

IT 314 – Software Engineering

Lab 7

Akshat Jindal | 202201299

Tower Of Hanoi Code

Program Inspection

Question 1. How many errors are there in the program? Mention the errors you have identified.

The only issue with this code is that the Java program doesn't accept the values such as `inter--`, `from+1` or `to+1`. On correcting those issues and on correcting the logic of replacing `topN++` with `topN - 1`, the code will work flawlessly.

Question 2. Which category of program inspection would you find more effective?

For this question, the Category E of the program inspection, the Control Flow errors were the most effective, as the failure of the code was due to the use of the recursion calls.

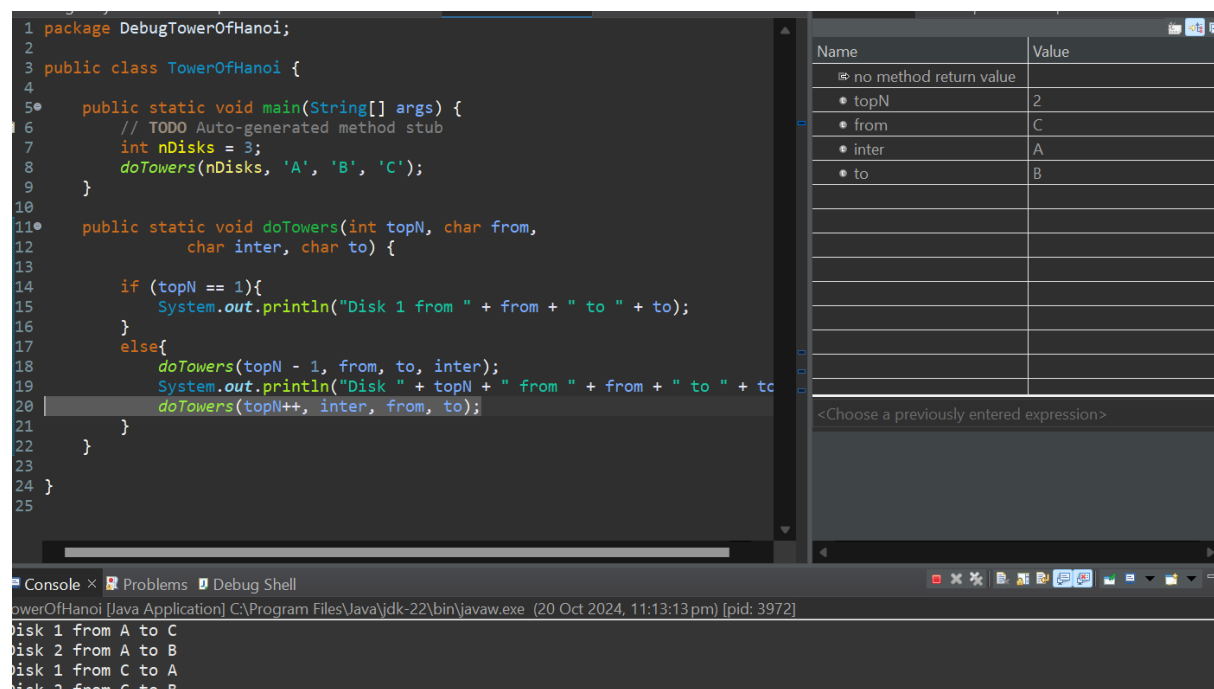
Question 3. Which type of error you are not able to identified using the program inspection?

In this example, all the errors in the document were identifiable using the program inspection method.

Question 4. Is the program inspection technique is worth applicable?

Due to the problem lying in recursion, it was a bit difficult to figure out the error with program inspection.

Code Debugging



The screenshot shows an IDE with a Java program for the Tower of Hanoi. The code is as follows:

```
1 package DebugTowerOfHanoi;
2
3 public class TowerOfHanoi {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int nDisks = 3;
8         doTowers(nDisks, 'A', 'B', 'C');
9     }
10
11     public static void doTowers(int topN, char from,
12                                 char inter, char to) {
13
14         if (topN == 1){
15             System.out.println("Disk 1 from " + from + " to " + to);
16         }
17         else{
18             doTowers(topN - 1, from, to, inter);
19             System.out.println("Disk " + topN + " from " + from + " to " + to);
20             doTowers(topN++, inter, from, to);
21         }
22     }
23 }
24 }
25 }
```

The debugger window on the right shows the following variables:

Name	Value
no method return value	
topN	2
from	C
inter	A
to	B

The console output at the bottom shows the following sequence of moves:

```
Disk 1 from A to C
Disk 2 from A to B
Disk 1 from C to A
Disk 2 from C to B
```

Question 1. How many errors are there in the program? Mention the errors you have identified.

After removing the syntax errors, there was only one error in the code. The value for topN in the second recursion call was increasing, whereas it should decrease and the recursion would take care of the code. Hence, with that one change, the code works perfectly.

Question 2. How many breakpoints you need to fix those errors? What are the steps you have taken to fix the error you identified in the code fragment?

With two break point on the 19th and the 20th line, we were able to figure out that the output of the code was not coming to be correct after the 2nd iteration. That's when it was possible to realize that the error is in the second recursion call, mainly due to the topN value being incremented.

Question 3. Submit your complete executable code.

```
1 package DebugTowerOfHanoi;
2
3 public class TowerOfHanoi {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int nDisks = 3;
8         doTowers(nDisks, 'A', 'B', 'C');
9     }
10
11     public static void doTowers(int topN, char from,
12                                char inter, char to) {
13
14         if (topN == 1){
15             System.out.println("Disk 1 from " + from + " to " + to);
16         }
17         else{
18             doTowers(topN - 1, from, to, inter);
19             System.out.println("Disk " + topN + " from " + from + " to " + to);
20             doTowers(topN - 1, inter, from, to);
21         }
22     }
23
24 }
```

Console × Problems Debug Shell
<terminated> TowerOfHanoi [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (20 Oct 2024, 11:17:33 pm – 11:17:33 pm) [pid: 12124]
Disk 1 from A to C
Disk 2 from A to B
Disk 1 from C to B
Disk 3 from A to C
Disk 1 from B to A
Disk 2 from B to C
Disk 1 from A to C