Change request log

Team
gitrams

Change Request

Change Request ID: #1 (JEdit)

Description of the change request: "In the File » Recent Files main menu of JEdit, the text box on top of the recent files list allows to highlight recent files names that match with a given string. 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."

Concept Location

- IDE Features used: **InstaSearch** and basic Search in Eclipse
- Queries used when searching: "Recent Files"
- Interactions with the system:
- Classes visited:
- The first class found to be changed (this is when concept location ends):

Step #	Description	Rationale
1	*We compiled and ran JEdit-5-4-0*	
2	*We interacted with Recent File Menu function of JEdit to understand how an input keyword is searched in the recent files names.*	*To get familiar with the existing features such as keyword search in Recent Files Menu. This is because, we wanted to change this functionality to search the keyword in the entire name and not just at the beginning of the file name.*
3	*We searched for "Recent Files" using the InstaSearch plugin installed in Eclipse Oxygen2*	*Because we wanted to modify the keyword search functionality under recent files menu.*
4	*We received 2 relevant results: the classes were* `RecentFilesProvider and GeneralOptionPane`*. The classes were inspected using the Eclipse IDE.*	*We wanted to find the place in the code where search was implemented.*
5	*We went through the class* `RecentFilesProvider and GeneralOptionPane` *to find out that search was implemented under public* `void update(JMenu menu) method` *of the* `RecentFilesProvider class.`	
6	*We noted that* `GeneralOptionPane` *Class was not relevant to our change request.*	*Because it was related to recent files menu panel.*
7	*We located the search functionality in the* `RecentFilesProvider` *class and decided to modify the 'regex' string that carried out the keyword search.*	*Changing the regex string would allow keyword to be searched within the entire name of the file.*
8	*This will require us to add another wild character (*) before the regex string.*	*We confirmed that the class* `RecentFilesProvider` *had to be modified.*

Time spent (in minutes): 65

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 looked for all classes related to RelatedFilesProvider class*	*To find classes that could be impacted by the change.*
2	*We inspected the interface DynamicMenuProvider and the class EnhancedMenu.*	*We realized these classes need not to be changed because the behavior of our target class was not changing these.*
3		
4		

Time spent (in minutes): 20

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	*We decided to add another wild character (*) before the 'regex' string inside the public* `void update()` *method in* `the RecentFilesProvider class.`	*This will allow a search hit if the keyword appears anywhere within the file name to be searched.*
2		

Time spent (in minutes): 20

Postfactoring (optional)

For this change request we did not require to postfactor any part of the code base except for adding a few comments. The changes that were made for this change request were isolated, few, and had very little impact on other classes. So postfactoring was not required except for adding comments for the purspose of documentation.

Step #	Description	Rationale
1	*Added comments regarding the newly implemented change*	*We wanted to keep a record/documentation in the form of comments in the source code.*

Time spent (in minutes): 10

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	*We performed manual testing by running the built software.*	*All the manual tests passed.*
2	*We opened a bunch of files to populate the recent files history.* *Then go to the File menu, select Recent Files.* *Search for a string.*	*If the highlighted filenames match the string, where the string is contained anywhere in the file name, the test is passed.*

Time spent (in minutes): 15

Timing

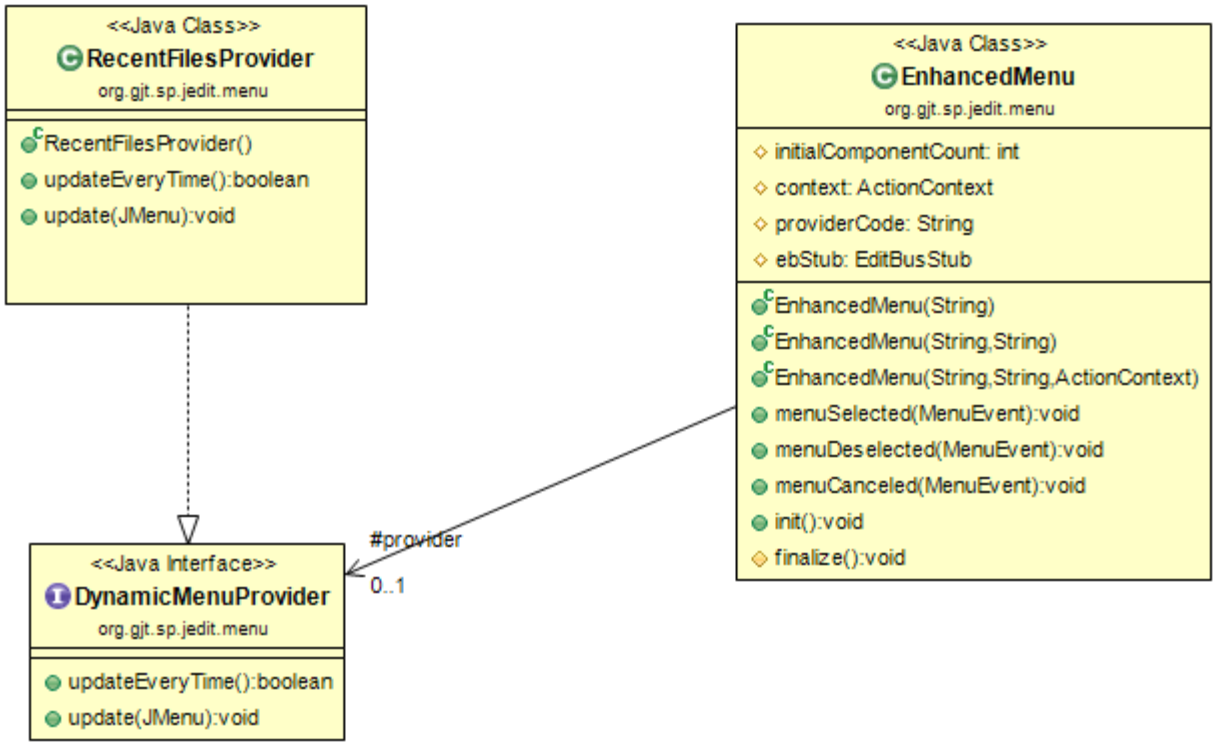
Summarize the time spent on each phase.

Phase Name	Time (in minutes)
Concept location	65
Impact Analysis	20
Prefactoring	0
Actualization	20
Postfactoring	10
Verification	15
Total	130

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.**



Conclusions

For this change, building and setting up the project in Eclipse was the hardest. The code was getting built fine on the command line (using the ant command) but it was not building in Eclipse. That took most of the time. Secondly the concept location was a bit tough too, because I did it for the first time. Testing was performed manually because it was difficult to add a test suite.

Classes and methods changed:

org/gjt/sp/jedit/memu/RecentFilesProvider
    void update(JMenu menu)