CS 420 Fall 2016 Laboratory 3
Binary I/O and Sequential I/O

Purpose

The last lab focused on basic I/O techniques in Java. We learned that while sequential text-driven I/O is simple to use, it does not necessarily provide a framework for efficiently storing data in a file. This lab provides practice exercising more advanced I/O concepts such as object serialization or random access in binary files using Java. We will rewrite the bully/nerd I/O code written for lab 2. My main goal for this lab is to demonstrate the convenience of object serialization and the use of random-access I/O for binary files.

Grading

This assignment is worth 5% of your final grade – it will be evaluated on a 20 point scale. The breakdown of points is shown at the end of this document.

Due Date

This lab is due Wednesday, September 21. You may turn it in on Laulima. Turn in a zipped or rarred copy of your NetBeans project folder. You may either work alone or in pairs for this project. If you choose to work with a partner, only one of you needs to turn in the file, but you must make the names of both partners obvious with your submission!

Task 1 – Rewrite I/O routines of Lab 2 using object serialization

You had three tasks in lab 2: write nerds to a text file, read the nerds from a text file, and update two nerds in the text file. This time, you will re-implement your code to write nerd objects directly to a *binary* file as in task 1. Close this file, then open the file, read in the objects, and display them to the screen (using their toString methods). You can skip the third task from lab 2. Use the `ObjectOutputStream` and `ObjectInputStream` classes for this task (see the slides for Java I/O that I posted on Laulima). Don't forget to modify your nerd classes so that they can be written to and read from files.

Task 2 – data I/O to a binary file

Write a method that uses a `DataOutputStream` object to save the contents of a two dimensional array to a binary file. Test your method with the following call (call the `task2_write` from the appropriate object if necessary):

```
double[][] x = {{10.34,23.567,61.2}, {-12.0,200.34,30e2}};
task2_write( x );
```

Write a second method `task2_read` that uses a `DataInputStream` object to read the values from this binary file and display them. Use a `JFileChooser` object to prompt the user for the choice of input file and output file.

Task 3 – manipulate a random-access binary file

Write a method `task3` that opens a file for random access, writes the values of the integers 10, 20, 30, … 100 into the file and closes it. Then re-open the file, retrieve the 5th (50) and 10th (100) items, replace them with their negative values, and close the file. Finally, open the file one final time and read and display the contents of the file on the screen. Again, use a `JFileChooser` object to prompt the user for the choice of file name. Use the Java I/O class `RandomAccessFile` to accomplish this task.

Note: I have posted a helpful set of slides: Exceptions and Advanced IO on Java. They will help you accomplish these tasks.

Points breakdown

[6 pts] task 1 – nerd objects are written to and read from the file correctly
[7 pts] task 2 – methods to read and write to binary files with data streams work correctly.
[7 pts] task 3 – methods for manipulating random-access binary files work correctly.

Comments count folks. Your java files should have headers with your name(s) and a description of the file. Methods should have useful comment blocks.

Submission

On Laulima, turn in a zip file containing either your java source code files or an entire NetBeans project folder containing your solution.

Hint

Here's a tutorial on using the `JFileChooser` class:

http://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html