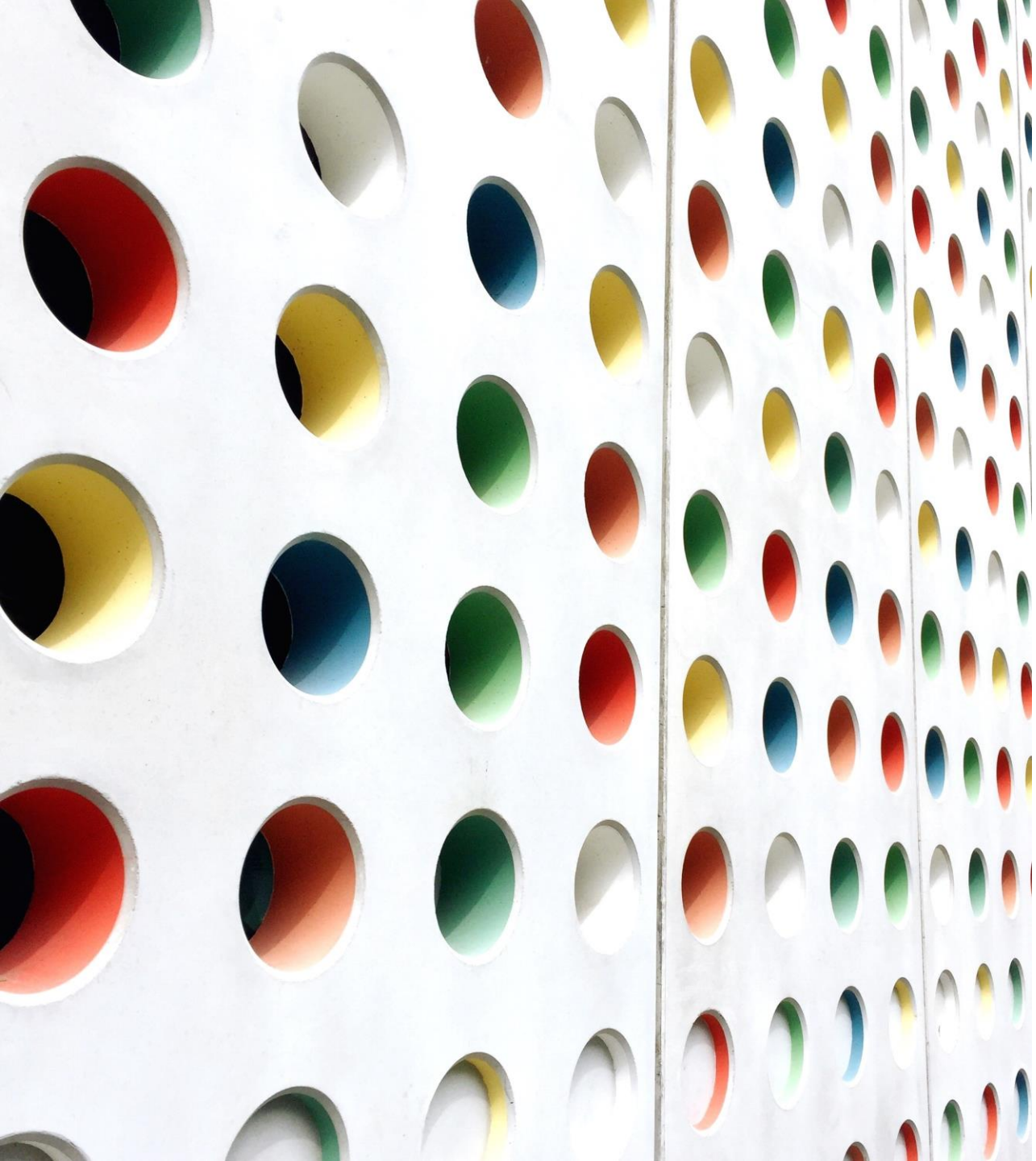


Word Occurances Application

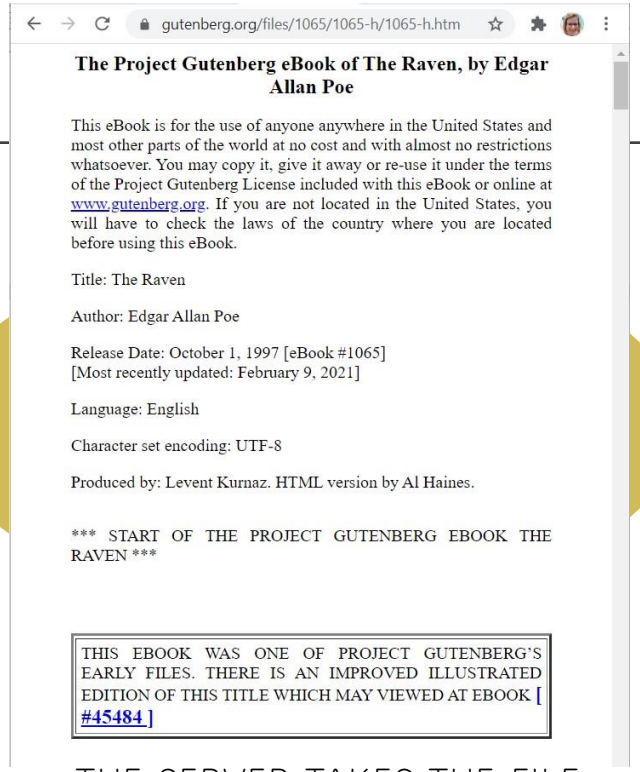
"Parsing a website one word at a time."

Using javaFX and mySQL database

BY: JOANNA SMITH

Client and Server Database Application

CLIENT FEEDS THE SERVER A WEBSITE ADDRESS AFTER IT IS VALIDATED



THE SERVER TAKES THE FILE AND SPLITS IT INTO INDIVIDUAL WORDS

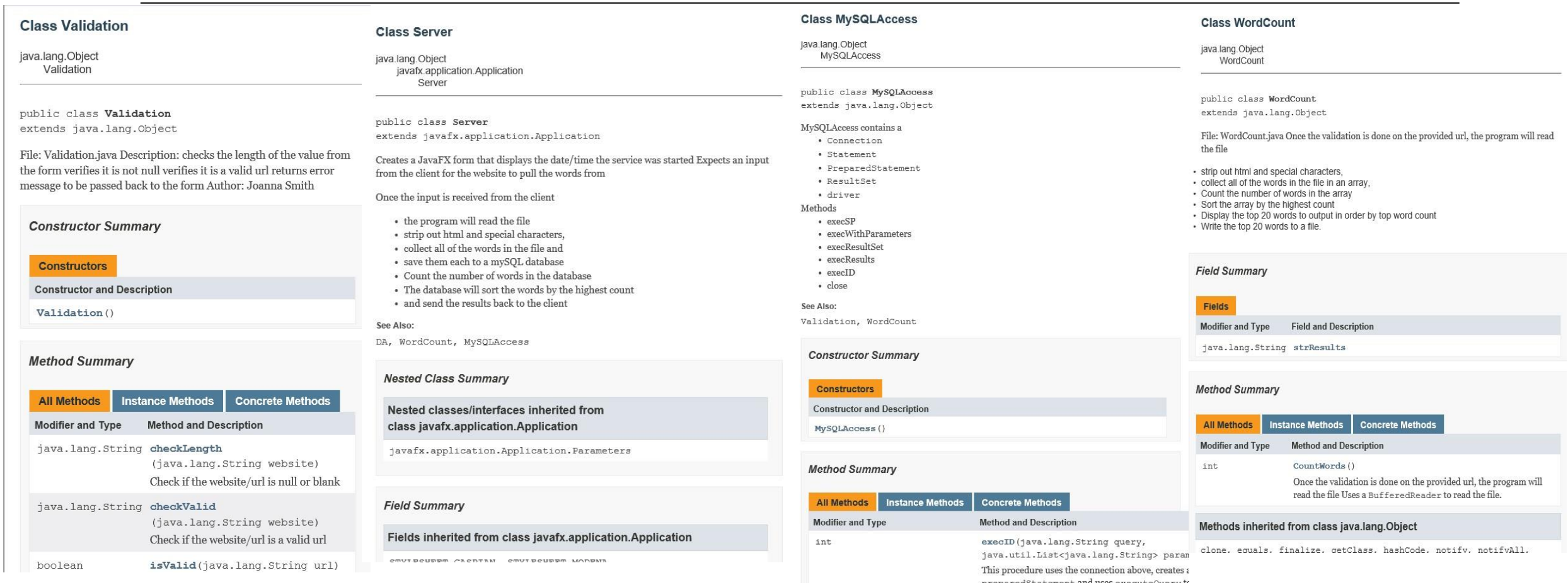
AND STORES EACH WORD IN A MYSQL DATABASE

THE SERVER GRABS A LIST OF THE TOP 20 WORDS SORTED BY WORD COUNT FROM THE DATABASE

THE CLIENT DISPLAYS THOSE RESULTS

JavaDoc

Documentation of the code behind the madness



Class Validation

java.lang.Object
Validation

public class **Validation**
extends java.lang.Object

File: Validation.java Description: checks the length of the value from the form verifies it is not null verifies it is a valid url returns error message to be passed back to the form Author: Joanna Smith

Constructor Summary

Constructors

Constructor and Description
`Validation()`

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
java.lang.String	**checkLength** (java.lang.String website) Check if the website/url is null or blank
java.lang.String	**checkValid** (java.lang.String website) Check if the website/url is a valid url
boolean	**isValid**(java.lang.String url)

Class Server

java.lang.Object
javafx.application.Application
Server

public class **Server**
extends javafx.application.Application

Creates a JavaFX form that displays the date/time the service was started Expects an input from the client for the website to pull the words from

Once the input is received from the client

- the program will read the file
- strip out html and special characters,
- collect all of the words in the file and
- save them each to a mySQL database
- Count the number of words in the database
- The database will sort the words by the highest count
- and send the results back to the client

See Also:
DA, WordCount, MySQLAccess

Nested Class Summary

Nested classes/interfaces inherited from class javafx.application.Application

javafx.application.Application.Parameters

Field Summary

Fields inherited from class javafx.application.Application

Class MySQLAccess

java.lang.Object
MySQLAccess

public class **MySQLAccess**
extends java.lang.Object

MySQLAccess contains a
- Connection
- Statement
- PreparedStatement
- ResultSet
- driver

Methods
- execSP
- execWithParameters
- execResultSet
- execResults
- execID
- close

See Also:
Validation, WordCount

Constructor Summary

Constructors

Constructor and Description
`MySQLAccess()`

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
int	**execID**(java.lang.String query, java.util.List<java.lang.String> param This procedure uses the connection above, creates a

Class WordCount

java.lang.Object
WordCount

public class **WordCount**
extends java.lang.Object

File: WordCount.java Once the validation is done on the provided url, the program will read the file

- strip out html and special characters,
- collect all of the words in the file in an array,
- Count the number of words in the array
- Sort the array by the highest count
- Display the top 20 words to output in order by top word count
- Write the top 20 words to a file.

Field Summary

Fields

Modifier and Type	Field and Description
java.lang.String	strResults

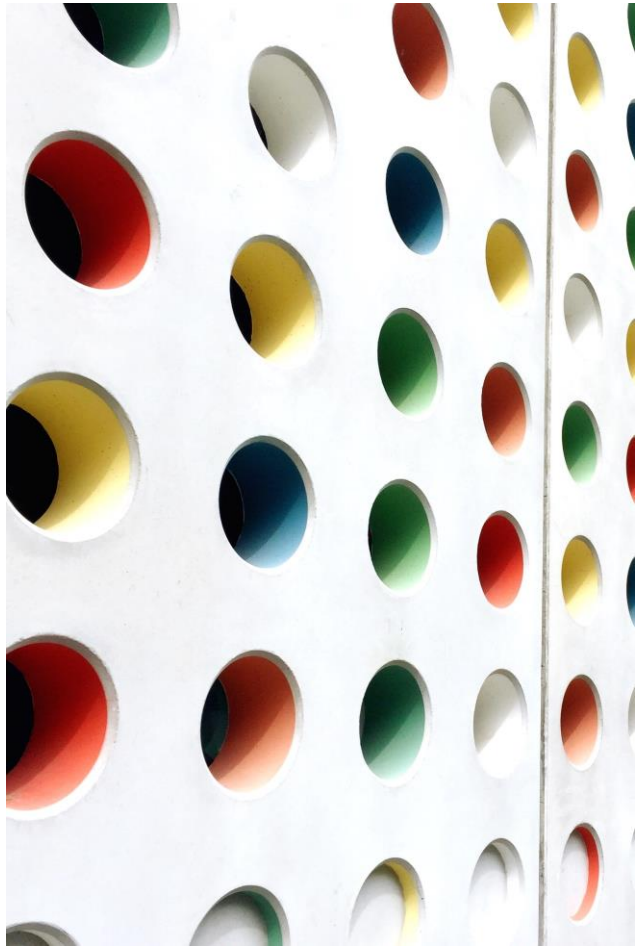
Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
int	**CountWords**() Once the validation is done on the provided url, the program will read the file Uses a `BufferedReader` to read the file.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll,



GitHub

joannashad / CountWords

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki ...

main CountWords / JavaFXModule2 / Go to file Add file ...

joannashad Added mySQL support to application ... 8 days ago History

..		
mySQL	Added mySQL support to application	8 days ago
nbproject	Added mySQL support to application	8 days ago
src	Added mySQL support to application	8 days ago
test	Add javadoc comments	15 days ago
Words.txt	Added mySQL support to application	8 days ago
build.xml	bring in the files for the program	2 months ago
javadoc1.jpg	Add javadoc comments	15 days ago
javadoc2.jpg	Add javadoc comments	15 days ago
javadoc3.jpg	Add javadoc comments	15 days ago
manifest.mf	bring in the files for the program	2 months ago

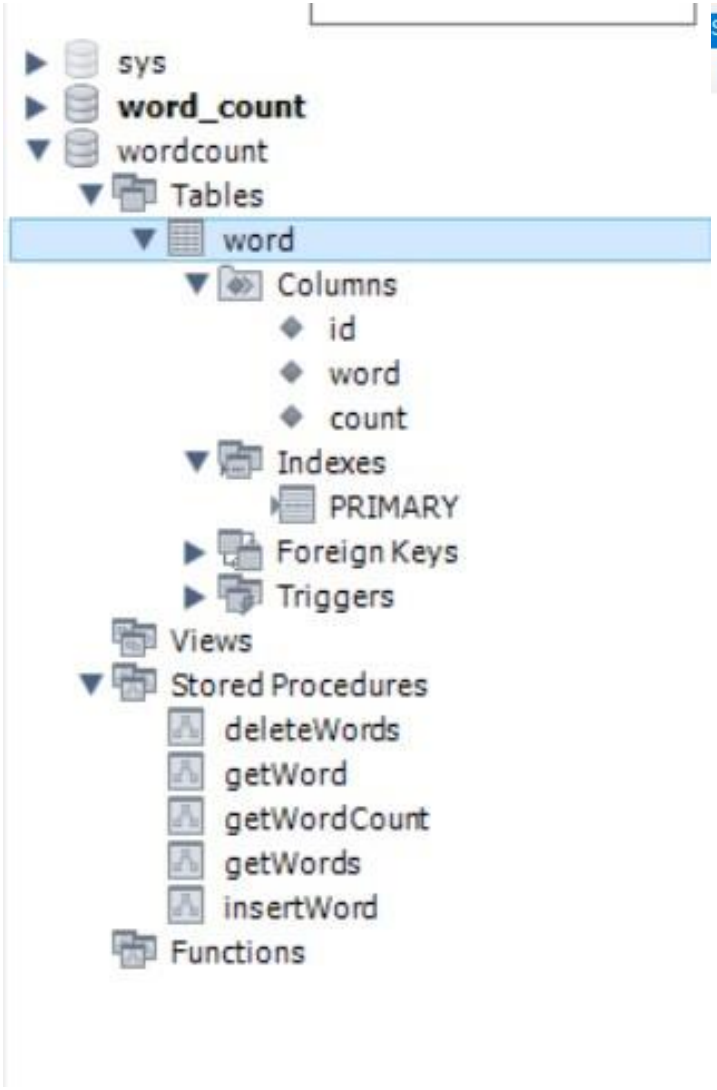
Files are stored, maintained, and organized on the cloud in GitHub.

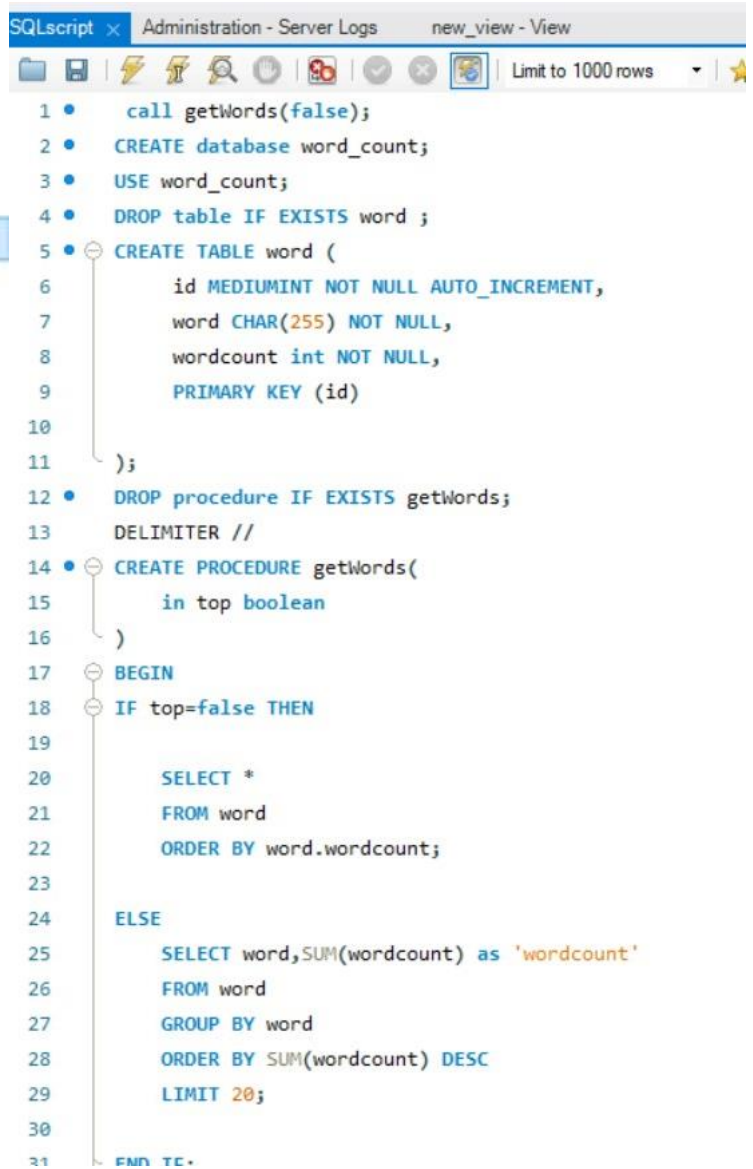
GitHub is a good free resource for source control. The source files from all the projects in this application can be found in GitHub

HTTPS://GITHUB.COM/ JOANNASHAD/COUNTWORDS.GIT

Database Development

- Add database support to your word occurrences application.
- Add a schema called “word occurrences”. Add a table called “word”. Then, as you parse the document, add new words that are not already in the database like this,
  - insert into wordOccurrences.word (word) values ('the’);
  - select * from wordOccurrences..word
- Add JDBC support to your Java project In your word occurrence application, instead of storing the frequencies in the array, store/read them from the database





Unit Testing

Unit testing for Word Count application proved to be very beneficial.

First of all, when creating a testing plan it proved essential that more code should be separated into classes than what was originally written. For example, error checking was built directly into the UI layer of the code. That was separated into it's own class. Doing this made it possible to check that those procedures were performing as expected through automated unit testing.

Second of all, when running the automated unit testing, several errors were caught that were not found in initial manual testing. For example, when testing that the website entered was valid, a null value was not tested and error handling for null exceptions was not built in. Another example was that word case was not considered, so the words were converted to lower case before searching and counting.