CMSC 315, Project 1

Alberth Matos

21 Jan 2026

1 Design

The goal of the project was to implement a program that checks for balanced delimiters within a given file, using a stack-based approach to track that each right-delimiter matched its corresponding left-delimiter. Of note, the project instructions state the following:

> It should then repeatedly call the method that returns the next character until it returns a null character indicating the end of the file or until a mismatch of delimiters is encountered. If the character is a left delimiter it should be pushed onto a delimiter stack. *If it is a right delimiter, the stack should be popped and a check should be made to ensure that the delimiters are of a matching type*. If the delimiters do not match, a message should be displayed indicating what delimiter was encountered and at what position.

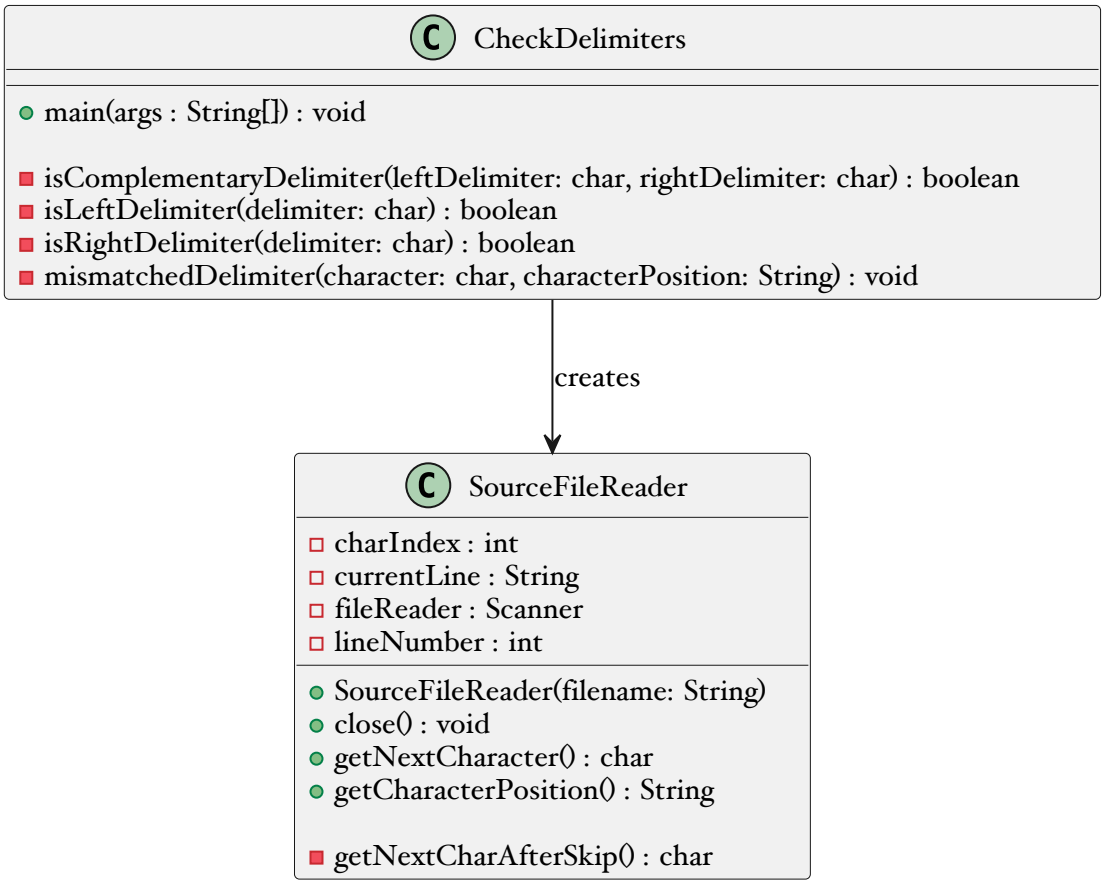
This can cause confusion when dealing with extraneous delimiters, as the instructions state that every time a right-delimiter is encountered, the stack should be popped, removing the last left-delimiter. In the case of an extraneous right-delimiter, a potentially valid left-delimiter and right-delimiter pair would not be matched if an extra right-delimiter is encountered.

The following code snippet illustrates the potential issue:

```
1 public class badExample {
2     ) // Bad delimiter
3 } // Unmatched good delimiter
```

Using this code snippet, following the project requirements as written, the program will display an error message indicating that an unmatched right-delimiter was encountered at line 3, position 1. This is actually incorrect, as the actual problem is that there is an unmatched left-delimiter at line 2, position 3.

1.1 UML Diagram



2 Test Plan

Test Plan:[1]

1. Invalid filename provided; program should display an error message indicating that the file does not exist, and re-prompt the user for a filename.

2. Valid input with matching delimiters; Several delimiters are included in the input file, *goodfile.java*, both inside and outside of comments. The program should correctly identify and match all delimiters, and not display any error messages. The program should especially not display any error messages related to unmatched delimiters inside of comments.

3. Valid input with mismatched delimiters; a total of six (6) mismatched delimiters are included in the input file, *badfile1.java*. The program should correctly identify and report all mismatched delimiters, and not display any error messages related to unmatched delimiters inside of comments.

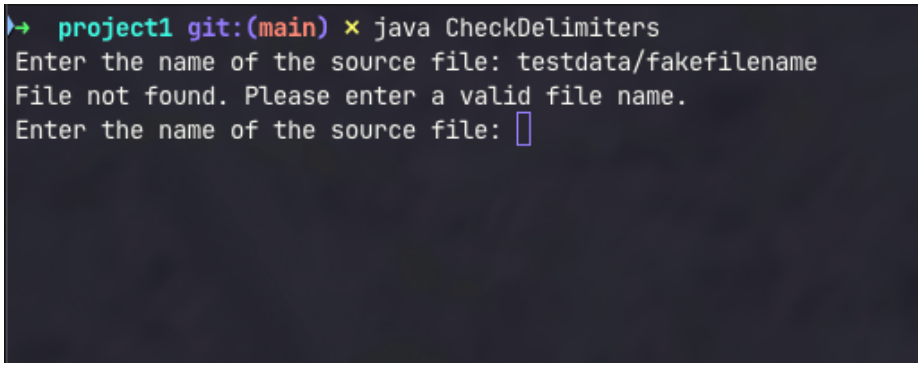
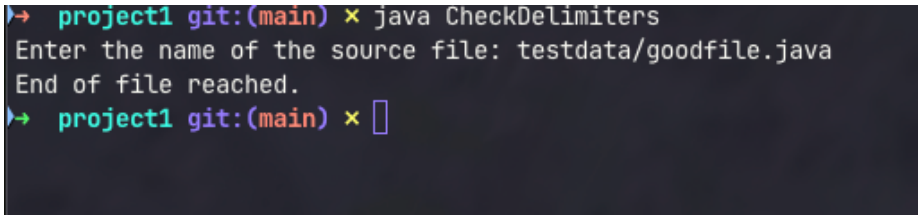
4. Valid input, *badfile2.java*, with an extraneous left-delimiter, as well as a left-delimiter inside a comment. The program should correctly identify and report the single left-delimiter.

[1]Source code files used as test data are provided in Section 4.

5. Valid input, *badfile3.java*, with an extraneous right-delimiter, as well as a right-delimiter inside a comment. The program should correctly identify and report the bad right-delimiter, as well as the right-delimiter that would normally be matched at the end of the file.

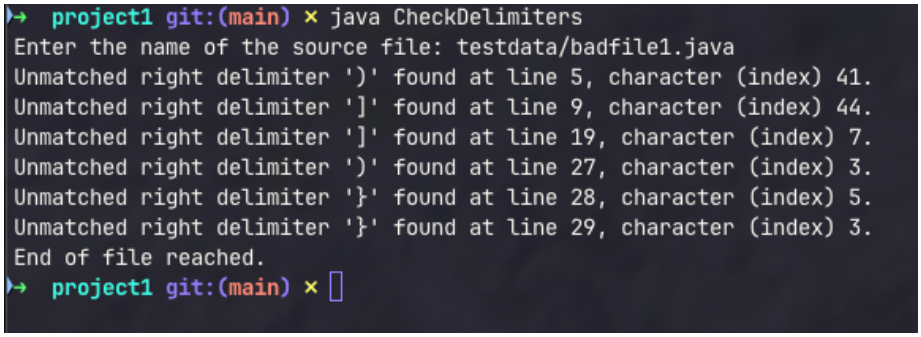
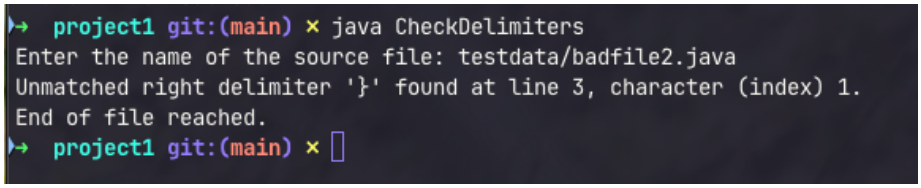
2.1 Test Results

Table 1: Test Cases

Test Case	Input	Expected Output
1	Invalid filename	Message displays that the file does not exist, and re-prompt the user for the filename.
		
2	Valid input with matching delimiters, file: goodfile.java 4.1	No error message, program completes successfully and displays "End of file reached."
		

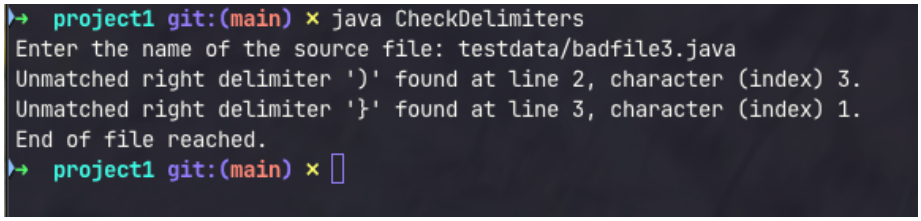
Continued on next page

Table 1: Test Cases (Continued)

Test Case	Input	Expected Output
3	Invalid input with mismatched left-delimiters, file: badfile1.java [4.2]	Program displays error messages for each mismatched delimiter. Unmatched right delimiter ')' found at line 5, character (index) 41. Unmatched right delimiter ']' found at line 9, character (index) 44. Unmatched right delimiter ']' found at line 19, character (index) 7. Unmatched right delimiter ')' found at line 27, character (index) 3. Unmatched right delimiter '}' found at line 28, character (index) 5. Unmatched right delimiter '}' found at line 29, character (index)) 3.
		
4	Invalid input with mismatched right-delimiters, file: badfile2.java [4.3]	Program displays error messages for each mismatched delimiter. Unmatched right delimiter '}' found at line 3, character (index) 1.
		
5	Invalid input with mismatched right-delimiters, file: badfile3.java [4.4]	Program displays error messages for each mismatched delimiter. Unmatched right delimiter ')' found at line 2, character (index) 3. Unmatched right delimiter '}' found at line 3, character (index) 1.

Continued on next page

Table 1: Test Cases (Continued)

Test Case	Input	Expected Output
		

3 Lessons Learned

The single most important lesson learned from the project was the importance of reading the requirements carefully and thoroughly. When initially creating the pseudocode framework of the project, this programmer initially assumed that one should trap each individual delimiter, and not necesserily pop a left-delimiter off of the stack if the next right-delimiter did not match. This assumption would have increased the complexity of the code, and would have discarded one of the requirements of the project, that each time a right-delimiter was encountered, one *must* pop the left-delimiter off of the stack.

A second read through the requirements allowed the programmer to catch this error in understanding, and correct the pseudocode before writing the actual program.

4 Test Data

4.1 goodfile.java

```java
/*
 * multi-line comment
 */
public class goodfile {
  public static void main(String[] args) {
    // single-line comment
    // single-line comment
    // Initialize scanner to read from standard input
    Scanner scanner = new Scanner(System.in);
    while (reader == null) {
      System.out.print("Enter the name of the source file: ");
      String filename = scanner.nextLine();
      // Try to create a SourceFileReader object with the
         ↪ provided filename.
      // If the file is not found, catch the exception and prompt
         ↪  the user.
      try {
        reader = new SourceFileReader(filename);
      } catch (FileNotFoundException e) {
        System.out.println("File not found. Please enter a valid
           ↪ file name.");
      }
    }
  }
  private static void mismatchedDelimiter(char character, String
     ↪ characterPosition) {
    // Print an error message indicating the unmatched right-
       ↪ delimiter and
    // its position.
    System.out.println("Unmatched right delimiter '" + character
       ↪ + "' found at " + characterPosition);
  }
  private static boolean isMatchingType(char leftDelimiter, char
     ↪ rightDelimiter) {
    /*
    Long comment explaining the purpose of the method and its
       ↪ parameters.
    */
    switch(leftDelimiter) {
      case '(':
        return (rightDelimiter == ')');
      case '{':
        return (rightDelimiter == '}');
      case '[':
```

```
37          return (rightDelimiter == ']');
38        default:
39          return false;
40      }
41    }
42    private static boolean isRightDelimiter(char rightDelimiter) {
43      /*
44       If character is (, {, or [, return true, otherwise return
            ↪ false.
45       As with isMatchingType, this could get consolidated into a
            ↪ single return
46       statement, but is expanded into a switch for clarity.
47      */
48    }
```

4.2 badfile1.java

```
1  /*
2   * multi-line comment
3   */
4  public class badfile1 {
5    public static void main(String[] args)) { // Bad delimiter )
6      // single-line comment )
7      // single-line comment
8      // Initialize scanner to read from standard input
9      Scanner scanner = new Scanner(System.in]; // Bad delimiter ]
10     while (reader == null) {
11       System.out.print("Enter the name of the source file: ");
12       String filename = scanner.nextLine();
13       // Try to create a SourceFileReader object with the
             ↪ provided filename}.
14       // If the file is not found, catch the exception and prompt
             ↪  the user.
15       try {
16         reader = new SourceFileReader(filename);
17       } catch (FileNotFoundException e) {
18         System.out.println("File not found. Please enter a valid
               ↪ file name.");
19       ] // Bad delimiter ]
20       }
21     }
22     private static void mismatchedDelimiter(char character,
23     String characterPosition) {
24       // Print an error message indicating the unmatched right-
             ↪ delimiter and
25       // its position. [
26       System.out.println("Unmatched right delimiter '" +
             ↪ character + "' found at " + characterPosition);
```

```
27   ) // Bad delimiter )
28     }
29   }
```

4.3 badfile2.java

```
1 public class badfile2 {
2   [ // Bad delimiter
3 } // Unmatched good delimiter
```

4.4 badfile3.java

```
1 public class badfile3 {
2   ) // Bad delimiter
3 } // Unmatched good delimiter
```