Lab 7

File I/O and Sorting

Introduction

Java is an extremely rich language when it comes to File I/O. There are two basic types: ByteStreams and CharacterStreams. The great majority of code has been written using ByteStreams. ByteStreams just assume that the file consists of a sequence of bytes. A textfile like one produced in Notepad is just a sequence of bytes, with each byte representing a character in Ascii code. So, if your file had the letter 'N' written in it, then its' representation in the actual file would be the number 78 (Ascii code for 'N') in one byte. Even the characters that represent a line-feed are represented in an Ascii code.

Of course, the file may also represent whatever you created it to be. It may be a ByteStream file full of numbers that are in 2's complement representation. This all gets confusing very fast and is why you must read the chapter!

The Lab

In the same directory you find this lab assignment, you will also find two files:
readfile.java
classlist.TXT

readfile.java is a java program I wrote that looked at a Synergy generated text file of class members and information about them. In the program it shows what one line of that file I used looked like. My program then looked at each byte in that file and pulled out the information needed. I used various string methods just by looking them up in Oracle (I googled java String methods) and used them to pull apart each line in the file. In the original file, each line was delineated by line-feed character (or Ascii 10). **Use this program as a learning tool to figure out how to write your program.** The output of this program was done using CharacterStreams. The input was done using a ByteStream. This was done just to show you the variety of I/O in java.

classlist.TXT is a file that my program **created** and holds on each line information about each student. You can read this file in Notepad if you want. This will be the **input** file to your program.

Program 1:

Write a program which reads in the classlist.TXT file. Count up the number of students in each grade level and print that number out. Also, count the number of girls vs. boys. Finally, count up how many of each 1st letter of the last name. (In other words, how many students' last names start with 'A', how many start with 'B', etc.)

Your output should look like this (with the correct numbers!):
Freshman : 15
Sophomores: 22

Juniors: 56
Seniors: 70

Boys: 83
Girls: 82

Last names beginning with:
A : 21
B: 15
C: 22
etc.

Hint: To keep track of the last name issue, create a 26 cell array of integers. The array must be indexed from 0 to 25 of course, but if you know the asci code for A which is 65, and each next capital letter is the next number, then suppose your input buffer has the character 'B' in it. It will be represented by the number 66. Subtract the Ascii number for 'A' off (66-65 = 1) which gives you the index into your counting array. This is a very common technique in Computer Science for counting frequencies of letters.

Program 2

This program is just like the first program, but you need to create an array of students and students should be defined exactly as how I did it in the program readfile.java (my sample program). Then, **sort** the program using the bubblesort algorithm shown in the book and the sort should be based on the student's last name. Create a new file with all the information preserved, but sorted correctly.

Also, sort the same classlist.TXT file but this time sort based on the student number. Print the new file out sorted by student number. These files must be produced to disk and they must be in character format so I can read them.