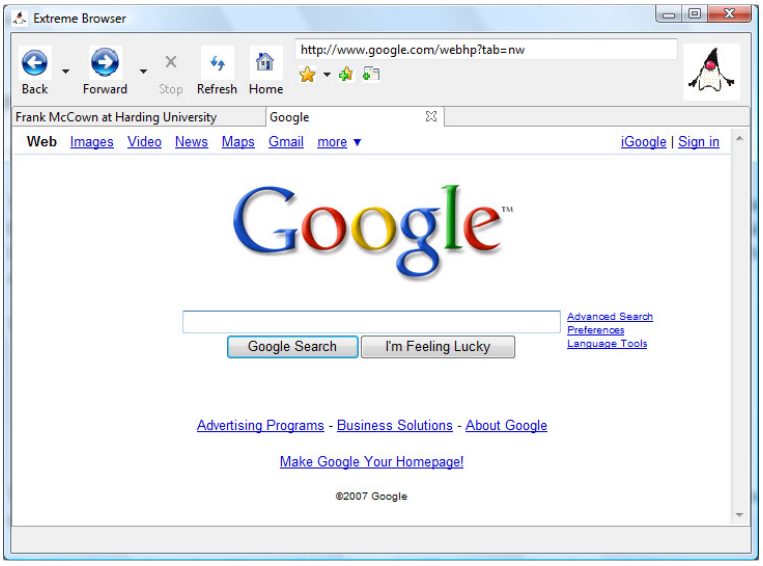
GUI Programming
Program #3 – **Web Browser**
Due Sun, Dec 9 at midnight
100 Points

Overview

You are to create a web browser using Java and SWT. The browser will allow the user to "surf" the Web just like other popular browsers like Internet Explorer and Firefox. The user can go forward or back through the browse history, and popular URLs can be bookmarked. The application window can hold up to six tabs, each with its own browser and browse history. A screenshot of what your browser may look like is shown below.



Requirements

The following specifications must be met in order to receive all 100 points:

1. The application should have a single window with:
   a. A custom icon in the title bar
   b. A toolbar containing various navigation buttons as shown above
   c. A textbox for entering URLs
   d. A portion of the screen that animates when pages are being downloaded
   e. A horizontal tab control where browser tabs are located
   f. A status bar at the bottom of the window that displays the currently selected browser's status and progress bar showing how much of a web page has been downloaded.

   The progress bar should only be visible when a page is downloading. The window should be resizable.

2. All buttons in the toolbar should be represented graphically. You can design your own graphics or "steal" them from another browser.

3. The program should have a single tab open to the home page when the application is first started. The home page can be set to any location the programmer wants (optionally, a button on the browser's toolbar could allow the user to change the home page to the currently viewed page). A button on the browser's toolbar should allow new tabs to be created, up to a maximum of 6 tabs. When a tab is created, it should become the active tab, and the home page should

be loaded into it. The tab's title should reflect the first 40 characters of the title of the tab's currently loaded web page. The user can close individual tabs using a close button on the tab.

4. The program should have the following 5 buttons that control navigation of the currently selected browser:
    a) Back – to go back one or more pages.
    b) Forward – to go forward one or more pages.
    c) Stop – to stop downloading the current web page.
    d) Refresh – to reload the current web page.
    e) Home – to go back to the home page.

5. The Back, Forward, and Stop buttons should be disabled if the actions they implement are not available. For example, if the user has not yet visited a second web page, there is no reason the Back button should be enabled. And if there is nothing downloading, the Stop button should not be enabled.

6. The Back and Forward buttons should have an arrow on the side that, if clicked, shows the pages (by title) the user can go back to or forward to in a popup menu. If the user selects an item from the menu, the browser should navigate to the selected page. The Back and Forward buttons' popup menus should also change based on which browser tab is selected, so only the current browser's history is accessible.

7. The user should be able to add new bookmarks using a button from the toolbar, and s/he should be able to access any of the previously bookmarked pages from a popup menu that appears when clicking on a button in the toolbar. The bookmarks are to be labeled according to the title of the page it refers to. A method for renaming, deleting, and re-ordering bookmarks is not necessary. All bookmarked pages should be stored to the file **bookmarks.txt** in the same directory as the local images so they are available when the application is restarted.

8. Your program must use the BrowserEx class (discussed later) which extends the org.eclipse.swt.browser.Browser implementation.

Grading

Your application should implement all the requirements given. You may want to mimic my Extreme Browser, IE, or Firefox.

5 bonus points are available for changing the text box for entering URLs into a combo box that maintains the browser history and shows the favicon (website icon) of the website being viewed. Both IE and Firefox implement this functionality, so you should mimic their behavior.

All files making up your Eclipse project should be zipped together and submitted to **Easel** before the due date. I will load your project into Eclipse and run it from there to test it.

We've all heard that a million monkeys banging on a million typewriters
will eventually reproduce the entire works of Shakespeare.
Now, thanks to the Internet, we know this is not true.

- Robert Wilensky

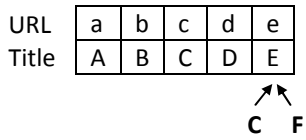
Implementation

An example web browser that implements many of the requirements of this project can be accessed at \\cs1\Classes\Comp445\Java\JavaBrowser. The application implements the SWT browser class and has functional back, forward, stop, and reload buttons. The app uses the FormLayout class to layout the various components in its window including the toolbar, text box, canvas, browser, and status bar. This is an excellent place to start.

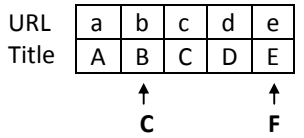
One problem with the SWT browser class is that it lacks access to the underlying browser's history. It only exposes back and forward methods for moving the browser to the pages stored in the history. We need to know all the pages in the history to display them when the user clicks on the Back and Forward buttons. To do this, we need to develop our own browser class by extending the SWT browser.

The BrowserEx class (next page) keeps two parallel arrays (`historyUrl` and `historyTitle`) which hold the browser's session history. The browser knows a new page should be added to the history when its progress listener's `completed` method is called. New URLs and their titles are added to the back of the arrays or on top of previously loaded URLs in `addLocationToHistory`.

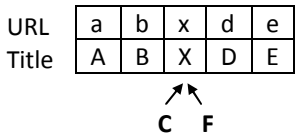
The following example illustrates how the history is maintained. The variables currentLoc and forwardLoc are used to keep track of where new URLs are inserted into the arrays and where the URLs reside that are in the browser's "forward area". currentLoc and forwardLoc are always incremented when new URLs are visited; currentLoc is decremented when going back. Suppose a user browsed to the following URLs in this order: a, b, c, d, e. The historyUrl and historyTitle arrays would look like this (**C** is currentLoc and **F** is forwardLoc):



The items in the "back area" would be everything to the left of where **C** points: a, b, c, and d. Everything to the right of **C** up to **F** would be in the forward area (in this example, nothing yet). If the user went back three pages, **C** would be decremented by 3, and the URLs c, d, and e would now be in the forward area:



If the user visited a new URL x, **C** would be incremented, and x would replace whatever **C** is pointing to. The forward area would be reset by making **F** point to the same location as **C**:



An alternative strategy would be to explicitly delete the d and e items from the arrays, but typically it is better to avoid freeing memory that will likely be needed again in the near future. When the user visits the new URL y, **C** and **F** are incremented as usual, and y replaces d.

There are several methods which have been overridden by BrowserEx: `back`, `forward`, and `checkSubClass` (to disable the check that prevents us from subclassing the Browser class). Other public methods have been created to extend the functionality of the Browser class: `getTitle`, `back(int times)`, `forward(int times)`, `getBackList`, and `getForwardList`. All the methods have been implemented for you except the last two; you need to complete these methods and call them from your program to display the back and forward history in your web browser.

```
import java.util.ArrayList;
import org.eclipse.swt.browser.Browser;
import org.eclipse.swt.browser.ProgressEvent;
import org.eclipse.swt.browser.ProgressListener;
import org.eclipse.swt.browser.TitleEvent;
import org.eclipse.swt.browser.TitleListener;
import org.eclipse.swt.widgets.Composite;


public class BrowserEx extends Browser {

	// Parallel arrays storing browser history Title and URL
	private ArrayList<String> historyTitle;
	private ArrayList<String> historyUrl;

	// Set to false to prevent the progress listener from adding the loaded page
	// to the browser history arrays
	private boolean visitNewUrl = true;

	// Title of currently loaded web page
	private String title;

	// Index into history arrays indicating which page is currently being viewed
	int currentLoc = -1;

	// Index into history arrays indicating which page is the last valid
	// page the user can forward to
	int forwardLoc = -1;

	public BrowserEx(Composite parent, int style) {
		super(parent, style);

		historyTitle = new ArrayList<String>();
		historyUrl = new ArrayList<String>();

		super.addProgressListener(new ProgressListener() {
			public void changed(ProgressEvent event) {
			}

			public void completed(ProgressEvent event) {
				// Only add this URL if user didn't use back and forward methods
				if (visitNewUrl)
					addLocationToHistory();
				else
					visitNewUrl = true;
			}
		});

		// Set tab title to browser's web page title
		super.addTitleListener(new TitleListener() {
			public void changed(TitleEvent event) {
				title = event.title;
			}
		});
	}

	protected void checkSubclass() {
		// Disable the check that prevents subclassing of SWT components
	}

	public String getTitle() {
		if (title == null)
			return "";
		return title;
	}

	public boolean back() {
		visitNewUrl = false;   // Prevent adding this URL to history
		currentLoc--;
		return super.back();
	}

	public boolean back(int times) {
		for (int i = 0; i < times; i++) {
			if (!back())
				return false;
```

```
                }
                return true;
        }

        public boolean forward() {
                visitNewUrl = false;   // Prevent adding this URL to history
                currentLoc++;
                return super.forward();
        }

        public boolean forward(int times) {
                for (int i = 0; i < times; i++) {
                        if (!forward())
                                return false;
                }
                return true;
        }

        public String[] getBackList() {
                // Return a list of items that are in the back area
        }

        public String[] getForwardList() {
                // Return a list of items that are in the forward area
        }

        private void addLocationToHistory(String title, String url) {
                currentLoc++;

                // See if there are any items in forward
                if (currentLoc == historyTitle.size()) {

                        if (currentLoc > 0) {
                                if (historyUrl.get(currentLoc - 1).equalsIgnoreCase(url)) {
                                        // Ignore duplicate URLs that are next to each other
                                        currentLoc--;
                                        return;
                                }
                        }

                        // Nothing in forward, so add to end of list
                        historyTitle.add(title);
                        historyUrl.add(url);
                }
                else {
                        // There are items in forward, so replace them
                        historyTitle.add(currentLoc, title);
                        historyUrl.add(currentLoc, url);
                }

                forwardLoc = currentLoc;

                printHistory();
        }

        private void addLocationToHistory() {
                addLocationToHistory(title, super.getUrl());
        }

}
```

Tabbed Browsing

To implement tabbed browsing, you'll need to create a tab folder (org.eclipse.swt.custom.CTabFolder) to hold each tab:

```
        tabFolder = new CTabFolder(parent, SWT.BORDER);
```

You will then create a tab (org.eclipse.swt.custom.CTabItem) which will hold each browser:

```
        final CTabItem tabItem = new CTabItem(tabFolder, SWT.CLOSE);
        final BrowserEx browser = new BrowserEx(tabFolder, SWT.NONE);
        tabItem.setControl(browser);
```

Each browser will need a:

1. LocationListener to, among other things, set the location text field to the browser's location.
2. ProgressListener to update the progress bar and animation and set the enable field of various navigation buttons.
3. StatusTextListener to update the status bar.
4. TitleListener to set the tab's title. Optionally this could be done with the ProgressListener using the getTitle method.

The tab folder will need a:

1. SelectionListener to (among other things) set the location text box to the tab's browser that was selected.
2. CTabFolder2Adapter to determine when a tab is closed.

The Forward and Back buttons should be implemented as drop-down ToolItems like so:

```
itemForward = new ToolItem(toolbar, SWT.DROP_DOWN);
```

Each will need a Listener to determine if the Back or Forward button is being pressed or if the drop-down arrow is being pressed:

```
if (event.detail == SWT.ARROW) {
        // Create a forward list for the active browser
}
else {
        // Go forward
}
```

See the Menu code snippets at http://www.eclipse.org/swt/snippets/ to see how to make dynamic menus pop-up at a particular location.

Running Extreme Browser

I have created a web browser that implements all of the required functions. You can access it at \\cs1\Classes\Comp445\Java\ExtremeBrowser. Just copy the entire folder to your hard drive, and add the swt.jar to your classpath. To see if swt.jar is already added to your classpath, open a command line window, and enter:

```
C:\>echo %classpath%
```

This will output your classpath. If you don't see swt.jar, then you can add it one of two ways:

1) You can change classpath temporarily for the command line window by entering the following command in at the command prompt:

```
set classpath=%classpath%;C:\path_to_swt\ org.eclipse.swt\swt.jar
```

2) You can set it for the entire system by editing your classpath environment variable by adding C:*path_to_swt*\ org.eclipse.swt\swt.jar. See http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/environment_variables.mspx?mfr=true for instructions on creating/modifying environment vars in XP.

Once you have the directory copied and the classpath set, change to the directory you just copied over in the command line window, and enter:

```
C:\projects\ExtremeBrowser>java ExtremeBrowser
```

You will now see the browser (or an error message (stack trace) in the command line window if there were any problems).