Implementation Specification

for

Microminer

Version 1.0 Approved

Prepared by Lynn Barnett and Victoria Potvin

University of Central Oklahoma

February 25, 2015

Table of Contents

[1. Introduction 3](#_Toc412734425)

[1.1 Overview 3](#_Toc412734426)

[1.2 Implementation Specifications 3](#_Toc412734427)

[2. Implementation Code 3](#_Toc412734428)

[2.1 Web Pages 3](#_Toc412734429)

[2.1.1 index.xhtml 3](#_Toc412734430)

[2.2 Package SharedData 4](#_Toc412734431)

[2.2.1 Kwic.java 4](#_Toc412734432)

[2.2.2 CircularShift.java 5](#_Toc412734433)

[2.2.3 Input.java 6](#_Toc412734434)

[2.2.4 Output.java 6](#_Toc412734435)

[2.2.5 LineHolder.java 6](#_Toc412734436)

[2.2.6 LineStorage.java 6](#_Toc412734437)

[2.2.7 Tree.java 7](#_Toc412734438)

[3. Testing Files 9](#_Toc412734439)

[3.1 CircularShiftTest.java 9](#_Toc412734440)

[3.2 InputTest.java 10](#_Toc412734441)

[3.3 OutputTest.java 11](#_Toc412734442)

# Introduction

## Overview

The Key Word In Context (KWIC\*) system takes a set of lines from user input and produces a new set of lines where the input has been circularly shifted and then the new set of lines alphabetically sorted. It is a web-based system in which the user has access to a single page where they can provide input and view output.

## Implementation Specifications

The system has been implemented utilizing a Shared Data and Object Oriented Architecture. This provides maintainability and reusability. The system has been implemented using Java Server Faces and a GlassFish Server. The project was tested using JUnit.

# Implementation Code

## Web Pages

### index.xhtml

<?xml version=**'1.0'** encoding=**'UTF-8'** ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=**"http://www.w3.org/1999/xhtml"**

xmlns:h=**"http://xmlns.jcp.org/jsf/html"**>

<h:head>

<link href=**"//maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css"** rel=**"stylesheet"**/>

<title>**KWIC**</title>

</h:head>

<h:body>

<div class=**"container"**>

<div class=**"row"**>

<div class=**"col-lg-5 col-lg-offset-3"**>

<h1>**Key Word In Context**</h1>

<h:form >

<div class=**"form-group"**>

<label for=**"input"**>**Input**</label>

<h:inputTextarea id="input" class="form-control" rows="10" value="#{kwic.input}"/>

</div>

<div class=**"form-group"**>

<label for=**"output"**>**Output**</label>

<h:inputTextarea id="output" class="form-control" rows="15" value="#{kwic.output}"/>

</div>

<h:commandButton value="Process" class="btn btn-primary" action="#{kwic.run()}"/>

</h:form>

</div>

</div>

</div>

</h:body>

</html>

## Package SharedData

### Kwic.java

package Microminer;

import java.sql.SQLException;

import javax.annotation.PostConstruct;

import javax.enterprise.context.RequestScoped;

import javax.inject.Inject;

import javax.inject.Named;

// Nameed Bean the takes input the input from the form starts connects all the

// Filter to their respective pipes and starts the process.

@Named

@RequestScoped

public class Kwic{

private String input;

private String output;

private LineStorage ls;

private Tree indexHolder;

@Inject

private Input in;

@PostConstruct

private void init(){

ls = new LineStorage();

indexHolder = new Tree(ls);

}

public String getInput() {

return input;

}

public void setInput(String input) {

this.input = input;

}

public String getOutput() {

return output;

}

public void setOutput(String output) {

this.output = output;

}

public void run() throws SQLException{

in.processInput(this.input, ls);

CircularShift.shiftLines(ls, indexHolder);

this.output = Output.CreateOuput(indexHolder);

System.out.println(this.output);

}

}

### CircularShift.java

package SharedData**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Collections**;**

**import** java**.**util**.**LinkedList**;**

public class CircularShift **{**

private static final String noiseWords**[]** **=** **{**"a"**,** "an"**,** "the"**,** "and"**,** "or"**,**

"of"**,** "to"**,** "be"**,** "is"**,** "in"**,** "out"**,** "by"**,** "as"**,** "at"**,** "off"**};**

public static void shiftLines**(**LineStorage ls**,** Tree indexHolder**){**

**for(**int i**=**0**;** i **<** ls**.**size**();**i**++){**

Line l **=** ls**.**getLine**(**i**);**

LinkedList**<**Integer**>** tempList **=** **new** LinkedList**<>();**

**for(**int j **=** 0**;** j **<** l**.**size**();** j**++){**

tempList**.**add**(**j**);**

**}**

**for(**int j**=**0**;** j **<** l**.**size**();** j**++){**

Collections**.**rotate**(**tempList**,** **-**1**);**

**if(!**Arrays**.**asList**(**noiseWords**).**contains**(**l**.**getWord**(**tempList**.**get**(**0**)))){**

Integer**[]** t **=** **new** Integer**[**tempList**.**size**()];**

tempList**.**toArray**(**t**);**

LineHolder tempHolder **=** **new** LineHolder**(**t**,** i**);**

indexHolder**.**insert**(**tempHolder**);**

**}**

**}**

**}**

**}**

**}**

### Input.java

package Microminer;

import java.io.Serializable;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.annotation.Resource;

import javax.enterprise.context.ApplicationScoped;

import javax.inject.Named;

import javax.sql.DataSource;

@Named(value = "input")

@ApplicationScoped

public class Input implements Serializable{

@Resource(name="jdbc/sa")

private DataSource ds;

// Splits input string based on new lines

public void processInput(String input, LineStorage ls) throws SQLException{

String[] lines = input.split("\\r?\\n");

int lineNum = 0;

String URLString = "(https|http)://([a-zA-Z0-9]+.)?([a-zA-Z0-9]+.)(com|edu|org|net)";

Pattern URLPattern = Pattern.compile(URLString);

for(String l: lines){

long key;

Matcher URLMatcher = URLPattern.matcher(l);

if(!URLMatcher.find()){

System.out.println("URL MATCH CONTINUE");

continue;

}

// TEST FOR UPPER AND LOWER CASE STUFF;

String sURL = URLMatcher.group();

String[] desArray = l.split(",");

if(desArray.length > 2){

System.out.println("ARRAY CONTINUE");

continue;

}

String description = desArray[1].trim();

if (ds == null) {

throw new SQLException("CANNOT GET DATASOURCE");

}

Connection conn = ds.getConnection();

if (conn == null) {

throw new SQLException("CANNOT GET DATABASE");

}

try {

boolean committed = false;

conn.setAutoCommit(false);

try{

String query = "INSERT INTO LINE\_STORAGE(URL, DESCRIPTION) "

+ "VALUES(?, ?)";

PreparedStatement stmt = conn.prepareStatement(query,

Statement.RETURN\_GENERATED\_KEYS);

stmt.setString(1, sURL);

stmt.setString(2, description);

stmt.executeUpdate();

ResultSet gk = stmt.getGeneratedKeys();

if (gk.next()) {

key = gk.getLong(1);

}else{

throw new SQLException("CANNOT GET GENERATED KEY");

}

conn.commit();

} finally {

if (!committed) {

conn.rollback();

}

}

}finally{

conn.close();

}

int wordNum = 0;

for (String word : l.split("\\s+")) {

ls.setWord(lineNum, wordNum, word, key);

wordNum++;

}

lineNum++;

}

}

}

### Output.java

package SharedData**;**

public class Output **{**

public static String CreateOuput**(**Tree indexHolder**){**

StringBuilder output **=** indexHolder**.**printTree**();**

**return** output**.**toString**();**

**}**

**}**

### LineHolder.java

package SharedData**;**

public class LineHolder **{**

int lineNumber**;**

Integer**[]** line**;**

public LineHolder**(**Integer**[]** l**,** int lineNumber**){**

**this.**line **=** l**;**

**this.**lineNumber **=** lineNumber**;**

**}**

**}**

### LineStorage.java

package SharedData**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**List**;**

public class LineStorage **{**

private List**<**Line**>** lines**;**

private int linesNum**;**

public LineStorage**(){**

lines **=** **new** ArrayList**<>();**

linesNum **=** 0**;**

**}**

public void setWord**(**int line**,** int wordNum**,** String word**){**

**if(**line **>=** lines**.**size**()){**

Line newLine **=** **new** Line**();**

newLine**.**setWord**(**wordNum**,** word**);**

lines**.**add**(**newLine**);**

linesNum**++;**

**}** **else** **{**

Line l **=** lines**.**get**(**line**);**

l**.**setWord**(**wordNum**,** word**);**

lines**.**set**(**line**,** l**);**

**}**

**}**

public String getWord**(**int line**,** int word**){**

**return** lines**.**get**(**line**).**getWord**(**word**);**

**}**

public int size**(){**

**return** **this.**linesNum**;**

**}**

public Line getLine**(**int i**){**

**return** lines**.**get**(**i**);**

**}**

public void add**(**Line l**)** **{**

lines**.**add**(**l**);**

linesNum**++;**

**}**

public int lineSize**(**int i**){**

**return** lines**.**get**(**i**).**size**();**

**}**

**}**

### Tree.java

package SharedData**;**

public class Tree **{**

private Node root**;**

private ShiftIndex si**;**

private LineStorage ls**;**

public Tree**(**LineStorage ls**)** **{**

root **=** **null;**

**this.**ls **=** ls**;**

**}**

public void insert**(**LineHolder data**)** **{**

root **=** insert**(**root**,** data**);**

**}**

private Node insert**(**Node node**,** LineHolder data**)** **{**

**if** **(**node**==null)** **{**

node **=** **new** Node**(**data**);**

**}** **else** **{**

String dataString **=** ""**;**

**for(**int l **=** 0**;** l **<** data**.**line**.**length**;** l**++){**

dataString **+=** ls**.**getWord**(**data**.**lineNumber**,** data**.**line**[**l**]);**

**}**

String nodeString **=** ""**;**

**for(**int l **=** 0**;** l **<** node**.**line**.**line**.**length**;** l**++){**

nodeString **+=** ls**.**getWord**(**node**.**line**.**lineNumber**,** node**.**line**.**line**[**l**]);**

**}**

**if(**dataString**.**compareToIgnoreCase**(**nodeString**)** **<=** 0**){**

node**.**left **=** insert**(**node**.**left**,** data**);**

**}** **else** **{**

node**.**right **=** insert**(**node**.**right**,** data**);**

**}**

**}**

**return(**node**);**

**}**

public StringBuilder printTree**()** **{**

StringBuilder output **=** **new** StringBuilder**();**

printTree**(**root**,** output**);**

**return** output**;**

**}**

private void printTree**(**Node node**,** StringBuilder output**)** **{**

**if** **(**node **==** **null)** **return;**

printTree**(**node**.**left**,** output**);**

**for(**int i **=** 0**;** i **<** node**.**line**.**line**.**length**;** i**++){**

output**.**append**(**ls**.**getWord**(**node**.**line**.**lineNumber**,** node**.**line**.**line**[**i**])** **+** " "**);**

**}**

output**.**append**(**"\n"**);**

printTree**(**node**.**right**,** output**);**

**}**

public static class Node **{**

private LineHolder line**;**

private Node left**;**

private Node right**;**

Node**(**LineHolder newLine**)** **{**

left **=** **null;**

right **=** **null;**

line **=** newLine**;**

**}**

**}**

**}**

# Testing Files

## CircularShiftTest.java

package SharedData**;**

**import** org**.**junit**.**After**;**

**import** org**.**junit**.**AfterClass**;**

**import** org**.**junit**.**Before**;**

**import** org**.**junit**.**BeforeClass**;**

**import** org**.**junit**.**Test**;**

**import** static org**.**junit**.**Assert**.\*;**

/\*\*

\*

\* @author Victoria

\*/

public class CircularShiftTest **{**

public CircularShiftTest**()** **{**

**}**

@BeforeClass

public static void setUpClass**()** **{**

**}**

@AfterClass

public static void tearDownClass**()** **{**

**}**

@Before

public void setUp**()** **{**

**}**

@After

public void tearDown**()** **{**

**}**

/\*\*

\* Test of shiftLines method, of class CircularShift.

\*/

@Test

public void testShiftLines**()** **{**

System**.**out**.**println**(**"shiftLines"**);**

LineStorage ls **=** **new** LineStorage**();**

Tree indexHolder **=** **new** Tree**(**ls**);**

Input**.**processInput**(**"I am a line."**,** ls**);**

CircularShift**.**shiftLines**(**ls**,** indexHolder**);**

assertEquals**(**3**,** ls**.**size**());**

**}**

**}**

## InputTest.java

package SharedData**;**

**import** org**.**junit**.**After**;**

**import** org**.**junit**.**AfterClass**;**

**import** org**.**junit**.**Before**;**

**import** org**.**junit**.**BeforeClass**;**

**import** org**.**junit**.**Test**;**

**import** static org**.**junit**.**Assert**.\*;**

/\*\*

\*

\* @author Victoria

\*/

public class InputTest **{**

public InputTest**()** **{**

**}**

@BeforeClass

public static void setUpClass**()** **{**

**}**

@AfterClass

public static void tearDownClass**()** **{**

**}**

@Before

public void setUp**()** **{**

**}**

@After

public void tearDown**()** **{**

**}**

@Test

public void testProcessInput**()** **{**

LineStorage ls **=** **new** LineStorage**();**

Input**.**processInput**(**"I am input"**,** ls**);**

System**.**out**.**println**(**"The value is: " **+** ls**.**getLine**(**0**).**toString**());**

assertTrue**(**ls**.**getLine**(**0**).**toString**().**equals**(**"I am input "**));**

assertEquals**(**1**,** ls**.**size**());**

assertEquals**(**3**,** ls**.**getLine**(**0**).**size**());**

**}**

**}**

## OutputTest.java

package SharedData**;**

**import** org**.**junit**.**After**;**

**import** org**.**junit**.**AfterClass**;**

**import** org**.**junit**.**Before**;**

**import** org**.**junit**.**BeforeClass**;**

**import** org**.**junit**.**Test**;**

**import** static org**.**junit**.**Assert**.\*;**

/\*\*

\*

\* @author Victoria

\*/

public class OutputTest **{**

public OutputTest**()** **{**

**}**

@BeforeClass

public static void setUpClass**()** **{**

**}**

@AfterClass

public static void tearDownClass**()** **{**

**}**

@Before

public void setUp**()** **{**

**}**

@After

public void tearDown**()** **{**

**}**

/\*\*

\* Test of CreateOuput method, of class Output.

\*/

@Test

public void testCreateOuput**()** **{**

System**.**out**.**println**(**"CreateOuput"**);**

LineStorage ls **=** **new** LineStorage**();**

Input**.**processInput**(**"aa Aa AA aA"**,** ls**);**

Tree indexHolder **=** **new** Tree**(**ls**);**

String expResult **=** "aa Aa AA aA"**;**

String result **=** Output**.**CreateOuput**(**indexHolder**);**

assertEquals**(**expResult**,** result**);**

**}**

**}**