<< SortFuncs.java >>

```
public class SortFuncs {
	/*
	 * This is a modified implementation of insertion sort enabled
	 * to sort chars instead of integers
	 * @param in array to be sorted
	 */
	public static void charSort(char[] in) {
		for(int i = 1,j; i<in.length; i++) {
			char temp = in[i];
			for(j = i; (j>0)&&(temp<in[j-1]); j--)
				in[j] = in[j-1];
			in[j] = temp;
		}
	}
	/*
	 * This is an adapted verison of the insertionsort from the lectures
	 * to be able to sort through String objects
	 * @param in object that contains the String to be sorted
	 */
	public static void insertionSort(CustomList in) {
		for(int i=1, j; i<in.size(); i++){
			String temp = in.getData(i);
			for(j=i; (j>0)&&((temp.compareTo(in.getData(j-1)))<0); j--){
				String tmp = in.getData(j-1);
				in.set(tmp,j);
			}
			in.set(temp,j);
		}
	}
	/*
	 * This is an adapted version of quicksort from the lectures
	 * to be able to sort through String objects
	 * @param lo first element
	 * @param hi last element
	 * @param in array to be sorted
	 */
	public static void quickSort(int lo, int hi, CustomList[] in) {
		int first = lo, last = hi;
		CustomList temp;
		// set a pivot element
		String pivot = in[(lo+hi)/2].getHead().data;
```

```
		// divide arrays
		while(first<=last){
			// identify a number greater than pivot value from left subarray
			while((in[first].getHead().data).compareTo(pivot) < 0){
				first++;
			}
			// identify a number less than pivot value from right subarray
			while((in[last].getHead().data).compareTo(pivot) > 0){
				last--;
			}
			// swap
			if(first<=last){
				temp = in[first];
				in[first] = in[last];
				in[last] = temp;
				first++;
				last--;
			}
		}
		// recursive method call
		if(lo < last)
			quickSort(lo, last, in);
		if(first < hi)
			quickSort(first, hi, in);
	}
}
```

<< CustomList.java >>

```
public class CustomList{
	/*
	 * Defines the contents of each element of the array
	 */
	public class Node{
		String data;
		Node next = null;
	}
	/*
	 * Head pointer
	 */
	private Node head;
	/*
	 * Size of the list
```

```
 */
private int size;
/*
 * Constructors
 */
public CustomList() {
	head = null;
	size = 0;
}
public CustomList(String text) {
	addFront(text);
}
/*
 * Returns the size of the list
 */
public int size() {return size;}
/*
 * Returns the head pointer
 */
public Node getHead() {return head;}
/*
 * Adds a node in the beginning of the list and increases
 * size by 1.
 * @param text specifies data inside the node
 */
public void addFront(String text) {
	Node temp = new Node();
	temp.data = text;
	temp.next = head;
	head = temp;
	size++;
}
/*
 * Retrieves the data of the node in the nth location
 * @param n specifies the location of the node
 */
public String getData(int n) {
	if((n<0)||(n>=size)) {
		System.err.println("Invalid access. Program will now exit");
		System.exit(0);
	}
	Node temp = head;
```

```
            for(int index = 0; index<n; index++)
                  temp = temp.next;
            return temp.data;
      }
      /*
       * Changes the data within the nth node to text
       * @param text the new data
       * @param n the position of the node
       */
      public void set(String text, int n) {
            if((n<0)||(n>=size)) {
                  System.err.println("Invalid index. Program will now exit.");
                  System.exit(0);
            }
            Node temp = head;
            for(int index = 0; index<n; index++)
                  temp = temp.next;
            temp.data = text;
      }
}
```

<< Anagram.java >>

```
import java.io.*;
import java.util.Arrays;

public class Anagram {
      /*
       * User defined list to contain anagram matrix
       */
      CustomList[] wordMat;
      /*
       * File name of the input text file
       */
      String fileIN;
      /*
       * File name of the output text file
       */
      String fileOUT;
      /*
       * Time measurement fields
       */
      double start, stop, totalStart, totalStop;
      /*
```

```
 * File printing field
 */
PrintWriter cursor;
/*
 * Input size and storage array size
 */
int arraySize, lines;
/*
 * Read input text file and store into custom list
 */
public void readInputFile() throws IOException{
	BufferedReader buffer = new BufferedReader(new FileReader(fileIN));
	String data;
	arraySize = 0;
	// scans the input file by checking if the next characters is an EOL
	double now, later;
	now = System.nanoTime();
	while((data = buffer.readLine()) != null) {
		if(!isAnagram(data)) {
			wordMat[arraySize] = new CustomList(data);
			arraySize++;
		}
	}
	later = System.nanoTime();
	cursor.println("The method to determine if two words are anagrams took "+(later-now)+" nanoseconds.");
	buffer.close();
}
/*
 * Identifies the number of words to read from the file
 */
public void numberOfWords() throws IOException {
	BufferedReader reader = new BufferedReader(new FileReader(fileIN));
	lines = 0;
	while (reader.readLine() != null) {
	    lines++;
	}
	reader.close();
}
/*
 * Determines if two words are anagrams of each other
 */
public boolean isAnagram(String text) {
```

```
		char[] inputAsChar = text.toCharArray();
		SortFuncs.charSort(inputAsChar);
		// String inputText = inputAsChar.toString();
		for(int i = 0; i<arraySize; i++) {
			char[] currentAsChar = wordMat[i].getHead().data.toCharArray();
			SortFuncs.charSort(currentAsChar);
			// String currentText = currentAsChar.toString();
			if(Arrays.equals(inputAsChar, currentAsChar)) {
				wordMat[i].addFront(text);
				return true;
			}
		}
		return false;
	}
	/*
	 * Prints the output to a file
	 */
	public void printToFile() {
		try {
			cursor.print("This is the sorted list of anagrams.\n");
			// goes through all the pointers
			for(int i = 0; wordMat[i] != null; i++) {
				// goes through all the contents within pointer[i]
				for(int j = 0; j < wordMat[i].size(); j++) {
					// prints to file
					cursor.print(wordMat[i].getData(j) + " ");
				}
				cursor.println();
			}
		}
		catch(Exception e) {
			e.printStackTrace();
			System.out.println("File does not exist.");
		}
	}

	public static void main(String[] args) throws IOException {
		Anagram sample = new Anagram();

		sample.fileIN= args[0];
		sample.fileOUT	= args[1];
		sample.cursor = new PrintWriter(sample.fileOUT);
```

```
            sample.totalStart = System.nanoTime();
            System.out.println("The program has started.");
            sample.cursor.println("The program has started.");

            sample.start = System.nanoTime();
            sample.numberOfWords();
            sample.wordMat = new CustomList[sample.lines];
            sample.readInputFile();
            sample.stop = System.nanoTime();
            sample.cursor.print("Reading the input file took "+(sample.stop-sample.start)/1000000000.0+" seconds.\n");

            sample.start = System.nanoTime();
            for(int i = 0; i<sample.arraySize; i++)
                  SortFuncs.insertionSort(sample.wordMat[i]);
            sample.stop = System.nanoTime();
            sample.cursor.print("Sorting each row took "+(sample.stop-sample.start)/1000000000.0+" seconds.\n");

            sample.start = System.nanoTime();
            SortFuncs.quickSort(0,sample.arraySize-1,sample.wordMat);
            sample.stop = System.nanoTime();
            sample.cursor.print("Sorting the rows took "+(sample.stop-sample.start)/1000000000.0+" seconds.\n");

            sample.start = System.nanoTime();
            sample.printToFile();
            sample.stop = System.nanoTime();
            sample.cursor.print("Printing output to file took "+(sample.stop-sample.start)/1000000000.0+" seconds.\n");

            System.out.println("The program has ended.");
            sample.totalStop = System.nanoTime();
            sample.cursor.print("Processing the input file that contains "+sample.lines+" words took "+(sample.totalStop-
sample.totalStart)/1000000000.0+" seconds.\n");

            sample.cursor.println("The program has ended.");
            sample.cursor.close();

            System.out.print("Processing the input file that contains "+sample.lines+" words took "+(sample.totalStop-
sample.totalStart)/1000000000.0+" seconds.\n");
      }
}
```