Jedit Change request log - 2

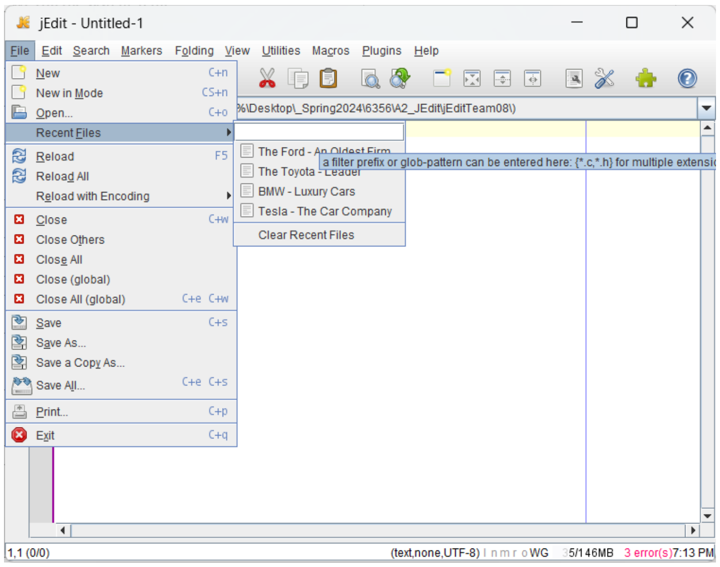
1. Concept Location

Step #	Description	Rationale
1	We ran the system, JEdit	We used IntelliJ to run the source code
2	We interacted with the system: We accessed Menu Bar.	To get familiar with some of the features of the system, and identify the screens or graphical elements we had to change.
3	We navigated to Search Field for Recent Files. Navigation: File>Recent Files	This was mentioned in the Change Request -2
4	We created four files with .txt extension	We saw a placeholder saying "No recent Files" Hence, we created files.
5	We made sure that all these four files have been named with single word, multi-word and with and without articles.	We need them in future for testing and validation
6	We also made sure that there are repetitive words in the file names.	We need them in future for testing and validation
7	We tested the existing functionality of Search and found that only file names that start with search word are being highlighted.	To get familiar with existing functionality, we needed to test this.
8	We performed a global search with 'a filter prefix or glob-pattern can be entered here', with these keys phrases and found the file jedit_en.props	We found a tooltip which says, "a filter prefix or glob-pattern can be entered here:{*.c,*.h} for multiple extensions"
9	In this file, on line 121, we saw this: recent-files.textfield.tooltip=a filter prefix or glob-pattern can be entered here	We started inspecting jedit_en.props as it was as the only result from global search.
10	We did a global search on, "recent-files.textfield.tooltip' variable	If there were no fruitful results, we planned to search for individual terms, i.e 'recent-files' (or) 'tooltip' (or) combination of them etc.
11	We found the java file, RecentFilesProvider.java among the 9 search results.	This is the only .java file among the search results.
12	On careful inspection, Update() is the method, where the change needs to be made.	We found the search ended up on line no. 93 which is in Update() method.
13	We marked class RecentFilesProvider as "located".	Hence we understood that we have located the concept in the class RecentFilesProvider

Time spent (in minutes): 120 minutes

Classes and methods inspected:

- jEditTeam08\org\jedit\localization\jedit_en.props
  - variable: defaultCatalog=<?xml….
- jEditTeam08\org\gjt\sp\jedit\menu\RecentFilesProvider.java
  - Class: public class RecentFilesProvider implements DynamicMenuProvider
    Methods:
  - public void update(JMenu menu),
  - public void keyReleased(KeyEvent e)



2. Impact Analysis

Step #	Description	Rationale
1	We made a list of methods called by RecentFilesProvider and classes interacting as well.	To track the methods and classes that could be impacted by the change.
2	We inspected the interface DynamicMenuProvider.	We realized this class had to be inspected because the RecentFilesProvider implements this class, and is the only class interacting with RecentFilesProvider.
3	The two methods found does the regular job of continually updating the search results and passes the JMenu object i.e, each time.	We needed to check what purpose they are serving.
4	The interface DynamicMenuProvider was discarded from the list of classes to change	Because these methods and class deals with expected behavior to continue, we did not change anything.
5	We inspected Update() method.	This is the method where the search result earlier was pointed to line no. 93
6	On analysing the code, this is the method where filtering and the highlighting of the files is happening and we need to make changes to Update() method itself.	We analysed and understood what is happening in those lines and the method.
7	Hence, we concluded only a method in RecentFilesProvider class needs to be changed	Because, crucial code for the search functionality was happening within a method of this class itself.

Time spent (in minutes): 30 minutes

Classes and methods inspected:

- jEditTeam08\org\gjt\sp\jedit\menu\
  Class: DynamicMenuProvider.java
- jEditTeam08\org\gjt\sp\jedit\menu\ DynamicMenuProvider.java
  Methods:
  - boolean updateEveryTime();
  - void update(JMenu menu);
- jEditTeam08\org\gjt\sp\jedit\menu\RecentFilesProvider.java

- Class:
    - public class RecentFilesProvider implements DynamicMenuProvider
- Method:
    - public void update(JMenu menu)
    - public void keyReleased(KeyEvent e)

3. Prefactoring (optional)

Step #	Description	Rationale
1	We performed initial inspection in the files to check if Prefactoring can be performed.	We still wanted to see if any pre-factoring can be done with respect to highlighting of file names. We did not find any serious code smell in the inspected class that needs immediate addressing.
2	No explicit Pre-factoring was performed. We made no changes with git.	Since, there was no scope found to perform Prefactoring.

Time spent (in minutes): 10 minutes

Classes and methods inspected:

- jEditTeam08\org\gjt\sp\jedit\menu\RecentFilesProvider.java
    - Class: public class RecentFilesProvider implements DynamicMenuProvider
    - Method:
        - public void update(JMenu menu)
        - public void keyReleased(KeyEvent e)

4. Actualization

Step #	Description	Rationale
1	We inspected the code in the class RecentFilesProvider and especially update() method.	We realized that the responsibility of the class is dealing with RecentFiles option in File menu.
2	The typed text in Search field is stored in typedText variable and the files are being highlighted based on the conditionals in filter variable, only if atleast 1 character is inserted	We understood the logic and code by careful inspection and drew insights.
3	Inside the if conditional for filter, we found regex variable, which denotes Regular Expression.	We analysed this, because we needed to know if highlighting is happening because of filter variable.
4	Previously: regex = regex + "*";	We understood highlighting of file names is happening based on the regular expression given.
5	Now: regex = "*" + regex + "*";	From our knowledge of using Regular expressions, the asterisk (*) is a quantifier that specifies zero or more occurrences of the preceding character or group.
6	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.
7	We modified the comments accordingly as well	Because comments help us in concept location and promote readability

8	Now: // New style (after jEdit 4.3pre18): Match any part of the file name	Previously: // Old style (before jEdit 4.3pre18): Match start of file name

Time spent (in minutes): 45 minutes

Classes and methods inspected:
- jEditTeam08\org\gjt\sp\jedit\menu\RecentFilesProvider.java
    - Class: public class RecentFilesProvider
    - Method:
        - public void update(JMenu menu)
        - public void keyReleased(KeyEvent e)

Classes and methods changed:
- jEditTeam08\org\gjt\sp\jedit\menu\RecentFilesProvider.java
    - Method:
        - public void update(JMenu menu)
        - public void keyReleased(KeyEvent e)

5. Postfactoring (optional)

Step #	Description	Rationale
1	We performed a final inspection in the files to check if Postfactoring can be performed.	Because we need to take an eagle view of all files involved.
2	We identified that code inside public void keyReleased(KeyEvent e) can be refactored.	Because this is where we are making change, we focused mainly on this method.
3	BEFORE: boolean filter = (typedText.length() > 0);	Takes time to use length() since it needs to compute length of entire string searched. Not optimal in cases with very long search string.
4	AFTER: boolean filter = (!typedText.isEmpty());	More readable to use NOT and promotes clean coding. Also easy to compute a Boolean.

5	BEFORE:
	
	for (JMenuItem recent : menuItems)
	{
	recent.setEnabled(filter ?
	pattern.matcher(recent.getText()).matches()
	: true);
	}
		Using Ternary Conditional Operator can be complex to understand

6	AFTER:
	
	for (JMenuItem recent : menuItems)
	{
	
	recent.setEnabled(!filter ||
	pattern.matcher(recent.getText()).matches());
	}
		Just an OR conditional is easy to compute and adds more readability

7	We found that we could use Extract Method to form a new class to encapsulate the filter and regex information.	Because there is code for filter, regex, and try and catch block within same method
8	We avoided using Extract Method to form a new class to encapsulate the filter and regex information.	Because this method is not doing other than filtering and highlighting, we didn't want to violate SRP – Single Responsibility Principle as well. Performing Extract method would only cause Over-factoring and is not ideal.
9	After the previous change, we ran the unit tests corresponding to the class Schedule and also we ran the system.	We tested everything was working as before, after the refactoring.
10	We committed and pushed our changes with git.	Just in case we need to revert our changes.

Time spent (in minutes): 30 minutes

Classes and methods inspected:

- jEditTeam08\org\gjt\sp\jedit\menu\RecentFilesProvider.java
    - Class: public class RecentFilesProvider
    - Method:
        - public void update(JMenu menu)
        - public void keyReleased(KeyEvent e)

Classes and methods changed:

- jEditTeam08\org\gjt\sp\jedit\menu\RecentFilesProvider.java
    - Method:
        - public void keyReleased(KeyEvent e)

6. Validation

Performed Manual Testing, by compiling and running the new code.

Step #	Description	Rationale
1	Test case defined: Search 'the' – small case Inputs: the Expected output: All file names with 'the' included to be highlighted	This is the regular expected behavior. The test passed.
2	Test case defined: Search 'The' – case Insensitivity Inputs: The Expected output: All file names with 'the' included to be highlighted	This is the regular expected behavior. The test passed.
3	Test case defined: Search 'car' – middle word in file name Inputs: car Expected output:	This is the required behavior and we have achieved it. The test passed.
4	Test case defined: Search 'firm' – last word in file name Inputs: firm Expected output:	This is the regular expected behavior. The test passed.
5	Test case defined: Search 'latest' – word not present in any file name Inputs: latest Expected output:	This is the regular expected behavior. The test passed.

6	Test case defined: Search ’-’ – Special character search Inputs: **-** Expected output:	This is the regular expected behavior. The test passed.

Time spent (in minutes): 10 minutes

7. Summary of the change request

Phase	Time (minutes)	No. of classes inspected	No. of classes changed	No. of methods inspected	No. of methods changes
Concept location	120	1	0	1	0
Impact Analysis	30	2	0	4	0
Prefactoring	10	1	0	2	0
Actualization	45	1	1	2	1
Postfactoring	30	1	1	2	1
Verification	10	1	0	1	0
Total	245				

8. Conclusions

Our team successfully implemented Change Request #2 for the jEdit application, enhancing the File » Recent Files feature to highlight file names containing the search string anywhere within the name rather than just at the beginning. Through a meticulous Concept Location, Impact Analysis, Prefactoring, Actualization, and Validation process focused on the RecentFilesProvider's Update() method, we updated the regular expression to enable this improved highlighting functionality. Manual testing confirmed that various test cases now pass as expected, validating the change's alignment with user expectations for more efficient navigation and identification of relevant files. With careful consideration for minimizing impact on existing code, we are confident this localized enhancement will contribute to a smoother overall user experience in jEdit.