COSC 102 Spring 2015

Speed Reader Lab 8

The first part of this lab is to setup you up with Bailey's *structure package*, as you will need to use it for the next homework.

- You will setup up a `.jar` file in DrJava (or Eclipse if it is your IDE)
- You will use the `structure5` API by doing a short example.

The second part of this lab is to write a **speed reader** application. Your tasks are

- to handle the command line arguments
- to provide the File IO: reading from an input file
- to process each string, aka token, so it is colored and centered in the application

The code for the graphics and the animation is provided, you need to understand it to be able to tweak it to the requirements listed below.

Structure Package

Setup

First download the bailey.jar from the Williams College site.

On your personal computer or in your home directory in the cs domain you have a directory for COSC102, such as `102`. In this directory:

- place the `bailey.jar` file you downloaded
- create a new directory for this week lab, `lab8` in your `lab` directory
- move the provided file `Spells.java` in this week lab folder
- open `Spells.java` in DrJava and try to compile it...

  It doesn't work: a yellow overlay appears on the `import structure5.*` statement.

DrJava can't find the `structure5` package yet: we need to add a link to its path, i.e., the location on disk where its `.jar` file is, so that the JVM (Java Virtual Machine) can find it.

Next in DrJava

1. search the `Edit` menu for the option `Preferences...` which opens a window
2. in the `Resource Locations` category click the `Add` button for `Extra Classpath`, which opens a file browsing window
3. navigate to the location where you can select `bailey.jar` in your `102` folder
4. compile and run `Spells.java`: you should see something but not much

5. look at the DrJava console which says `Specify a spell at the command line`
6. read the code and figure out a command line argument so that a spell is returned as output in the interaction window

   Remember previous labs you can use the interaction window as a Unix console

   ```
   > java ClassName str1 str2
   ```

   or in the DrJava syntax

   ```
   > run ClassName str1 ..
   ```

Short Example

First try the following experiment with `Spells.java`

- comment the line
  `spell[9] = new Association<String, String>("Incendio","Start a fire");`
- uncomment the line below
  `spell[9] = = new Association<String, String>(null,"Start a fire");`
- compile and run (with command line argument) the program
- observe the error
  - why is it happening?
  - look back at the `class Association` code distributed in class
  - ask us if you are not sure

Array to Vector

Modify the code so that instead of using an array for storing the list of association, i.e. the spells, you use a linear collection that dynamically expands: the `Vector` data structure of the `structure5` package, which we studied the implementation in class.

- Open `Lab8Ex.java` in DrJava
- Compile it and run it: it is similar to `Spells.java`
- Complete `step 1` by following labels `1.a, 1.b, 1.c`: try to test each subpart, by uncommenting/commenting out relevant part of code

Make sure you test the `Vector` object creation and initialization (adding the content of the init method).

Method `addUnique`

At home to help you with the next homework correct the `addUnique` method, so that only an association, whose key is not already in the `Vector`, is actually added to the collection.

- Complete `step 2` by writing the `addUnique` method to only inserts unique key.
- Test your method implementation using the print method after some `addUnique` calls.

Speed Reader

Everyone has wished they could read faster at some point in their life, for example, to get through the required reading for their English class, to cram for an exam, or to simply get through their ever-growing list of novels to read. Since the 1950s, psychologists, linguists, and educators have devoted significant efforts into speed reading techniques that can dramatically increase your reading speed with relatively little loss of comprehension.

In this lab, you will prototype one approach to speed reading called Rapid Serial Visual Presentation (RSVP)–recently popularized by Spritz Inc. and apps like Spreed. At the end of this lab you should be able to test it on yourself and try it on your friend!

See some demos to understand what you are implementing.

At its core, a speed reader app:

1. Reads in a text file provided by the user, which requires you to put into application your reading on File Input Techniques.
2. Breaks the text file into tokens. Recall a token is by default a single word separate by whitespace (which includes both spaces and line breaks).
3. Displays each token in succession with some sort of delay between tokens.

First run, observe and read to understand the provided code `SpeedReader.java`. The starter code does two relevant things.

1. After 5 sec there is a blinking square, which color alternates between black and red every 1 sec. This animation is achieved thanks to a `Timer` object connected to the `SpeedReader` object, `this`, as it implements the `actionPerformed` method of the `ActionListener` interface.
2. Pressing the `spacebar` key changes the word displayed from the instance variable array, `fixStrings`.

What should you do?

1. Implement the command line arguments. See comments: `COMMAND LINE ARGS`
2. Process the words of the file. See comments: `WORDS`
3. Display them using the font size and at the speed requested: each word is displayed at the center of the frame and with the middle letter colored. See comments: `DISPLAY`

Once you implement your speed reader, have fun! You and your partner should train on using the speed reader. You can gather sample text from a variety of sources, for example, Wikipedia and Project Gutenberg.

Submit

Show the lab instructor the work you completed before leaving the lab.

Submit on Moodle by the deadline a `zip` file called `lab08` containing your two files (You should delete the *.class* files before creating the `zip` so it is a smaller file).

Credit: Speed Reader is a Nifty Assignments written by Peter-Michael Osera from the University of Pennsylvania.