

Chase Franse  
Dr. Hong  
COM 301  
20 November 2020

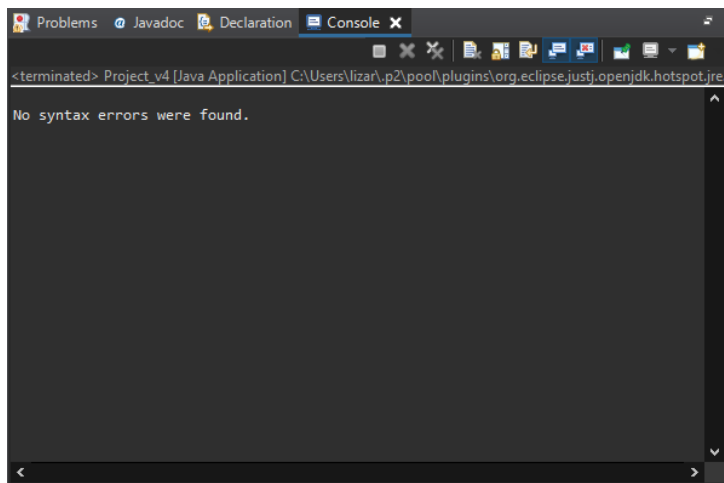
**Final Project Screenshots:**

- [Project v4](#)
- [Project v5 EXE](#)
- [Project v6](#)

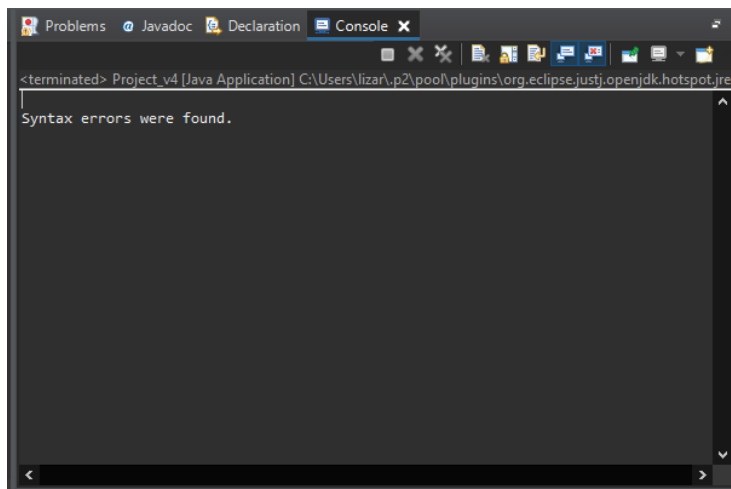
## Project\_v4:

```
10 //COM 301 Final Project v4.5 - Bracket Checker
11 //By: Chase S. xxxxxx
12 //File Manipulation Imports, et. al
13
14
15 import java.nio.file.*;
16 import java.io.*;
17 import java.util.*;
18 import static java.nio.file.StandardOpenOption.*;
19
20 public class Project_v4 {
21     /**
22      * ADDITIONAL FUNCTIONALITY:
23      * - CONVERT TO JFRAME [Done] {Version 5}
24      * -> PROMPT USERS FOR I/O PATHS [Done] {Version 5}
25      * -> Automatically know that the specified input file is a ".java" [Done] {Version 5}
26      * -> Automatically name the output file [Done] {Version 5}
27      * - DETERMINE AND OUTPUT WHERE THE ERRORS OCCUR
28      * -> Using the point in the stack, re-read the file and find that position's line number,
29      *     display that line number along with the error message(s).
30      * -> Display the remaining items in the stack to the output file. [Done] {Version 6}
31      */
32
33     public static void main(String[] args) {
34         //Get and create variables for the input and output paths
35         Path file = Paths.get("F:\\3 SENIOR YEAR\\1 FALL 20\\COM 301\\06\\PROJECT\\Helloworld.java");
36         //ADDITIONAL FUNCTIONALITY: Allow for the user to provide the file paths.
37         Path outputPath = Paths.get("F:\\3 SENIOR YEAR\\1 FALL 20\\COM 301\\06\\PROJECT\\Bracket Syntax Results.txt");
38
39         //Checks to see if the output file already exists or not. If so, it deletes it, and will replace it with a new one later.
40         File f = new File(outputPath.toString());
41         if (f.exists()) {
42             f.delete();
43         }
44
45         InputStream input = null;
46         OutputStream output = null;
47         byte[] data;
48         boolean CHECK = false;
49
50         //Use a TRY-CATCH just in-case there are any I/O issues
51         try {
52             //Create a Buffered Reader and Output Stream for reading and writing to the files specified above
53             input = Files.newInputStream(file);
54             output = new BufferedOutputStream(Files.newOutputStream(outputPath, CREATE));
55             BufferedReader reader = new BufferedReader(new InputStreamReader(input));
56             String s = null;
57             s = reader.readLine();
58             data = s.getBytes();
59
60             /*****
61              * PROGRAM MUST:
62              * - Read through EVERY line of the provided file. [WHILE LOOP]
63              * - Determine if there are any issues with the brackets. [IF STATEMENTS]
64              * - Bracket Syntax to check Include:
65              *
66              * -> [
67              * -> ]
68              * -> (
69              * -> )
70              * -> {
71              * -> }
```

*HelloWorld.java:*

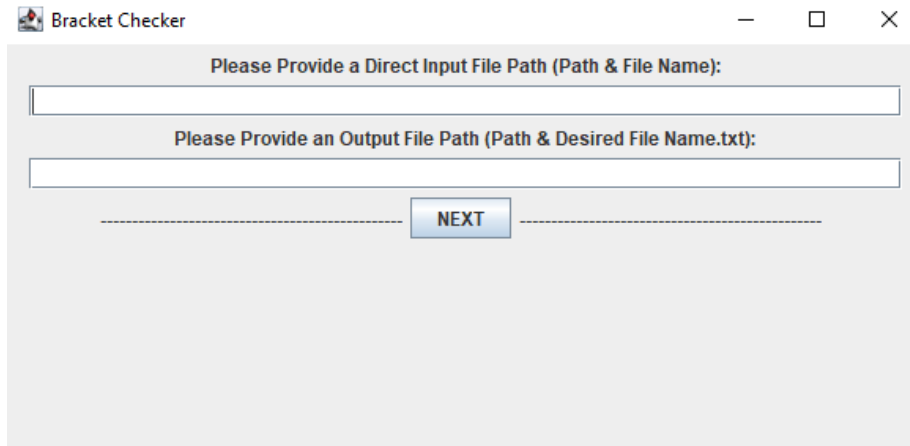


*HelloWorld\_wrong.java:*



## Project\_v5:

```
1 //COM 301 Final Project v5.6 - Chase S. France
2 //Project Imports
3 import java.nio.file.*;
4 import java.io.*;
5 import java.util.*;
6 import static java.nio.file.StandardOpenOption.*;
7
8 //JFrame Imports
9 import javax.swing.*;
10 import java.awt.*;
11 import java.awt.event.*;
12
13 public class Project_v5_EXE extends JFrame implements ActionListener{
14     /**
15      * ADDITIONAL FUNCTIONALITY:
16      * - CONVERT TO JFrame [Done] {Version 5}
17      * -> PROMPT USERS FOR I/O PATHS [Done] {Version 5}
18      * -> Automatically know that the specified input file is a ".java" [Done] {Version 5}
19      * -> Automatically name the output file [Done] {Version 5}
20      * - DETERMINE AND OUTPUT WHERE THE ERRORS OCCUR
21      * -> Using the point in the stack, re-read the file and find that position's line number,
22      *     display that line number along with the error message(s).
23      * -> Display the remaining items in the stack to the output file. [Done] {Version 6}
24     */
25
26     //JFrame Creation Elements
27     final int WIDTH = 600;
28     final int HEIGHT = 300;
29     JPanel section1 = new JPanel();
30     JLabel inputPathRequestDia = new JLabel("Please Provide a Direct Input File Path (Path & File Name):");
31     JTextField inputPathRequest = new JTextField((WIDTH/12));
32     JLabel outputPathRequestDia = new JLabel("Please Provide an Output File Path (Path & Desired File Name.txt):");
33     JTextField outputPathRequest = new JTextField((WIDTH/12));
34     JLabel leftSpacing = new JLabel("-----");
35     JLabel rightSpacing = new JLabel("-----");
36     JButton button = new JButton("NEXT");
37     JLabel result = new JLabel("");
38     JLabel test = new JLabel("TEST");
39
40
41     //JFrame Itself
42     public Project_v5_EXE() {
43         super("Bracket Checker");
44         setSize(WIDTH, HEIGHT);
45         setLayout(new FlowLayout(FlowLayout.CENTER));
46
47         add(inputPathRequestDia);
48         add(inputPathRequest);
49         add(outputPathRequestDia);
50         add(outputPathRequest);
51         add(leftSpacing);
52         add(button);
53         add(rightSpacing);
54         add(result);
55
56         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
57         button.addActionListener(this);
58     }
59
60     @Override
61     public void actionPerformed(ActionEvent x) {
62         String inputPath = inputPathRequest.getText();
63         String outputPath = outputPathRequest.getText();
64
65         inputPath.concat(".java");
66         outputPath.concat("Bracket Checker Results.txt");
67
68         //Get and create variables for the input and output paths
69         Path file = Paths.get(inputPath);
70         Path outPath = Paths.get(outputPath);
71
72         //Checks to see if the output file already exists or not. If so, it deletes it, and will replace it with a new one later.
73         File f = new File(outPath.toString());
74         if (f.exists()) {
75             f.delete();
76         }
77
78         InputStream input = null;
79         OutputStream output = null;
80         byte[] data;
81         boolean CHECK = false;
82     }
```



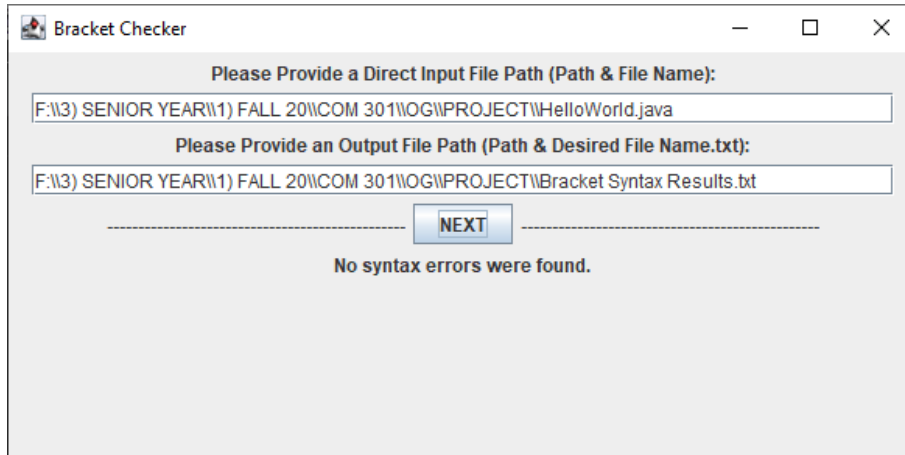
Bracket Checker

Please Provide a Direct Input File Path (Path & File Name):

Please Provide an Output File Path (Path & Desired File Name.txt):

NEXT

*HelloWorld.java:*



Bracket Checker

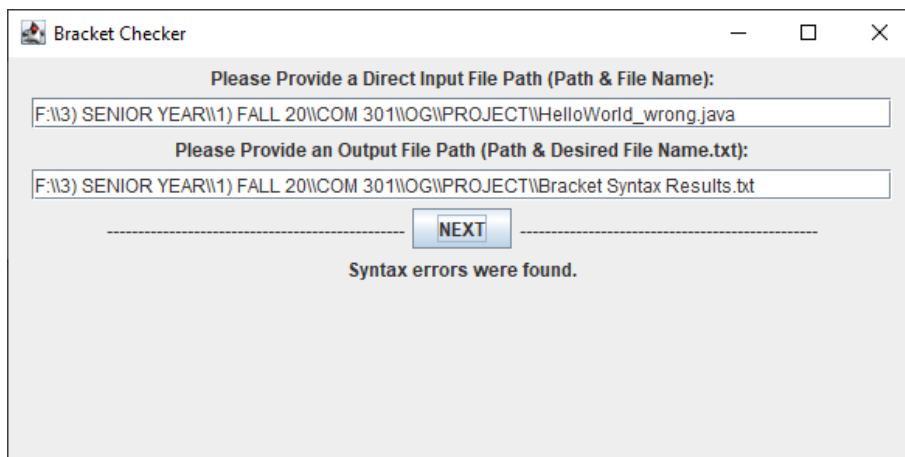
Please Provide a Direct Input File Path (Path & File Name):

Please Provide an Output File Path (Path & Desired File Name.txt):

NEXT

No syntax errors were found.

*HelloWorld\_wrong.java:*



Bracket Checker

Please Provide a Direct Input File Path (Path & File Name):

Please Provide an Output File Path (Path & Desired File Name.txt):

NEXT

Syntax errors were found.



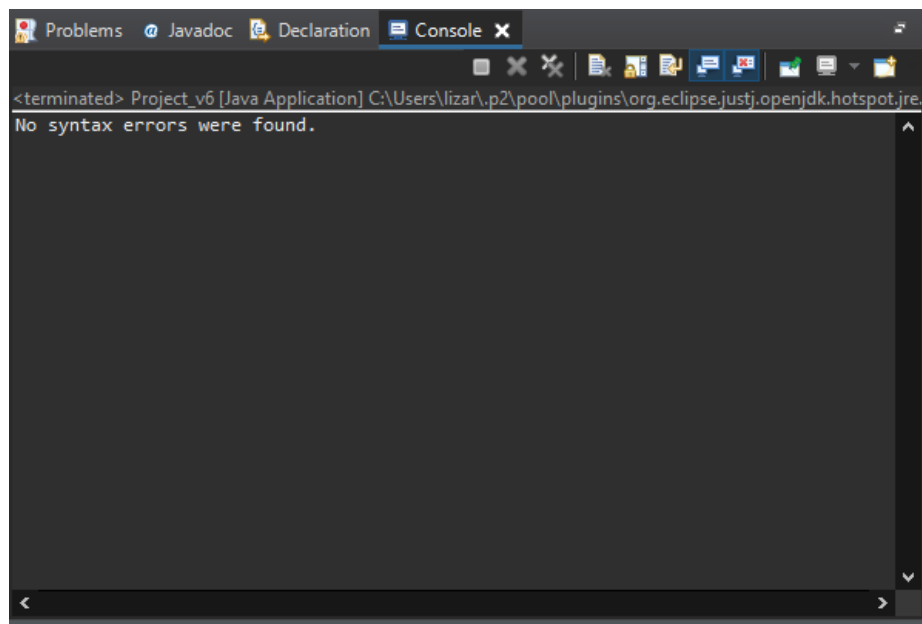
## Project\_v6:

```
10 //COM 301 Final Project v6.2 - Bracket Checker
11 //By: Chase S. Farnsworth
12 //File Manipulation Imports, etc.
13
14 import java.nio.file.*;
15 import java.io.*;
16 import java.util.*;
17 import static java.nio.file.StandardOpenOption.*;
18
19 public class Project_v6 {
20     /**
21      * ADDITIONAL FUNCTIONALITY:
22      * - CONVERT TO JFrame [Done] {Version 5}
23      * - PROMPT USERS FOR I/O PATHS [Done] {Version 5}
24      * - Automatically know that the specified input file is a ".java" [Done] {Version 5}
25      * - Automatically name the output file [Done] {Version 5}
26      * - DETERMINE AND OUTPUT WHERE THE ERRORS OCCUR
27      * - Using the point in the stack, re-read the file and find that position's line number,
28      *   display that line number along with the error message(s).
29      * - Display the remaining items in the stack to the output file. [Done] {Version 6}
30     */
31
32     public static void main(String[] args) {
33         //Get and create variables for the input and output paths
34         Path file = Paths.get("F:\\3 SENIOR YEAR\\1 FALL 20\\COM 301\\OG\\PROJECT\\HelloWorld.java");
35         //ADDITIONAL FUNCTIONALITY: Allow for the user to provide the file paths.
36         Path outputPath = Paths.get("F:\\3 SENIOR YEAR\\1 FALL 20\\COM 301\\OG\\PROJECT\\Bracket Syntax Results.txt");
37
38         //Checks to see if the output file already exists or not. If so, it deletes it, and will replace it with a new one later.
39         File f = new File(outputPath.toString());
40         if (f.exists()) {
41             f.delete();
42         }
43
44         InputStream input = null;
45         OutputStream output = null;
46         byte[] data;
47         boolean CHECK = false;
48
49         //Use a TRY-CATCH just in-case there are any I/O issues
50         try {
51             //Create a Buffered Reader and Output Stream for reading and writing to the files specified above
52             input = Files.newInputStream(file);
53             output = new BufferedOutputStream(Files.newOutputStream(outputPath, CREATE));
54             BufferedReader reader = new BufferedReader(new InputStreamReader(input));
55             String s = null;
56             s = reader.readLine();
57             data = s.getBytes();
58
59             /*****
60              * PROGRAM MUST:
61              * - Read through EVERY line of the provided file. [WHILE LOOP]
62              * - Determine if there are any issues with the brackets. [IF STATEMENTS]
63              * - Bracket Syntax to check Include:
64              * -> [
65              * -> ]
66              * -> (
67              * -> )
68              * -> {
69              * -> }
```

...

```
125
126 //Generates and fills output file when there is an error.
127 public static void publishStack(Stack<String> st, Path outputPath, OutputStream output) {
128     try {
129         byte[] data;
130         String errorMsg = "Syntax errors were found. \n\nItems Remaining in the Stack:\n";
131         data = errorMsg.getBytes();
132
133         for (int m = 0; m < st.size(); m++) {
134             String temp;
135             temp = st.pop();
136             temp = temp + "\n";
137
138             output.write(data);
139             data = temp.getBytes();
140         }
141
142         output.flush();
143     }
144     catch (IOException e) {
145         System.out.println(e);
146     }
147 }
148
149 }
```

*HelloWorld.java:*



*HelloWorld\_wrong.java:*

