Assignment 3: Tag Clouds

Weight: 15%
Due: Friday, March 20, 2009 at 11:59pm

Assignment Goals

The purpose of this assignment is to give you practical experience in the analysis, design, and implementation of an object-oriented system.

Problem

In this assignment, you will read in a text document from a file and draw the corresponding tag cloud.

Input

Your program will take as input two files: a document, and a list of stop words. The document can contain any list of English words, separated by whitespace of some kind (spaces, new lines, tabs, etc.). Some of these words can (and will) be repeated. The stop words file will contain a list of non-repeated *lowercase* words. The stop words are a list of words that will be ignored in your tag cloud.

The names of these files will be specified on the command line, along with the following parameters:

- (*p*) The smallest point size used to draw text to the screen
- (*P*) The largest point size used to draw text to the screen
- (*c*) The colour to use for words with the lowest frequency (1)
- (*C*) The colour to use for words with the highest frequency
- (*h*) The amount of horizontal space to add between words (in pixels)
- (*v*) The amount of vertical space to add between words (in pixels)
- (*f*) The minimum frequency of a word that is to be drawn (words with a smaller frequency will not be drawn)
- (*w*) The width of the entire tag cloud

NOTE: the labels used in this list are only for making this assignment description easier to read. **You should consider using different variable names in your code**.

Output

Your program will display a tag cloud of all of the words in the document that are also not in the stop list and occur more than the minimum frequency (*f*). Each word should be drawn without overlapping any other word. Words should be laid out from left to right with the specified amount of empty space (*h*) between words until the width of the tag cloud (*w*) is reached. Once this boundary is reached, words should be drawn on the next line from left to right with the specified amount of vertical spacing (*v*) separating each line. NOTE: You are not required to check whether too many rows are drawn. Words in each row should be drawn at the same baseline (i.e., using the same y-coordinate in quickdraw).

The size and colour of each word should be interpolated between the minimum (*p* or *c*) and maximum (*P* or *C*) specified values using the word's frequency and the maximum frequency for any word. A class method in A3Helper has been provided to help with this interpolation. For example, if the minimum colour is white (r=255, g=255, b=255), the maximum colour is blue (r=0, g=0, b=255), a word "hello" has a frequency of 5, and the maximum frequency for any word in the document is 20, you would calculate the green value of the colour to draw "hello" with as follows:

```
int green = A3Helper.interpolate(0, 255, 5, 20);
```

Example

Here is an example of the appropriate output for Edgar Allen Poe's "The Raven" (raven.txt) as the document and a short stop list (short-stopwords.txt) with the following parameters:

- Smallest point size = 12 points
- Largest point size = 72 points
- Min frequency colour = r: 255, b: 255, g: 255 (white)
- Max frequency colour = r: 0, b: 0, g: 255 (green)
- Horizontal space = 50 pixels
- Vertical space = 50 pixels
- Minimum frequency = 4
- Tag cloud width = 790 pixels

Support

While this may initially seem like a daunting task, I have provided some code to help you with this assignment. You are also free to use the code that we wrote in class. There are two files of code to help you with this assignment: Display.java and A3Helper.java.

A3Helper.java

This class provides three public class methods that you can use in your assignment:

- getStringBounds(String, int, String, String)
    - This function takes three font parameters (name, size, and style) and a string parameter and returns a bounding rectangle for the string.
- getFile(String)
    - This function takes a filename and returns a list of strings.
- interpolate(float, float, int, int)
    - This function interpolates size and colour values (as described above).

This class also contains a main method with some sample code for how to use the getFile function.

Display.java

This class provides an interface to the quickdraw program. There are two instance methods that are of particular importance:

- Display()
    - This constructor creates a quickdraw window that you can send commands to.
- writeCommand(String)
    - This function sends a command to the quickdraw window.

This class also contains a main method with some sample code for how to use this class as well as the A3Helper.getStringBounds method.

Expectations

For CPSC 217 and for assignments 1 and 2, you likely created your programs as a single file. All of your variable declarations and all of your code went into these files. You learned about modules, but you didn't have to create any yourself. At this point in CPSC 219, we are now creating much more modular programs. Each public class in Java must be in its own file. For a simple program with four classes, this now requires that you create four files and this requires that you now have to make decisions about where to place code. Getting this right is an absolute priority for this assignment!

You are expected to make use of the online documentation for Java. You are welcome to make use of any of the classes available in this specification. Of particular interest are the ArrayList, String, and Rectangle classes. If any part of this documentation or anything in this assignment description is unclear, or even if you just get stuck, I am EXPECTING you to come see me or the TA.

Evaluation

Your mark for each part will be calculated as follows:

	Excellent	Satisfactory	Unsatisfactory
Documentation	(15 marks) Your documentation is effective, concise and includes all of the components listed below.	(10-14 marks) Your documentation is missing one or two of the components below, you are overly verbose in a few places, or it is difficult to understand what you have written for one or two descriptions.	(0-9 marks) Most of your documentation is missing the below components; your comments are extremely long and usually difficult to understand.
Programs output correct values	(15 marks) Your program produces correct output for every possible input (according to the specifications).	(10-14 marks) Your program produces mostly correct output, with the exception of up to four types of input.	(0-9 marks) Your program produces mostly incorrect output. The range of 0-9 will depend on how close the output is to being correct.
Implementation	(15-20 marks) It is very easy to follow the flow of your program and it is clear why each step is performed and each class was included. You divide your code across several classes with methods of relatively small size.	(10-14 marks) The TA has some difficulty understanding why you chose the classes and methods that you did, but is able to eventually figure it out. Some of your methods are long and could be broken down into two or more functions.	(0-9 marks) The TA has a very difficult time understanding your implementation (or cannot understand it at all). Most of your methods are too long and could be broken down into two or more.
Class, method, and variable names	(15 marks) The names you chose make your code clear and easy to read.	(10-14 marks) Up to six names aren't clear (e.g., x, foo, bar, class A).	(0-9 marks) More than six names aren't clear.
Demonstration	(15 marks) Your program works exactly according to the specification for all test cases. You are also able to clearly explain your code and answer questions about the effect of changing it.	(10—14 marks) Your program works according to the specification, with the exception of up to three test cases. You have some difficulty explaining your code or require some prompting from the TA to be able to describe the effect of changes to your code.	(0-9 marks) Your program produces incorrect output for most of the test cases. You cannot explain your code or cannot answer questions about the effect of changing it.
Analysis and design	(15-20 marks) Your class model has a minimal number of classes and makes your code clear and concise.	(10-14 marks) You use 3-5 classes that are unnecessary, redundant, or in some way complicate your design.	(0-9 marks) In more than five places, you choose inappropriate classes.

The demonstration must be completed within 1 week of the due date. The student must demo the code which was submitted to the TA. The TA has the right to assign a mark as low as 0 as a final grade for the whole assignment if he is not satisfied with the demonstration portion of the assignment. The TA may deduct up to 5% from the assignment's final mark for errors in spelling and grammar.

Demonstration

Your TA will execute your program several times to ensure that it is behaving as specified above using a variety of test documents (all under 500 KB in size).

Documentation

For each class, you should minimally document:

- your name;
- the purpose of the class;
- the date you started writing the class;
- each instance variable;
- each class variable;

For each method, you should minimally document:

- the purpose of the method;
- the input parameters;
- the output/return values; and the algorithm used if it is not obvious from the code.

Working Together

If you decide to work with someone from the class or to use resources that you found online or in a book (besides the course textbooks), you **must** cite these sources. When handing in your assignment, please specify who you worked with and list these sources. You will be required to demonstrate your knowledge of how the code performs its task to the TA to get full marks on the assignment. If the TA feels that you do not fully understand what you have written, he may decide to reduce your assignment mark. An example question that the TA could ask would be "what would happen if I changed this line of code to this?" (explains the change) or "why did you choose to include this class in your design?"

Handing in your assignment

For this assignment, email your programs to your TA on or before the due date. Be sure to include all files with the .java extension (including those provided to you that you make use of) as well as a description of how to run your program (i.e., which class contains your 'main' method). Make sure that your email client program saves a copy of the email you send to your TA. In the event of email problems, we need the header information from your original email to ensure that you submitted your assignment on time.