Lab 7 - Question 4

Question

Write a program that display the number of characters, words and lines in a text file. Assume that each word is separated by one space character.

Solution

To solve the question, we need to:

1. Read the text from file
2. Count the number of characters
3. Count the number of words
4. Count the number of lines

Step 1 - Declare function

To start, let's start with a `countTextFromFile` function.

```
public static void countTextFromFile() {}
```

Noted that the function should read the file from given path. So, the function should take `String` as parameter.

Therefore, the method header should be modified as follows:

```
public static void countTextFromFile(String filePath) {}
```

Step 2 - Try-Catch

Now, let's write the method body. Use `BufferedReader` to read `.txt` file and use `try-catch` to handle `IOException`.

```

```

```
public static void CountTextFromFile(String filePath) {
    try {
        BufferedReader reader = new BufferedReader(new
FileReader('reference.txt'));
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

Step 3 - Using `.readLine()`

L et's declare a `String`. Use `BufferedReader` to read the line in `reference.txt` using `.readLine()` and pass the value to `line`.

```
public static void CountTextFromFile(String filePath) {
    try {
        BufferedReader reader = new BufferedReader(new
FileReader('reference.txt'));
        String line = reader.readLine();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

Step 4 - Using `while` loop

So far, our code can read only **ONE LINE** in the `reference.txt`. However, `reference.txt` may contains multiple lines. For illustration:

```
This is the first line
This is the second line
This is the third line
This is the last line
```

To read multiple lines without knowing the exact number of line existed in the file, we should use `while` loop.

```
public static void CountTextFromFile(String filePath) {
    try {
        BufferedReader reader = new BufferedReader(new
FileReader('reference.txt'));
```

```
        String line = "";
        while((line = reader.readLine()) != null) {

        }

    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

The boolean condition in `while` loop is set to `(line = reader.readLine()) != null)`. It means that:

1. First, assign line with `reader.readLine()`
2. Second, check if `line != null`. If yes, keep reading the next line and vice versa.

Step 5 - Counting

```
public static void CountTextFromFile(String filePath) {
    int numOfWords = 0, numOfChars = 0, numOfLines = 0;

    try {
        BufferedReader reader = new BufferedReader(new
FileReader('reference.txt'));

        String line = "";
        while(line = reader.readLine()) != null) {
            numOfLines++;
            numOfChars += line.length;
            numOfWords += line.split(" ").length();
        }

        System.out.printf("Lines: %d\nChars: %d\nWords: %d", numOfLines,
numOfChars, numOfWords);

    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

1. Since we want to count on lines, characters and words, we have to first declare them first.
2. Every time we successfully read a line, it means there is a line. So we can use `numOfLines++` to indicate that.
3. The number of character in each line equals the length of that line. We use `numOfChars += line.length` here.

4. We can use `.split(" ")` to get a String array and use `.length()` to get the size of the array. The array, in fact, stores every word in the line. So the size of the array indicates the number of word in each line. Hence, we can use `numOfWords += line.split(" ").length()` to mean that.