Lab #3 CoSc 20203 Spring 2021

Word Verifier
Due Tue, 8 April

This assignment is designed to give you experience in using recursion and in using RandomAccessFile. The program must accept a string of letters and determine whether the string is a word in a dictionary.

Since we want our program to be able to run on low-end machines without much memory, the dictionary will not be loaded into an internal data structure. Instead, the dictionary will be implemented as a random access, binary file of char, called *words*. We want to be able to search the *words* file without having to read a large portion of the file each time. This is possible by using a random access, binary file of int, called *index*, that indexes into the *words* file; the *n-th* value in *index* gives the byte offset of the *n-th* word in the *words* file. Since ints in Java are always 4 bytes long, we can do binary search using the index file to specify a word in the *words* file. It is a requirement of this lab that you do a recursive binary search of the *words* file using the *index* file.

To provide an example of these files, consider a dictionary that contains only the words "a", "list", and "the". (All characters are lower case.) Characters are stored in the two-byte unicode format. The following shows the hex equivalent of the raw bytes stored in the *words* file. The character equivalent of each pair is shown beneath that.

words file

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
hex	00	61	00	6C	00	69	00	73	00	74	00	74	00	68	00	65
char	a		l		i		s		t		t		h		e	

The corresponding *index* file would contain three int values specifying the byte offest of the beginning of each word plus an additional int value specifying the byte offset of where the next word would begin if one were added to the end of the *words* file. The following shows the hex equivalent of the raw bytes stored in the *index* file. The decimal equivalent of each int is shown beneath that.

index file

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
hex	00	00	00	00	00	00	00	02	00	00	00	0A	00	00	01	00
dec	0				2				10				16			

Code for the GUI is attached. As usual, do not change anything above the row of asterisks. Your program will be graded on documentation and code quality as well as functionality.

Requirements:

- A recursive binary search that searches the dictionary on disk.
- Do not read the entire dictionary into memory.

```
import javax.swing.*;
import java.awt.event.*;
import java.io.*;

public class Lab3 extends JFrame implements ActionListener {
	static final long serialVersionUID = 1;
	JLabel stringLabel = new JLabel("String");
	JTextField stringText = new JTextField(10);
	JButton findButton = new JButton("Find");
	JTextArea result = new JTextArea(20,40);
	JScrollPane scroller = new JScrollPane();
	RandomAccessFile indexFile = null;
	RandomAccessFile wordsFile = null;
	public Lab3() {
		setTitle("Word Check");
		setLayout(new java.awt.FlowLayout());
		setSize(500,430);
		setDefaultCloseOperation(DISPOSE_ON_CLOSE);
		add(stringLabel);
		add(stringText);
		add(findButton); findButton.addActionListener(this);
		scroller.getViewport().add(result);
		add(scroller);
	}
	public static void main(String[] args) {
		Lab3 display = new Lab3();
		display.setVisible(true);
	}
	void getFiles() {
		String indexName = getFileName("Index File");
		if (indexName == null) {
			result.setText("No index file chosen");
			return;
		}
		String wordsName = getFileName("Words File");
		if (wordsName == null) {
			result.setText("No words file chosen");
			return;
		}
		try {
			indexFile = new RandomAccessFile(indexName, "r");
		} catch (FileNotFoundException e) {
			result.setText("Index file not found");
			return;
		}
		try {
			wordsFile = new RandomAccessFile(wordsName, "r");
		} catch (FileNotFoundException e) {
			result.setText("Words file not found");
			return;
		}
	}
	String getFileName(String title) {
		JFileChooser fc = new JFileChooser();
		fc.setDialogTitle(title);
		int returnVal = fc.showOpenDialog(this);
		if (returnVal == JFileChooser.APPROVE_OPTION)
			return fc.getSelectedFile().getPath();
		else
			return null;
	}
	public void actionPerformed(ActionEvent evt) {
		if (indexFile == null || wordsFile == null)
			getFiles();
		try {
			search();
		} catch (IOException e) {
			result.setText("I/O eror occurred");
		}
	}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/* Do not change anything above this line                  */
/* You must implement the search method as described in    */
/* the assignment                                          */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

	void search() throws IOException {

	}
}
```