IT 314 – Software Engineering

Lab 7

Akshat Jindal | 202201299

Armstrong Code

Program Inspection

Question 1. How many errors are there in the program? Mention the errors you have identified.
Data reference Error for the string args given as a command line argument. The code is not able to tackle the situation when there are no input args.

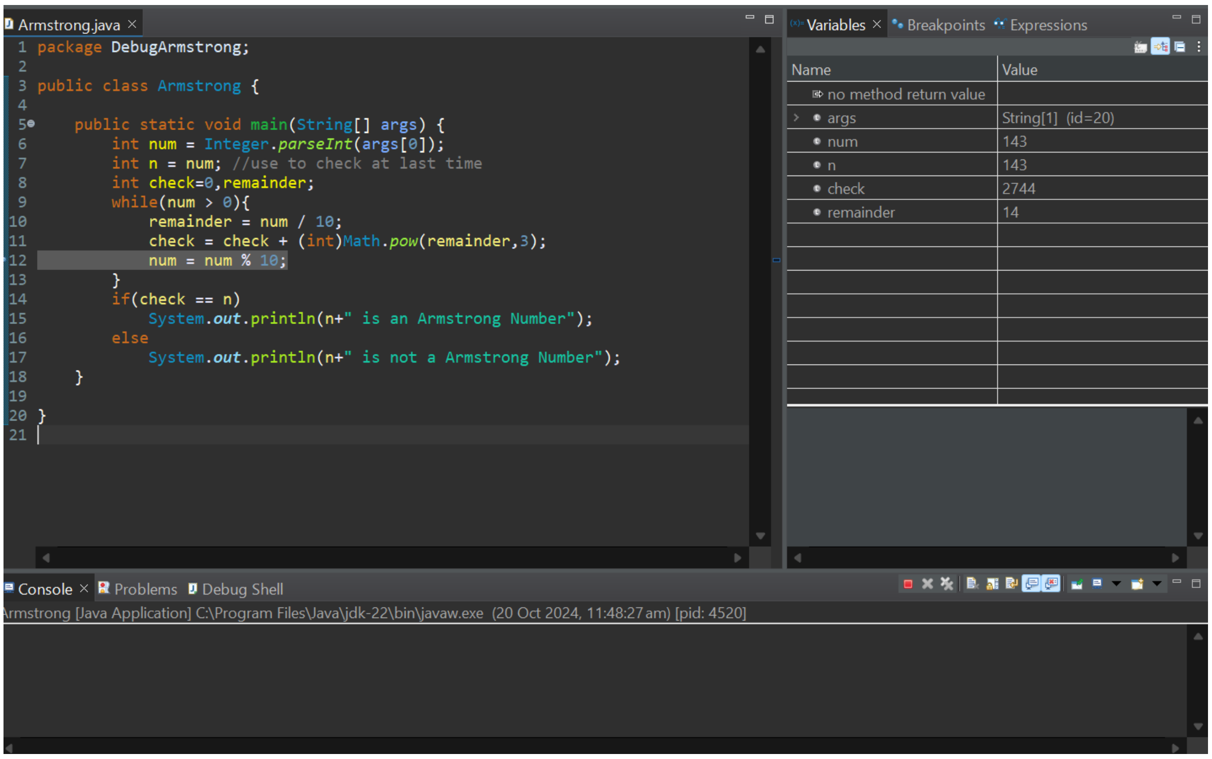
Computational Error in calculating the remainder and the quotient in the for loop, with the modulus and the division operator interchanged.

Question 2. Which category of program inspection would you find more effective?
For this question, the Category C of the program inspection, the Computational errors were the most effective, as the failure of the code was due to the use of wrong operators.

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 small length of the code and easy to find computational errors, the program inspection technique is worth applicable here.

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]
```

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

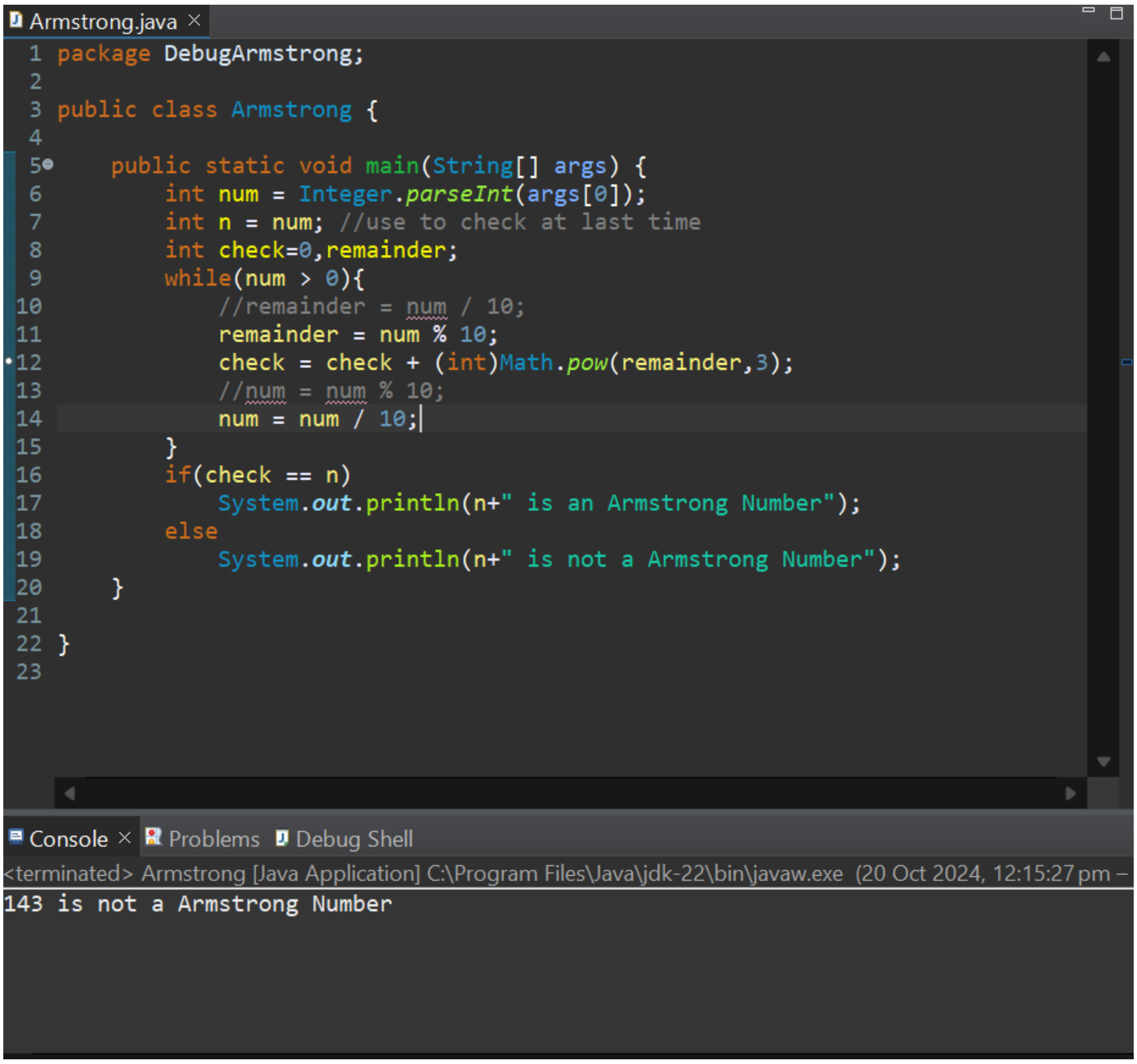
There were 2 errors in this program. The calculation of the remainder as well as the quotient was flawed due to the interchanged operators.

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 just one break point of line 12, as shown in the figure, we can see the error in the code on the first run of the while loop, when the values for remainder and check come out incorrect.

These errors can be fixed by simply using the correct operators in the initial code to correctly calculate the values for remainder, check and num.

Question 3. Submit your complete executable code.



Armstrong.java

```
 1 package DebugArmstrong;
 2 
 3 public class Armstrong {
 4 
 5     public static void main(String[] args) {
 6         int num = Integer.parseInt(args[0]);
 7         int n = num; //use to check at last time
 8         int check=0,remainder;
 9         while(num > 0){
10             //remainder = num / 10;
11             remainder = num % 10;
12             check = check + (int)Math.pow(remainder,3);
13             //num = num % 10;
14             num = num / 10;
15         }
16         if(check == n)
17             System.out.println(n+" is an Armstrong Number");
18         else
19             System.out.println(n+" is not a Armstrong Number");
20     }
21 
22 }
23 
```

Console Problems Debug Shell

```
<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
```