Searches and Sorts

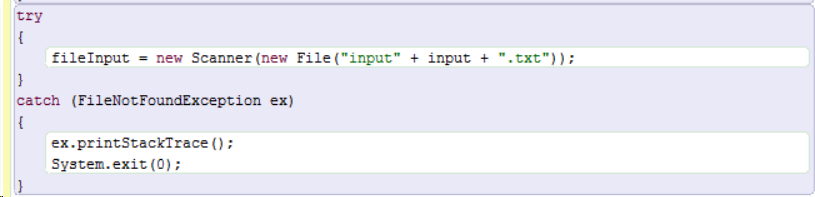
Part I: Sorts

**Overview**: Create a program that sorts an input file using selection, insertion and merge sorting algorithms. TO GET THE FULL BENEFIT OF THIS ACTIVITY DO NOT DIVE INTO TO LOOKING FOR CODE THAT HAS ALREADY BEEN CREATED, TRY TO FIGURE IT OUT YOURSELF. This will give you stronger foundational understanding of the algorithms. Of course if you get stuck you can look to some examples…but always look – put away – and come back to the code you are writing. Don’t just transcribe.

**Interface**: The program should ask the user which input file to use and which algorithm they would like to use to sort the file data.

**Implementation**:

-The program must read in an input file. (To practice for work we will do later in the year, it should handle input that would produce an error. Also catch exceptions; see snip:



import java.io.\* would be necessary for this to run.)

-Use an array to hold the incoming numbers

-Provide methods for a selection sort, insertion sort and merge sort.

-Output the first 10 numbers of the sorted entry, the last 10 numbers of the sorted entry, the sort method used and how long the sort took from after reading the file in to having the numbers sorted. (Use System.currentTimeMillis( ); this is type long)

-Output a table showing the input file (columns) and the sort method (the rows) and how long it took to sort the data in the file, for each file and each sort method.

-If you have time, look up another sort method and implement that into this comparison.

Part II: Searches (optional: for those that have time to do the logical extension)

Let me check to see that you have the sorts working before going on to searches.

Create a Search class with two search methods, sequentialSearch and binarySearch. Both search methods receive an array and a target String (for this assignment it can be a target int if you prefer) and return an index value for the target in the array. The methods return -1 if the target does not appear in the array. The binarySearch can accept ONLY sorted arrays (you should run a check for this in the method); whereas, sequential search can handle any array.

(If you want to get fancy, you can return an ordered list of indices for all of the places the target appears in the array.)

After you have them all working, output a table as you did for sorts with the search type as rows, the fileNames as columns and the cells holding the length of time of the searches.