Change request log

1. Team

Joshua – Driver

Supraj – Navigator

2. Change Request

Change Request #2: fixing recent file search so that recent files are highlighted when the searched text appears anywhere in the file name

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	We ran the system	
2	Under File>Recent Files, we hovered over the search bar and read the tool tip “a filter prefix or glob-pattern…”	Similar to our rationale with CR 1, we figured that the string in this tool tip would lead us to where we needed the change
3	Searching project files for “a filter prefix or glob” lead us to the string “recent-files.textfield.tooltip” in jedit_en.props	Now that we have found a variable name for this string we can use it to find files associated with this search bar
4	Searching project files for “recent-files.textfield.tooltip”, we found RecentFilesProvider.java	The name of this class seems to suggest that this may be where we need to make some changes

Time spent (in minutes): 40

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.

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 details 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	We made a list of the methods in RecentFilesProvider.jave – there were only 2: updateEveryTime() and update()	Since we'll be changing one of these methods, we need to check where these methods are called
2	Right clicking each method and selecting "Find Usages", we see that updateEveryTime() and update() are each only called once in EnhancedMenu.init()	We figured that looking at where these methods were called would help us determine whether or not changing either of these methods would have impacts elsewhere in the system
3	We determined that making changes to updateEveryTime() could affect how EnhancedMenu.init() would run	Because updateEveryTime() is a boolean method called within a conditional expression, parts of EnhancedMenu.init() run or don't run depending on what this method returns
4	We determined that making changes to update() would be safe and not impact EnhancedMenu.init()	

Time spent (in minutes): 15

5. 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.

Step #	Description	Rationale
1	Reading through update() in RecentFilesProvider.java, we came across line 100 that read "String regex = typedText;"	Since we knew we were dealing with a search bar, it made sense that we were looking to make a change somewhere that involved typed text. We also knew that regular expressions are used to search through chunks of text.
2	Editing this regular expression to include wildcards before and after the typed text ensures that the search will find file names with the typed text located anywhere in the file name.	

Time spent (in minutes): 30

6. 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.

Step #	Description	Rationale
1	Test case defined: We opened 3 files, named file 1, this is a file 2, this is not a fil 3 (file spelled incorrectly on purpose for file 3) In the search bar we entered fil	This is the regular expected behavior. all 3 should be highlighted The test passed.
2	Test case defined: We opened 3 files, named file 1, this is a file 2, this is not a fil 3 (file spelled incorrectly on purpose for file 3) In the search bar we entered file	Expected Behavior: files 1 and 2 should be highlighted, file 3 should not be highlighted The test passed.

Time spent (in minutes): 10

7. Timing

Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	40
Impact Analysis	15
Actualization	30
Verification	10
Total	95

8. Reverse engineering

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

Create a partial UML class diagram of the classes visited while navigating through the code. Include the associations between classes (e.g., inheritance, aggregations, compositions, etc.), as well as the important fields and methods of each class that you learn about. The diagram may have disconnected components. Use the UML tool of your preference. When a significant fact about a class or method is learned, indicate it via annotations on the diagram. **For each change request, start with the diagram produced in the previous change request. For the first, you will start from scratch.**

9. Conclusions

For this change, locating where the change needed to be made was probably the most challenging part of the code. There were many class names that it could have been, but ultimately, we found it to be the RecentFilesProvider.java class. The only thing that was changed was a variable, so implementing the change was very easy. We tested the code by introducing multiple file names and checking to see if the search bar looks at the entire string.

Classes and methods changed:

- org/gjt/sp/jedit/RecentFilesProvider.java/update()
    - regex variable adjusted to search entire string