Computing and Algorithms II: Programming Assignment 4

CS-102 **Fall 2008**

In this programming assignment, you will demonstrate your knowledge of dynamic data structures (in particular, binary search trees) by modifying the airport abbreviation code program from Programming Assignment 3 to incorporate a new dynamic data structure.

New Internal Requirements

For this assignment, you should replace your Database class with another implementation which uses *two dynamic binary search trees* to organize the information within the database. Each tree will be organized as before; one will be sorted by abbreviation, while the other will be sorted by name. As before, you may use any variations on linked lists or binary search trees (dummy head nodes, extra pointers, *etc.*) which you might find useful.

New Functional Requirements

Your program will be an extension of Programming Assignment 3, and thus should operate in the same manner as that assignment, unless otherwise specified herein. In particular, this means that any errors present in your submissions for Programming Assignment 3 should be fixed for this assignment.

The following new requirements should be implemented as well:

- A new option should be added to the main menu which allows the user to write the entire contents of the database to disk. If selected, the program should prompt the user for the name of a file in the current directory to be used for writing the desired information. The program should then write all records from the database to those files, *in the same format used for input.* That is, it should be possible to use the output of this command as input to the program at some future time.

  The data should be written using a *pre-order traversal of the abbreviation tree.* (This permits the read-from-file command to recreate that tree using the same structure.)

- A new option should be added to the main menu which allows the user to re-initialize the database from a file. If selected, the program should prompt the user for a file name. If a file does not exist, an error message should be issued and the program should leave the current database unmodified. If the file does exist, the program should delete the current (internal) database and create a new database based upon the file, as if the program had been originally started with that file as its command-line argument.

Submitting Your Program

Before 11:59:59 p.m., Monday, 1 December 2008 (9th Monday), you must send a MIME-encoded email message to `jhuggins@kettering.edu` containing all source code files for your program, including a class named `Prog4` containing a `main` method.

In addition, you must deliver to the instructor a printout of your program files at the start of class on Tuesday, 2 December 2008 (9th Tuesday).

Notes

1. Note that there is no class on 28 November (8th Friday), due to the Thanksgiving holiday. The program is due immediately after the holiday weekend; get an early start on the program before then.

2. In order to write to a file, the `java.io.FileWriter` and `java.io.BufferedWriter` classes may be used. See the Java API for details. Note the following hints:

   - To write a line to a file, you should use a combination of `write()` (which can write a string to the file) and `newLine()` (which writes an end-of-line character to the file).
   - Once you are finished writing to a file, use the `close()` method to flush the output buffer to the disk. (Otherwise, data may be lost.)
   - You may find it useful to use the `flush()` method to flush the output buffer to the disk without closing the file. (If your program crashes before reaching a `flush()` or `close()`, your output file may not give you an accurate picture of how many records have been processed.)