactoring (optional)

The table below describes each step we followed to postfactor the code.

Step #	Description	Rationale
1	*We committed and pushed our changes with git.*	*Just in case we need to revert our changes.*

Time spent (in minutes): 10

8. Validation

The table below describes the validation activities (e.g., testing, code inspections, etc.) we performed for this change request.

Step #	Description	Rationale
1	*We manually tested the searching functionality on multiple PDF documents.*	*To verify the expected behavior and see if we can search for phrases contained anywhere in the PDF title.*
2	*We repeated this process multiple times across multiple computers.*	*To verify behavior.*
3	*We add and remove different files and test whether it is still working or not.*	*To verify behavior.*
4	*We found that our implementation is working without causing any issues.*	

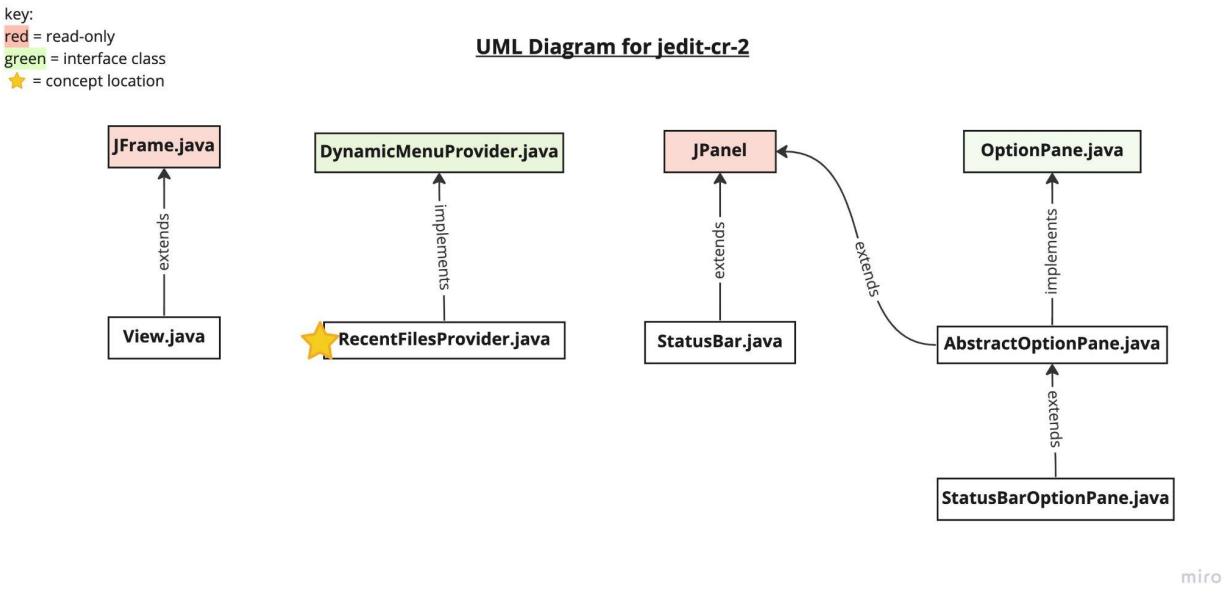
Time spent (in minutes): 60

9. Timing

The table shows the summary of the time we spent on each phase.

Phase Name	Time (in minutes)
Concept location	120
Impact Analysis	60
Prefactoring	30
Actualization	90
Postfactoring	10
Verification	60
Total	370

10. Reverse engineering



11. Conclusions

The concept location of this change request was not too difficult. We were able to find out the required file by searching with the keywords. Our plan was to understand how the highlighting operation in the recent files is implemented in the software. We noticed that when we are typing any character, a variable named ‘regex’ is created and the highlighting operation is performed with that keyword. This realization made our implementation easier. We implemented this change request in two different ways. Both of them are working perfectly without impacting other components of the software.