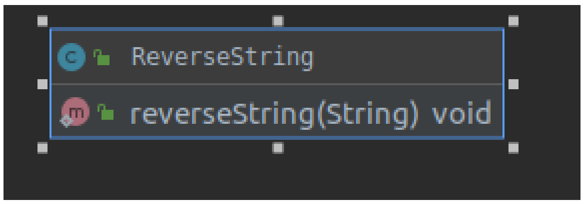
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 4 Question 3 Report

Melihcan Çilek
1801042092

Class Diagrams

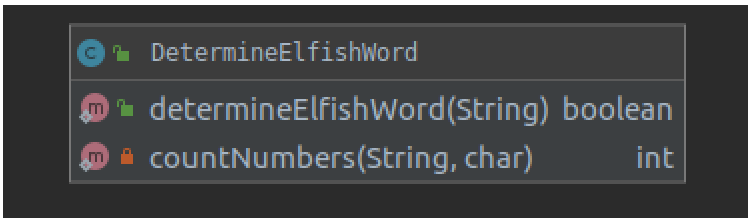
Question 1:



Base Case: If String send’s length is equal to length of last word splited by ‘ ’

Smaller Problems And Solution: In this problem, we have to split words by ‘ ’ and add them to a String array that includes all words by splited by ‘ ’ so we can print words and send that string to same function by last word deleted.

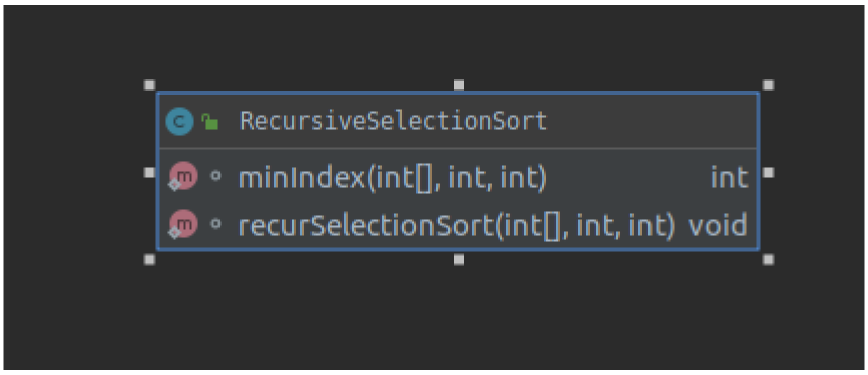
Question 2:


DetermineElfishWord
determineElfishWord(String) boolean
countNumbers(String, char) int

Base Case: If String send’s length is zero or not.

Smaller Problems And Solution: In this problem, we have to detect if a string contains characters ‘e’, ‘l’ and ‘f’ so we have to travel all characters in the string one by one so we can count number of these characters and determine whether this word sent is elfish or not.

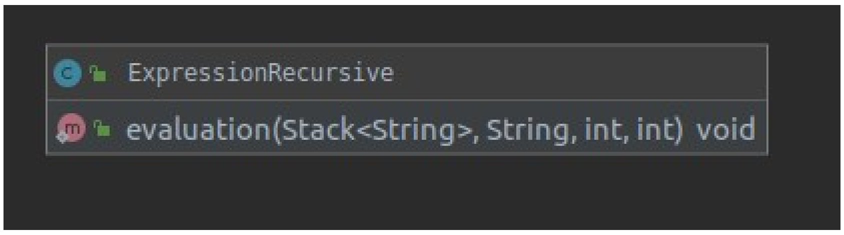
Question 3:


RecursiveSelectionSort
minIndex(int[], int, int) int
recurSelectionSort(int[], int, int) void

Base Case: If we reach end ot the index of array we sent

Smaller Problems And Solution: For solve that problem, we need to find smallest number of that array according to index sent so we can switch numbers if number that we found is less than our index.

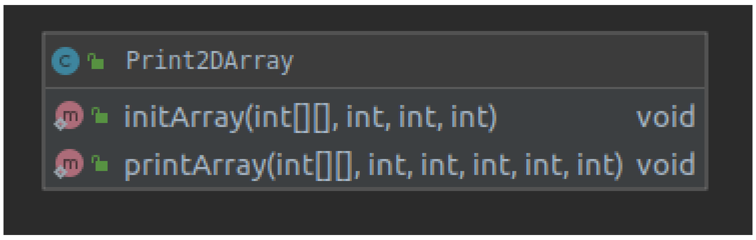
Question 4-5:



Base Case: If our expression that is sent to method as String is empty or not, did we reach end of the expression and our next character is ‘ ’ or not are our base cases.

Smaller Problems And Solution: For solve that problem, we have to do same thing that we did at second problem which is sending substrings and looking first character. If first character is a number, we push it to stack, if not, we pop two element from stack and do the expression and push solution to the stack. We can do this thing by using same method except changing evaluation mode. If evaluation mode is 1, this method doing postfix expression evaluation, if we use another number, this method evaluating prefix expression evaluation because postfix and prefix evaluation is nearly same. Differences between these evaluations are differentiated.

Question 6:


```
Print2DArray
initArray(int[][], int, int, int)             void
printArray(int[][], int, int, int, int, int)  void
```

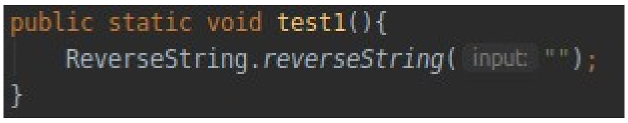
Base Case: There are 6 base case for this problem because while traveling around array, there are 6 point that we have to stop and control our position which are if we are at the beginning, if we reached top right, bottom right, bottom left corner or not, if we completed our one full loop or not and did we finish all loops or not. These are our control points.

Smaller Problems And Solution: For solve that problem, The way that I used is do drawing of array matrix and find the control points. After than writing recursive calls are really easy.

Problem Solution Approach:

In this question of the 4th assignment, we were asked to try to write the recursive codes of the given problems. If I need to explain it in the simplest form, I had to write the iterative states of these functions first for being able to write recursive functions. So I got help from the internet and tried to write iterative versions of these iterative functions by myself. After writing these functions, I found out how to convert these functions to recursive functions and I converted these functions to recursive functions.

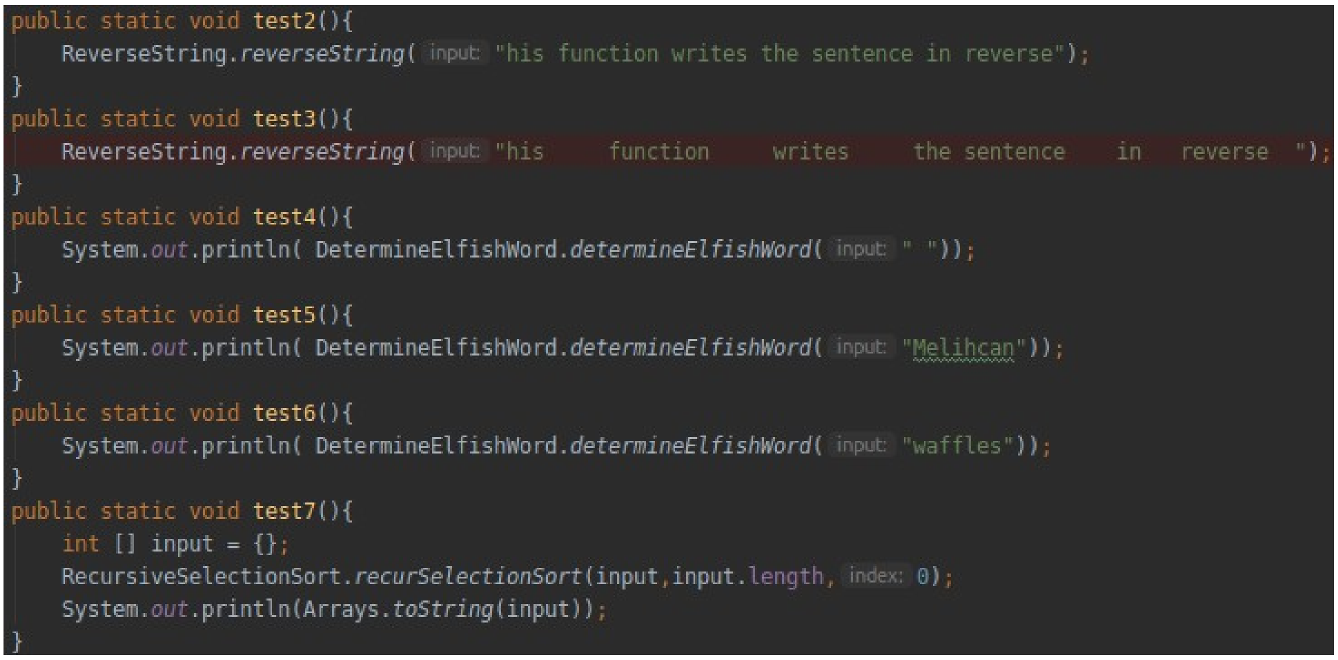
Test 1:



```java
public static void test1(){
    ReverseString.reverseString( input: "");
}
```

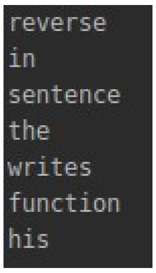
Result 1:

Test 2 – 3 – 4 – 5 – 6 – 7:



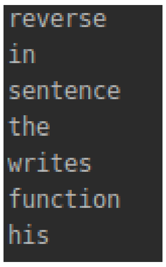
```java
public static void test2(){
    ReverseString.reverseString( input: "his function writes the sentence in reverse");
}
public static void test3(){
    ReverseString.reverseString( input: "his     function     writes     the sentence    in   reverse  ");
}
public static void test4(){
    System.out.println( DetermineElfishWord.determineElfishWord( input: " "));
}
public static void test5(){
    System.out.println( DetermineElfishWord.determineElfishWord( input: "Melihcan"));
}
public static void test6(){
    System.out.println( DetermineElfishWord.determineElfishWord( input: "waffles"));
}
public static void test7(){
    int [] input = {};
    RecursiveSelectionSort.recurSelectionSort(input,input.length, index: 0);
    System.out.println(Arrays.toString(input));
}
```

Result 2:



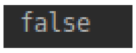
```
reverse
in
sentence
the
writes
function
his
```

Result 3:



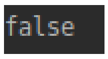
```
reverse
in
sentence
the
writes
function
his
```

Result 4:



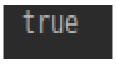
```
false
```

Result 5:



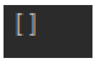
```
false
```

Result 6:



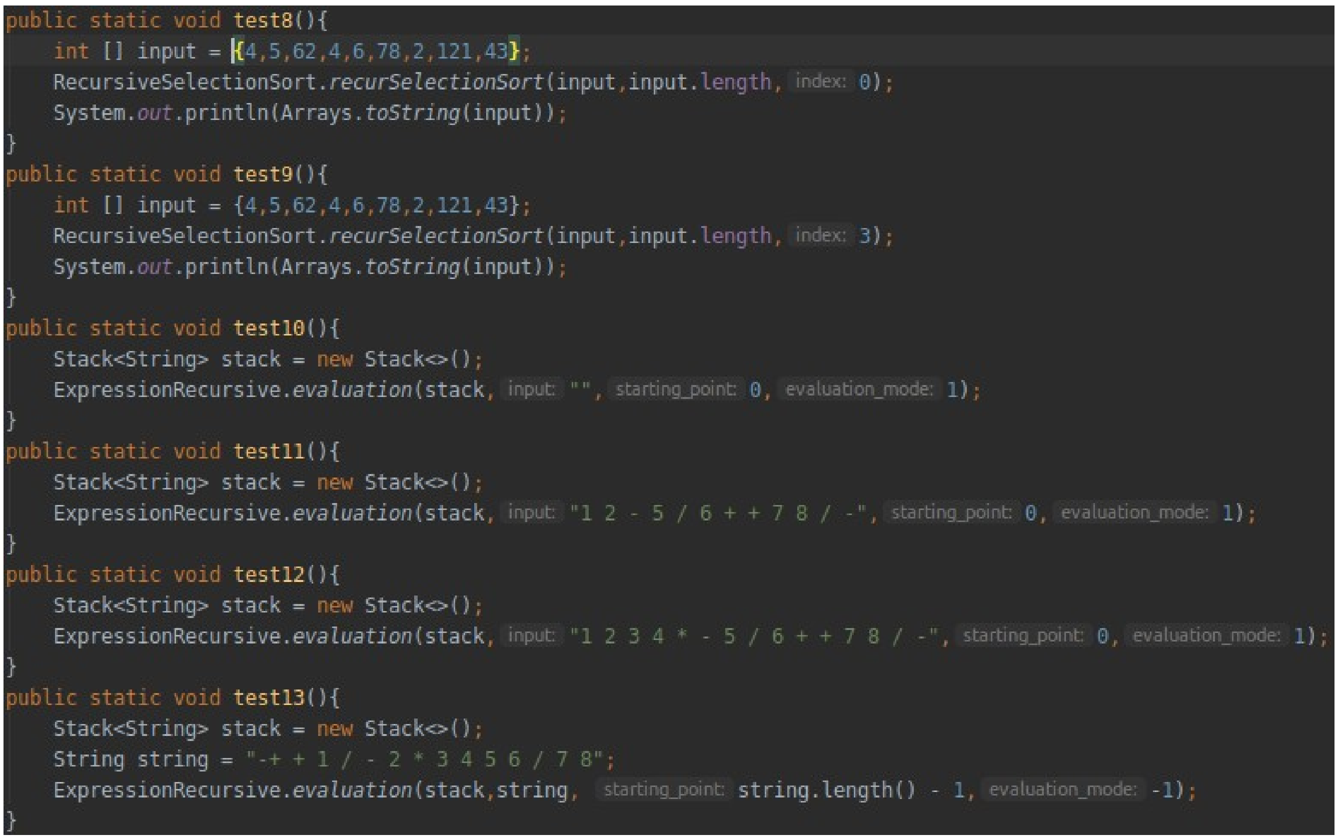
```
true
```

Result 7:



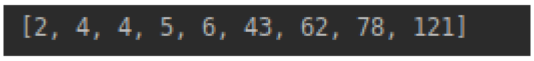
```
[]
```

Test 8 – 9 – 10 – 11 – 12 – 13:



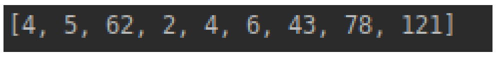
```
public static void test8(){
    int [] input = {4,5,62,4,6,78,2,121,43};
    RecursiveSelectionSort.recurSelectionSort(input,input.length, index: 0);
    System.out.println(Arrays.toString(input));
}
public static void test9(){
    int [] input = {4,5,62,4,6,78,2,121,43};
    RecursiveSelectionSort.recurSelectionSort(input,input.length, index: 3);
    System.out.println(Arrays.toString(input));
}
public static void test10(){
    Stack<String> stack = new Stack<>();
    ExpressionRecursive.evaluation(stack, input: "", starting_point: 0, evaluation_mode: 1);
}
public static void test11(){
    Stack<String> stack = new Stack<>();
    ExpressionRecursive.evaluation(stack, input: "1 2 - 5 / 6 + + 7 8 / -", starting_point: 0, evaluation_mode: 1);
}
public static void test12(){
    Stack<String> stack = new Stack<>();
    ExpressionRecursive.evaluation(stack, input: "1 2 3 4 * - 5 / 6 + + 7 8 / -", starting_point: 0, evaluation_mode: 1);
}
public static void test13(){
    Stack<String> stack = new Stack<>();
    String string = "-+ + 1 / - 2 * 3 4 5 6 / 7 8";
    ExpressionRecursive.evaluation(stack,string, starting_point: string.length() - 1, evaluation_mode: -1);
}
```

Result 8:



```
[2, 4, 4, 5, 6, 43, 62, 78, 121]
```

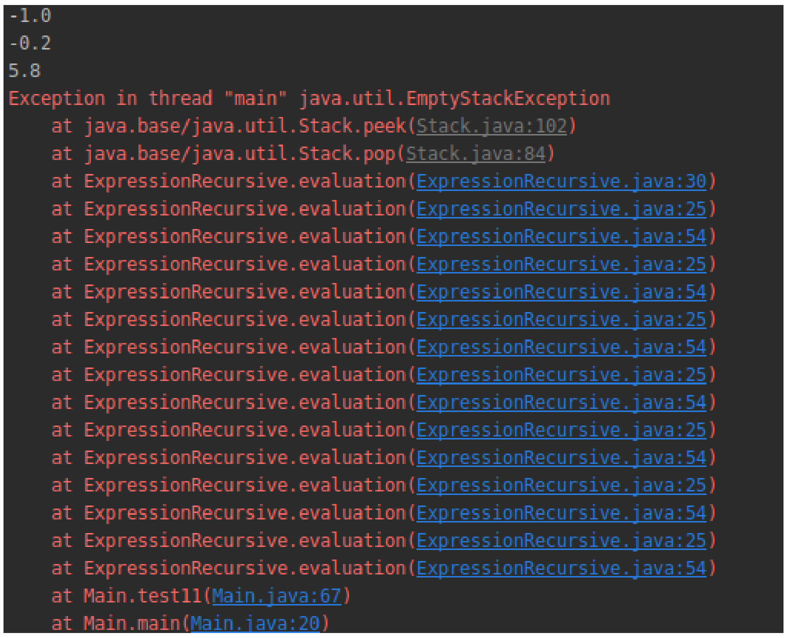
Result 9:



```
[4, 5, 62, 2, 4, 6, 43, 78, 121]
```

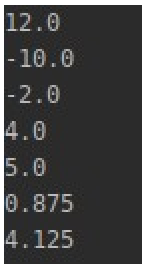
Result 10:

Result 11:



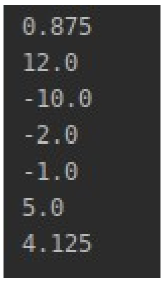
```
-1.0
-0.2
5.8
Exception in thread "main" java.util.EmptyStackException
    at java.base/java.util.Stack.peek(Stack.java:102)
    at java.base/java.util.Stack.pop(Stack.java:84)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:30)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:54)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:54)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:54)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:54)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:54)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:54)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:54)
    at Main.test11(Main.java:67)
    at Main.main(Main.java:20)
```

Result 12:



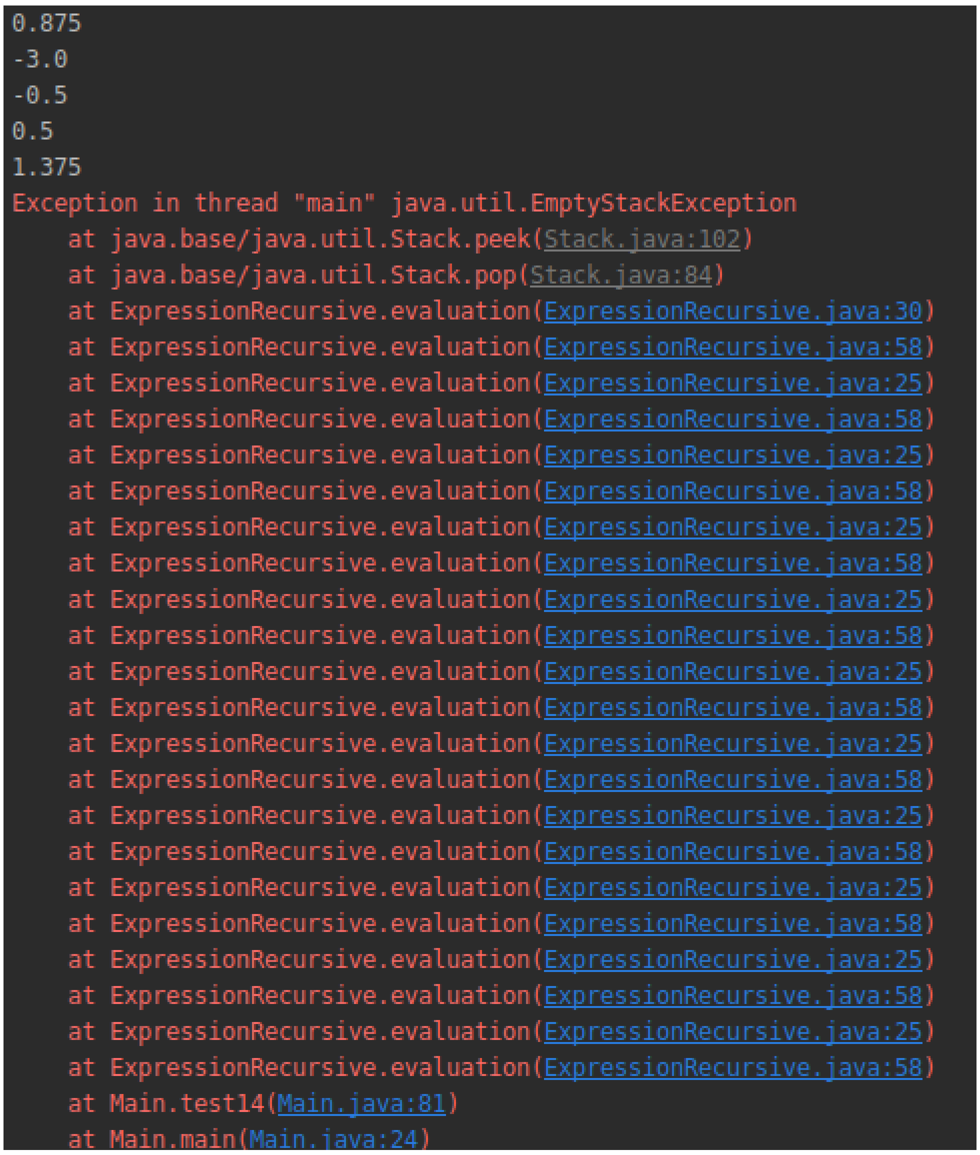
```
12.0
-10.0
-2.0
4.0
5.0
0.875
4.125
```

Result 13:



```
0.875
12.0
-10.0
-2.0
-1.0
5.0
4.125
```

Result 14:



```
0.875
-3.0
-0.5
0.5
1.375
Exception in thread "main" java.util.EmptyStackException
    at java.base/java.util.Stack.peek(Stack.java:102)
    at java.base/java.util.Stack.pop(Stack.java:84)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:30)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:25)
    at ExpressionRecursive.evaluation(ExpressionRecursive.java:58)
    at Main.test14(Main.java:81)
    at Main.main(Main.java:24)
```