Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

20.17 Ritchie Program 3: Tokenizer

Program 3

Due: Friday 10/12/19 , 11:59PM

Grade: 90% Zybooks unit tests, 10% I will hand grade your assignments, I will be looking for appropriate comments, and correct implementation of the classes (IE no hard coding the answers).

In this assignment you will be creating a tokenizer. A tokenizer refers to the process of turning a section of characters in a string into a single entity called a token. A single string can in fact have multiple tokens associated with it. A token is usually determined by something called a delimiter. A delimiter is a specific character that will be used as the location (or locations) where the string is split into tokens. Let us look at an example:

Given the following string: “The cat in the hat” And let the delimiter for our tokenization process be a single space character: ‘ ‘ Then the resulting tokens from the above string would be:

“The” , “cat”, “in”, “the”, “hat”

In which case we would have 5 tokens. What if the string was the following:

“TheCat inthe Hat”

Notice the lack of a space character between some of the words in the string. Therefore, based on the same delimiter, the resulting tokens would be:

“TheCat”, “inthe”, “Hat”

In this case there are only three tokens that resulted from that string.

For this programming assignment you will be creating a class that is able to read input from a text file, tokenize the content of that file based on the space character as a delimiter, store the tokens in an ArrayList, and write the tokens back to a file “output.txt”.

Program to turn in:

You will turn in multiple files: Token.java Tokenizer.java

The Token.java file should contain:

- A class called: Token

- Token should have a private String variable that holds the value associated with a particular token
    - For example, if the string “The cat in the hat” gets tokenized then the value of one Token object might be: “The” , or might be “cat” etc..
- Create getters and setters for the private String variable called:
    - getValue, and setValue
- Token should have a constructor that has one String parameter, this parameter should be used to set the value of private string variable
- Override the toString method inherited from the Object class, and have it return the value of the private String variable.

The Tokenizer.java file should contain:

- A class called: Tokenizer
- Tokenizer should have a private variable that is an ArrayList of Token objects.
- Tokenizer should have a private variable that is an int, and keeps track of the number of keywords encountered when parsing through a files content.
- In this case there will only be one keyword: public.
- Tokenizer should have a default constructor that initializes the ArrayList of Token objects
- Tokenizer should have a public method called: tokenizeFile
    - tokenizeFile should have a string parameter thatwill be a file path
    - tokenizeFile should return void
    - tokenizeFile should throw an IOException
    - tokenizeFile should take the contents of the file and tokenize it using a space character as a delimiter
        - If there are multiple spaces between tokens then ignore those spaces and only retrieve the words, so given the string: “The cat” should still only produce two tokens: “The”, and “cat”
        - A newline should also serve to separate tokens, for example:
            - If the input contains multiple lines such as:

```
The cat in the hat
Red fish blue fish
```

            - There is no space between “hat” and “Red”, just a newline, which means the resulting tokens are “hat” and “Red”, NOT a single token: “hatRed”
    - Each token should be added to the private ArrayList variable as a Token object
    - If the value of a token is the keyword: public , then increment the private int counter that is keeping track of the number of keywords encountered when parsing the content of a file. -

Remember that “public” is the only keyword you have to worry about - Any letter in the word: public , can be capitalized and you must still increment the counter - If public is part of another token, then it should NOT count as a keyword - For example if a file contained: - Blahblahpublic - The above public does not count as a keyword since public in this case is not its own token

- Every call to tokenizeFile should first clear the previous tokens from the ArrayList of Tokens, as well as reset the private int counter of keywords (public) - For example: if a user calls tokenizeFile(“someFile.txt”) and it generates these tokens: “The” , “cat”, “in” , and then the user calls tokenizeFile again but with a different input file, such as: tokenizeFile(“somenewfile.txt”), the ArrayList should no longer hold the previous tokens, and should instead hold only the new tokens generated by the new file.

- Tokenizer should have only a getter for the ArrayList of Token objects (not a setter)
- The getter should be called: getTokens and should return an Array of Token objects, NOT an ArrayList of Token objects
- Tokenizer should have a method called: writeTokens
  - writeTokens should have no input parameter
  - writeTokens should have a return type of void
  - writeTokens should throw an IOException
  - writeTokens should write the tokens out to a file called: “output.txt”, and should reside in your current working directory. Each token should be written on a newline, for example - Given tokens: “The”, “cat”, “in”, “the”,”hat” The result in output.txt should be:

```
The
cat
in
the
hat
```

    - Notice the order the tokens should be written to file are in order from which they are read from the input file, which should be from left to right and from top to bottom
- Tokenizer should have a public method called: getKeywordCount , that has no input arguments and returns the private int keyword counter field
- Override the toString method inherited from the Object class, have it return the same value as calling the toString method on the ArrayList of Token objects

Suggestions and Hints

- If you are feeling unsure of how to use an ArrayList or some of the File IO objects, be sure to reread zybooks: ch 7.12 and ch 10.5.
  - You can also search on the internet for java’s documentation on these objects (remember that java is owned by Oracle, so the documentation website url should be something along the lines of: docs.oracle.com/…etc)

- You may find the Scanner class's "next" method useful
- You may find the String class's "split" method useful
- Remember you can test your code by creating your own file, let's say for example: Application.java, creating a class called: Application, and creating a "main" method. Then you can create your own tokenizer object and test some of the methods. (Just remember not to turn in this extra file)
- Do not turn in Token.java or Tokenize.java with a "main" method in either of these files.
    - So don't have this: public static void main(String[] args) … etc

LAB ACTIVITY | 20.17.1: Ritchie Program 3: Tokenizer | 56 / 56

Submission Instructions

Downloadable files

`example.txt` **Download**

Compile command

```
javac Tokenizer.java Token.java -Xlint:all -encoding utf-8
```

We will use this command to compile your code

Upload your files below by dragging and dropping into the area or choosing a file on your hard drive.

Tokenizer.java Drag file here or **Choose on hard drive.**

Token.java Drag file here or **Choose on hard drive.**

Submit for grading

Latest submission - 5:08 PM on 10/13/19 — Submission passed all tests ✓ — Total score: 56 / 56

☐ Only show failing tests **Download this submission**

1: Unit Test 1 ^ 2 / 2

Test Correct File Input

Your output

```
No IOException should be thrown... GOOD!! No
```

2: Unit Test 2 2 / 2

Test Incorrect File Input

Your output

```
IOException should be thrown... GOOD! The fo
wrongfile.txt (No such file or directory)
```

3: Unit Test 3 2 / 2

Test Token toString

Your output

```
Congrats! Your Tokens's toString method retu
SOMESTRING
```

4: Unit Test 4 2 / 2

Test Token getValue

Your output

```
Congrats! Your Tokens's toString method retu
SOMESTRING
```

5: Unit Test 5 2 / 2

Test Token getValue

Your output

```
Congrats!! Your Tokens's setValue method set
```

6: Unit Test 6 4 / 4

Test Tokenizer toString with example.txt file

Your output

```
Congrats! Your Tokenizer's toString method r
[The, cat, in, the, hat, TheCat, inThe, hat,
```

7: Unit Test 7 4 / 4

Test Tokenizer toString with test.txt file

Your output

```
Congrats! Your Tokenizer's toString method r
[The, cat, in, the, hat, The, Cat, In, The,
```

8: Unit Test 8 4 / 4

Test Tokenizer getKeywordCount with example.txt file

Your output

```
Correct!! Special Token Count for example.tx
```

9: Unit Test 9 4 / 4

Test Tokenizer getKeywordCount with test.txt file

Your output

```
Correct!! Special Token Count for example.tx
```

10: Unit Test 10 4 / 4

Test Tokenizer getTokens with example.txt file

Your output

```
Correct!! Your tokens are accurate and you h
```

11: Unit Test 11 6 / 6

Test Tokenizer getTokens with test.txt file

Your output

```
Correct!! Your tokens are accurate and you h
```

12: Unit Test 12 6 / 6

Test Tokenizer clears ArrayList and Special Token Count with example.txt file

Your output

```
Correct!! Your tokens are accurate and you h
```

13: Unit Test 13 6 / 6

Test Tokenizer writeTokens with example.txt file

Your output

```
Correct!! You have correctly written all of
```

14: Unit Test 14 8 / 8

Test Tokenizer writeTokens with test.txt file

Your output

```
Correct!! You have correctly written all of
```

Previous submissions

3:34 PM on 10/13/19	56 / 56	View
3:21 PM on 10/13/19	28 / 56	View

Trouble with lab?