IT-314: Software Engineering

Lab Assignment: 07

Title: Program Inspection, Debugging and Static Analysis

Lab Group: G2

Name: Parth Vadodaria

ID: 202201174

Q1. Program Inspection: (Submit the answers of following questions for each code fragment)

Ans.

Code Fragment 1 (Client Code):

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

- Total errors identified: 4
    1. Unnecessary boolean initialization (receive=true; in the constructor without a clear need for early initialization).
    2. Variable scope inconsistency (parameter Socket s in sendFile shares the same name as the Socket s in the main() method, which may cause confusion).
    3. Comparison of characters in flag == 'n' may lead to unexpected behavior.
    4. Potential infinite loop in listenClient() without termination condition (no exit condition for the thread).

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

- Category E: Control-Flow Errors: This category helped identify the potential infinite loop in the listenClient() method, which could lead to serious performance issues and unresponsive threads.
- Category G: Input/Output Errors: This category flagged the potential file overwrite issue, which could cause data loss.

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

- Thread safety issues were not fully explored using this inspection method. For example, if multiple threads are reading/writing data simultaneously (especially for network I/O), potential race conditions or deadlocks might occur, but these are hard to catch with basic inspection.

4. Is the program inspection technique worth applying?

- Yes, program inspection helps catch several issues early on, including potential infinite loops and I/O errors, which can severely impact the program. It's particularly effective in uncovering control-flow problems and data handling issues.

However, runtime behavior and concurrency problems would require dynamic testing or deeper analysis.

Code Fragment 2 (Server Code):

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

- Total errors identified: 7
    1. Unnecessary self-assignment (currentClientHandler = this; is redundant).
    2. Scope of socket (Socket s) is unclear due to repeated declarations in different contexts.
    3. Potential infinite loop in ClientHandler.run() without proper exit conditions.
    4. Inconsistent exception handling (exceptions are caught but not meaningfully handled).
    5. Lack of synchronization for clientHandlers access, which could lead to race conditions.

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

- Category F: Interface Errors: This category helped catch synchronization issues with clientHandlers and inconsistent exception handling, both of which could severely affect system behavior.
- Category E: Control-Flow Errors: This category was useful in identifying the potential infinite loop in ClientHandler.run() and ensuring the thread has a proper exit mechanism.

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

- File transmission errors and concurrency issues related to network I/O operations are difficult to spot with static inspection. Dynamic testing would be needed to simulate real-world conditions like high traffic, multiple simultaneous connections, and error conditions like dropped connections.

4. Is the program inspection technique worth applying?

Yes, the inspection technique proved valuable in catching structural and control-flow issues, especially regarding client handling, file transmission, and thread management. However, it may not be sufficient for identifying deep concurrency problems or issues that arise only under specific runtime conditions, which would require more thorough testing and profiling.

Project link for this question: https://github.com/Parth3105/File-Share-Stream

Q2. CODE DEBUGGING: Debugging is the process of localizing, analyzing, and removing suspected errors in the code (Java code given in the .zip file)

Ans.

I. Armstong

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

- Error 1: Incorrect operator for extracting the remainder.
  - Original code: remainder = num / 10;
  - Corrected code: remainder = n % 10;
  - Reason: The / operator is used for division, but % is needed to extract the remainder.
- Error 2: Incorrect operator for removing the last digit from num.
  - Original code: num = num % 10;
  - Corrected code: n = n / 10;
  - Reason: The / operator is used to divide by 10 and remove the last digit. The % operator was mistakenly used here.

2. How many breakpoints do you need to fix those errors?

- 2 breakpoints are needed to fix the two errors.

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Fixing remainder calculation:
   - Change remainder = num / 10; to remainder = n % 10;

    - This ensures the remainder is calculated correctly to get the last digit of n.
2. Fixing digit removal:
    - Change num = num % 10; to n = n /10;
    - This ensures the code correctly removes the last digit by dividing n by 10.

II. GCD & LCM

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

- Error 1: Incorrect assignment for a and b in the gcd() function.
    - Original code: a = (x > y) ? y : x;, b = (x < y) ? x : y;
    - Corrected code: a = (x > y) ? x : y;, b = (x < y) ? x : y;
    - Reason: a should hold the larger number and b the smaller one.
- Error 2: Incorrect while loop condition in the gcd() function.
    - Original code: while(a % b == 0)
    - Corrected code: while(a % b != 0)
    - Reason: The loop should continue until the remainder is 0, meaning a % b != 0 is the correct condition.
- Error 3: Incorrect condition in the lcm() function.
    - Original code: if(a % x != 0 && a % y != 0)
    - Corrected code: if(a % x == 0 && a % y == 0)
    - Reason: The condition should check when a is divisible by both x and y to determine the least common multiple (LCM).

2. How many breakpoints do you need to fix those errors?

- 5 breakpoints are needed to fix the errors.

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Fixing gcd() function:
    - Change the assignment of a and b to ensure a is the larger number and b is the smaller number.

- Change the while condition to while(a % b != 0) so the loop continues until the remainder becomes 0.
- Ensure the remainder calculation r = a % b is correct.

2. Fixing lcm() function:
   - Change the condition in the if statement to if(a % x == 0 && a % y == 0) to ensure that a is divisible by both x and y.
   - Ensure the loop continues until the least common multiple is found.

III. Knapsack

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

- Error 1: Incorrect index used in the option1 calculation.
  - Original code: int option1 = opt[n++][w];
  - Corrected code: int option1 = opt[n-1][w];
  - Reason: n++ was incrementing n unnecessarily. The correct index is n-1 because we are considering the previous item when deciding whether to include the current item or not.
- Error 2: Incorrect condition in the if statement for option2.
  - Original code: if (weight[n] > w)
  - Corrected code: if (weight[n] <= w)
  - Reason: The condition should check whether the current item's weight is less than or equal to the remaining capacity w before considering its inclusion.
- Error 3: Incorrect indices for calculating option2 in the opt and profit arrays.
  - Original code: option2 = profit[n-2] + opt[n-1][w-weight[n]]
  - Corrected code: option2 = profit[n] + opt[n-1][w-weight[n]]

    - Reason: The original code referenced an incorrect index (n-2) for profit, which would incorrectly compute the profit value of a different item.
- Error 4: Missing condition in selecting the better option (opt[n][w]).
    - The original code did not have an explicit error here, but the logic in selecting opt[n][w] = Math.max(option1, option2) needs to be made clearer.

2. How many breakpoints do you need to fix those errors?

- 4 breakpoints are needed to fix the errors.

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Fixing option1:
    - Changed int option1 = opt[n++][w]; to int option1 = opt[n-1][w]; to correctly use the previous item in the calculation.
2. Fixing option2 condition:
    - Updated the if statement from if (weight[n] > w) to if (weight[n] <= w) so that the item can only be considered if its weight is less than or equal to the current capacity.
3. Fixing option2 calculation:
    - Changed option2 = profit[n-2] + opt[n-1][w-weight[n]] to option2 = profit[n] + opt[n-1][w-weight[n]] so that the current item's profit is correctly added when calculating the value of taking the item.

IV. <u>Magic number</u>

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

- Error 1: Incorrect condition in the inner while loop.
    - Original code: while(sum==0)
    - Corrected code: while(sum!=0)
    - Reason: The loop should run as long as sum is not 0 to break the number into digits. The original condition

sum==0 would cause the loop to terminate immediately, skipping the digit extraction.

- Error 2: Incorrect operation in calculating the sum of digits.
  - Original code: s=s*(sum/10)
  - Corrected code: s=s+(sum%10)
  - Reason: The operation should add the last digit of sum (calculated using %) to s. The original multiplication and division were incorrect operations.
- Error 3: Incorrect update to sum after extracting a digit.
  - Original code: sum=sum%10
  - Corrected code: sum=sum/10
  - Reason: The number should be reduced by dividing by 10 to remove the last digit after processing it. The original modulus operation did not update sum correctly.

2. How many breakpoints do you need to fix those errors?

- 4 breakpoints are needed to fix the errors.

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Fixing the inner while loop condition:
   - Changed while(sum==0) to while(sum!=0) so that the loop continues until all digits of sum are processed.
2. Fixing the sum of digits calculation:
   - Updated s=s*(sum/10) to s=s+(sum%10) to correctly accumulate the digits of sum by extracting the last digit using modulus.
3. Fixing the update to sum:
   - Replaced sum=sum%10 with sum=sum/10 to remove the last digit after it has been processed.

V. Merge sort

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

- Error 1: In the mergeSort function, the left and right arrays were incorrectly created by using arithmetic on the array itself:
    - Original Code: int[] left = leftHalf(array+1);
    - Corrected Code: int[] left = leftHalf(array);
    - Reason: In Java, you cannot perform arithmetic operations directly on an array variable like array+1. Java doesn't use pointers in the way languages like C do. Hence, the array should be passed as-is, and the method leftHalf will handle the splitting.
- Error 2: Similar issue with the creation of the right array.
    - Original Code: int[] right = rightHalf(array-1);
    - Corrected Code: int[] right = rightHalf(array);
    - Reason: As with the left array, we cannot modify the array reference directly. The method rightHalf will split the array correctly without requiring any manual modification.

2. How many breakpoints do you need to fix those errors?

- 0 breakpoints are needed to fix the errors.

VI. Multiply matrix

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

- Error 1: Incorrect indexing for accessing elements during matrix multiplication.
    - Original Code: sum = sum + first[c-1][c-k]*second[k-1][k-d];
    - Corrected Code: sum = sum + first[c][k]*second[k][d];
    - Reason: The original code incorrectly accessed elements with decremented indices, which would lead to an ArrayIndexOutOfBoundsException and incorrect calculations. The correct indices should directly use c, k, and d.

2. How many breakpoints do you need to fix those errors?

- 1 breakpoint is needed to fix the identified error

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Fixed Indexing in Matrix Multiplication:
    - Changed the line from sum = sum + first[c-1][c-k]*second[k-1][k-d]; to sum = sum + first[c][k]*second[k][d]; to correctly access the elements of the first and second matrices based on the current indices.

VII. Quadratic Probing

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

- No Error: Probably there is no error. The problem is the way of input into program which can be can be encountered in two ways:
    - Make a switch case for display and after giving input display the output. Accordingly, the display code should also be changed.
    - Create a loop for insert such that the user can input the key-value till he requires and then all are passed to the insert function in an array. Accordingly, insert code should be changed.

VIII. Sorting array

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

- Error 1: Incorrect loop condition for the outer loop.
    - Original Code: for (int i = 0; i >= n; i++);
    - Corrected Code: for (int i = 0; i < n; i++)
    - Reason: The original loop condition (i >= n) is incorrect and will not execute the loop. It should be i < n to iterate over all elements of the array.
- Error 2: Incorrect comparison operator in the sorting condition.

- Original Code: if (a[i] <= a[j])
- Corrected Code: if (a[i] > a[j])
- Reason: To sort the array in ascending order, the condition should check if a[i] is greater than a[j] to swap them. The original condition (<=) would lead to incorrect ordering.

2. How many breakpoints do you need to fix those errors?

- 2 breakpoints are needed to fix the identified errors

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Fixed Outer Loop Condition:
   - Changed for (int i = 0; i >= n; i++); to for (int i = 0; i < n; i++) to ensure the loop iterates through all array elements.
2. Corrected Sorting Logic:
   - Updated the comparison condition from if (a[i] <= a[j]) to if (a[i] > a[j]) so that elements are swapped correctly for ascending order.

IX. Stack Implementation

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

- Error 1: Incorrect increment of top in the push method.
  - Original Code: top--;
  - Corrected Code: top++;
  - Reason: The top variable should be incremented to point to the next available position in the stack after adding a value. The original code incorrectly decremented top, which would lead to an index out of bounds error.
- Error 2: Incorrect decrement of top in the pop method.
  - Original Code: top++;
  - Corrected Code: top--;
  - Reason: When popping an element from the stack, the top index should be decremented. The original code

incorrectly incremented top, resulting in an incorrect pop operation.

- Error 3: Incorrect loop condition in the display method.
  - Original Code: for(int i=0;i>top;i++)
  - Corrected Code: for(int i=0;i<=top;i++)
  - Reason: The loop should iterate from 0 to top, inclusive, to display all elements currently in the stack. The original condition (i > top) would never execute the loop.

2. How many breakpoints do you need to fix those errors?

- 3 breakpoints are needed to fix the identified errors

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Fixed Increment in push Method:
   - Changed top--; to top++; to ensure the top index points to the next free position for the value being pushed.
2. Corrected Decrement in pop Method:
   - Updated top++; to top--; to correctly decrement the top index when popping an element from the stack.
3. Updated Loop Condition in display Method:
   - Changed the loop condition from i > top to i <= top so that all elements in the stack are printed.

X. <u>Tower of Hanoi</u>

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

- Error 1: Incorrect modification of topN and variable parameters in the recursive call.
  - Original Code: doTowers(topN ++, inter--, from+1, to+1)
  - Corrected Code: doTowers(topN - 1, inter, from, to)
  - Reason: The original code incorrectly attempted to increment and decrement the parameters using ++ and --, which can lead to undefined behavior. Instead, the recursive call should simply decrement topN to reflect the number of disks being processed.

2. How many breakpoints do you need to fix those errors?

- 2 breakpoints are needed to fix the identified errors

3. What are the steps you have taken to fix the error you identified in the code fragment?

1. Corrected the Recursive Calls:
    - Updated the second recursive call from doTowers(topN ++, inter--, from+1, to+1) to doTowers(topN - 1, inter, from, to) to ensure that we correctly manage the disk count without modifying the parameters improperly.

Q3. Static Analysis Tools (PMD Tool and Java code is used)

PMD report

Problems found

#	File	Line	Problem
1	D:\File Transfer Socket\src\Client.java	5	All classes, interfaces, enums and annotations must belong to a named package
2	D:\File Transfer Socket\src\Client.java	5	Class comments are required
3	D:\File Transfer Socket\src\Client.java	6	Field comments are required
4	D:\File Transfer Socket\src\Client.java	6	To avoid mistakes add a comment at the beginning of the receive field if you want a default access modifier
5	D:\File Transfer Socket\src\Client.java	6	Use explicit scoping instead of the default package private level
6	D:\File Transfer Socket\src\Client.java	7	Field comments are required
7	D:\File Transfer Socket\src\Client.java	7	To avoid mistakes add a comment at the beginning of the writeData field if you want a default access modifier
8	D:\File Transfer Socket\src\Client.java	7	Use explicit scoping instead of the default package private level
9	D:\File Transfer Socket\src\Client.java	8	Field comments are required
10	D:\File Transfer Socket\src\Client.java	8	To avoid mistakes add a comment at the beginning of the readData field if you want a default access modifier
11	D:\File Transfer Socket\src\Client.java	8	Use explicit scoping instead of the default package private level
12	D:\File Transfer Socket\src\Client.java	10	A method/constructor should not explicitly throw java.lang.Exception
13	D:\File Transfer Socket\src\Client.java	10	Avoid variables with short names like s
14	D:\File Transfer Socket\src\Client.java	10	Parameter 's' is not assigned and could be declared final
15	D:\File Transfer Socket\src\Client.java	10	To avoid mistakes add a comment at the beginning of the Client constructor if you want a default access modifier
16	D:\File Transfer Socket\src\Client.java	15	The method 'listenClient()' has a cognitive complexity of 23, current threshold is 15
17	D:\File Transfer Socket\src\Client.java	15	To avoid mistakes add a comment at the beginning of the listenClient method if you want a default access modifier
18	D:\File Transfer Socket\src\Client.java	15	Use explicit scoping instead of the default package private level
19	D:\File Transfer Socket\src\Client.java	16	To be compliant to J2EE, a webapp should not use any thread.
20	D:\File Transfer Socket\src\Client.java	28	Avoid using Literals in Conditional Statements
21	D:\File Transfer Socket\src\Client.java	29	Avoid instantiating new objects inside loops
22	D:\File Transfer Socket\src\Client.java	29	Local variable 'filePath' could be declared final
23	D:\File Transfer Socket\src\Client.java	30	Avoid instantiating new objects inside loops
24	D:\File Transfer Socket\src\Client.java	30	Local variable 'fileName' could be declared final
25	D:\File Transfer Socket\src\Client.java	32	Local variable 'pathSize' could be declared final

26	D:\File Transfer Socket\src\Client.java	37	Local variable 'nameSize' could be declared final
27	D:\File Transfer Socket\src\Client.java	44	Avoid catching generic exceptions such as NullPointerException, RuntimeException, Exception in try-catch block
28	D:\File Transfer Socket\src\Client.java	45	System.out.println is used
29	D:\File Transfer Socket\src\Client.java	52	Avoid variables with short names like s
30	D:\File Transfer Socket\src\Client.java	52	Parameter 'fileLocation' is not assigned and could be declared final
31	D:\File Transfer Socket\src\Client.java	52	Parameter 's' is not assigned and could be declared final
32	D:\File Transfer Socket\src\Client.java	52	To avoid mistakes add a comment at the beginning of the sendFile method if you want a default access modifier
33	D:\File Transfer Socket\src\Client.java	52	Use explicit scoping instead of the default package private level
34	D:\File Transfer Socket\src\Client.java	54	Local variable 'file' could be declared final
35	D:\File Transfer Socket\src\Client.java	55	Avoid instantiating FileInputStream, FileOutputStream, FileReader, or FileWriter
36	D:\File Transfer Socket\src\Client.java	55	Ensure that resources like this FileInputStream object are closed after use
37	D:\File Transfer Socket\src\Client.java	55	Local variable 'fileRead' could be declared final
38	D:\File Transfer Socket\src\Client.java	61	Local variable 'locationParts' could be declared final
39	D:\File Transfer Socket\src\Client.java	62	Local variable 'fileName' could be declared final
40	D:\File Transfer Socket\src\Client.java	63	Potential violation of Law of Demeter (object not created locally)
41	D:\File Transfer Socket\src\Client.java	69	Local variable 'buffer' could be declared final
42	D:\File Transfer Socket\src\Client.java	70	Found 'DD'-anomaly for variable 'bytes' (lines '70'-'71').
43	D:\File Transfer Socket\src\Client.java	70	The initializer for variable 'bytes' is never used (overwritten on line 71)
44	D:\File Transfer Socket\src\Client.java	71	Avoid assignments in operands
45	D:\File Transfer Socket\src\Client.java	71	Found 'DU'-anomaly for variable 'bytes' (lines '71'-'80').
46	D:\File Transfer Socket\src\Client.java	77	Avoid catching generic exceptions such as NullPointerException, RuntimeException, Exception in try-catch block
47	D:\File Transfer Socket\src\Client.java	78	Avoid printStackTrace(); use a logger call instead.
48	D:\File Transfer Socket\src\Client.java	82	Parameter 'fileName' is not assigned and could be declared final
49	D:\File Transfer Socket\src\Client.java	82	Parameter 'filePath' is not assigned and could be declared final
50	D:\File Transfer Socket\src\Client.java	82	To avoid mistakes add a comment at the beginning of the receiveContents method if you want a default access modifier
51	D:\File Transfer Socket\src\Client.java	82	Use explicit scoping instead of the default package private level
52	D:\File Transfer Socket\src\Client.java	84	Local variable 'folder' could be declared final
53	D:\File Transfer Socket\src\Client.java	88	Avoid instantiating FileInputStream, FileOutputStream, FileReader, or FileWriter
54	D:\File Transfer Socket\src\Client.java	88	Ensure that resources like this FileOutputStream object are closed after use

Also html and pdf format provided separately.

Note: I tried some of the Github codes but PMD was not giving any errors. At last, I pick up one of my project code for static analysis.

Project link: https://github.com/Parth3105/File-Share-Stream