Implementation Specification

for

Key Word In Context

Version 1.0 Approved

Prepared by Lynn Barnett and Victoria Potvin

University of Central Oklahoma

February 4, 2015

Table of Contents

[1. Introduction 3](#_Toc410858680)

[1.1 Overview 3](#_Toc410858681)

[1.2 Implementation Specifications 3](#_Toc410858682)

[2. Implementation Code 3](#_Toc410858683)

[2.1 Web Pages 3](#_Toc410858684)

[2.1.1 index.xhtml 3](#_Toc410858685)

[2.2 Package com.barnett.kwic 4](#_Toc410858686)

[2.2.1 Kwic.java 4](#_Toc410858687)

[2.3 Package pipeandfilter 5](#_Toc410858688)

[2.3.1 CircularShiftFilter.java 5](#_Toc410858689)

[2.3.2 Filter.java 6](#_Toc410858690)

[2.3.3 InputFilter.java 7](#_Toc410858691)

[2.3.4 OutputFilter.java 7](#_Toc410858692)

[2.3.5 Pipe.java 8](#_Toc410858693)

[2.3.6 SortFilter.java 9](#_Toc410858694)

[3. Testing Files 10](#_Toc410858695)

[3.1 KwicTest.java 10](#_Toc410858696)

[3.2 CircularShiftFilterTest.java 12](#_Toc410858697)

[3.3 InputFilterTest.java 13](#_Toc410858698)

[3.4 OutputFitlerTest.java 14](#_Toc410858699)

[3.5 SortFitlerTest.java 15](#_Toc410858700)

# 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 Pipe and Filter 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.startFilter}"/>

</h:form>

</div>

</div>

</div>

</h:body>

</html>

## Package com.barnett.kwic

### Kwic.java

package com**.**barnett**.**kwic**;**

**import** javax**.**enterprise**.**context**.**RequestScoped**;**

**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**{**

// Attributes that hold the input and out, used by the JSF framework

private String input**;**

private String output**;**

public String startFilter**()** **throws** InterruptedException**{**

// Pipes the connect the filters

pipeandfilter**.**Pipe p1 **=** **new** pipeandfilter**.**Pipe**();**

pipeandfilter**.**Pipe p2 **=** **new** pipeandfilter**.**Pipe**();**

pipeandfilter**.**Pipe p3 **=** **new** pipeandfilter**.**Pipe**();**

// Takes input as string, parses out individual lines and sends the

// lines into pipe p1

pipeandfilter**.**Filter f1 **=** **new** pipeandfilter**.**InputFilter**(**p1**,** **this.**input**);**

// Takes lines from pipe p1 and cicularly shifts them to take create new

//lines that are sent into pipe p2

pipeandfilter**.**Filter f2 **=** **new** pipeandfilter**.**CircularShiftFilter**(**p1**,** p2**);**

// Takes strings from pipe p2 and stores them until is recieves the

// teminating string, then it sorts the lines and send them to the

// pipe p3

pipeandfilter**.**Filter f3 **=** **new** pipeandfilter**.**SortFilter**(**p2**,** p3**);**

// take the output from pipe p3 and build an output strine while it

// has not recieved the terminating string.

pipeandfilter**.**Filter f4 **=** **new** pipeandfilter**.**OutputFilter**(**p3**);**

//Thread t1 = new Thread(f1); // Start

Thread t2 **=** **new** Thread**(**f2**);**

Thread t3 **=** **new** Thread**(**f3**);**

Thread t4 **=** **new** Thread**(**f4**);**

//t1.start();

t2**.**start**();**// Start Cicular Shift Filter

t3**.**start**();**// Start Sort Filter

t4**.**start**();** // Start output Filter

// waits while output is still builing up the string, once the

// terminating string is found t4 while no long be alive, and

// the program will move on.

t4**.**join**();**

pipeandfilter**.**OutputFilter f5**;**

**if** **(**f4 **instanceof** pipeandfilter**.**OutputFilter**)** **{**

f5 **=** **(**pipeandfilter**.**OutputFilter**)** f4**;**

**}else{**

f5 **=** **null;**

**}**

String test **=** f5**.**getOutput**();**

**this.**output **=** test**;**

**return** "index"**;**

**}**

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**;**

**}**

**}**

## Package pipeandfilter

### CircularShiftFilter.java

package pipeandfilter**;**

public class CircularShiftFilter **extends** Filter**{**

public CircularShiftFilter**(**Pipe inPipe**,** Pipe outPipe**){**

**super(**inPipe**,** outPipe**);**

**}**

// Meant to run in a seperate thread

@Override

public void run**(){**

**while(true){**

String input **=** inPipe**.**PullData**();**

**if(**input**.**equals**(**"#DONE"**)){**

**this.**outPipe**.**PushData**(**input**);**

**break;**

**}**

String output **=** ""**;**

String lastWord **=** ""**;**

String firstWords**;**

// Test if line only has one word;

**if(!(**input**.**indexOf**(**" "**)** **==** **-**1**)){**

// remove white space so circular shift doen't match an empty

// first word to last word

input **=** input**.**trim**();**

// Get the first word of the line so you can test with the

// shift has made all posible line

String begin **=** input**.**substring**(**0**,**input**.**indexOf**(**" "**));**

// Shift and create new string until all strings have been

// created

**while(!**lastWord**.**equals**(**begin**)){**

firstWords **=** input**.**substring**(**0**,** input**.**lastIndexOf**(**" "**));**

lastWord **=** input**.**substring**(**input**.**lastIndexOf**(**" "**)** **+** 1**);**

output **=** lastWord **+** " " **+** firstWords**;**

input **=** output**;**

// Push new string into output pipe

**this.**outPipe**.**PushData**(**input**);**

**}**

**}** **else{**

// push single word string into pipe

**this.**outPipe**.**PushData**(**input**);**

**}**

**}**

**}**

**}**

### Filter.java

package pipeandfilter**;**

public abstract class Filter **implements** Runnable **{**

Pipe inPipe**;**

Pipe outPipe**;**

public Filter**(**Pipe inPipe**,** Pipe outPipe**){**

**this.**inPipe **=** inPipe**;**

**this.**outPipe **=** outPipe**;**

**}**

public abstract void run**();**

**}**

### InputFilter.java

package pipeandfilter**;**

public class InputFilter **extends** Filter **{**

private String input**;**

// calls super class constructor with null input pipe, becuase input will

// come from a string.

public InputFilter**(**Pipe outPipe**,** String input**){**

**super(null,** outPipe**);**

**this.**input **=** input**;**

processInput**();**

**}**

// Splits input string based on new lines

private void processInput**(){**

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

**for(**String l**:** lines**){**

**this.**outPipe**.**PushData**(**l**);**

**}**

**this.**outPipe**.**PushData**(**"#DONE"**);**

**}**

@Override

public void run**(){}**

**}**

### OutputFilter.java

package pipeandfilter**;**

public class OutputFilter **extends** Filter**{**

private String output**;**

public OutputFilter**(**Pipe inPipe**){**

**super(**inPipe**,** **null);**

**this.**output **=** ""**;**

**}**

@Override

public void run**(){**

**while(true){**

String value **=** **this.**inPipe**.**PullData**();**

**if(**value**.**equals**(**"#DONE"**)){**

**break;**

**}** **else{**

output **+=** value **+** "\n"**;**

**}**

**}**

**try** **{**

Thread**.**sleep**(**500**);**

**}** **catch** **(**InterruptedException ex**)** **{**

ex**.**printStackTrace**();**

**}**

**}**

public String getOutput**(){**

**return** **this.**output**;**

**}**

**}**

### Pipe.java

package pipeandfilter**;**

**import** java**.**util**.**concurrent**.**BlockingQueue**;**

**import** java**.**util**.**concurrent**.**LinkedBlockingQueue**;**

public class Pipe **{**

private BlockingQueue**<**String**>** dataBuffer**;**

public Pipe**(){**

dataBuffer **=** **new** LinkedBlockingQueue**<>();**

**}**

public void PushData**(**String input**){**

**try{**

**this.**dataBuffer**.**put**(**input**);**

**}** **catch(**Exception e**){**

System**.**out**.**println**(**"Push Data Exception"**);**

e**.**printStackTrace**();**

**}**

**}**

public String PullData**(){**

**try{**

**return** dataBuffer**.**take**();**

**}catch(**Exception e**){**

System**.**out**.**println**(**"Pull Data Exception"**);**

e**.**printStackTrace**();**

**}**

**return** **null;**

**}**

**}**

### SortFilter.java

package pipeandfilter**;**

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

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

**import** java**.**util**.**Iterator**;**

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

**import** java**.**util**.**logging**.**Level**;**

**import** java**.**util**.**logging**.**Logger**;**

public class SortFilter **extends** Filter **{**

public SortFilter**(**Pipe inPipe**,** Pipe outPipe**){**

**super(**inPipe**,** outPipe**);**

**}**

@Override

public void run**(){**

String data**;**

// container for recieved strings

List values **=** **new** ArrayList**<**String**>();**

**while(true){**

data **=** **this.**inPipe**.**PullData**();**

// when terminating string is found sort the array and

// push the values to the output pipe

**if(**data**.**equals**(**"#DONE"**)){**

Collections**.**sort**(**values**);**

// add the terminating string to the end of the array

// so that the next filter will know when to end it's

// execution.

values**.**add**(**"#DONE"**);**

Iterator iterator **=** values**.**iterator**();**

**while(**iterator**.**hasNext**()){**

String element **=** **(**String**)** iterator**.**next**();**

**this.**outPipe**.**PushData**(**element**);**

**}**

**break;**

**}**

values**.**add**(**data**);**

**}**

**}**

**}**

# Testing Files

## KwicTest.java

package com**.**barnett**.**kwic**;**

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

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

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

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

/\*\*

\*

@author lynnbarnett

\*/

public class KwicTest **{**

public KwicTest**()** **{**

**}**

@BeforeClass

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

**}**

@AfterClass

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

**}**

/\*\*

Test of startFilter method, of class Kwic.

\*/

@Test

public void testStartFilter**()** **throws** InterruptedException **{**

System**.**out**.**println**(**"MAIN TEST"**);**

pipeandfilter**.**Pipe p1 **=** **new** pipeandfilter**.**Pipe**();**

pipeandfilter**.**Pipe p2 **=** **new** pipeandfilter**.**Pipe**();**

pipeandfilter**.**Pipe p3 **=** **new** pipeandfilter**.**Pipe**();**

String testInput **=** "A test\nB test"**;**

pipeandfilter**.**Filter f1 **=** **new** pipeandfilter**.**InputFilter**(**p1**,** testInput**);**

pipeandfilter**.**Filter f2 **=** **new** pipeandfilter**.**CircularShiftFilter**(**p1**,** p2**);**

pipeandfilter**.**Filter f3 **=** **new** pipeandfilter**.**SortFilter**(**p2**,** p3**);**

pipeandfilter**.**Filter f4 **=** **new** pipeandfilter**.**OutputFilter**(**p3**);**

Thread t2 **=** **new** Thread**(**f2**);**

Thread t3 **=** **new** Thread**(**f3**);**

Thread t4 **=** **new** Thread**(**f4**);**

t2**.**start**();**// Start Cicular Shift Filter

t3**.**start**();**// Start Sort Filter

t4**.**start**();** // Start output Filter

t4**.**join**();**

pipeandfilter**.**OutputFilter f5**;**

**if** **(**f4 **instanceof** pipeandfilter**.**OutputFilter**)** **{**

f5 **=** **(**pipeandfilter**.**OutputFilter**)** f4**;**

**}else{**

f5 **=** **null;**

**}**

assertTrue**(**f5**.**getOutput**().**equals**(**"A test\nB test\ntest A\ntest B\n"**));**

**}**

/\*\*

Test of getInput method, of class Kwic.

\*/

@Test

public void testGetInput**()** **{**

// System.out.println("getInput");

// Kwic instance = new Kwic();

// String expResult = "";

// String result = instance.getInput();

// assertEquals(expResult, result);

// // TODO review the generated test code and remove the default call to fail.

// fail("The test case is a prototype.");

**}**

/\*\*

Test of setInput method, of class Kwic.

\*/

@Test

public void testSetInput**()** **{**

// System.out.println("setInput");

// String input = "";

// Kwic instance = new Kwic();

// instance.setInput(input);

// // TODO review the generated test code and remove the default call to fail.

// fail("The test case is a prototype.");

**}**

/\*\*

Test of getOutput method, of class Kwic.

\*/

@Test

public void testGetOutput**()** **{**

// System.out.println("getOutput");

// Kwic instance = new Kwic();

// String expResult = "";

// String result = instance.getOutput();

// assertEquals(expResult, result);

// // TODO review the generated test code and remove the default call to fail.

// fail("The test case is a prototype.");

**}**

/\*\*

Test of setOutput method, of class Kwic.

\*/

@Test

public void testSetOutput**()** **{**

// System.out.println("setOutput");

// String output = "";

// Kwic instance = new Kwic();

// instance.setOutput(output);

// // TODO review the generated test code and remove the default call to fail.

// fail("The test case is a prototype.");

**}**

**}**

## CircularShiftFilterTest.java

package pipeandfilter**;**

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

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

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

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

/\*\*

\*

\* @author lynnbarnett

\*/

public class CircularShiftFilterTest **{**

public CircularShiftFilterTest**()** **{**

**}**

@BeforeClass

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

**}**

@AfterClass

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

**}**

/\*\*

\* Test of run method, of class CircularShiftFilter.

\*/

@Test

public void testRun**()** **{**

System**.**out**.**println**(**"Circular Shfit Run"**);**

Pipe inPipe **=** **new** Pipe**();**

Pipe outPipe **=** **new** Pipe**();**

CircularShiftFilter instance **=** **new** CircularShiftFilter**(**inPipe**,** outPipe**);**

inPipe**.**PushData**(**"Test One"**);**

inPipe**.**PushData**(**"Test Two"**);**

inPipe**.**PushData**(**"test three"**);**

inPipe**.**PushData**(**"#DONE"**);**

instance**.**run**();**

assertTrue**(**outPipe**.**PullData**().**equals**(**"One Test"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"Test One"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"Two Test"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"Test Two"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"three test"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"test three"**));**

// TODO review the generated test code and remove the default call to fail.

**}**

**}**

## InputFilterTest.java

package pipeandfilter**;**

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

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

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

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

/\*\*

\*

\* @author lynnbarnett

\*/

public class InputFilterTest **{**

public InputFilterTest**()** **{**

**}**

@BeforeClass

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

**}**

@AfterClass

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

**}**

/\*\*

\* Test of run method, of class InputFilter.

\*/

@Test

public void testRun**()** **{**

System**.**out**.**println**(**"run InputFilterTest"**);**

Pipe outPipe **=** **new** Pipe**();**

String testString **=** "This is Test One\nAnd Another\nThird Try\nForth"**;**

InputFilter instance **=** **new** InputFilter**(**outPipe**,**testString**);**

instance**.**run**();**

assertTrue**(**outPipe**.**PullData**().**equals**(**"This is Test One"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"And Another"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"Third Try"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"Forth"**));**

// TODO review the generated test code and remove the default call to fail.

**}**

**}**

## OutputFitlerTest.java

package pipeandfilter**;**

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

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

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

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

/\*\*

\*

\* @author lynnbarnett

\*/

public class OutputFilterTest **{**

public OutputFilterTest**()** **{**

**}**

@BeforeClass

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

**}**

@AfterClass

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

**}**

/\*\*

\* Test of run method, of class OutputFilter.

\*/

@Test

public void testRun**()** **{**

System**.**out**.**println**(**"run Output Filter Test"**);**

Pipe inPipe **=** **new** Pipe**();**

inPipe**.**PushData**(**"TEST ONE"**);**

inPipe**.**PushData**(**"TEST TWO"**);**

inPipe**.**PushData**(**"#DONE"**);**

OutputFilter instance **=** **new** OutputFilter**(**inPipe**);**

instance**.**run**();**

assertTrue**(**instance**.**getOutput**().**equals**(**"TEST ONE\nTEST TWO\n"**)** **);**

**}**

/\*\*

\* Test of getOutput method, of class OutputFilter.

\*/

@Test

public void testGetOutput**()** **{**

// System.out.println("getOutput");

// OutputFilter instance = null;

// String expResult = "";

// String result = instance.getOutput();

// assertEquals(expResult, result);

// // TODO review the generated test code and remove the default call to fail.

**}**

**}**

## SortFitlerTest.java

package pipeandfilter**;**

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

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

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

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

/\*\*

\*

\* @author lynnbarnett

\*/

public class SortFilterTest **{**

public SortFilterTest**()** **{**

**}**

@BeforeClass

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

**}**

@AfterClass

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

**}**

/\*\*

\* Test of run method, of class SortFilter.

\*/

@Test

public void testRun**()** **{**

System**.**out**.**println**(**"run Sort Filter"**);**

Pipe inPipe **=** **new** Pipe**();**

Pipe outPipe **=** **new** Pipe**();**

SortFilter instance **=** **new** SortFilter**(**inPipe**,** outPipe**);**

inPipe**.**PushData**(**"Z"**);**

inPipe**.**PushData**(**"Y"**);**

inPipe**.**PushData**(**"C"**);**

inPipe**.**PushData**(**"B"**);**

inPipe**.**PushData**(**"A"**);**

inPipe**.**PushData**(**"#DONE"**);**

instance**.**run**();**

assertTrue**(**outPipe**.**PullData**().**equals**(**"A"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"B"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"C"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"Y"**));**

assertTrue**(**outPipe**.**PullData**().**equals**(**"Z"**));**

**}**

**}**