CSC 345 Lab – Scanner

Overview

Write and use a scanner class to recognize data from an input file.

Part 1

Create a class named LabScanner and use it in main to scan an input file.

- Create a class named LabScanner.
    - Token Enum. Add an enum for the tokens as a member variable. It should contain enum values for +, -, and end of file.
    - PushbackReader.
        - Add a member variable for a PushbackReader. This will be the input stream that is read from by the scanner.
        - Add a set method for the PushbackReader.
    - readNextChar Method. Write a method to read the next character from the input stream and return that value. It should read data from the PushbackReader member variable. Here is the method header: int readNextChar()
    - scan Method. Write the scan method. Here is the method header: TOKEN scan()
        - There should be a while loop that keeps going until the end of file. A character value of -1 means the end of file has been reached.
        - The scan method should ignore all white space. You can write a method to check if a character is white space and call it from scan() as necessary. If a white space character is encountered it should read the next character and go back to the loop test.
        - Check if + was the character just read in. If it is, then return the token corresponding to it.
        - Check if - was the character just read in. If it is, then return the token corresponding to it.
        - Add code after the loop to return the end of file token.
    - Create an input file in the project. This file should only contain +, - and whitespace characters.
    - Write code in main to use the LabScanner class.
        - Open the input file and connect it to a PushbackReader instance.
        - Create an instance of LabScanner.
        - Set the PushbackReader on the LabScanner instance.
        - Write a loop will keep scanning tokens until it reaches the end of file token. Print each token that it scans on screen.

Part 2

Update the scan method and the token enum so that the scanner can recognize (and). Change the input file so that it contains (and).

Part 3

Update the scan method and the token enum so that the scanner can recognize an intliteral. An intliteral contains multiple digits. After it reaches the end of the intliteral it must push the last character read back on to the input stream. The last character read is not part of the intliteral. It must be added back to the input stream so that it is there for the next call to scan.

Hints: You can use the static isDigit method of the Character class to help with this. You can write an unreadChar method.

Part 4

Add a buffer to hold the token characters that were read in.

- Add a StringBuilder member variable to the LabScanner (call it tokenString).
- It should clear tokenString as necessary in the scan method. Call the setLength method on StringBuilder passing in 0 to clear it.
- Keep adding characters to the end of the token string as necessary.
- You should also add a method named getTokenString to the LabScanner class that returns the string being stored in tokenString.
- Add code to main that prints the token string on screen (the token string is what is in the token string buffer).

Part 5

Write JUnit tests for plus and intliteral (there need to be two test methods in the test class). Each test method needs to do the following:

- Setup a PushbackReader and a LabScanner. Use a StringReader instead of a FileReader when setting up the PushbackReader. The StringReader constructor takes a String as a parameter. Pass in the code to test to the StringReader constructor. For example, new StringReader(“+”) for plus and new StringReader(“123”) for an intliteral. The
- Call the scan method on the LabScanner instance.
- Execute assertions to make sure that the correct token was returned and that the token string buffer has the correct string in it.

Part 6

Update the scan method and the token enum so that the scanner can recognize an id (an identifier). The first character in an identifier should be a character (not a digit or anything else). Characters after the

first can be alphanumeric (letter or digit). It should save characters to the token string buffer just like for intliteral. It should also show the token string buffer in main for id tokens. Hint: You can use the static isLetter and isDigit methods of the Character class to help with this.