Names and NSIDs: Ruida Xie (rux793); Hao Li (hal356); Yinsheng Dong (yid164)

Project: jEdit

Change Number: [7]

Date: 04/02/2018

**Search List**

1. **Change Request:**

Currently jEdit allows users to access the text that was previously search by pressing page\_up or right-click keys in Search Dialog. Display in a listbox the last 5 text fragments that were previously search.

1. **Concept Location:**

**Table 1. The list of all the classes visited during concept location.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **File name** | **Tool used** | **Located?** | **Comments** |
| 1 | SearchDialog.java | Eclipse Searching for “Search Dialog” | Changed | This file is what I actually want to change, because the search list should be added in this dialog, so I should create a search list and added it to the panel |
| 2 | SearchAndReplace.java | Reading | Propagating | This class implemented regular expression and literal search within JEdit buffers |
| 3 | SearchBar.java | Reading | Unchanged | This is incremental search bar, so it should be unrelated with what we need |
| 4 | SearchMatcher.java | Reading | Unchanged | This is an abstract class for matcher the searching, but it is not related with what we want to do. |

1. **Impact Analysis:**

**Table 2. The list of all the classes visited during impact analysis.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Class name** | **Tool used** | **Impacted?** | **Comments** |
| 1 | SearchDialog.java | Eclipse (cmd+shift+T) Searching for “Search Dialog” | Yes | This dialog will be added a search list |
| 2 | SearchAndReplace.java | Reading | No | The function that I created will using search and replace method, but nothing will be impacted |
| 3 | SearchBar.java | Reading | No | This is an unrelated class |
| 4 | SearchMatcher.java | Reading | No | This is an unrelated class |

1. **Learning process:** *Include a partial UML* ***class diagram.*** *The class diagram must contain all the classes that you visited and all the relations (dependencies) between the classes that you understood. (see Figure 1). Mark the classes that were changed with red text as in the example.*

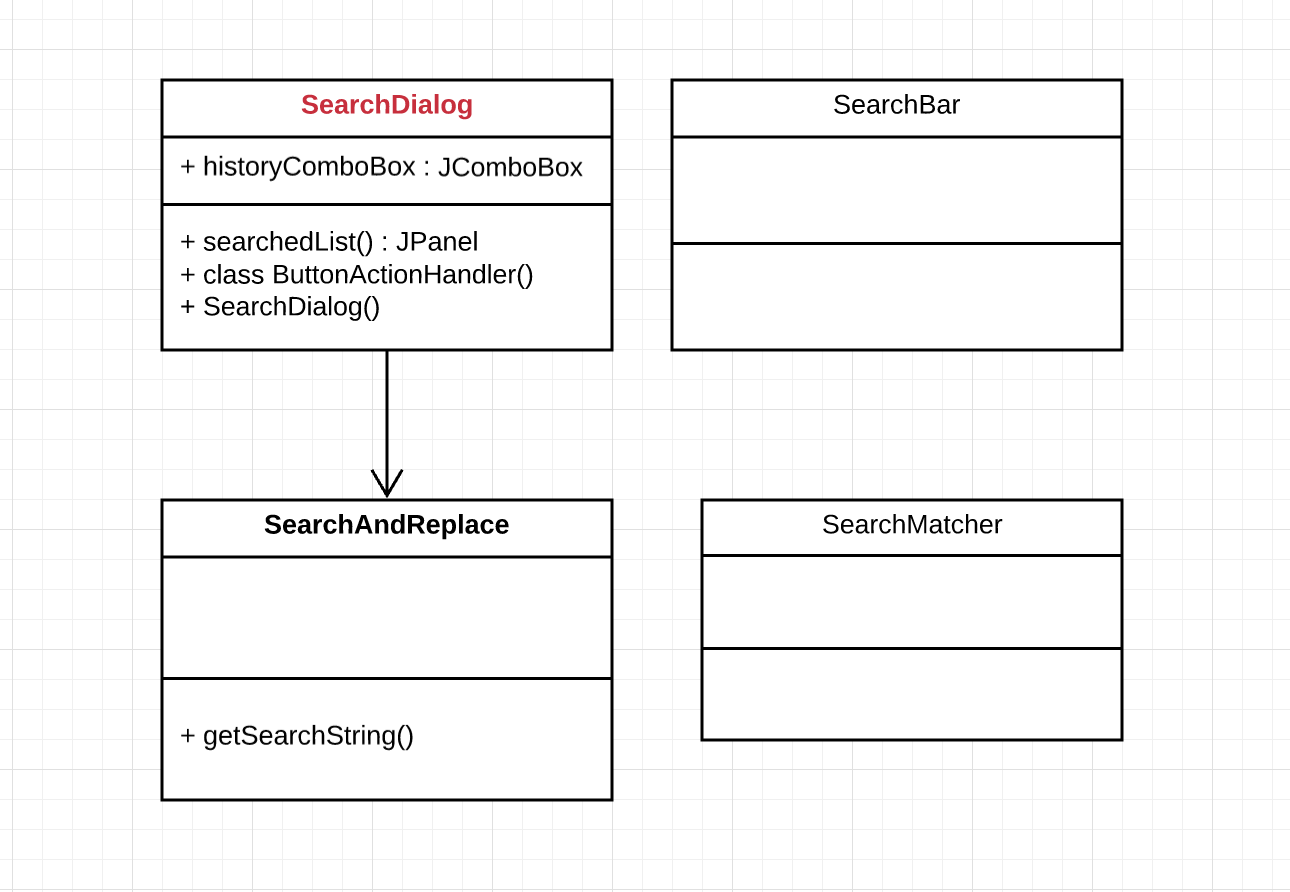
****

Figure class diagram

1. **Description of the implementation:**

I created new JComoBox global variable in SearchDialog class, and make initialization in the searchedList() method

I made some modifications in the ButtonActionHandler() inner class, actually add searched list as a parameter in that class, so it can change every time

I added searchedList() to the SearchDialog (modified)

1. **Sources:**
   1. <https://docs.oracle.com/javase/tutorial/uiswing/events/actionlistener.html>
   2. <https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html>
2. **Highlighted Source Code:**

<SearchDialog.java>

private JComboBox historyComboBox;

/\*\*

\* THe panel that showing the history list

\* @return JPanel of searchedList

\*/

public JPanel searchedList()

{

JPanel panel = new JPanel();

JLabel label = new JLabel("Searched List: ");

historyComboBox = new JComboBox();

panel.add(label);

panel.add(historyComboBox);

return panel;

}

//{{{ SearchDialog constructor

/\*\*

\* Creates a new search and replace dialog box.

\* @param view The view

\*/

private SearchDialog(View view)

{

super(view,jEdit.getProperty("search.title"),false);

this.view = view;

JPanel content = new JPanel(new BorderLayout());

content.setBorder(new EmptyBorder(0,12,12,12));

setContentPane(content);

JPanel centerPanel = new JPanel(new BorderLayout());

centerPanel.add(BorderLayout.CENTER,createFieldPanel());

centerPanel.add(BorderLayout.SOUTH,createSearchSettingsPanel());

content.add(BorderLayout.CENTER,centerPanel);

content.add(BorderLayout.SOUTH,createMultiFilePanel());

content.add(BorderLayout.EAST,createButtonsPanel());

// add the search history panel to the dialog

content.add(BorderLayout.AFTER\_LAST\_LINE,searchedList());

pack();

jEdit.unsetProperty("search.width");

jEdit.unsetProperty("search.d-width");

jEdit.unsetProperty("search.height");

jEdit.unsetProperty("search.d-height");

GUIUtilities.loadGeometry(this,"search");

load();

EditBus.addToBus(this);

} //}}}

//{{{ ButtonActionHandler class

class ButtonActionHandler implements ActionListener

{

public void actionPerformed(ActionEvent evt)

{

Object source = evt.getSource();

if(source == closeBtn)

cancel();

else if(source == findBtn || source == find

|| source == replace)

{

ok();

//only in this situation, the search list will store the search string

int i = historyComboBox.getItemCount();

//size is not larger than 5

if(i>=5)

{

String a = (String)historyComboBox.getItemAt(1);

String b = (String)historyComboBox.getItemAt(2);

String c = (String)historyComboBox.getItemAt(3);

String d = (String)historyComboBox.getItemAt(4);

//historyComboBox.repaint();

historyComboBox.removeAllItems();

historyComboBox.addItem(a);

historyComboBox.addItem(b);

historyComboBox.addItem(c);

historyComboBox.addItem(d);

// add the item from SearchAndReplace class

historyComboBox.addItem(SearchAndReplace.getSearchString());

}else {

historyComboBox.addItem(SearchAndReplace.getSearchString());

}

}

else if(source == replaceAndFindBtn)

{

save(false);

if(SearchAndReplace.replace(view))

{

ok();

//only in this situation, the search list will store the search string

int i = historyComboBox.getItemCount();

//size is not larger than 5

if(i>=5)

{

String a = (String)historyComboBox.getItemAt(1);

String b = (String)historyComboBox.getItemAt(2);

String c = (String)historyComboBox.getItemAt(3);

String d = (String)historyComboBox.getItemAt(4);

//historyComboBox.repaint();

historyComboBox.removeAllItems();

historyComboBox.addItem(a);

historyComboBox.addItem(b);

historyComboBox.addItem(c);

historyComboBox.addItem(d);

// add the item from SearchAndReplace class

historyComboBox.addItem(SearchAndReplace.getSearchString());

}else {

historyComboBox.addItem(SearchAndReplace.getSearchString());

}

}

else

getToolkit().beep();

}

else if(source == replaceAllBtn)

{

if(searchSelection.isSelected() &&

view.getTextArea().getSelectionCount()

== 0)

{

GUIUtilities.error(view,"search-no-selection",null);

return;

}

setCursor(Cursor.getPredefinedCursor(Cursor.WAIT\_CURSOR));

if(!save(false))

{

setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT\_CURSOR));

getToolkit().beep();

return;

}

if(searchSelection.isSelected())

{

if(SearchAndReplace.replace(view))

closeOrKeepDialog();

else

getToolkit().beep();

}

else

{

if(SearchAndReplace.replaceAll(view))

closeOrKeepDialog();

else

getToolkit().beep();

}

setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT\_CURSOR));

}

}

} //}}}