CS3283 Assignment #3

12th March 2004

Due by 5pm on Friday, April 9, 2004
Delivered to Hugh

You may work in groups of (upto) three students. Do not bother asking for a group of four. The group members do not have to be from your tutorial group, and it is up to you to ensure that everyone contributes equally. Email Hugh with your group members before March 19th 2004, and select your task from the list below.

This assignment is worth 40% of your assignment mark. It consists of the development or modification of a Java program, with some documentation.

Your task...

You have three possible tasks. You can do any one of them, but I should tell you now that you may find some of them HARD, and I am giving them to you as a challenge:

Task #1: Reimplement *your* Tcl/Tk assignment #2 as a Java/Swing program. It must look and feel as close to your Tcl/Tk program as possible.

> *If you intend to take up this challenge, come and talk to me first.*

Task #2: Implement a program to assist in the (most likely 2D) visualization of some large dataset. The program will show a display plotting values of two related variables selected from a large dataset, and then allow you to select the size of a hexagonal tile (by moving a slider). The display then plots the tiles, shaded according to the number of points contained within the tile. If the tile is 1 pixel wide, then the display will be as for the original display, but if you set the tile size to (say) 100, then all datapoints in that hexagonal region will be coloured a shade of grey determined by the number of points.

> *If you intend to take up this challenge, come and talk to me first.*

Task #3: Your task is to implement and document a Java *application* or a Java *applet* which provides a user interface for a tool to assist in the management of *large* numbers of images. This application/applet is likely to principally display TEXT information (perhaps in a spreadsheet form), but may also display small (thumbnail) versions of the images if you wish. The file saved and restored by this application will be a database of some sort, which records the file names of images, and other information, as described below.

The interface is a special purpose editor, intended to manage the process of classifying, annotating and querying a large number of images. Each image can be classified into one or more named sections. Each section may be classified into one or more other named sections. For example, the image DSCN0100.JPG may be a picture of my friend Tim at a party. This may be classified in the following three sections:

- "*Friends*"
- "*Trip to NZ in Dec 2003*", which is itself in the section "*Trips*"
- "*Hooligans*"

The user of your application will be using a main screen which shows a list of images. This screen may just show a text listing of a file used to store the database, or it may be done in a spread-sheet like manner. The user of your application will be able to load new databases (which may of course just be files), to insert new databases, to create new, copy and delete database entries and to save back the resultant database. In addition you must have a help menu or button.

Each image/section has editable and fixed annotation fields which provide extra background information about that image. The information should include:

- The date and time the image was entered into the section (not editable).
- A unique identifier for the image
- A scrollable text box with (say) 5 visible lines of text description.

The main screen should allow you to display the images in a section, one line per image, with the filename, section identifiers and with truncated versions of the above fields. The images will have an ordering, which you can change using some sort of cut and paste technique. For fields that are too long, you may truncate the field when you display it.

Field editing: When you select a particular line on the main display, a Java/Swing dialog box is to be created with editable fields for each of the editable information fields given above. After editing the fields, you can choose to save the data back, and this is reflected in the original screen.

At any time you can query the database, and produce a (TEXT) display which shows all images that have annotations that match some specified query. This query may just be a text substring match query - For example if I had annotated the image DSCN0100.JPG in Hooligans with a description "Here is Tim wearing a silly hat.", and then I queried for "Tim", I should retrieve and display this database entry. The resultant display should show the image information as for the normal display.

Notes:

The following points form part of the functional specification of the user interface.

1. At any time you should be able to save and restore the state (i.e. the system is persistent)
2. The program should be safe on failure - that is - if some part of your program dies for some reason, you can re-run the program, and get back to where-you-were.
3. The minimum flow of operation of the interface is that you can
   (a) create, locate and delete new sections,
   (b) import image(s) into a section, using selection or cut and paste.
   (c) edit image/section information annotations,
   (d) save and load new databases,
   (e) query the system with a text search.

Deliverables:

You are to present your assignment as a single (zipped) file containing the sourcecode and a README file outlining how to run your program, along with an electronic version of the documentation in PDF format. The documentation can be minimal, must also be presented on paper, and should contain:

- A title page containing your names and matriculation numbers.
- Table of contents...
- A one page introduction describing the application function in a brief non-technical style.
- A one page section describing the system, (such as how you store the database, file formats and so on).
- A one to three page section describing the interface design (what are the screens, what is the general flow of operation of each screen.

Note that this assignment *does* require you to *implement* the application, and it must be implemented in Java/Swing, and (preferably) runnable on the Suns.

Assessment:

The assessment is as follows:

Documentation	15%
Code style/quality	35%
Operation of the interface	50%

Try to achieve clarity in your writing and take care in the structuring of the document.

COOPERATING VERSUS CHEATING

You are allowed to discuss the problems with your friends, and to study any background material with them, but the assignment *should be your own group's work*. **Copying** and **cheating** will be grounds for failing the assignment.

REUSE

I am amenable to your re-use of existing code that you might find somewhere, as long as you

- Have permission from the author
- Clearly credit the author of the code, both in the source, and in your documentation
- It contributes less than 50% of the code you present to me (measured by lines-of-code)