Name : Siddhant Kotak

Student ID : 202201410

Course : IT 314 Software Engineering

Professor : Saurabh Tiwari

Semester : Autumn 2024

Lab : Program Inspection, Debugging and Static Analysis

Armstrong Number

Given code :

```
//Armstrong Number

class Armstrong{

        public static void main(String args[]){

                int num = Integer.parseInt(args[0]);

                int n = num; //use to check at last time

                int check=0,remainder;

                while(num > 0){

                        remainder = num / 10;

                        check = check + (int)Math.pow(remainder,3);

                        num = num % 10;

                }

                if(check == n)

                        System.out.println(n+" is an Armstrong Number");

                else

                        System.out.println(n+" is not a Armstrong Number");

        }
```

Input: 153

Output: 153 is an armstrong Number.

Program Inspection for Armstrong numbers

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

- **Errors Identified: 3**
    - **Incorrect Calculation of Remainder:** In the loop, the calculation of remainder should be done using num % 10 instead of num / 10. The current implementation incorrectly computes the next digit and does not correctly extract the digits of the number.
    - **Updating the Number Incorrectly:** The line num = num % 10; should be replaced with num = num / 10; to reduce the number correctly for the next iteration of the loop.
    - **Inconsistent Output Format:** The output statement should use "an" instead of "a" before "Armstrong Number" when referring to "Armstrong Number" because it starts with a vowel.

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

- **Effective Category:**
    - **Logic Errors:** This category is essential here since the Armstrong number check involves mathematical calculations and correctly iterating through the digits of a number. Identifying logical flaws is critical to ensure the correct behavior of the program.

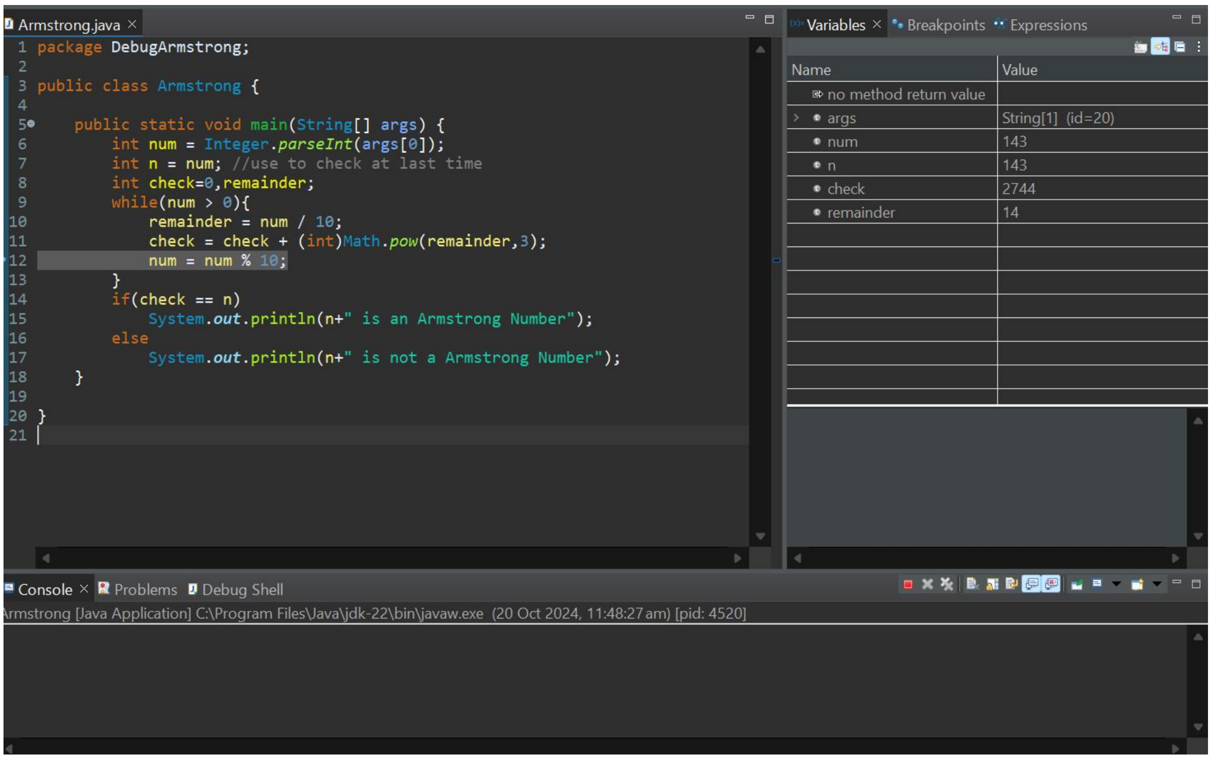
3. **Which type of error you are not able to identify using the program inspection?**

- **Errors Not Identified:**
    - **Edge Cases:** While program inspection can reveal logical and syntactical errors, it may not fully capture potential issues with edge cases, such as negative numbers or non-integer inputs.

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

- **Applicability of Program Inspection:**
    - Yes, program inspection is valuable as it helps identify logical and syntactical errors before runtime. However, it should be used alongside unit tests to verify the program’s correctness across a variety of input scenarios.

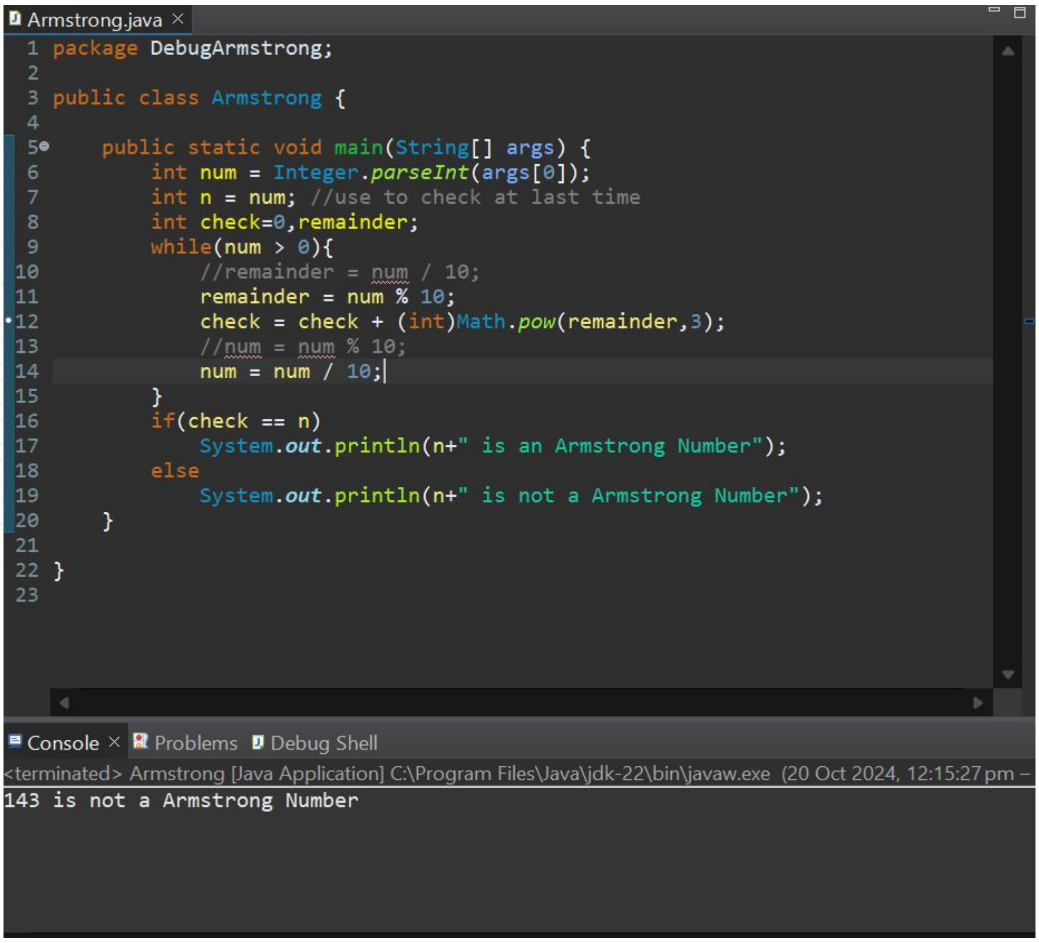
Code Debugging



```
package DebugArmstrong;

public class Armstrong {

    public static void main(String[] args) {
        int num = Integer.parseInt(args[0]);
        int n = num; //use to check at last time
        int check=0,remainder;
        while(num > 0){
            remainder = num / 10;
            check = check + (int)Math.pow(remainder,3);
            num = num % 10;
        }
        if(check == n)
            System.out.println(n+" is an Armstrong Number");
        else
            System.out.println(n+" is not a Armstrong Number");
    }

}

```

Name	Value
no method return value	
args	String[1] (id=20)
num	143
n	143
check	2744
remainder	14

```
rmstrong [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (20 Oct 2024, 11:48:27 am) [pid: 4520]
```



```java
package DebugArmstrong;

public class Armstrong {

    public static void main(String[] args) {
        int num = Integer.parseInt(args[0]);
        int n = num; //use to check at last time
        int check=0,remainder;
        while(num > 0){
            //remainder = num / 10;
            remainder = num % 10;
            check = check + (int)Math.pow(remainder,3);
            //num = num % 10;
            num = num / 10;
        }
        if(check == n)
            System.out.println(n+" is an Armstrong Number");
        else
            System.out.println(n+" is not a Armstrong Number");
    }

}
```

```
<terminated> Armstrong [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe  (20 Oct 2024, 12:15:27 pm –
143 is not a Armstrong Number
```

Errors Identified

1. **Incorrect Calculation of Remainder:**
   - **Original Line:** remainder = num / 10;
   - **Correction:** remainder = num % 10; (This extracts the last digit of the number.)
2. **Incorrect Update of num:**
   - **Original Line:** num = num % 10;
   - **Correction:** num = num / 10; (This removes the last digit from the number.)
3. **Missing Closing Bracket:**
   - Ensure that the class and main method have matching closing brackets.

Breakpoints Needed

To effectively debug the code, consider setting breakpoints at the following locations:

- After int num = Integer.parseInt(args[0]); to check the initial value of num.
- After remainder = num % 10; to verify the value of remainder.
- After num = num / 10; to see how num changes after extracting the last digit.

Steps to Fix the Errors

1. Change the line for calculating the remainder to remainder = num % 10;.
2. Change the line for updating num to num = num / 10;.
3. Ensure the closing brackets for the class and the main method are properly placed.

Fixed Code

```
// Armstrong Number
class Armstrong {
  public static void main(String args[]) {
    int num = Integer.parseInt(args[0]);
    int n = num; // use to check at last time
    int check = 0, remainder;

    while (num > 0) {
      remainder = num % 10; // Extract the last digit
      check = check + (int) Math.pow(remainder, 3); // Sum of cubes of digits
      num = num / 10; // Remove the last digit
    }

    if (check == n)
      System.out.println(n + " is an Armstrong Number");
    else
      System.out.println(n + " is not an Armstrong Number");
  }
}
```

Input and Output

- **Input:** 153
- **Output:** 153 is an Armstrong Number