Change Request #2 Log: (Recent Files Main Menu of jEdit)

Team 18: Teena Thomas, Nazeera Siddiqui, and John Scalley

Scripter	Coder
John Scalley	Teena Thomas

Table of Contents

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*	*To initiate the change request process*
2	*We interacted with the system*	*To get familiar with some of the features of the system, and identify the screens or graphical elements we had to change.*
3	*Look at files in org.gjt.sp.jedit.menu folder*	*In the change request instructions it mentions "main menu" and the naming convention of the folder matched the change request.*
4	*Looked in* RecentFilesProvider.java	*Naming conventions match those in change request*
5	*Interacted with source code in current file*	*To get familiar with the specific class we think we will be changing and because it was short enough*
6	*Focused on update method*	*Contains comment pertaining to number of documents in input*
7	*Focused on keyReleased method within the update method*	*Contained comments pertaining to pattern matching and location to make change found*

Time spent (in minutes): 61

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 detail 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	*Inspected RecentFilesProvider class*	*Regex to be changed contained in this class*
2	*Make list of areas Regex variable is used*	*Location of change in in Regex variable assignment.*
3	*Inspected globToRE method call*	*Uses REGEX variable*
	Removed globToRE from impact set	*REGEX variable still valid regular expressions to be used.*

Time spent (in minutes): 35

5.Prefactoring (optional)

Using the table below, describe each step you follow to prefactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that were modified, added, removed, renamed, etc.

Step #	Description	Rationale
1	*N/A*	*N/A*

Time spent (in minutes): 0

6.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	*Added * wildcard to beginning of REGEX*	*Allow to match with any string preceding the input string*
2		
3		

Time spent (in minutes): 15

7.Postfactoring (optional)

Use the table below to describe each step you followed to postfactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that were modified, added, removed, renamed, etc.

Step #	Description	Rationale
1	*We committed and pushed our changes with git.*	*Just in case we need to revert our changes.*
2	*...*	

Time spent (in minutes): 17

8.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:*** *Input string at beginning of recent file* ***Inputs:*** *string at beginning of recent file* ***Expected output:*** *highlight of specified file*	*This is the regular expected behavior.* *The test passed.* *Did not change old behavior*
2	***Test case defined:****Input string in middle of recent file* ***Inputs:*** *string in middle of recent file* ***Expected output:****highlight of specified file*	*This is the regular expected behavior.* *The test passed.* *Did not change old behavior*
3	***Test case defined:****Input string at end of recent file* ***Inputs:*** *string at end of recent file* ***Expected output:****highlight of specified file*	*This is the regular expected behavior.* *The test passed.* *Did not change old behavior*
4	***Test case defined:****Input string not in recent file* ***Inputs:*** *random string* ***Expected output:*** *highlight of no files*	*This is the regular expected behavior.* *The test passed.* *Did not change old behavior*

Time spent (in minutes): 63

9.Timing

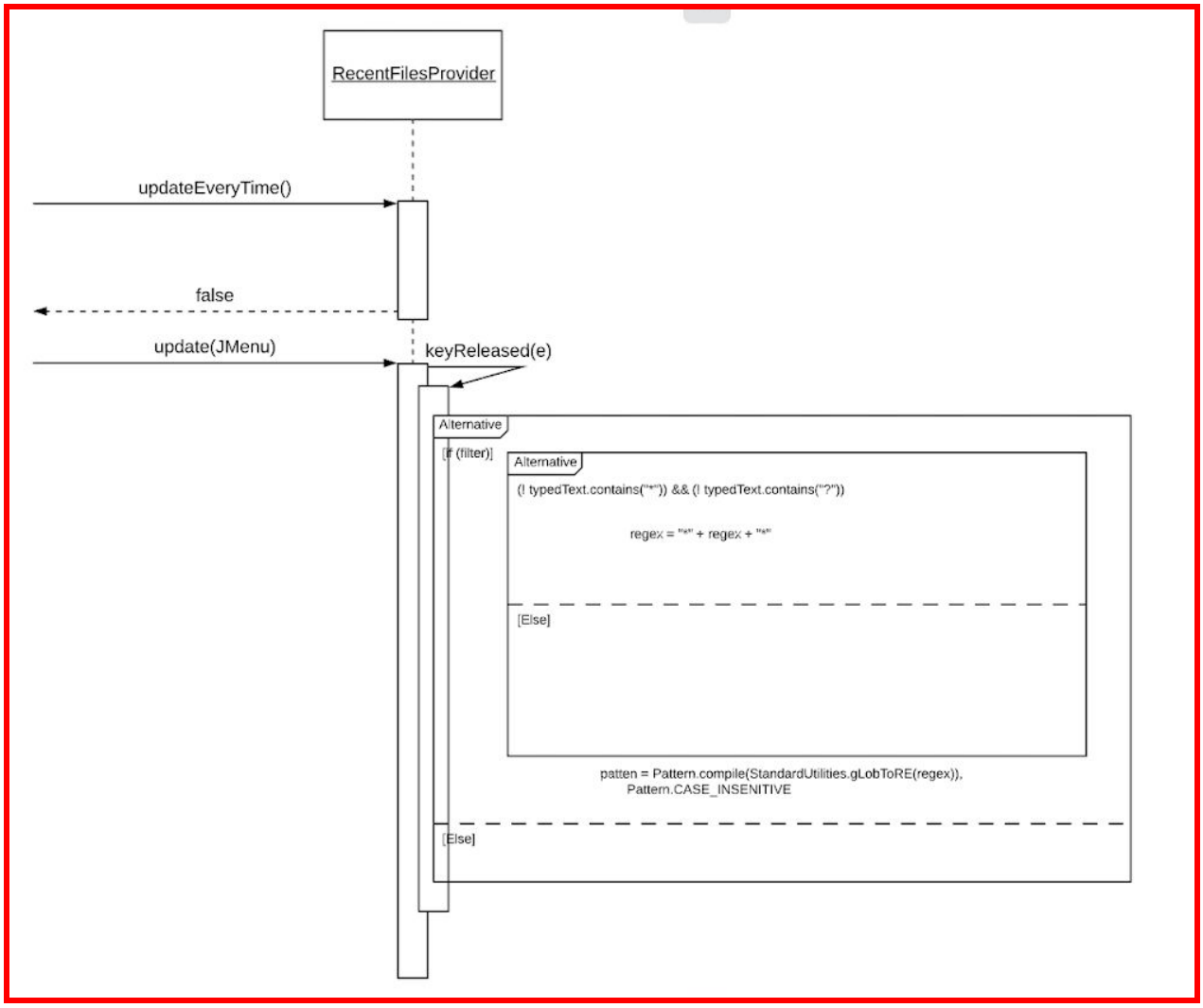
Summarize the time spent on each phase.

Phase Name	**Time (in minutes)**
Concept location	61
Impact Analysis	35
Prefactoring	0
Actualization	15
Postfactoring	17
Verification (Validation)	63
Total	**191**

10.Reverse engineering

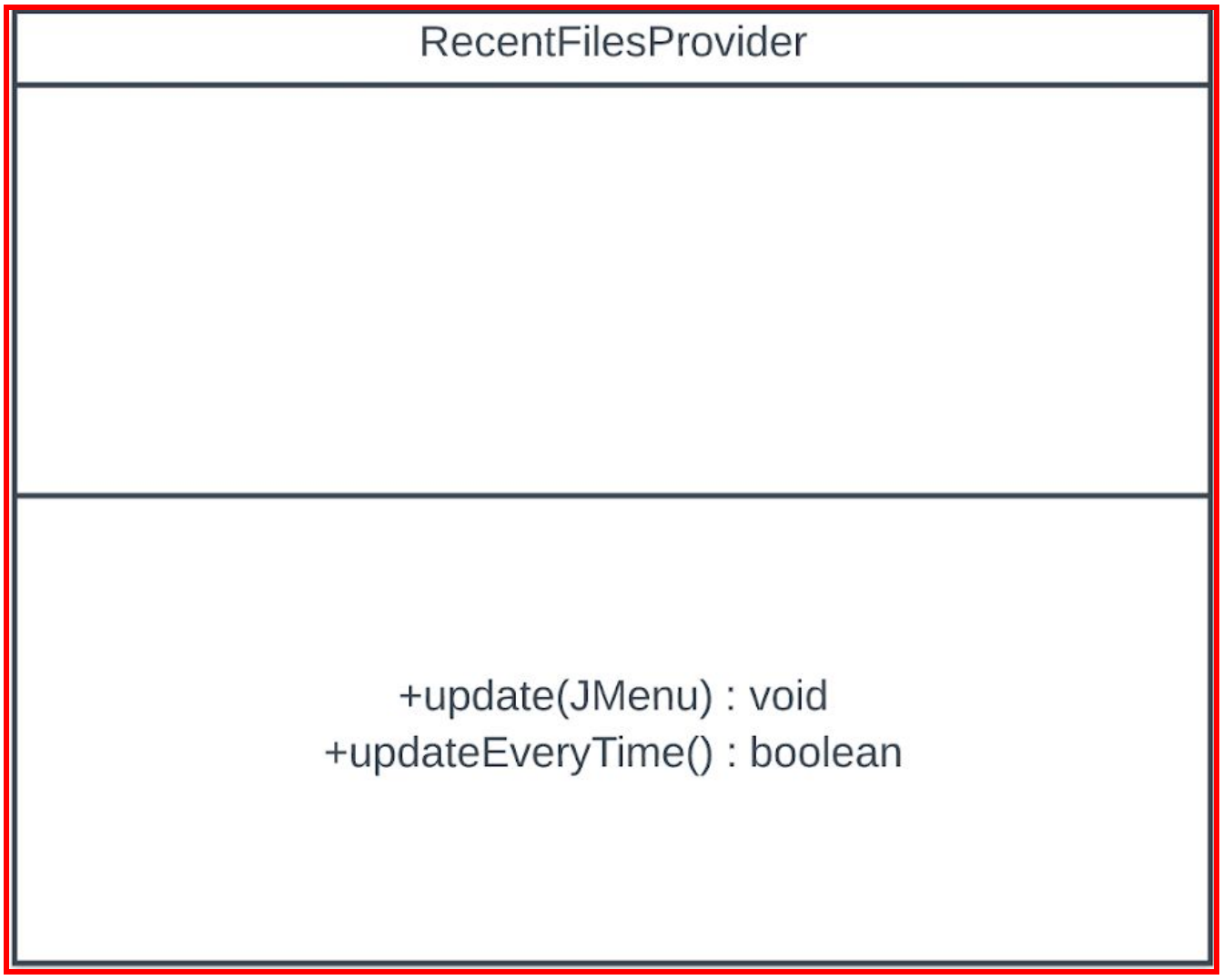
UML Sequence Diagram

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



UML Class Diagram

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.**



11.Conclusions

Perform and analysis of the change requests and the change process. List the major challenges this change request posed.

- The biggest problem we faced was in trying to locate where to make the change. There were many files and navigating through them was a bit challenging. In order to build JEdit on my device we had to install Apache Ant 1.9.14 and JDK 1.8.0_22. Although, we had installed the described software, we were not able to run JEdit on my computer due to some unknown issues. As a result of the previously described issues, I used John's computer to run the changes we made to JEdit. Once the location of the change was found the actual implementation of the change was simple. Concept location, impact analysis, actualization (and change propagation) was done manually. Testing could not be performed using JUnit and Abbot.

List all the classes and methods you have changed.

- public class RecentFilesProvider implements DynamicMenuProvider
- public void update(JMenu menu)

Classes and methods changed (Path):

- jEdit\WorkingTree\org\gjt\sp\jedit\menu\RecentFilesProvider.java
- jEdit\WorkingTree\org\gjt\sp\jedit\menu\RecentFilesProvider.java\update(JMenu menu)

Changes made to RecentFilesProvider.java :

```
public void update(JMenu menu)
	{
		final View view = GUIUtilities.getView(menu);

		//{{{ ActionListener...
		ActionListener actionListener = new ActionListener()
		{
			public void actionPerformed(ActionEvent evt)
			{
				jEdit.openFile(view,evt.getActionCommand());
				view.getStatus().setMessage(null);
			}
		}; //}}}

		//{{{ ChangeListener...
		ChangeListener changeListener = new ChangeListener()
		{
			public void stateChanged(ChangeEvent e)
			{
				JMenuItem menuItem = (JMenuItem) e.getSource();
```

```
view.getStatus().setMessage(menuItem.isArmed()?menuItem.getActionCommand():null);
                        }
                }; //}}}

                List<BufferHistory.Entry> recentVector = BufferHistory.getHistory();

                if(recentVector.isEmpty())
                {
                        JMenuItem menuItem = new JMenuItem(
                                jEdit.getProperty("no-recent-files.label"));
                        menuItem.setEnabled(false);
                        menu.add(menuItem);
                        return;
                }

                final List<JMenuItem> menuItems = new ArrayList<JMenuItem>();
                final JTextField text = new JTextField();
                text.setToolTipText(jEdit.getProperty("recent-files.textfield.tooltip") +
                        ": " + jEdit.getProperty("glob.tooltip"));
                menu.add(text);
                text.addKeyListener(new KeyAdapter()
                {
                        public void keyReleased(KeyEvent e)
                        {
                                String typedText = text.getText();
                                boolean filter = (typedText.length() > 0);
                                Pattern pattern = null;
                                if (filter)
                                {
                                        String regex = typedText;
                                        if ((! typedText.contains("*")) && (! typedText.contains("?")))
                                        {
                                                // Old style (before jEdit 4.3pre18): Match start of file name
                                                regex = "*"+ regex + "*";
                                        }
                                        pattern = Pattern.compile(StandardUtilities.globToRE(regex),
                                                Pattern.CASE_INSENSITIVE);
                                }
                                try
                                {
                                        for (JMenuItem recent : menuItems)
                                        {
                                                recent.setEnabled(filter ?
                                                        pattern.matcher(recent.getText()).matches() : true);
                                        }
                                }
                                catch(PatternSyntaxException re)
                                {
                                  Log.log(Log.ERROR,this,re.getMessage());
                                }
                        }
                });

                boolean sort = jEdit.getBooleanProperty("sortRecent");

                int maxItems = jEdit.getIntegerProperty("menu.spillover",20);

                Iterator<BufferHistory.Entry> iter = recentVector.iterator();
                while(iter.hasNext())
                {
                        String path = iter.next().path;
                        if (jEdit.getBooleanProperty("hideOpen") && jEdit.getBuffer(path) != null)
                                continue;
                        JMenuItem menuItem = new JMenuItem(MiscUtilities
                                .getFileName(path));
```

```
                        menuItem.setToolTipText(path);
                        menuItem.setActionCommand(path);
                        menuItem.addActionListener(actionListener);
//                      menuItem.addMouseListener(mouseListener);
                        menuItem.addChangeListener(changeListener);

                        menuItem.setIcon(FileCellRenderer.fileIcon);

                        menuItems.add(menuItem);
                        if (!sort)
                        {
                                if (menu.getMenuComponentCount() >= maxItems
                                    && iter.hasNext())
                                {
                                        JMenu newMenu = new JMenu(
                                                        jEdit.getProperty("common.more"));
                                        menu.add(newMenu);
                                        menu = newMenu;
                                }

                                menu.add(menuItem);
                        }
                }

                if(sort)
                {
                        Collections.sort(menuItems, new MenuItemTextComparator());
                        for(int i = 0; i < menuItems.size(); i++)
                        {
                                if(menu.getMenuComponentCount() >= maxItems
                                        && i != 0)
                                {
                                        JMenu newMenu = new JMenu(
                                                jEdit.getProperty("common.more"));
                                        menu.add(newMenu);
                                        menu = newMenu;
                                }

                                menu.add(menuItems.get(i));
                        }
                }
                JMenuItem menuItem = new JMenuItem(jEdit.getProperty("clear-recent-files.label"));
                menuItem.addActionListener(new ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                BufferHistory.clear();
                        }
                });
                menu.addSeparator();
                menu.add(menuItem);
        } //}}}
}
```

Reference

"Java - Regular Expressions." *Tutorialspoint*, TutorialsPoint, www.tutorialspoint.com/java/java_regular_expressions.htm.

Jenkov, Jakob. "Java Regex - Java Regular Expressions." *Jenkov.com*, Jakob Jenkov, 3 May 2019, tutorials.jenkov.com/java-regex/index.html.

"UML 2 Class Diagrams: An Agile Introduction." *UML 2 Class Diagrams: An Agile Introduction*, Agile Modeling, agilemodeling.com/artifacts/classDiagram.htm.

"Unified Modeling Language (UML): Sequence Diagrams." *Unified Modeling Language (UML) | Sequence Diagrams*, GeeksforGeeks , 12 Feb. 2018, www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/.

Vogel, Lars. "Regular Expressions in Java - Tutorial." *Vogella.com*, Vogella, 2019, www.vogella.com/tutorials/JavaRegularExpressions/article.html.