Name : Siddhant Kotak

Student ID : 202201410

Course : IT 314 Software Engineering

Professor : Saurabh Tiwari

Semester : Autumn 2024

Lab : Program Inspection, Debugging and Static Analysis

GCD and LCM of two Number

Given code :

```
//program to calculate the GCD and LCM of two given numbers
import java.util.Scanner;

public class GCD_LCM
{
  static int gcd(int x, int y)
  {
    int r=0, a, b;
    a = (x > y) ? y : x; // a is greater number
    b = (x < y) ? x : y; // b is smaller number

    r = b;
    while(a % b == 0) //Error replace it with while(a % b != 0)
    {
      r = a % b;
      a = b;
      b = r;
    }
    return r;
  }

  static int lcm(int x, int y)
  {
    int a;
    a = (x > y) ? x : y; // a is greater number
    while(true)
    {
      if(a % x != 0 && a % y != 0)
          return a;
      ++a;
    }
  }



  public static void main(String args[])
  {
    Scanner input = new Scanner(System.in);
    System.out.println("Enter the two numbers: ");
    int x = input.nextInt();
    int y = input.nextInt();

    System.out.println("The GCD of two numbers is: " + gcd(x, y));
    System.out.println("The LCM of two numbers is: " + lcm(x, y));
    input.close();
  }
}
```

```
Input:4 5
Output: The GCD of two numbers is 1
        The GCD of two numbers is 20
```

Program Inspection for GCD and LCM of two numbers

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

- **Errors Identified: 3**
  - **GCD Logic Error:** In the gcd method, the condition in the while loop should be while (a % b != 0) instead of while (a % b == 0). The current condition results in an infinite loop if a is not divisible by b at the start.
  - **LCM Logic Error:** In the lcm method, the logic to find the LCM is incorrect. The condition if (a % x != 0 && a % y != 0) should be changed to if (a % x == 0 && a % y == 0) to correctly return the LCM when both conditions are satisfied.
  - **Unnecessary Variable:** The variable r in the gcd method is not needed. You can return b directly once the loop exits.

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

- **Effective Category:**
  - **Logic Errors:** This category is highly relevant because it deals with the correctness of the algorithms used to calculate GCD and LCM. Ensuring that mathematical operations are performed correctly is crucial for the expected output.

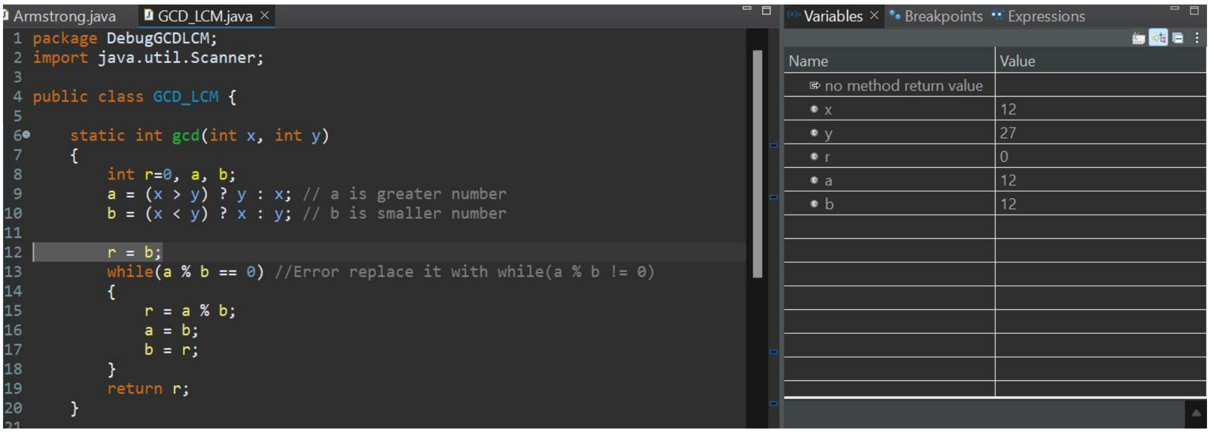
3. **Which type of error you are not able to identify using the program inspection?**

- **Errors Not Identified:**
  - **Performance Issues:** While program inspection can identify logical and syntactical errors, it may not highlight performance concerns such as inefficient algorithms, particularly with larger numbers.

4. **Is the program inspection technique worth applicable?**

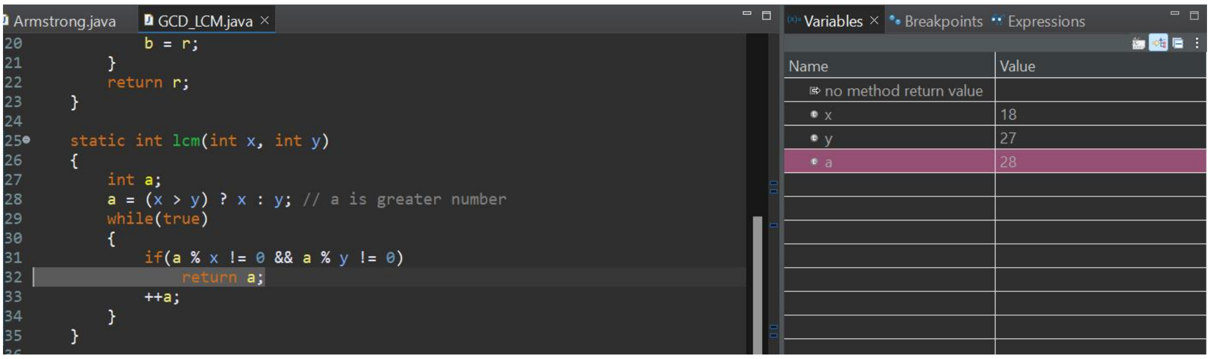
- **Applicability of Program Inspection:**
  - Yes, program inspection is a valuable technique as it helps catch logical and syntactical errors before runtime. It ensures that the code behaves as expected. However, it should be combined with testing to verify correctness across various input scenarios.

Code Debugging :



```
1  package DebugGCDLCM;
2  import java.util.Scanner;
3  
4  public class GCD_LCM {
5  
6      static int gcd(int x, int y)
7      {
8          int r=0, a, b;
9          a = (x > y) ? y : x; // a is greater number
10         b = (x < y) ? x : y; // b is smaller number
11 
12         r = b;
13         while(a % b == 0) //Error replace it with while(a % b != 0)
14         {
15             r = a % b;
16             a = b;
17             b = r;
18         }
19         return r;
20     }
```

Name	Value
no method return value	
x	12
y	27
r	0
a	12
b	12



```
20             b = r;
21         }
22         return r;
23     }
24 
25     static int lcm(int x, int y)
26     {
27         int a;
28         a = (x > y) ? x : y; // a is greater number
29         while(true)
30         {
31             if(a % x != 0 && a % y != 0)
32                 return a;
33             ++a;
34         }
35     }
```

Name	Value
no method return value	
x	18
y	27
a	28

Errors Identified

1. **GCD Calculation:**
   - **Original Condition:** while(a % b == 0)
   - **Correction:** Change to while(a % b != 0) (This allows the loop to continue until b is 0, which is when the GCD is found.)
2. **LCM Calculation:**
   - **Original Condition:** if(a % x != 0 && a % y != 0)
   - **Correction:** Change to if(a % x == 0 && a % y == 0) (This checks for the least common multiple correctly.)

Breakpoints Needed

You can set breakpoints at the following locations for effective debugging:

- **Line 13:** To check the loop logic for GCD.
- **Line 24:** To check the condition in the if statement for LCM.
- **Line 31:** To verify the final values of GCD and LCM.

Steps to Fix the Errors

1. **Fix GCD Calculation:**
   - Change the condition in the while loop to while(a % b != 0).

2. **Fix LCM Calculation:**
    - Change the condition in the if statement to if(a % x == 0 && a % y == 0).

Fixed Code

```
// Program to calculate the GCD and LCM of two given numbers
import java.util.Scanner;

public class GCD_LCM {
  // Method to calculate GCD using the Euclidean algorithm
  static int gcd(int x, int y) {
    int r = 0, a, b;
    a = (x > y) ? x : y; // a is the greater number
    b = (x < y) ? x : y; // b is the smaller number
    r = b;

    while (a % b != 0) { // Correct condition: loop until remainder is 0
      r = a % b;
      a = b;
      b = r;
    }

    return r; // The last non-zero remainder is the GCD
  }

  // Method to calculate LCM
  static int lcm(int x, int y) {
    int a;
    a = (x > y) ? x : y; // a is the greater number

    while (true) {
      if (a % x == 0 && a % y == 0) // Correct condition: divisible by both x and y
        return a; // Return the LCM
```

```
            ++a; // Increment a if not divisible
        }
    }

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the two numbers: ");
        int x = input.nextInt();
        int y = input.nextInt();
        System.out.println("The GCD of two numbers is: " + gcd(x, y));
        System.out.println("The LCM of two numbers is: " + lcm(x, y));
        input.close();
    }
}
```

Input and Output

- **Input:** 4 5
- **Output:**

```
The GCD of two numbers is: 1
The LCM of two numbers is: 20
```