**Individual Project:**

**Project Goals**

* Develop a Java application for analyzing text within a file.

**Details**

For this project, you must develop, using the Java language, a simple **command-line** utility called *textsummary*.

## Concise Specification of the textsummary Utility

* NAME:  
  textsummary - helps to analyze a text file.
* SYNOPSIS  
  textsummary OPT <filename>  
    
  where OPT can be **zero or more** of
  + -c <string>
  + -d [integer]
  + (-l|-s) [integer]
* COMMAND-LINE ARGUMENTS AND OPTIONS  
    
  <filename>: the file on which the textsummary operation has to be performed.

-c <string>: if specified, the textsummary utility will add the number of times the provided <string> appears in the line to the start of each line in the file.

-d [integer]: if specified, the textsummary utility will output the words (where a word is any sequence of alphanumeric characters) which are duplicated the most times throughout the file. By default it will output the single most common word. If a positive integer is provided in the optional parameter [integer], the utility will output that number of the most common words, starting from the most common. This option will always be executed first.

(-l|-s) [integer]: if specified, the textsummary utility will keep only the longest(-l) or shortest(-s) [integer](a required positive integer) lines in the file. Options -l and -s are mutually exclusive.

If **none** of the OPT flags is specified, textsummary will simply output the longest line in the file.

* NOTES
  + While the last command-line parameter provided is always treated as the filename, OPT flags can be provided in any order; though no matter the order of the parameters, if provided, -d will be applied first.
* EXAMPLES OF USAGE  
    
  **Example 1:**   
  textsummary file1.txt  
  File content:

3 dogs

a cat

Outputs “3 dogs”.  
  
**Example 2:**   
textsummary -d 2 -c “d” file1.txt  
File content:

dog bird cat cat

cat dog fish

Result file:

2 dog bird cat cat

1 cat dog fish

Outputs “cat dog”

**Example 3:**   
textsummary -l 1 file1.txt  
File content:

dog bird cat cat

cat dog fish

Result file:

dog bird cat cat

**Example 4:**   
textsummary -d -s 2 file1.txt  
File content:

dog bird cat cat

cat dog fish cat

dog

bird fish

Result file:

dog

bird fish

Outputs “cat”

It **must** pass the following JUNIT tests:

import org.junit.After;  
import org.junit.Before;  
import org.junit.Rule;  
import org.junit.Test;  
import org.junit.rules.TemporaryFolder;  
  
import java.io.\*;  
import java.nio.charset.Charset;  
import java.nio.charset.StandardCharsets;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
  
import static org.junit.Assert.\*;  
  
  
public class MainTest {  
  
 private ByteArrayOutputStream outStream;  
 private ByteArrayOutputStream errStream;  
 private PrintStream outOrig;  
 private PrintStream errOrig;  
 private Charset charset = StandardCharsets.*UTF\_8*;  
  
 @Rule  
 public TemporaryFolder temporaryFolder = new TemporaryFolder();  
  
 @Before  
 public void setUp() throws Exception {  
 outStream = new ByteArrayOutputStream();  
 PrintStream out = new PrintStream(outStream);  
 errStream = new ByteArrayOutputStream();  
 PrintStream err = new PrintStream(errStream);  
 outOrig = System.*out*;  
 errOrig = System.*err*;  
 System.*setOut*(out);  
 System.*setErr*(err);  
 }  
  
 @After  
 public void tearDown() throws Exception {  
 System.*setOut*(outOrig);  
 System.*setErr*(errOrig);  
 }  
  
 /\*  
 \* TEST UTILITIES  
 \*/  
  
 // Create File Utility  
 private File createTmpFile() throws Exception {  
 File tmpfile = temporaryFolder.newFile();  
 tmpfile.deleteOnExit();  
 return tmpfile;  
 }  
  
 // Write File Utility  
 private File createInputFile(String input) throws Exception {  
 File file = createTmpFile();  
  
 OutputStreamWriter fileWriter =  
 new OutputStreamWriter(new FileOutputStream(file), StandardCharsets.*UTF\_8*);  
  
 fileWriter.write(input);  
  
 fileWriter.close();  
 return file;  
 }  
  
  
 //Read File Utility  
 private String getFileContent(String filename) {  
 String content = null;  
 try {  
 content = new String(Files.*readAllBytes*(Paths.*get*(filename)), charset);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return content;  
 }  
   
 /\*  
 \* TEST FILE CONTENT  
 \*/  
 private static final String *FILE1* = "3 dogs" + System.*lineSeparator*() + "a cat";  
 private static final String *FILE2* = "dog bird cat cat" + System.*lineSeparator*() + "cat dog fish";  
 private static final String *FILE3* = "dog bird cat cat" + System.*lineSeparator*() + "cat dog fish cat"   
 + System.*lineSeparator*() + "dog" + System.*lineSeparator*() + "bird fish";  
 private static final String *FILE4* = "Log: 123 abc" + System.*lineSeparator*() +   
 "Error: 456123123 123 xyz" + System.*lineSeparator*() + "Log: 456 cde" + System.*lineSeparator*() +   
 "Log: 1231 cde" + System.*lineSeparator*() + "Error: 123123 ab c";  
  
  
 /\*  
 \* TEST CASES  
 \*/  
  
@Test  
 public void mainTest1() throws Exception {  
 File inputFile = createInputFile(*FILE1*);  
  
 String args[] = {inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = *FILE1*;  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 *assertEquals*("3 dogs", outStream.toString().trim());  
 }  
  
 // Purpose: To provide an example of a test case format  
 // Frame #: Instructor example 2 from assignment directions  
 @Test  
 public void mainTest2() throws Exception {  
 File inputFile = createInputFile(*FILE2*);  
  
 String args[] = {"-b", "2", "-a", "d", inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = "2 dog bird cat cat" + System.*lineSeparator*()   
 + "1 cat dog fish";  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 *assertEquals*("cat dog", outStream.toString().trim());  
 }  
  
 // Purpose: To provide an example of a test case format  
 // Frame #: Instructor example 3 from assignment directions  
 @Test  
 public void mainTest3() throws Exception {  
 File inputFile = createInputFile(*FILE2*);  
  
 String args[] = {"-l", "1", inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = "dog bird cat cat";  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 }  
  
 // Purpose: To provide an example of a test case format  
 // Frame #: Instructor example 4 from assignment directions  
 @Test  
 public void mainTest4() throws Exception {  
 File inputFile = createInputFile(*FILE3*);  
  
 String args[] = {"-d", "-s", "2", inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = "dog" + System.*lineSeparator*()   
 + "bird fish";  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 *assertEquals*("cat", outStream.toString().trim());  
 }  
  
 // Purpose: To provide an example of a test case format  
 // Frame #: Extra example  
 @Test  
 public void mainTest5() throws Exception {  
 File inputFile = createInputFile(*FILE4*);  
  
 String args[] = {"-s", "1", "-d", "4", inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = "Log: 123 abc";  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 *assertEquals*("Log 123 Error cde", outStream.toString().trim());  
 }  
  
 // Purpose: To provide an example of a test case format  
 // Frame #: Extra example  
 @Test  
 public void mainTest6() throws Exception {  
 File inputFile = createInputFile(*FILE4*);  
  
 String args[] = {"-c", "123", "-d", inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = "1 Log: 123 abc" + System.*lineSeparator*() +   
 "3 Error: 456123123 123 xyz" + System.*lineSeparator*() + "0 Log: 456 cde" + System.*lineSeparator*() +   
 "1 Log: 1231 cde" + System.*lineSeparator*() + "2 Error: 123123 ab c";  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 *assertEquals*("Log", outStream.toString().trim());  
 }  
  
  
 // Purpose: To provide an example of a test case format (no arguments passed)  
 // Frame #: Instructor error example  
 @Test  
 public void mainTest7() {  
 //if no arguments are entered on the command line it will pass an array of length 0 to the application, not null.  
 String args[] = new String[0];  
 Main.*main*(args);  
 *assertEquals*("Usage: textsummary [-d [int]] [-c string] [-l int | -s int] <filename>", errStream.toString().trim());  
 }  
  
  
 // Purpose: To provide an example of a test case format  
 // Frame #: Extra example  
 @Test  
 public void mainTest8() throws Exception {  
 File inputFile = createInputFile(*FILE4*);  
  
 String args[] = {"-c", "o", "-s", "2", inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = "1 Log: 123 abc" + System.*lineSeparator*() + "1 Log: 456 cde";  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 }  
  
  
 // Purpose: To provide an example of a test case format  
 // Frame #: Extra example  
 @Test  
 public void mainTest9() throws Exception {  
 File inputFile = createInputFile(*FILE4*);  
  
 String args[] = {"-c", "\*", "-d", "2", "-l", "1", inputFile.getPath()};  
 Main.*main*(args);  
  
 String expected = "0 Error: 456123123 123 xyz";  
  
 String actual = getFileContent(inputFile.getPath());  
  
 *assertEquals*("The files differ!", expected, actual);  
 *assertEquals*("Log 123", outStream.toString().trim());  
 }  
  
  
 // Purpose: To provide an example of a test case format (error in options)  
 // Frame #: Extra example  
 @Test  
 public void mainTest10() throws Exception {  
 File inputFile = createInputFile(*FILE4*);  
  
 String args[] = {"-c", "log", "-l", "1", "-z", inputFile.getPath()};  
 Main.*main*(args);  
 *assertEquals*("Usage: textsummary [-d [int]] [-c string] [-l int | -s int] <filename>", errStream.toString().trim());  
 }  
}

}