```java
//kwicParms.java
package playingWithJava;

public class kwicParms {

	public static boolean excludeArticles = false;
	public static boolean showFrequency = false;
}
```

```java
package playingWithJava;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import com.google.common.collect.Multimap;
import com.google.common.collect.TreeMultimap;

public class KWIC extends kwicParms {

	private List<String> wordList;
	private Multimap<String, String> dictionary;
	private final String articles = "a an A An AN the The THE";

	public KWIC() {
		wordList = new ArrayList<String>();
		dictionary = TreeMultimap.create();
	}

	public void applyParameters(String[] args) throws Exception {
		for (String arg : args) {
			if (arg.contains("-na")) {
				kwicParms.excludeArticles = true;
			} else if (arg.contains("-f")) {
				kwicParms.showFrequency = true;
			} else if (arg.contains("-h")) {
				System.out.println("Read from standard input and print either a key word in context table");
				System.out.println("or frequency table to standard output.\n");
				System.out.println("-na:\tRemove articles\n");
				System.out.println("-f :\tDisplay frequency table\n");
				System.out.println("-h :\tHelp\n");
			}
		}
	}

	public void readInput(Scanner scan) throws Exception {
		wordList.clear(); //clear the list in the event that buildKWIC branched here (lines 93-95)
		System.out.print(":");
		String input = scan.nextLine(); //read in text from keyboard
		String words[] = input.split(" "); //parse the input to create an array of strings

		for (String word : words) {

			// use Regex to remove the punctuation and then convert to lower-case
			word = word.replaceAll("[^a-zA-Z ]", "").toLowerCase();

			if (kwicParms.excludeArticles && articles.contains(word)) {
				continue;
			} else {
				wordList.add(word);
			}
		}

		buildKWIC(scan);

	}

	public String getContext(int startIndex) throws Exception {
		String first, second, third, fourth, fifth, context;

		first = wordList.get(startIndex);
		second = wordList.get(startIndex + 1);
		third = wordList.get(startIndex + 2);
		fourth = wordList.get(startIndex + 3);
		fifth = wordList.get(startIndex + 4);

		context = first + " " + second + " " + third + " " + fourth + " " + fifth;
		return context;
	}





	public void buildKWIC(Scanner scan) throws Exception {
```

```java
		String context = "";
		Integer i = 0; //used to signal the front and back of input

		for (String key : wordList) {

			if (wordList.size() >= 5 && i <= 3) {
				context = getContext(0);
				dictionary.put(key, context);
			} else if (wordList.size() >= 5 && i >= wordList.size() - 3) {
				context = getContext(wordList.size() - 5);
				dictionary.put(key, context);
			} else if (wordList.size() >= 5) {
				context = getContext(i - 2);
				dictionary.put(key, context);
			} else {
				System.out.println("There are not enough words to create a KWIC. Use 5 or more words.");
				readInput(scan);
			}

			i++;
		}
	}

	public void printKWIC() throws Exception {
		for (String key : dictionary.keySet()) {
			for (String context : dictionary.get(key)) {
				System.out.println(key + "\t" + context);
			}
		}
	}

	public void printFrequency() throws Exception {
		for (String key : dictionary.keySet()) {
			System.out.println(key + "\t" + dictionary.get(key).size());
		}
	}
}
```

```java
//PlayingWithJava.java
package playingWithJava;

import java.util.Scanner;

public class PlayingWithJava extends KWIC {

	public static void main(String[] args) throws Exception {
		Scanner scan = new Scanner(System.in);

		KWIC object = new KWIC();
		object.applyParameters(args);
		object.readInput(scan);

		if (kwicParms.showFrequency) {
			object.printFrequency();
		} else {
			object.printKWIC();
		}

		scan.close();
	}
}
```

For this assignment, also include a short summary of what changed in your design in this new version of the program.

Based off the feedback from the C++ implementation of the KWIC, I knew I needed to make it command line driven. So I implemented that and so when I run my program in cmd, it executes regardless of any parameters; the parameters just give it more direction on how to behave. Java also offers some really cool methods based off the data structure you choose. In C++ I had to create my own split method, but Java has that built into the language. Also, the foreach loops are more intuitive and don't require as much explicit syntax as in C++. In Java, there are different types of Maps you can implement. I started out with my original Map<String, List<String>> implementation but switched to TreeMap<String, List<String>> so that it would order off the key which is what C++ did automatically. However, I found myself switching to a MultiMap<String, String> because it supports duplicates. I also was able to take advantage of Regex to strip away punctuation versus all the checking I had to do in C++. Overall, I was able to simplify and cut my code in half using much of Java's methods. I'm much happier with my implementation in Java than in C++.

Test Cases

1) Basic command no other parameters

```
Command Prompt

C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava
:This is a very basic example to show how my KWIC program works. I'm running my
program in command line which was a pain to get to work.
a       this is a very basic
a       which was a pain to
basic   a very basic example to
command program in command line which
example very basic example to show
get     pain to get to work
how     to show how my kwic
im      program works im running my
in      my program in command line
is      this is a very basic
kwic    how my kwic program works
line    in command line which was
my      im running my program in
my      show how my kwic program
pain    was a pain to get
program my kwic program works im
program running my program in command
running works im running my program
show    example to show how my
this    this is a very basic
to      a pain to get to
to      basic example to show how
to      pain to get to work
very    this is a very basic
was     line which was a pain
which   command line which was a
work    pain to get to work
works   kwic program works im running

C:\Users\melan_000\workspace\PlayingWithJava>
```

2) Command with –na

```
C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava -na
:This is another example where I'm going to put articles like a, AN, and The to
show that it removes them.
and     articles like and to show
another this is another example where
articles        to put articles like and
example this is another example where
going   where im going to put
im      example where im going to
is      this is another example where
it      show that it removes them
like    put articles like and to
put     going to put articles like
removes show that it removes them
show    and to show that it
that    to show that it removes
them    show that it removes them
this    this is another example where
to      im going to put articles
to      like and to show that
where   another example where im going

C:\Users\melan_000\workspace\PlayingWithJava>
```

3) Command with –f

```
C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava -f
:The snow storm had a lot of snow. I made a snowman out of snow. I would like an
 apple. No, I want an apple pie. I'm going to eat an apple pie in the snow.
a       2
an      3
apple   3
eat     1
going   1
had     1
i       3
im      1
in      1
like    1
lot     1
made    1
no      1
of      2
out     1
pie     2
snow    4
snowman 1
storm   1
the     2
to      1
want    1
would   1

C:\Users\melan_000\workspace\PlayingWithJava>
```

4) Command with –na and –f

```
C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava -f -na
:The snow storm had a lot of snow. I made a snowman out of snow. I would like an
 apple. No, I want an apple pie. I'm going to eat an apple pie in the snow.
apple   3
eat     1
going   1
had     1
i       3
im      1
in      1
like    1
lot     1
made    1
no      1
of      2
out     1
pie     2
snow    4
snowman 1
storm   1
to      1
want    1
would   1

C:\Users\melan_000\workspace\PlayingWithJava>
```

5) Command with –f and –na

```
C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava -na -f
:The snow storm had a lot of snow. I made a snowman out of snow. I would like an
 apple. No, I want an apple pie. I'm going to eat an apple pie in the snow.
apple   3
eat     1
going   1
had     1
i       3
im      1
in      1
like    1
lot     1
made    1
no      1
of      2
out     1
pie     2
snow    4
snowman 1
storm   1
to      1
want    1
would   1

C:\Users\melan_000\workspace\PlayingWithJava>
```

6) Command with –h

```
C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava -h
Read from standard input and print either a key word in context table
or frequency table to standard output.

-na:    Remove articles

-f :    Display frequency table

-h :    Help
```

7) Command with misspelling a parameter (i.e. –n) Expect: perform regular quick

```
C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava -n
:The snow storm had a lot of snow. I made a snowman out of snow. I would like an
 apple. No, I want an apple pie. I'm going to eat an apple pie in the snow.
a       i made a snowman out
a       storm had a lot of
an      i want an apple pie
an      to eat an apple pie
an      would like an apple no
apple   eat an apple pie in
apple   like an apple no i
apple   want an apple pie im
eat     going to eat an apple
going   pie im going to eat
had     the snow storm had a
i       apple no i want an
i       of snow i made a
i       of snow i would like
im      apple pie im going to
in      apple pie in the snow
like    i would like an apple
lot     had a lot of snow
made    snow i made a snowman
no      an apple no i want
of      a lot of snow i
of      snowman out of snow i
out     a snowman out of snow
pie     an apple pie im going
pie     an apple pie in the
snow    apple pie in the snow
snow    lot of snow i made
snow    out of snow i would
snow    the snow storm had a
snowman made a snowman out of
storm   the snow storm had a
the     apple pie in the snow
the     the snow storm had a
to      im going to eat an
want    no i want an apple
would   snow i would like an

C:\Users\melan_000\workspace\PlayingWithJava>
```

8) Give input less than 5 words Expect: Message saying that 5 or more words are required

```
C:\Users\melan_000\workspace\PlayingWithJava>java -classpath "C:\Program Files (
x86)\eclipse\plugins\google-collections-1.0-rc1.jar";  playingWithJava.PlayingWi
thJava
:Bob and Sarah
There are not enough words to create a KWIC. Use 5 or more words.
:I like monster drinks
There are not enough words to create a KWIC. Use 5 or more words.
:
```