```
/**
  * This is my code!  It's goal is to accept command line arguments from the use
r.
  * CS 312 - Assignment 9
  * @author Mitchell Bardsley
  * @version 1.1  12/5/2020
  */

import java.nio.file.Paths;
import java.nio.file.Path;

import java.util.HashSet;
import java.util.HashMap;

import java.util.Scanner;

public class CLI
{

    protected String userQuery;

    protected InvertedIndex ii = new InvertedIndex( );

    /* gives the user a usage message when they try to run the CLI (O(1))
     * @param none
     * @return none
     */
    public void usage( )
    {
        System.out.println( "Usage: java CLI <stoplist> <docs>" );
    }

    /* constructs a CLI object (O(n^2))
     * @param a string array of arguments
     * @return none
     */
    public CLI( String [] args )
    {
        if ( args.length == 0 )
        {

            usage( );
            return;

        }

        if ( args[1].contains( ".txt" ) )
        {

            int i = 1;
            while ( i < args.length )
            {

                Path p = Paths.get( "/home/mabardsley/cs312/r-for-retrieval-mitchb25j/testing/"
+ args[i] );
                String fileName = p.getFileName( ).toString( );

                Document newDoc = new Document( fileName );

                for ( String word : newDoc )
                {
                    ii.addDocument( word, newDoc );
                }
```



```
                i++;

            }

        }

        Scanner scan = new Scanner( System.in );

        while ( userQuery != null )
        {

            userQuery = scan.nextLine( );

            if ( userQuery == "@@debug" )
            {

                System.out.println( "The inverted index contains " + ii.displayIndex( ) )
;

            }

            else
            {

                if ( userQuery.contains( " " ) )
                {

                    String[] queryStrings = userQuery.split( " " );

                    HashSet<String> userQuerySet  = new HashSet<String>( );

                    for ( String word : queryStrings )
                    {
                        userQuerySet.add( word );
                    }

                    HashSet<Document> multiWordQuery = new HashSet<Document>( );

                    multiWordQuery = ii.buildMultiWordQuery( userQuerySet, ii.do
cuments );

                    for ( Document doc : multiWordQuery )
                    {

                        System.out.print( doc.myName( ) + " " );

                        System.out.println( "--- found in " +
                        (multiWordQuery == null ? 0 : multiWordQuery.size( ) ) +
 "documents" );
                    }

                    long startTime = System.currentTimeMillis( );
                    long stopTime = System.currentTimeMillis( );
                    long elapsedTime = stopTime - startTime;

                    System.out.println( "@@ multi-word query took " + elapsedTime + "
ms" );

                }

                else if ( ! userQuery.contains( " " ) )
                {

                    HashSet<Document> singleWordQuery = new HashSet<Document>( )
;
```

```
                    singleWordQuery = ii.buildSingleWordQuery( userQuery, ii.doc
uments );

                    for ( Document doc : singleWordQuery )
                    {

                        System.out.print( doc.myName( ) + " " );

                    }

                    long startTime = System.currentTimeMillis( );
                    long stopTime = System.currentTimeMillis( );
                    long elapsedTime = stopTime - startTime;

                    System.out.println( "@@ single-word query took " + elapsedTime + "
ms" );

                }

            }

        }

    }

    /* runs the program (O(1))
     * @param a string array of arguments
     * @return none
     */
    public static void main( String [] args )
    {

        CLI retrievalCLI = new CLI( args );

    }

}
```



```
/**
  * This is my code!  It's goal is to create a document and an iterator for its
words.
  * CS 312 - Assignment 9
  * @author Mitchell Bardsley
  * @version 1.1  12/5/2020
  */

import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;

import java.util.Iterator;
import java.util.HashSet;

public class Document implements Iterable<String>
{

    protected String name;

    protected HashSet<String> fileWords;

    protected String asRead;

    /* constructs a Document object with a name and a HashSet of the file's orig
inal text (O(n))
     * @param a String name of the document
     * @return none
     */
    public Document( String name )
    {

        this.name = name;

        fileWords = new HashSet<String>( );

        try
        {

            BufferedReader br;
            br = new BufferedReader(new FileReader( name ));

            while ( asRead != null )
            {

                fileWords.add( asRead );

                asRead = new Scanner( br ).useDelimiter( "\\A" ).next( );

            }

            br.close( );

        }

        catch (Exception ex)
        {

            System.err.println( "Error occurred while reading file." );

        }
    }

    /* creates an Iterator for Documents for use in other classes (O(1))
     * @param none
     * @return an Iterator of Strings for Document objects
     */
    public Iterator<String> iterator( )
    {
```

```
        return new Scanner( asRead ).useDelimiter("[^a-zA-Z]+");

    }

    /* gives the name of the Document (O(1))
     * @param the String queryWord and a HashSet of documents to look through
     * @return a HashSet of documents for the user's multi-word query
     */
    public String myName( )
    {

        return name;
    }


}
```



```
/**
 * This is my code!  It's goal is to create an inverted index of documents by w
hat query words they contain.
  * CS 312 - Assignment 9
  * @author Mitchell Bardsley
  * @version 1.3  12/5/2020
  */

import java.nio.file.Paths;
import java.nio.file.Path;

import java.util.HashMap;
import java.util.HashSet;

public class InvertedIndex
{

    protected HashMap<String, HashSet<Document>> indexOfDocs;

    protected HashSet<Document> documents = new HashSet<Document>( );

    protected StopList stopList = new StopList( );

    /* constructs an InvertedIndex object with a HashMap index (O(1))
     * @param none
     * @return none
     */
    public InvertedIndex( )
    {

        indexOfDocs = new HashMap<String, HashSet<Document>>( );

    }

    /* adds a document to the document HashSet, then the HashSet to the index (O
(1))
     * @param a String word and a Document document object
     * @return none
     */
    public void addDocument( String word, Document doc )
    {

        documents.add( doc );

        indexOfDocs.put( word, documents );

    }

    /* builds a HashSet of documents using a single-word query for them (O(n^2))
     * @param the String queryWord and a HashSet of documents to look through
     * @return a HashSet of documents for the user's single-word query
     */
    public HashSet<Document> buildSingleWordQuery( String queryWord, HashSet<Doc
ument> documents )
    {

        HashSet<Document> singleWordQuery = new HashSet<Document>( );

        for ( Document doc : documents )
        {

            for ( String docWord : doc )
            {

                if ( docWord.equals( queryWord ) )
                {

                    if ( ! stopList.isStopWord( queryWord ) )
                        singleWordQuery.add( doc );
```

```
                }

            }

        }

        return singleWordQuery;

    }

    /* builds a HashSet of documents using a multi-word query for them (O(n^3))
     * @param the String queryWord and a HashSet of documents to look through
     * @return a HashSet of documents for the user's multi-word query
     */
    public HashSet<Document> buildMultiWordQuery( HashSet<String> queryWords, Ha
shSet<Document> documents )
    {

        HashSet<Document> multiWordQuery = new HashSet<Document>( );

        for ( Document doc : documents )
        {

            for ( String docWord : doc )
            {

                for ( String queryWord : queryWords )
                {

                    if ( docWord.equals( queryWord ) )
                    {

                        if ( ! stopList.isStopWord( queryWord ) )
                            multiWordQuery.add( doc );

                    }

                }

            }

        }

        return multiWordQuery;

    }

    /* translates the index into a printable String for the CLI (O(n))
     * @param none
     * @return a String of the index
     */
    public String displayIndex( )
    {

        String indexString = "";

        for ( String name : indexOfDocs.keySet( ) )
        {

            String wordKey = name;
            String docsValue = indexOfDocs.get( name ).toString( );

            indexString += ( wordKey + ":" + docsValue ) ;

        }

        return indexString;
```



```
    }

}
```

```
/**
  * This is my code!  It's goal is to create a list of stopwords as a stoplist.
  * CS 312 - Assignment 9
  * @author Mitchell Bardsley
  * @version 1.0  12/5/2020
  */

import java.util.HashSet;

public class StopList
{

    protected HashSet<String> stopWords;

    /* constructs a StopList object with a HashSet of stopwords (O(1))
     * @param none
     * @return none
     */
    public StopList( )
    {

        stopWords = new HashSet<String>( );

    }

    /* determines whether a word is a stopword (O(n))
     * @param a String word from a document
     * @return true or false
     */
    public boolean isStopWord( String docWord )
    {

        for ( String stopWord : stopWords )
        {

            if ( stopWord.equals( docWord ) )
                return true;

        }

        return false;

    }

}
```

use hash table
lookup O(1)
rather than this
O(n) search