SCREEN CAPTURE ONE

SHOW VARIABLES WINDOW

A computer screen shot of a program

Description automatically generated

TABLE ONE

|  |
| --- |
| Question: What is the Eclipse keyboard shortcut for toggling a breakpoint? |
| Answer: Ctrl-Shift-B |
| Question: What is the difference between “Step-Over”, and “Step-Into”, and “Step-Return”? |
| Answer:  Step-Over: Execute the next line of code without examining its details.  Step-Into: Go into the details of each line of codes, for example, examining a variable value, or all the statements of a method.  Step-Return: Step out of the current function and return to the current line of code. |
| Task: Practice tracing through the DebugStar sample program. It is ok if you don’t understand all of the java code; but you should be able to trace the order in which statements are executed. Based on your best understanding of the program, provide a list of methods that are called when the program executes (from start to end, in order of being called). You can skip library methods (like println, for example). HINT: Use a combination of “Step-Into” “Step-Over” and “Step-Return”. Use the “Stack Trace” window |
| Step-Into:    DebugStar.run(String, int, int)  DebugStar.getOperation(String)  String.equals(Object)  DebugStar.getOperation(String)  Add<init>()  Add(Onject)<init>()  Add<init>()  DebugStar.getOperation(String)  DebugStar.run(String, int, int)  Add.perform(int, int)  DebugStar.run(String, int, int)  Step-Return  Step-Over  Step-Into:    DebugStar.run(String, int, int)  DebugStar.getOperation(String))  DebugStar.getOperation(String)  Subtract<init>()  Subtract(Object)<init>()  Subtract<init<()  DebugStar.getOperation(String)  DebugStar.run(String, int, int)  Subtract.perform(int, int)  DebugStar.run(String, int, int)  Step-Return  Step-Over  Step-Intro    DebugStar.run(int)  Factorial<init>()  Factorial(Object)<init>()  Factorial<init>()  DebugStar.run(int)  Factorial.perform(int)  DebugStar.run(int) |

The Debug Challenge

1. SCREEN CAPTURE: Original Code with line numbers  
   Screen Capture of Original Code

|  |
| --- |
| Screen Capture of Original Code |

1. TABLE TWO

|  |  |  |  |
| --- | --- | --- | --- |
| Line Number | Type of error (compile-time, run-time, or logical) | Description | Correction |
| 26 | Syntax error | Missing “ ;” | Add “;” |
| 27, 31 | Compile-time error | The local variable n has not been initialized | Initialize n = 0; |
| 27 | Run-time error and logical error | “Index 2 out of bounds for length 2”  The condition is wrong | (n <= 2) |
| 47 | Logical error | Iteration starts at index 2 which is not desireable | i = 0; |
| 48 | Run-time error | Index out of bound due to condition (i <= fiboList.size()) | (i < fiboList.size()) |
| 40 | Logical | The result of adding the same element in the array list is a desired value in the Fibonacci numbers | f.add(f.get(i-1) + f.get(i - 2)); |
| 25 | Resource leak | Scanner is not closed | Add scanner.close(); below printList(getFiboList(n)); |

3. SCREEEN CAPTURE: Fixed code with line numbers

|  |
| --- |
|  |