

ADS LAB 1 REPORT

Elsayed, Ahmed
TEAM 15 MEMBER B

PREPARATION

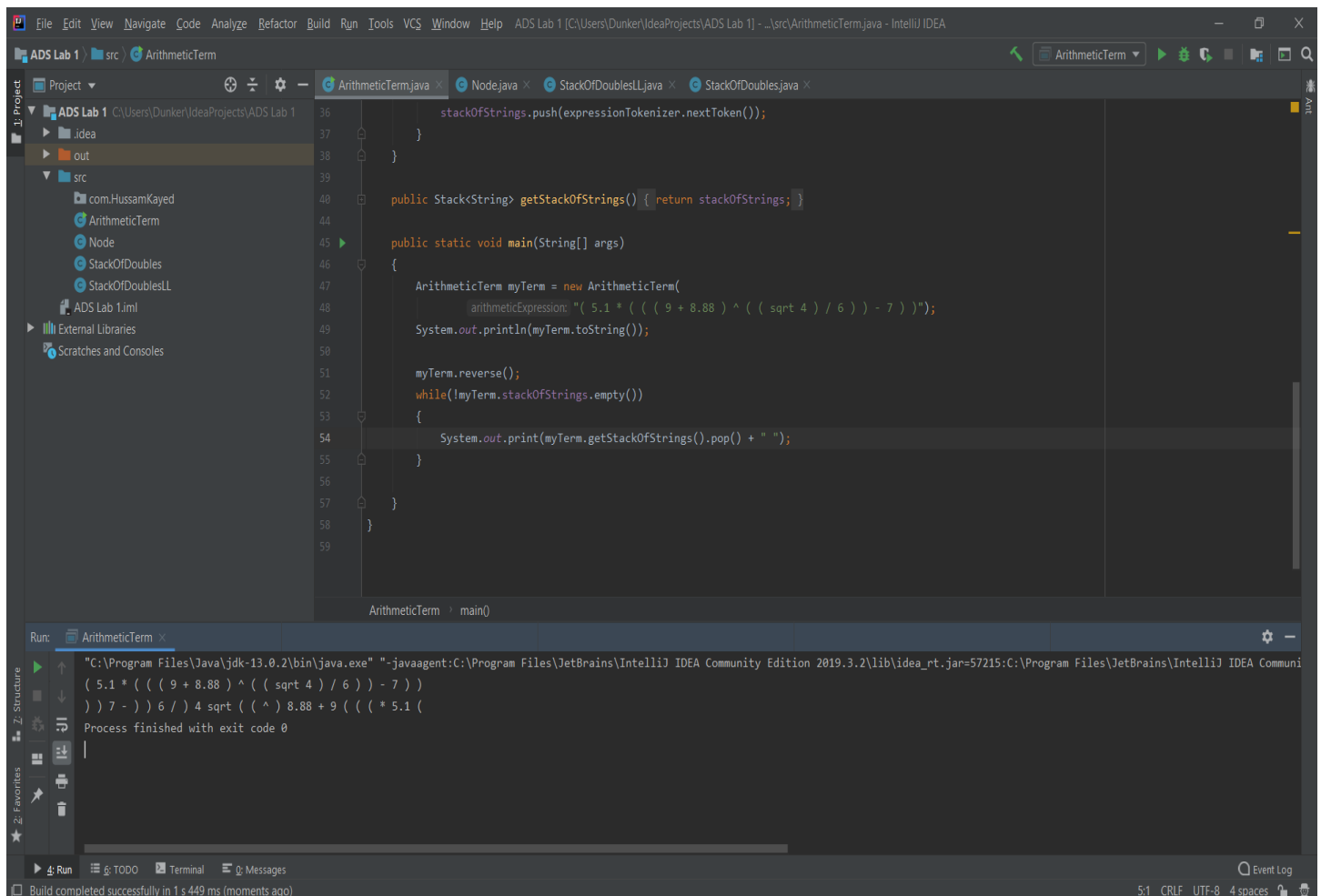
Problem 1:

For both the arithmetic expressions given in die introductions to problems 2 and 3 determine the maximum stack size needed, i.e. the size of the corresponding stack in that problem at it's maximum usage. For example, the maximum stack size is two for evaluating $5 \ 4 \ +$ and it is one for converting $(5 \ + \ 4)$ into the former postfix term. **Maximum Stack Size is 3**

Algorithm Run	Stack
1
2
3	"*"
4	"*"
5	"*"
6	"*"
7	"*"
8	"*", "+"
9	"*", "+"
10	"*"
11	"*", "^"
12	"*", "^"
13	"*", "^"
14	"*", "^", "sqrt"
15	"*", "^", "sqrt"
16	"*", "^"
17	"*", "^", "/"
18	"*", "^"
19	"*"
20	"*", "-"
21	"*", "-"
22	"*"
23

2. Arithmetic Term, Array and Linked List stack Implementation

The following screenshot shows a part of the ArithmeticTerm class implementation, for more information on this class, StackOfDoubles class as an array and Linked list implementation for the StackOfDoubles, please check out the source code in the same ZIP file as this report.



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help ADS Lab 1 [C:\Users\Dunker\IdeaProjects\ADS Lab 1] - IntelliJ IDEA
ADS Lab 1 > src > ArithmeticTerm
ArithmeticTerm.java Node.java StackOfDoublesLL.java StackOfDoubles.java
36 stackOfStrings.push(expressionTokenizer.nextToken());
37 }
38 }
39
40 public Stack<String> getStackOfStrings() { return stackOfStrings; }
41
42 public static void main(String[] args)
43 {
44     ArithmeticTerm myTerm = new ArithmeticTerm(
45         arithmeticExpression: "( 5.1 * ( ( 9 + 8.88 ) ^ ( ( sqrt 4 ) / 6 ) ) - 7 ) )";
46     System.out.println(myTerm.toString());
47
48     myTerm.reverse();
49     while(!myTerm.stackOfStrings.empty())
50     {
51         System.out.print(myTerm.getStackOfStrings().pop() + " ");
52     }
53 }
54
55 }
56
57 }
58
59
ArithmeticTerm -> main()
Run: ArithmeticTerm
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.3.2\lib\idea_rt.jar=57215:C:\Program Files\JetBrains\IntelliJ IDEA Communi
( 5.1 * ( ( ( 9 + 8.88 ) ^ ( ( sqrt 4 ) / 6 ) ) - 7 ) )
) ) 7 - ) 6 / ) 4 sqrt ( ( ^ ) 8.88 + 9 ( ( ( * 5.1 (
Process finished with exit code 0
Build completed successfully in 1 s 449 ms (moments ago)
S:1 CRLF UTF-8 4 spaces
```

3. Resizing Strategies and Number of Copy Operations (UPDATED)

Starting size $n = 8$, growing to $N = 32$.

First strategy: increasing by a doubling:

The strategy is to somehow obtain the number of copies in a recursive way of its calculation which means $\text{nextSize} = \text{currentSize} * 2$ (N)

In order to get such operation to happen, the number of copy operations will be doubled each time we increase the size, which means in our case the difference between the currentSize

and the targetSize + the number of previous copies is the number of copy operations so we have such an equation $C(N,n) = N-n$, which gives us the number of copy operations for increasing N and constant n. By substituting $N = n \cdot 2^k$, we get: $C(k) = n(2^k - 1)$

Second strategy: increasing by constant number (in this case $n = 8$): in this case, since we are increasing the currentSize by a constant number, the previous copying operations are taken into consideration and added to the other copying operations, so we get:

$currentSize = PreviousSize + constantSizeNumber;$

Question now is how can we describe the number of copying operations happening in the process? Well basically we have the previous number of copying operations + the constant number of operations we add each time to increase the size, so we get:

$C1(k) = Co(k) + n$; where n is the constant number in which case, we have $n = 8$.

Now last question is, how can we describe the previous number of copy operations? This can be done through thinking about the previous items in terms of N (wanted size) and n (constant size increment). Since we are increasing with a constant number, we can describe the relationship in terms of $1+2+3+4+..... +N = N^2/2$. But the thing is, we don't increase the size with 1, we increase it with 8, so our case is $8+16+32+48+56+64+..... +N(n) = N(N-n)/2n$

Therefore after simplification: $C1(N,n) = N(N-n)/2n$.

LAB

Problem 3:

1. String convert() was implemented as per what is required, please check the source code for more information on the method implementation
2. The following 5 test cases were tested, and all were successful
 - $((((6 + 4)^2) - 5) / 2)$
 - $((3 * 8) + ((\sqrt{9}) / 3))$
 - $((4 * (\sqrt{4})) - 1) + (\sqrt{2})$
 - $(((((3 - 2) * 4) / 2) * (\sqrt{4}))^2)$
 - $((((((5^4)^2)^4) + 9) - 8) / 2)$
 - $(((((\sqrt{4}) * (\sqrt{9})) - 6) * 6) + 4)$

All cases were successful in conversion of the FPAE from PostFix to InFix

3. Please find the documentation in the same ZIP file as the source code as well as a separate class called Main which has the test of one of the test cases