ExtractLine

```
ExtractLine test case: 0
Basically:1->[0]
I:1->[0]
scanning:1->[0]

ExtractLine test case: 1
a:1->[1]
Basically:1->[0]
file:1->[1]
from:1->[1]
fscanf:1->[1]
I:1->[0]
in:1->[1]
scanning:1->[0]
strings:1->[1]
the:1->[1]
using:1->[1]

ExtractLine test case: 2
a:1->[1]
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]
```

```
String fileName1="files/testfile1";
          String fileName2="files/testfile2";
          WordProcessor wp = new WordProcessor();

          //MyLinkedList allWords = wp.extractAll(fileName1);


          //Test ExtractLine
          MyLinkedList allWords = new MyLinkedList();
          ArrayList<String> x = wp.fileRead(fileName1);
         String[] str = new String[x.size()];

          for(int i = 0; i < 3; i++) {
             str[i] = x.get(i);
               MyLinkedList testExtractLine =
wp.extractLine(str[i], i, allWords);
```

Alphabetical and Occurrence Sorting

```
//allWords.MergeSortAlpha();
              //allWords.MergeSortOcc();
              //System.out.println("mergeSortOcc Test Case:
" + i + "\n" + allWords);

          System.out.println("mergeSortOcc test Case: 4");
          MyLinkedList extractAll = wp.extractAll(fileName1);
          System.out.println(extractAll);
```

```
mergeSortOcc Test Case: 0
I:1->[0]
Basically:1->[0]
scanning:1->[0]

mergeSortOcc Test Case: 1
Basically:1->[0]
I:1->[0]
fscanf:1->[1]
from:1->[1]
file:1->[1]
a:1->[1]
using:1->[1]
the:1->[1]
strings:1->[1]
scanning:1->[0]
in:1->[1]

mergeSortOcc Test Case: 2
for:1->[2]
each:1->[2]
calling:1->[2]
Basically:1->[0]
in:1->[1]
scanning:1->[0]
strings:1->[1]
the:1->[1]
using:1->[1]
then:1->[2]
string:1->[2]
my:1->[2]
a:1->[1]
file:1->[1]
from:1->[1]
fscanf:1->[1]
I:1->[0]
function:1->[2]

mergeSortOcc test Case: 4
a:2->[1, 3]
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:3->[1, 3, 4]
for:1->[2]
from:2->[1, 3]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]
```

```
mergeSortAlpha Test Case: 0
Basically:1->[0]
I:1->[0]
scanning:1->[0]

mergeSortAlpha Test Case: 1
a:1->[1]
Basically:1->[0]
file:1->[1]
from:1->[1]
fscanf:1->[1]
I:1->[0]
in:1->[1]
scanning:1->[0]
strings:1->[1]
the:1->[1]
using:1->[1]

mergeSortAlpha Test Case: 2
a:1->[1]
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]
```

}

ContainWord

```
System.out.println("Test containWord: " + allWords);
          System.out.println("1. Basically " +
allWords.containWord("Basically", 0));
          System.out.println("2. scanning " +
allWords.containWord("scanning", 0)); System.out.println("3.
I " + allWords.containWord("I", 0));
```

```
Test containWord: a:1->[1]
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]

1. Basically true
2. scanning true
3. I true
```

RemoveFirst

```
System.out.println("Test removeFirst: \n" + allWords);
          allWords.removeFirst();
          System.out.println("Test removeFirst: \n" +
allWords);
          allWords.removeFirst();
          System.out.println("Test removeFirst: \n" +
allWords);
allWords.removeFirst();System.out.println("Test removeFirst:
\n" + allWords);
```

```
Test removeFirst:
a:1->[1]
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]

Test removeFirst:
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]

Test removeFirst:
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]

Test removeFirst:
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]
```

AddFirst

```
System.out.println("Test addFirst: \n" + allWords);
          allWords.addFirst("TestCase1");
          allWords.addFirst("TestCase2");
allWords.addFirst("TestCase3");
System.out.println(allWords);
```

```
Test addFirst:
a:1->[1]
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]

TestCase3
TestCase2
TestCase1
a:1->[1]
Basically:1->[0]
calling:1->[2]
each:1->[2]
file:1->[1]
for:1->[2]
from:1->[1]
fscanf:1->[1]
function:1->[2]
I:1->[0]
in:1->[1]
my:1->[2]
scanning:1->[0]
string:1->[2]
strings:1->[1]
the:1->[1]
then:1->[2]
using:1->[1]
```

Queue, enqueue, dequeue

```
// used for mergesort
Queue q = new Queue();
          MyLinkedList queueTestCase1 = new MyLinkedList();
          MyLinkedList queueTestCase2 = new MyLinkedList();
          MyLinkedList queueTestCase3 = new MyLinkedList();

          System.out.println("test enqueue");
          q.enqueue(queueTestCase1);
          q.enqueue(queueTestCase2);
          q.enqueue(queueTestCase3);

          System.out.println("test dequeue");
          q.dequeue();
          q.dequeue();
          q.dequeue();
```

compareTo

```
WordItem testWord1 = new WordItem("testCase1", 0, 9999);
        WordItem testWord2 = new WordItem("testCase2", 0,
9999);
        WordItem testWord3 = new WordItem("testCase3", 0,
9999);

        System.out.println("testCase1 " + testWord1);
        System.out.println("testCase2 " + testWord2);
        System.out.println("testCase3 " + testWord3 +
"\n");

        System.out.println("Compare testWord1 to testWord2:
" + testWord1.compareTo(testWord2));
        System.out.println("Compare testWord2 to testWord1:
" + testWord2.compareTo(testWord1));
        System.out.println("Compare testWord3 to testWord3:
" + testWord3.compareTo(testWord3));

        System.out.println("testCase1 " + testWord1);
        System.out.println("testCase2 " + testWord2);
        System.out.println("testCase3 " + testWord3 +
"\n");
```

```
testCase1 testCase1:0->[9999]
testCase2 testCase2:0->[9999]
testCase3 testCase3:0->[9999]

Compare testWord1 to testWord2: -1
Compare testWord2 to testWord1: 1
Compare testWord3 to testWord3: 0
```

updateItem

```
System.out.println("test updateItem");
          testWord1.updateItem(0);
          System.out.println("testCase1 " + testWord1);
          testWord2.updateItem(10);
          System.out.println("testCase2 " + testWord2);
          testWord3.updateItem(100);
          System.out.println("testCase1 " + testWord3);

          System.out.println("testCase1 " + testWord1);
          System.out.println("testCase2 " + testWord2);
          System.out.println("testCase3 " + testWord3 +
"\n");
```

```
testCase1 testCase1:0->[9999]
testCase2 testCase2:0->[9999]
testCase3 testCase3:0->[9999]

test updateItem
testCase1 testCase1:1->[9999, 0]
testCase2 testCase2:1->[9999, 10]
testCase1 testCase3:1->[9999, 100]
```

equals

```
System.out.println("Test equals on testWord1 and testWord2:
" + testWord1.equals(testWord2));
        System.out.println("Test equals on testWord2 and
testWord1: " + testWord2.equals(testWord1));
        System.out.println("Test equals on testWord3 and
testWord3: " + testWord3.equals(testWord3));
```

```
testCase1 testCase1:0->[9999]
testCase2 testCase2:0->[9999]
testCase3 testCase3:0->[9999]

Test equals on testWord1 and testWord2: false
Test equals on testWord2 and testWord1: false
Test equals on testWord3 and testWord3: true
```