DUE DATES:

Monday recitations: 8:00 PM on 11/13
Wednesday recitations: 8:00 PM on 11/15
Thursday recitations: 8:00 PM on 11/16
Friday recitations: 8:00 PM on 11/17
Saturday recitations: 8:00 PM on 11/18

Preliminaries

This lab is similar to lab 8, and gives you additional practice with some of the same concepts. At the same time it gives you an opportunity to explore using fonts and reading data from files, and (if you want) create an executable jar file so you can easily run your program outside Eclipse.

NOTE THAT THE DUE TIME FOR LAB 9 IS SLIGHTLY DIFFERENT FROM OTHER LABS. This was done to ensure nobody's lab is due during the second exam.

Your TAs will give you an overview of the lab and hints of how to proceed in recitation. You should expect to have to put in time outside of recitation to finish this lab. We therefore recommend that you bring your laptop to recitation so you can more easily continue to work outside of Baldy 21.

During office hours TAs will give priority to students who have attended recitation. It is not acceptable to skip recitation and then expect one-on-one assistance in office hours. Some aspects of this lab will ONLY be discussed in recitation, and TAs will NOT answer basic questions about these topics during office hours.

Because this lab gives you a great deal of freedom in how you write the code there is NO AUTOMATED GRADING. When you submit to Web-CAT the only score you will see is your early bonus or late penalty. The functionality of this lab will be manually graded by the TAs.

Keep in mind that any work you submit must be your own. Submitting work done by someone else as your own is academically dishonest, and will result in immediate failure in the course. We will use software tools to detect inappropriate collaboration.

Introduction

Noam Chomsky, a quite famous linguist who has made significant contributions to the study of both human and formal languages, came up with the following sentence to demonstrate the difference between syntactic and semantic well-formedness:

Colorless green ideas sleep furiously.

This sentence has a perfectly acceptable structure, but its meaning is nonsensical. It consists of two adjectives modifying a plural noun, an intransitive verb, and finally an adverb. Here are some sentences with the same structure that does have a sensible meaning:

Sweet warm cookies disappear quickly.

Inquisitive young kittens play quietly.

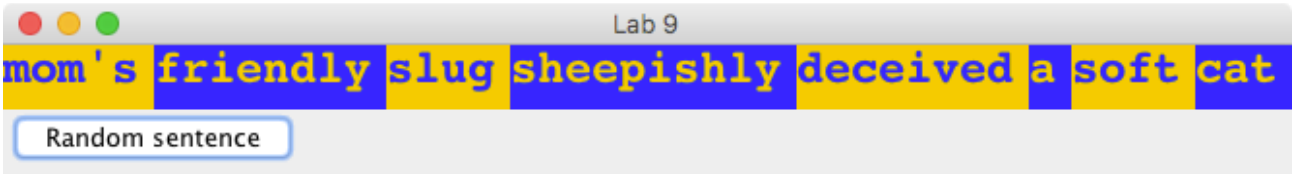
In this lab you will apply what you have learned about Collections and control structures in lecture to write a small program to generate random sentences. You will create a small Graphical User Interface (GUI) which generates sentences by choosing adjectives, adverbs, determiners, nouns, and verbs at random.

Preparatory tasks

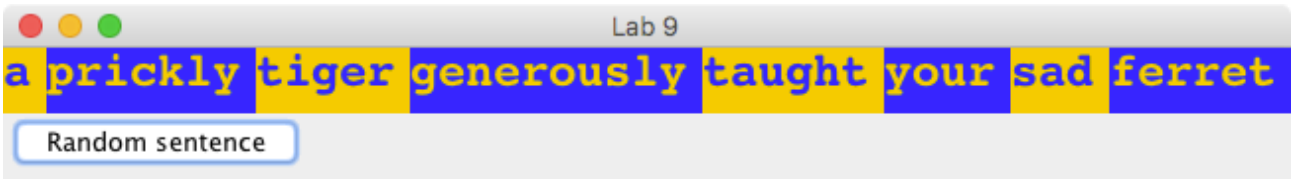
1. Log in
2. Start Eclipse
3. Switch to the CVS Repository Exploring perspective
4. Check out the FA16-CSE115-Lab9 project from the Labs repository
5. Switch to the Java perspective

User Interface

The user interface is very simple: it will consist of eight JLabels and one JButton. Each JLabel will hold one word of the sentence. When the JButton is pressed a new sentence must be generated. Here's an example of what it might look like:



or, after clicking the 'Random sentence' button, possibly this:



Each of the words 'a', 'friendly', 'eagle', 'cowardly', 'hugged', etc. is being displayed in its own distinct JLabel.

HINT #1 will be given here in recitation.

Displaying text on a JLabel

To set the text that is displayed on a JLabel, you can use the setText method; it has the following header:

```
public void setText(String text)
```

Notice that the JLabels alternate in foreground and background color. Make sure yours do too.

HINT #2 will be given here in recitation.

Notice that the font of the JLabels is not the usual one. I chose to use Courier, bold, 24 point. Choose Courier or some other font you like, in bold or italic, and in either 24 or 36 point.

HINT #3 will be given here in recitation.

Reading lines of text from a file using LineReader

In the project you will find five files, in the folder 'WordLists', with the names 'adjectives.txt', 'adverbs.txt', 'determiners.txt', 'nouns.txt' and 'verbs.txt'. These files contain adjectives, adverbs, determiners, nouns and verbs, respectively. Your program needs to read the words from each file and put them into an ArrayList<String>. Thus, your program will have five such ArrayList<String> objects: one for adjectives, one for adverbs, one for determiners, one for nouns and one for verbs. Things work out quite easily if you keep track of them all in an ArrayList (which will then be an ArrayList<ArrayList<String>> !)

HINT #4 will be given here in recitation.

Also included in the project is the class FileReader, a class we have defined to make reading Strings from a file a little easier that it would be otherwise. To create an instance of this class supply the path to the file whose contents you wish to read as a String argument to the constructor. As an example, to create a FileReader that reads line of text from the 'adjectives.txt' file contained in the 'WordLists' folder you would do the following:

```
new FileReader("WordLists/adjectives.txt");
```

The LineReader is set up as an iterator, with *boolean hasNext()* and *String next()* methods, so you can use an iterator-controlled while loop to read all the Strings from a given file.

Shuffling the contents of an ArrayList

The Collections class has a method named 'shuffle' which randomizes the order of elements in an ArrayList. This is a static method of the Collections class, and requires as its argument a reference to an ArrayList object (the ArrayList whose contents we're going to shuffle).

The 'Random sentence' button

Define an event handler for the JButton (which should be labeled 'Random sentence') so that when it is clicked a new random sentence is generated and displayed on in the user interface.

HINT #5 will be given here in recitation.

Running your program from within Eclipse

Define a Driver class with a main method,

```
public static void main(String[] args)
```

In the body of the main method start your program using the SwingUtilities.invokeLater method, as demonstrated in the sample code in the repository. Look in particular in the graphics_am code in the LectureCode project.

Don't forget to make the main class of your program implement Runnable!

Running your program from outside Eclipse (optional – not graded)

Make sure you can run your program from within Eclipse by defining the main method as described immediately above. A common way to deploy Java applications is to package up the required code into a so-called 'jar' file (Java ARchive file). You can do this by doing the following:

1. Right-click on the project name in the package explorer and selecting "Export…"
2. From the 'Export' dialog that appears, select Java->Runnable JAR file.
3. From the 'Runnable JAR File Export' dialog that appears next, select the Launch configuration you used to run your lab (it will probably be named "Driver (??) – FA16-CSE115-Lab9", where '??' is some number.
4. Choose an "Export Destination".
5. In the Library handling section, choose "Package required libraries into generated JAR".
6. Click "Finish".

Now you need to export the data files:

1. Right-click on the project name in the package explorer and selecting "Export…"
2. From the 'Export' dialog that appears, select General->File System.
3. From the 'Export' dialog that appears next, select just the 'words' folder to export. This automatically selects the four files inside the 'words' folder. Make sure all other check boxes are cleared; the FA16-CSE115-Lab9 checkbox will probably contain a minus sign ('-') – this is OK.
4. Specify the same directory (in the "To directory:" slot) that contains the jar file you just created.
5. Make sure the "Create only selected directories" option is selected.

6. Click “Finish”.

You should now have a jar file icon that you can just double-click to run. It on your operating system double-clicking doesn’t work, you can open a terminal/command prompt, navigate to the directory that contains the jar file, and manually run it. If your jar file is named Lab9.jar you can run it by typing,

```
java –jar Lab9.jar
```

Required functionality

[5 points] Display a JFrame whose title is your first and last name, followed by “’s Lab 9”. For example, the title of my JFrame must be “Carl Alphonce’s Lab 9”. Notice that the screenshots above do NOT show the correct title.

[10 points] Display eight JLabels in one row, each displaying a String of the appropriate part of speech (determiner-adjective-noun-adverb-verb-determiner-adjective-noun).

[10 points] Display one JButton in by itself in the row below the JLabel. The text of the JButton must be “Random sentence”.

[30 points] Handle a button-click on the JButton by generating and displaying a random sentence.

[20 points] Make appropriate use of data structures and control structures to minimize code duplication.

[25 points] Structure the code to achieve a model-view separation.

Submitting your project to Web-CAT

Make sure you submit your work on time; due dates are listed at the beginning of this lab description. There is NO automated grading for this lab. You may submit as many times as you wish. Your last submission is the one that counts (so consider carefully whether you want to make any late submissions, as the late penalty is 20 points per day or portion thereof late).
