Documentation

Overview Description

My web browser is made up of 3 main classes: ***webBrowser***, ***browserToolBar*** and ***progress***. These 3 classes compile to give you the web browser as you see before you. As a summary of my classes; my *webBrowser* class *extends* a *JFrame* and *implements* a *TreeSelectionListener* and a *MouseListener*. These use of these interfaces allows it to do the main basics like creating the *frame* and making all the *JMenus*. However, it also creates a lot of functionality. This class implements everything to do with *tabs* and uses *trees* to display *history* and *favourites*. The constructor runs the program and does all the communication with the other classes; all the main does is make a new instance of the *webBrowser* class. My second class, being *browserToolBar,* concentrates on the rest of the features required by the web browser such as: the use of *toolbars*, *copy/paste*, the use of *image icons*, *navigation*, the use of *JEditor panes* and the use of *hyperLinkListeners* and *hyperlink updates*. As a result, the class *extends* a *JToolBar* and *implements* a *MouseListener* and a *HyperLinkListener*. This class controls all the web pages by storing them in to an array list. The class validates and moves to different web pages according to the end users input via the navigation buttons and navigates to new web pages. This class also has the hyper link listener and is responsible for being able to read HTML and displaying it onto the JEditor pane. My third and final class is a simple class. This class is purely used as a means for my *JProgress bar* to run in a different *thread*. This class *extends Thread* and *implements runnable* and the thread is started in my *browserToolBar* class were appropriate.

The 3 Classes Broken down

WebBrowser Class

This class has 3 main parts: the constructor and windowBar method; the Creation/ closing/ updating of tabs; and the favourites and history.
To begin with I will demonstrate each of these 3 parts by including screenshots of code from my program and explaining what it does, beginning with the constructor and windowBar method.

These are 2 of the most important methods, as they make the whole program run. The constructor does all the work and I decided to keep the constructor short and put most of the data in an outside method known as windowBar. I recursively call this method in the constructor so as all the code inside it is run.

```
public webBrowser(){
    JFrame frame = new JFrame();
    JPanel jpanel = new JPanel(new BorderLayout());
        createTab();
        getContentPane().add(tabPane);
            popTab();
            windowBar();
                this.setTitle("G52GUI - Browser");
                    this.setSize(1024,768);
                        this.setVisible(true);

    }//constructor
```

This is my webBrowser constructor. As you can see the constructor creates most of the main features required such as the tabs, popup menu for my tabs, recursively calls the windowBar method and sets the basics such as title, size and visibility. The main information comes from

the createTab and windowBar method. I will speak of the tab section in the following break down of this class but the use of windowBar was just so as to make the constructor as short as possible as windowBar contains all the lengthy code needed to create JMenus, their shortcuts and actions assigned to those menu items.

The next main section to this class is the in the use of a JTabbedPane and the opening, closing and updating of tabs.

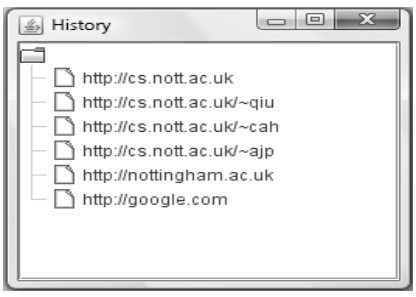
```
public void updateTab(){
    if(tabPane.getTabCount() >= 0){
        noTabsLeft = false;
        menuitemCloseTab.setEnabled(true);
        itemCloseTab.setEnabled(true);
    }//if
        }//method
public void createTab(){
    JPanel jpanel = new JPanel(new BorderLayout());
        browserToolBar toolbar = new browserToolBar();
        jpanel.add(toolbar, BorderLayout.NORTH);
        jpanel.add(new JScrollPane(toolbar.editorPane), BorderLayout.CENTER);
        tabPane.addTab("Browser "+tabPane.getTabCount(), jpanel);

}//method
public void closeTab(){
    tabPane.getTabCount();
    tabIndex = tabPane.getSelectedIndex();
    tabPane.remove(tabIndex);
    if(tabPane.getTabCount() <= 0){
        noTabsLeft = true;
        menuitemCloseTab.setEnabled(false);
        itemCloseTab.setEnabled(false);
        }//if

    }//method
```

These are the only three methods used in with tabs. I have a global variable tabPane made at the top of the program and the most important thing about the tabs is the creation of one because when I create a tab it is my main communication point between the webBorwser and browserToolBar class. When I create a tab I also create a JPanel in which I add my toolbar and then add my tabs onto the JPanel. I can open a numerate amount of tabs and closing them is done by using the getSelectedIndex and remove method. The tabs are checked whenever the last tab is closed and the close tab button is disabled and then re-enabled when a new one is made. This happens the same way with my JPopup menu for closing and opening tabs. Instead of having to click file and close tab, all one has to do is right click on the tabbed pane and close the tab or an even quicker way would be to CTRL+W and likewise, is the same with opening a tab (CTRL+T).

The last major part to the webBrowser class is the use of displaying your favourites and your history.

```
public void createHistory(){
    String[] history = getAllHistory();
        JFrame showHistory = new JFrame();
            DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode();
                for(int i = 0;i<history.length;i++){
                    rootNode.add(new DefaultMutableTreeNode(history[i]));
                }//for
                    root = new JTree(rootNode);
                        root.addTreeSelectionListener(this);
                        showHistory.add(new JScrollPane(root));
                        showHistory.setSize(300,200);
                        showHistory.setTitle("History");
                        showHistory.pack();
                        showHistory.setVisible(true);
                }//method
```

As shown by the coded section to the left. I have made use of the JTree class to be able to view ones own history. The bottom left screenshot shows the result after using a buffered reader class to read in the pages one has visited from the 'History.txt' text file. Likewise, I have displayed my favourites in very much the same way using a JTree.



browserToolBar class

The *browserToolBar* class is the main heart of the program when it comes to the functionality side. This class extends a *JToolBar* and implements the both the hyper link listener and the mouse listener. This class contains the ability to move from web page to web page, the ability to store web pages and the means to copy and paste from the *JEditor pane*. It also has code to import icons such as: the navigation buttons, the home button and the refresh button.

The following class can be broken down into functions. The 2 main functions are: copy/paste and navigation. Beginning with Navigation as it is the longest part. I have 7 methods that are used in navigation. These are: *actionBack*, *actionForward*, *showUrl(URL, Boolean)*, *actionUpdate*, *actionHome*, *validation and Go*. By the names you can derive what they all do. My *actionBack* and *forward* commands the ability to move backwards and forwards from your initial web pages. The *actionHome* has a specific URL that is classed as the home webpage which is: *http://nottingham.ac.uk.* The *validation and Go* methods go hand in hand as validation allows the user to not have to continuously write 'http://' and as a result it enters it into the *JTextfield* for you automatically; as you can guess the Go button is responsible for taking you to the desired webpage. So far all these methods are actions and are linked to all the toolbar buttons. Therefore, the *actionUpdate* method is responsible for updating all the buttons as appropriate e.g. disabling and enabling the backwards and forwards buttons when appropriate. I save the *showUrl(URL, boolean)* method till last because it is one that will be that I have provided the code for below as it requires explanation.

```
public void showUrl(URL page, boolean addPagetoList){
    bar.loading=true;
    System.out.println("showURL");
    // Show progress cursor while loading is under way.
    setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
    try {
        // Get URL of page currently being displayed.
        URL currentUrl = editorPane.getPage();
        // Load and display specified page.
        editorPane.setPage(page);
        // Get URL of new page being displayed.
        URL newUrl = editorPane.getPage();
        // Add page to list if specified.
            if (addPagetoList) {
                System.out.println("addPagetoList");
                int listSize = pageList.size();
                    if (listSize > 0) {
                        int currentPageIndex =pageList.indexOf(currentUrl.toString());
                            if (currentPageIndex <= listSize - 1) {
                                for (int i = listSize - 1; i > currentPageIndex; i--) {
                                    pageList.remove(i);
                    }//for
                }//3rd if
            }//2nd if
            pageList.add(newUrl.toString());
        }//1st if
    // Update location text field with URL of current page.
    webPageEditor.setText(newUrl.toString());

    // Update buttons based on the page being displayed.
    actionUpdate();
    } catch (Exception exception) {
        showError("Invalid URL");
    }finally{
    // Return to default cursor.
        setCursor(Cursor.getDefaultCursor());
    }//finally
        bar.loading=false;
}//method
```

This method is the mechanism by which the web pages change. Each of the 6 above methods listed all call this one method as it is through this method that they can do what they have been designed to do. Each of the 6 methods have separate URL variables and when calling this method each other method enters their designated *URL variable* and often false as for most of the time web pages are not to be removed from my *pageList ArrayList*, as the end user would like to navigate through all the web pages. One of the only times the Boolean value is true is when *showUrl* is called in the hyperlink update and the pages with the same index are removed to make navigation more efficient as so as to stop one for having to click

backwards and forwards multiple times and viewing the same webpage if you have clicked the same web page multiple times. This method creates a *current URL* and a *new URL variable* which is used to move from the current to the new web page and once done the *JEditor pane* is set to the appropriate page. Throughout the process, progress is tracked with the wait mouse cursor if you notice the '*bar.loading = true*' and the '*bar.loading = false*' is the progression of my progress bar which is being called from my *progress clas*s. This method is the most important when it comes to basic usability of the browser.

The next major function that this class carries out is the copy and paste function carried out through a JPopup menu. I have 2 methods: one called copy and the other paste. The creation of the JPopup menu is declared at the top of the class and my menu items, and accelerators are made in the browserToolBar constructor. I call both copy and paste into my constructor. The coding for both copy and paste is shown below:

```
public void copy(){
    itemCopy.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent h){
            editorPane.copy();
            webPageEditor.copy();
            }//method
        });
}comment
public void paste(){
    itemPaste.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent h) {
            webPageEditor.paste();
        }//method
    });
```

As shown by the code you can see I used Java's built in copy and paste functions. My webPageEditor variable is my JTextfield. This means that one can copy from both the JTextfield and the editor pane. You can paste to the JTextField and you can paste to all other external applications such as: notepad. The next fragment of code is the creation of the JPopup menu used for the copy and paste. As you can see the webPageEditor being my JTextField must have a mouseListener and so to must my JEditor pane for the JPopup menu to be fired. As a result, I required a popup trigger and a mouseListener to be implemented into this class and this is all evident and well described and commented within the code.

```
//Copy and paste popup menu!
    itemCopy = new JMenuItem("Copy");
    itemPaste = new JMenuItem("Paste");
    itemCopy.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, ActionEvent.CTRL_MASK));
    itemPaste.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V, ActionEvent.CTRL_MASK));
    pop.add(itemCopy); pop.addSeparator(); pop.add(itemPaste);
    pop.addMouseListener(this);

    webPageEditor.addMouseListener(this);
    editorPane.addMouseListener(this);

    copy();
    paste();
```

Progress Class

This class is simple. The class extends Thread and implements runnable. The main purpose for this class is to provide a separate thread for my JProgress bar to run in. The progress bar is made as a global variable and within the run method I do all of the progress within the bar. I have a loading variable that you have seen in my showUrl method in the browserToolBar class. The sole purpose for this is so as to start and stop the loading of the progress bar. It is placed within my ShowUrl method because this is the mechanism by which web pages change from one to the other.

Conclusion

The classes that I have described are aimed at emphasis on the key point within each of these classes. There is a lot more code than just the code that I have provided in this document. Therefore, I urge the one marking this to simply use it as a guide and follow the thoroughly commented code to get a better understanding of how the code works. Also, there are a few functions that have been created but are incomplete due to a lack of time or not knowing how to implement them. These functions include: the saving of favourites and the inputting of history. I have implemented these functions, and have the ability to view both history and favourites using JTrees. Also, I have created an options menu with the ability to re-enter your home page but currently I cannot change the home page. With extra time I perhaps would have completed this piece of functionality. But none the less the option menu is there.

User Manual

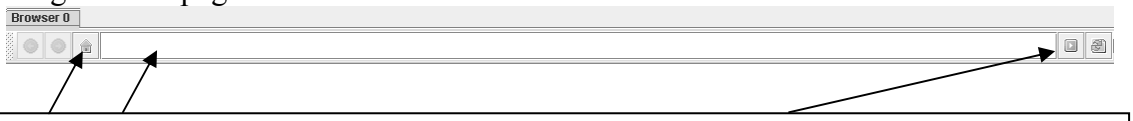
To begin to show how the web browser works we will start with the basics. I will show what the browser looks like when it is first loaded and all its major functionality.

Start Up

First off, the web browser starts up with a blank home page as shown below:

As shown the navigation buttons are all disabled and the progress bar is at 0%. Therefore, to get started you may enter a web page into the address bar then press go or simply press the home button.

To get a web page:



Step 1: Type the desired web page one would like to visit. **Step 2:** click the go button or click the home button directly to the left of the address bar.

Navigation

Then once the webpage is loaded and you have directed to several web pages you will notice that the navigation buttons will become available if you would like to navigate back or forward to your previously viewed web pages; as shown below. All one has to do is click on the navigation buttons to go back or forward.

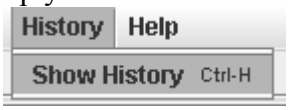
Step 1: Click either backwards or forwards or the home page button. **Step 2**: You will be directed to the correct page. The home page is currently **http://nottingham.ac.uk**

The progress bar will change when one changes a web page, this is a normal process and you can see how much is loaded depending upon the how much the webpage has loaded. This is shown here:

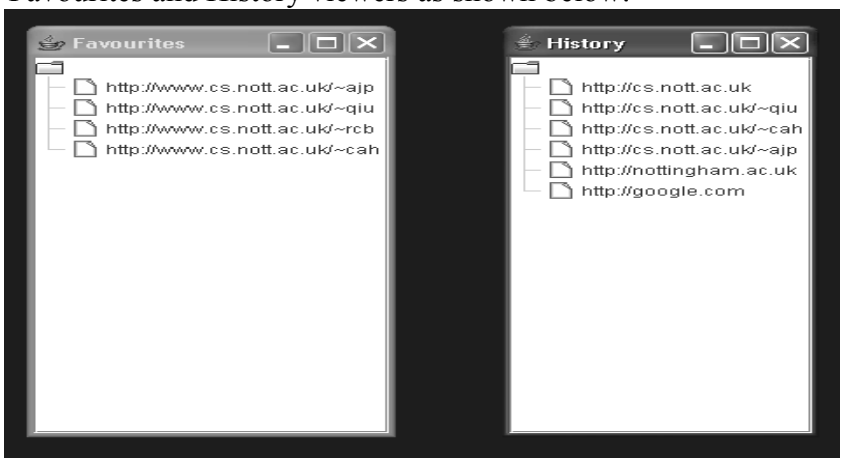
Viewing of History and Favourites

To view history and ones favourites follow the following guidelines:

Step 1: click History or Favourites on the menu bar as shown below or to speed things up you can use a shortcut Ctrl +H (for history) or Ctrl+S (for Favourites):



Step 2: Click on the icon or if one used the hotkey what should appear are the Favourites and History viewers as shown below:



Creating new tabs within the Browser

Follow the following guidelines to creating new tabs within this browser:

Step 1: you have 3 options in which you can create a new tab or close existing ones. The first way is if you click file then new tab, the second way is by right clicking the tab pane and then left clicking new tab and the 3rd and final way is to use shortcuts to create a tab you press Ctrl + T and to close a tab you press Ctrl+W.

Step 2: once the above is completed a new tab is created and one can use it. This is shown here:

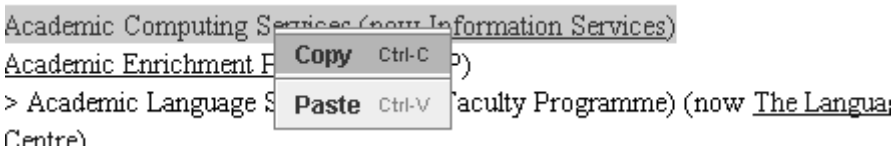
Creating a new Window

All one has to do to create a new window is when the browser has started up then **step 1** is to click on '**file**' and **step 2** is click on the button '**New Window'** or if you want to do it quicker all one has to do is use the shortcut **Ctrl+ N**.

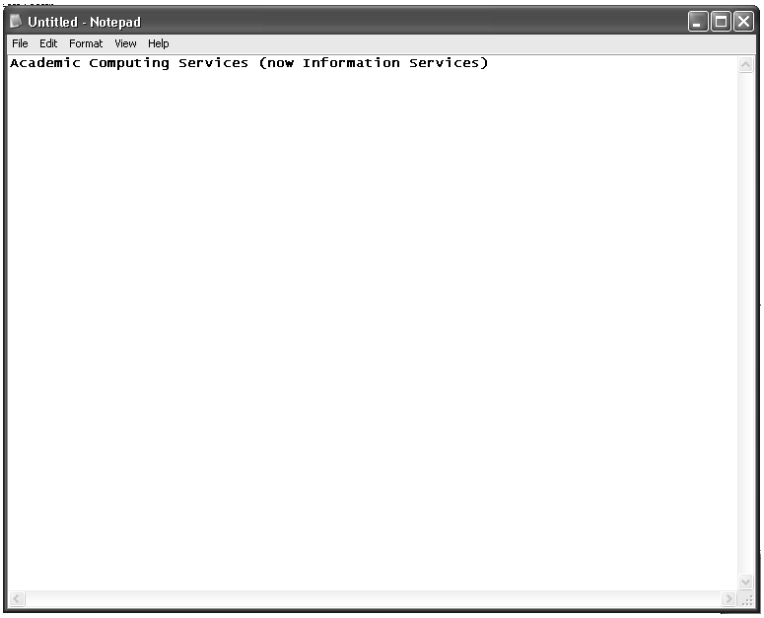
How to Copy and Paste

The web browser provides copy and paste functionality. To copy and paste follow the following guidelines:

Step 1: highlight the required text that one would like to copy then right click the mouse and click the button Copy. An alternative way of doing this is by using the shortcut Ctrl+C.



Step 2: When you have copied the required text, you can paste that text into any application. E.g. if you have copied a paragraph from a website it can be copied into a document. If you have copied a URL from a web page it can be pasted into the address bar.



The Options Menu

To access the options menu follow the following guideline:

Step 1: Click the edit menu then go down to 'Options'. If you want to accelerate the process use the following shortcut: Ctrl+C.

Step 2: Add your desired web page into the text field, if you want to cancel then click cancel. The options menu for the changing of the homepage is shown below: