MSc Java Lab - Using the API - Day 2

Robert Chatley and William Lee

1 Aims

The aim of this lab is to allow you to gain experience using some of the classes from the Java standard library, and also to make use of a more sophisticated development environment, Eclipse. We will perform some data manipulation tasks, reading, processing and outputting data in a number of different ways. We suggest you work in pairs.

2 Getting started

Use a Linux machine (Eclipse is available for Windows but we are not sure if it installed in the lab. For simplicity and consistency we will specify the lab exercise in the Linux environment).

Throughout the exercise you will find it helpful to refer to the Java API documentation. This is available at http://www.doc.ic.ac.uk/csg/java/1.4docs/api/

Launch Eclipse by typing `eclipse &`
When prompted for a workspace, specify a new directory somewhere inside your home directory.
If this is the first time you have used Eclipse you will see a screen with several icons, choose **launch workbench**.
From the File menu choose New `->` Project... and choose a Java Project.
Enter a project name, click Next and Finish.
Your project should appear in the Navigator, open the folder.
Right click on your project, click New `->` Class.
Give the class a name, tick creation of a main method and click Finish.
As a test, add code to print out a message.
Save your file (ctrl + s).
From the Run menu, select Run As... `->` Java Application.
Your message should be displayed in the console.
From now on you can run your program by just pressing the Run button (green circle with white arrow).

3 Reading a File

We have prepared some data files for this exercise. You can download them from

http://www.doc.ic.ac.uk/~rbc/java.

File1.txt contains a list of words, one on each line.

Using the techniques shown in the lecture, write some code to open the file, read each word, store it in some sort of collection, then print out the contents of the collection on the console. You may wish to look at the API documentation for the following classes: File, FileReader, BufferedReader, List, ArrayList, Iterator.

Look at the documentation for the class Collections. Can you print out the words from the file in alphabetical order?

Try making use of Eclipse's suggestions when your code doesn't compile. This can speed up the development process.

3.1 A more complex file

File2.txt contains more text, but with more than one word on each line. Change your program to read and sort the words from this file instead. In order to divide each line of the file up into separate words, try using a StringTokenizer.

Can you find the total number of words?
What if we wanted only distinct words?

3.2 A difference source

You might like to see if you can change your program to read File2.txt directly from the web rather than from your downloaded copy of the file. Look at java.net.URL and how to obtain an InputStream from a URL.

4 GUI

Instead of printing the results out on the screen, how about putting them in a window? Look at the slides and documentation for JFrame, JPanel, JTextArea. Can you create a window that shows the output?

How about using a JFileChooser to identify which file you want to process?