CS 1331 Homework 3

Due: Monday, September 17, 2012

Introduction

In this homework, you will work with the String class to better familiarize yourself with it. You also will do more with applets and graphics, utilizing the Random class to help in the second part.

On the last HW assignment, you began to learn about javadoc. As you'll recall, Javadoc is an application that is part of the JDK that you downloaded. You can run it on the command line by typing "javadoc *.java" which will create documentation for all the classes in your current directory. If you are using JGrasp, you can invoke javadoc by using Project->Generate Documentation from the pull-down menu. Javadoc creates the nice web-page based documentation that we have been using in class (the API). Remember from class we said that javadoc comments were a special form of comment. Javadoc recognizes a comment that starts with /** as a javadoc comment. You should continue to use Javadoc comments on this assignment. All classes you create from now on should have a descriptive Javadoc comment, just like in homework 2.

Assignment 3.1: Leetspeak Translator

[From Wikipedia]: Leet or Eleet (sometimes rendered l33t, 1337 or 31337), also known as Leetspeak or Leetzorz (1337Z0l2Z), is an alphabet used primarily on the Internet, which uses various combinations of ASCII characters to replace Latinate letters (see http://en.wikipedia.org/wiki/Leet_speak for more details).

Your assignment is to write a program that translates English into simple leetspeak. Call this program LeetTranslator.java. The user will enter a sentence in English and you are to translate it into leetspeak given the following set of rules:

- "@" is equal to "a"
- "3" is equal to "e"
- "1" (one) is equal to "i"
- "$" is equal to "s"
- "0" (zero) is equal to "o"

We have not covered conditionals in Java yet, but you do not need them on this assignment. Also, you do not need to handle any other translations apart from those five.

Example (user input in bold):

Enter string to translate: **johannes thinks he is so cool, but he's not**
j0h@nn3$ th1nk$ h3 1$ $0 c00l, but h3'$ n0t

Assignment 3.2: Mood Ring

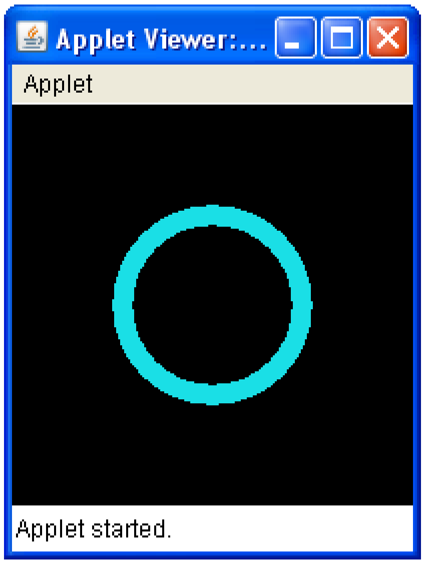
JohannesCorp is considering manufacturing mood rings and Marketing wants you to simulate them using an applet. The mood rings are to be very cheap and of poor quality, so they turn a random color and change randomly regardless of the user's actual mood. Thus, Marketing has given you the following specifications for your applet:

- The ring's color should be completely random (use the Color constructor with random parameters, and be sure to check the API if you need it).
- The ring should be displayed with a random diameter.
- The ring's color and diameter should change every time the user runs the program or resizes the applet.
- The center of the ring should be at the center of the applet, even if the window is resized.
- The ring should not extend beyond the edge of the applet window (you can assume the applet will always be square).
- The ring should have a reasonable minimum size so it always displays properly (that is, the ring should still be visible even at the smallest size) so you will need to think about your range of random numbers when choosing a diameter.

Call the program MoodRingApplet.java. Also, make an HTML page (MoodRingApplet.html) to display your applet.

Because the background drawing code behavior has changed in Java 6, you can use a fillRect which covers the entire applet to mimic a background (the background does not have to be a random color). Remember to handle resizing windows!

Here is an example of the mood ring applet:



Note that if I resize the applet above, the ring should be redrawn so it remains in the center (and it would be drawn with a new random color and random size).

Ambitious people: you can optimize the ring to fit in any size window, square or not, by using the Math.min method (look at the Java API for details) to determine whether width or height is the constraint, but it is *not* required. As long as your ring fits within a square applet when resized, you will receive full credit for that criteria.

Turn-in Procedure

Turn in the following files on T-Square:

- LeetTranslator.java
- MoodRingApplet.java
- MoodRingApplet.html

Don't forget to javadoc all .java files and include a collaboration statement.

Extra Credit

Extra credits are advanced topics that will test your programming mettle! In total, these will be worth 5 extra points each. These will not be covered in class, so you'll have to figure out how to do them on your own (of course, you may consult friends, TA's, and the Internet). If you attempt these, and your normal program doesn't work properly, or the extra credit is broken and interrupts a required feature, you won't get points for either. Attempt this only after you've submitted a working assignment. If you do the extra credit, make sure to mention it in your javadocs!

How difficulty ratings work:

0: Trivially easy.

5: Challenging. Involves topics you might not know yet.

10: Involves topics from all across the course. Designed to be difficult.

Homework 3 Extra Credits:

1. (Difficulty: 3/10) Allow your program to translate from Leet back to normal. When your program starts up, you'll need to ask the user which type of conversion they would like to do. To do this, you need to know how **if()** conditionals work.
2. (Difficulty: 8/10) The mood ring should change colors periodically. To do this, you'll need to understand the **Timer** and **Actionlistener** classes, as well as how to force an applet to **repaint**. You should generate a random color for the mood ring to switch to. 1 second is the suggested delay for the Timer.