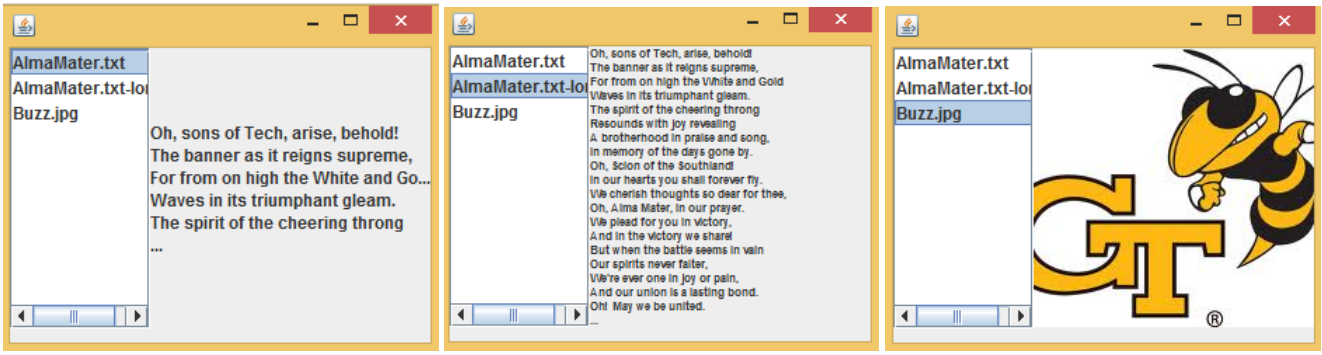
CS 1331: Introduction to Object Oriented Programming

Homework 6

Due: March 7, 2014 @ 8:00 PM

File Preview Panel

In this program, you will develop a GUI that emulates the file system previewer that can be found in OSX. Specifically, it will have a list of files on the left-hand side of the GUI, and a preview of the currently selected file on the right-hand side.



You don't currently know how to display a list of items, so we will be giving you this section of the code. In real life, you will often have to build on code written by other people. For this homework, we are giving you a class called PreviewListPanel. This class expects you to provide it with objects that implement the Previewable interface, which is described below.

We've given you the Javadocs for the PreviewListPanel. You can find them by looking at index.html. In it, the behavior of the constructors and both of the methods are described. You'll also find the Javadocs to another useful class, FileScanner, which we've written for you.

Once you've written the Main GUI with a list of files and a panel to preview the files, we expect you to instantiate at least 2 copies of each type of Previewable file (see below) and add them to the PreviewListPanel. Feel free to add files of your own!

Previewable

Interfaces are helpful when you are working with other programmers and you have to agree that certain pieces of code will take inputs and provide outputs, without knowing necessarily how that code will be implemented. This is why interfaces are often referred to as contracts.

In this homework, you have to write and implement the Previewable interface. This interface will be used by different classes that represent files to notify other programs that they can display themselves on JPanels. It must require the following method:

```
public void preview(JPanel panel);
```

In all Previewable objects, this item will take in a JPanel and display the Previewable object on the JPanel. You should remove any components that are already on the JPanel. You should not change the size of the JPanel.

You must write classes for at least 3 types of Previewable files:

- A short text file, like ugaJoke.txt. This file should be previewed by displaying the first 5 lines from the text file. This should be able to handle files with less than 5 lines.
- A long text file, like almaMater.txt. This file should be previewed by displaying the first 20 lines from the text file in a slightly smaller font. This should be able to handle files with less than 20 lines. Hint: use JLabel.setFont(Font newFont), JLabel.getFont(), and Font.deriveFont(float size).
- An image, like Buzz.jpg. Hint: JLabel has a constructor which accepts an ImageIcon.

We require that you implement Previewable in at least three different ways. So, your GUI should be able to display at least three different types of files, one for each class that you write. However, if you want to get creative, feel free to experiment with different file types! (Can you figure out a smart way to preview .java files?)

Notes and Hints

- Don't change PreviewListPanel.java - it works, and changing it will only make life harder for you.
- Use LayoutManager's to make your GUI look good.
- To read in a text file, we've provided you with a FileScanner class. This works exactly the same way as Scanner. Just create one and pass it the name of the file to read, then use .nextLine() to get the lines of the file one-by-one.

Javadocs

We are going to have you do Javadocs again for this assignment (and for all assignments here on out). The samples from the last HW writeup are below for your reference.

```
import java.util.Scanner;

/**
* This class represents a Dog object.
* @author Ethan Shernan
* @version 1.0
*/

public class Dog {
....
}

/**
* This method takes in two ints and returns their sum
* @param a, b
* @return their sum
*/

public int add(int a, int b)){
...
}
```

Checkstyle

Just in case you forget how to do Checkstyle, here are the instructions again!

You have been provided with a Checkstyle Jar file and the CS 1331 Checkstyle configuration file (`cs1331-checkstyle.xml`). We do this to make sure you are following the proper style guidelines as posted in the Style Guide under T-Square resources. Please refer to these guidelines for any help as you are writing your programs.

We will run checkstyle on all the Java source files you submit. You can run Checkstyle with the command below (in the directory containing all your Java source files):

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by using the command below and subtracting 2 from the number printed (which is how we will deduct points). For example:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | wc -l
2
```

Also means no errors.

If you have any questions about a Checkstyle error, please let a TA know! They should be pretty self-explanatory and you should be able to fix them easily. We will be taking points off for checkstyle on this assignment, so please make sure you run it!

Turn-in Procedure

Submit all of the Java source files you created to T-Square. Do not submit any compiled bytecode (`.class` files). When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code with Checkstyle!

Verify the Success of Your Submission to T-Square
Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is 8PM Friday. Do not wait until the last minute!