```java
/**
  * This is my code! ItâM-^@M-^Ys goal is to search through documents and return
 which
  * documents contained the word given
  * CS 312 - Assignment 9
  * @George Haff
  */

import java.util.HashMap;
import java.util.HashSet;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Collections;
import java.io.File;
import java.io.IOException;

// Creates a list of words not to be seen by the search engine
class Stoplist {

    protected ArrayList<String> stoplist;

    /*
     * purpose: Creates the stoplist from the file
     * input: String filename
     * result: updates the stoplist
     */
    Stoplist(String filename) throws IOException
    {

        stoplist = new ArrayList<>();
        File f = new File(filename);
        Scanner file = new Scanner(f);

        try {
            while (file.hasNextLine())
            {
                file.useDelimiter("[^a-zA-Z]+");
                stoplist.add(file.nextLine().toLowerCase());
            }
            file.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    /*
     * purpose: Retrieves the stoplist
     * result: returns the stoplist as a list
     */
    public ArrayList<String> getStoplist()
    {
        return stoplist;
    }
}

// Contains all of the documents and stores which words have been in
// which documents
class InvertedIndex {

    protected HashSet<String> containIn;
    protected HashMap<String, HashSet<String>> docIndex;
    protected Stoplist stoplist;
    protected ArrayList<Document> documents;

    /*
     * purpose: Initializes all of the lists and hashes
     * input: String filename
```

```java
     * result: updates of the all lists and hashes
     */
    InvertedIndex(String filename) throws IOException
    {
        docIndex = new HashMap<>();
        stoplist = new Stoplist(filename);
        containIn = new HashSet<>();
        documents = new ArrayList<>();
    }

    /*
     * purpose: Adds document to the list
     * input: Document d
     */
    public void addDocument(Document d)
    {
        d.ridStopWords(stoplist);
        documents.add(d);
    }

    /*
     * purpose: Searches each document for a single word, updates a list
     *   if a word is found and in which document, and outputs
     *   contents of the documents if debug is true
     * input: String query, boolean debug
     * result: updating inverted index if word is found
     */
    public String singleWordQuery(String query, boolean debug)
    {

        query = query.toLowerCase();
        containIn = new HashSet<>();
        StringBuilder contents = new StringBuilder();

        for (Document d : documents)
        {
            if (d.hasQuery(query)) {
                containIn.add(d.documentName());
                if (debug)
                    contents.append(d.debug());
            }
        }

        if (!(containIn.size() == 0))
            docIndex.put(query, containIn);

        StringBuilder s = new StringBuilder("--- found in "
                + containIn.size() + " documents\n");

        if (containIn.size() != 0)
        {
            for (String s1 : containIn)
            {
                s.append(s1 + ",");
            }
        }

        return s.toString() + "\n" + contents.toString();
    }

    /*
     * purpose: Searches each document for a phrase, updates a list
     *   if the phrase is found and in which document, and outputs
     *   contents of the documents if debug is true
     * input: String query, boolean debug
     * result: updating inverted index if phrase is found
     */
    public String multiWordQuery(String query, boolean debug)
    {
        String[] list = {};
        list = query.toLowerCase().split(" ");
        containIn = new HashSet<>();
        StringBuilder contents = new StringBuilder();

        for (Document d : documents)
        {
            if (d.hasQuery(list)) {
                containIn.add(d.documentName());
                if (debug)
                    contents.append(d.debug());
            }
        }

        if (!(containIn.size() == 0))
            docIndex.put(query, containIn);

        StringBuilder s = new StringBuilder("--- found in "
                + containIn.size() + " documents\n");

        if (containIn.size() != 0)
        {
            for(String s1: containIn)
            {
                s.append(s1 + ",");
            }
        }

        return s.toString() + "\n" + contents.toString();
    }

    /*
     * purpose: Dumps the contents of the inverted index
     * result: Outputting the contents of the inverted index
     */
    public void debug()
    {
        System.out.println("The inverted index contains " + docIndex);
    }
}

// Class that breaks down the file into a list of Strings
class Document
{

    protected String name = "";
    protected ArrayList<String> originalText;
    protected ArrayList<String> editedText;

    /*
     * purpose: Creates the document from the filename
     * input: String fileName
     * result: lists and name of the document are updated
     * if the file exists
     */
    Document(String fileName) throws IOException{
        name = fileName;
        editedText = new ArrayList<>();
        originalText = new ArrayList<>();

        File f = new File(name);
        Scanner file = new Scanner(f);
        int i = 0;
        String [] temp = {};
        String s = "";

        try {
            while (file.hasNextLine())
            {
```

```
                originalText.add(file.nextLine());
                s = originalText.get(i).toLowerCase();

                // The Delimiter didn't work with this for some reason
                // It also doesn't work on the bee movie script for some reason
                temp = s.replaceAll("[^a-zA-Z]", "").split(" ");
                Collections.addAll(editedText, temp);
                i++;
            }
            file.close();
        }

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    /*
     * purpose: Strips the document of useless words form the stoplist
     * input: Stoplist stoplist
     * result: editedText list is either smaller or the same size
     */
    public void ridStopWords(Stoplist stoplist)
    {
        ArrayList<String> stop = stoplist.getStoplist();
        for (String s : stop)
        {
            editedText.remove(s);
        }
    }

    /*
     * purpose: Checks if the documents contain the single word query
     * input: String query
     * result: true if it is found, false otherwise
     */
    public boolean hasQuery(String query)
    {
        return editedText.contains(query);
    }

    /*
     * purpose: Checks if the document contains the multi word query
     * input: String array list
     * result: true if all are found, false otherwise
     */
    public boolean hasQuery (String [] list)
    {
        for (String s : list) {
            if (!this.hasQuery(s))
                return false;
        }
        return true;
    }

    /*
     * purpose: Retrieves the name of the document
     * result: name of the document is returned
     */
    public String documentName()
    {
        return name;
    }

    /*
     * purpose: Dumps the original content of the document
     * result: returns the original content of the document
     */
```

```java
    public String debug()
    {
        StringBuilder s = new StringBuilder(name + " contains: \n");
        for (String s1 : originalText)
        {
            s.append(s1);
        }
        return s.toString();
    }
}

// The command line interface of the program
public class CLI
{

    protected boolean debug;
    protected InvertedIndex ii;

    /*
     * purpose: Process the user’s commands
     * input: The command arguments
     * result: display what further the user must do
     */
    public void process(String[] args) throws IOException
    {

        debug = false;

        if (args.length > 1)
        {
            long startTime = System.currentTimeMillis();

            if (args[0].equals("-d"))
            {
                debug = true;
                ii = new InvertedIndex(args[1]);
                for (int i = 2; i < args.length; i++)
                {
                    Document d = new Document(args[i]);
                    ii.addDocument(d);
                }
            }

            else
            {
                ii = new InvertedIndex(args[0]);
                for (int i = 1; i < args.length; i++)
                {
                    Document d = new Document(args[i]);
                    ii.addDocument(d);
                }
            }

            // Taking how long it took to handle the documents
            long stopTime = System.currentTimeMillis();
            long elapsedTime = stopTime - startTime;
            System.out.println("@@ Adding documents took "
                    + elapsedTime + " ms");
            Scanner scan =  new Scanner (System.in);
            String taken = "";

            while (!taken.equals("-stop"))
            {
                System.out.println("What word or phrase would you like to look for? > ");
                taken = scan.nextLine();

                if (!taken.equals("-stop")) {
                    if (taken.equals("@@debug"))
                        ii.debug();
```

```java
                    else if (taken.contains(" "))
                        System.out.println(ii.multiWordQuery(taken, debug));
                    else
                        System.out.println(ii.singleWordQuery(taken, debug));
                }
            }
        }

        else
        {
            System.out.println("Usage: java CLI [-d] stoplist documents" +
                    "\n\t-d         Displays contents of the document" +
                    "\n\tstoplist    List that contains words not needed" +
                    "\n\tdocument(s) Documents to be searched through" +
                    "\n\nTo stop the program, type -stop");
        }
    }

    /*
     * purpose: Run the program
     * input: Commands from the user
     * result: Documents are searched and outputted
     */
    public static void main (String [] args) throws IOException
    {
        CLI cli = new CLI();
        cli.process(args);
    }
}
```