IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-3, Mar 2019 E-ISSN: 2347-2693

Deviation of Remainder from Euclidean Definition, Java's Perspective, Reason(s) and Suggestion(s)

Rishi Saxena

Dept. of Computer Science, Sophia Girls' College, Ajmer (Autonomous), India.

Corresponding author: rishi.522.in@gmail.com

DOI: https://doi.org/10.26438/ijcse/v7i3.10791083 | Available online at: www.ijcseonline.org

Accepted: 16/Mar/2019, Published: 31/Mar/2019

Abstract- Current Era is the age of Information Technology. No single domain is untouched by digital systems. The backbone of this era is the programming languages which make such high quality software that can solve the most challenging problems to human mind and eliminate the chances of error. Java is such a language which spans across standalone software programming to web development to mobile application development to networking to cognitive programming to data mining and etc, which is the foundation of all modern programming.

The remainder operation is a part of every programming language, so in Java, is a very important operation and has its use from mundane computing to the most technical implementations. This paper addresses the issue that remainder operation in mathematics is achieved by the definition given by the great mathematician Euclid, founder of geometry, whereas in Java the implementation of remainder division does not exactly follow the rules of mathematics, and due to this deviation and also for a programmer who does not know this fact can unknowingly program a software with wrong calculations which may result in catastrophe.

Besides raising this issue this paper also provides suggestion(s) and solution(s) to overcome the matter and align properly with the underlying mathematics of remainder operation.

***Keywords*:** Remainder, Java, Euclidean theorem, Arithmetic Operators

I. INTRODUCTION

Java is a pure object oriented programming language designed & developed by a computer scientist Mr. James Arthur Gosling and was released in the year 1995 CE by SUN Microsystems [1].

Java is a Multi-paradigm language i.e. object-oriented, structured, imperative, generic, reflective and concurrent.
When Java was created, the primary goals and objectives were, which are now referred as the key features of Java, "Simple, object-oriented, & familiar", "robust & secure", "architecture-neutral & portable", "high performance", and "interpreted, threaded, & dynamic"

Java provides implementation of several programming platforms in form of Java Development Kit, versions ranging from JDK 1.0 released on January 23, 1996 to Java SE 11 released on September 25, 2018.
Currently Java is owned by Oracle Corporation, their official website is https://www.oracle.com/in/java/

Operators are basically constructs in programming languages; their behavior is similar to that of functions with difference in their syntax as well as in semantics. Syntax of functions are derived from the normal polish notation in which operators are written before operands whereas syntax of operators are in infix notation in which left operand precedes the operator which precedes the right operand.
To perform basic mathematical operations in the same way as in Algebra Java has a set of operators named as Arithmetic operators:

Table 1

S. No.	Symbol of Operator	Description of the Operation Performed
1	+	Result returned as addition of left operand with right operand

2	-	Result returned as subtraction of right operand from left operand
3	*	Result returned as multiplication of left operand with right operand
4	/	Result returned as division of right operand from left operand
5	%	Result returned as the remainder after dividing left operand by right operand

In arithmetic the remainder is the left-over quantity after performing the division operation on two integers and obtaining an integer quotient.

In computer science numbers are internally represented by using positional notation system with radix 2 referred as binary numeral system which has only two symbols zero and one represented as 0 and 1. In binary numeral system each digit, which is the basic unit of information, is referred as a Bit (a word formed by joining two words binary & digit).

The paper is organized as follows, Section I introduces the various aspects of the paper – Java programming language, Arithmetic operators, Remainder operation and representation of numbers in computer, Section II is about the concept of remainder in the field of mathematics, Section III comprises of How java language has implemented remainder operation for integer operands, Section IV contains IEEE specification about the remainder operation, Section V comprises of Java's remainder implementation for floating point operands, Section VI contrasts the Java's implementation with Euclidean definition, Section VII addresses the importance of remainder in computing, Section VIII acts as a suggestion to programmer for effective & aligned to Euclidean definition solution, Section IX states the conclusion about the research study and specifies the future scope.

II. CONCEPT OF REMAINDER

In the field of Mathematics the remainder is the quantity, which is left over after division, and this happens when dividend is not exactly divided by the divisor.
For example, 18 are not exactly divided by 5. If we divide 18 in equal parts of 5 then after 3 divisions the left over quantity is 3, so the remainder is 3.

In computer science and mathematics, modulo operation or sometimes modulus operations is used to find the remainder after the execution of division operation over two numbers.
Modulo / modulus operation is based on Euclidean Division algorithm [2]. The theorem states that, let 'a' and 'b' be two given integers, with b is not equal to zero, then a = b*q + r, where q and r are unique integers.
The names have been given to the four integers used in the above equation are: a stands for dividend, b stands for divisor, q stands for quotient, and r stands for remainder. The main property of Euclidean division theorem is the uniqueness of quotient and remainder.

III. JAVA'S IMPLEMENTATION FOR INTEGER OPERANDS

The operator with symbol '%', works on binary operands, yields the remainder of the division performed by left hand operand as numerator and right hand operand as denominator.
Java's implementation of % operator allows operands to be of floating point type as opposed to C & C++ where operands are restricted to integer data type only.

The result obtained after applying the % operator over integer operands 'x' and 'y', is such that (x / y) * y + (x % y) = x. for example let x = 7 and y = 3, then 7/3 is 2, 2*3 is 6, 7%3 is 1, and 6+1 is 7. The sign of the remainder obtained after applying the % operator depends upon the sign of numerator, if numerator is positive the remainder thus obtained is positive and if numerator is negative the remainder returned with negative sign. And the result's magnitude is always less than the divisor's magnitude.
Java's run time environment throws an exception of type ArithmeticException if the values of the numerator are zero.

IV. IEEE 754 REMAINDER SPECIFICATION

The result obtained as remainder by division operation performed over two floating point numbers follows the following rules:
If either left hand operand or right hand operand is Not a Number (NaN) or the numerator is infinite, or the divisor is zero, then the result is Not a Number (NaN) and the sign of the result is the sign of the numerator.

The result becomes the numerator if the numerator is finite and the denominator is infinite, or if the numerator is zero and the denominator is finite.
Beside the aforementioned cases in which zero or NaN or infinite values are not involved in the division operation the remainder is calculated by using the following equation, i.e. remainder = numerator – (denominator * quotient), in which sign of quotient is positive/negative if the numerator/denominator is positive/negative.

V. JAVA'S IMPLEMENTATION FOR FLOATING POINT OPERANDS

In Java modulus operator % is allowed to be applied on floating point operands and it returns the remainder of the division operation, but the result obtained after applying the modulus operator % over floating point operands is not in accordance to the specifications defined by IEEE 754 remainder operation because the IEEE 754 remainder operation follows the IEEE 754 rounding rules, which specify five rules, that are Rule 1 - Round to nearest, ties to even, Rule 2 - Round to nearest, ties away from zero, Rule 3 - Round toward 0, Rule 4 - Round toward $+\infty$, Rule 5 - Round toward $-\infty$, whereas Java's remainder operation on floating point operands follows truncation division which limits the number of digits after the decimal point and its behavior is very much analogous to the remainder operation on integer operands.

VI. JAVA'S IMPLEMENTATION VERSUS EUCLIDEAN DEFINITION

The following illustration on literal values establishes the contrast between Euclidean definition and Java's Implementation:
Let the question be is to find out the remainders of all the four positive & negative combinations number 7 divided by number 3, that are (7 mod 3), (7 mod -3), (-7 mod 3) and (-7 mod -3).
When the following code segment is tested in NetBeans IDE 8.2,

```
public class Program{
    public static void main(String[] args) {

        System.out.println(7%3);
        System.out.println(7%-3);
        System.out.println(-7%3);
        System.out.println(-7%-3);
    }
}
```

Gives the output as:

```
1
1
-1
-1
```

Whereas Euclidean theory finds the remainder by following:
For case 7 % 3:
If x = 7 & y = 3, then for 7 divided by 3, quotient will be 2 and remainder will be 1, as $7 = 3 \times 2 + 1$.
The answer is same as Java's output.

For case 7 % -3:
If x = 7 & y = -3, then for 7 divided by -3, quotient will be -2 and remainder will be 1, as $7 = -3 \times -2 + 1$.
The answer is same as Java's output.

For case -7 % 3:
If x = -7 & y = 3, then for -7 divided by 3, quotient will be -3 and remainder will be 2, as $-7 = 3 \times -3 + 2$.
The answer is different from Java's output.

For case -7 % -3:
If x = -7 & y = -3, then for -7 divided by -3, quotient will be 3 and remainder will be 2, as $-7 = -3 \times 3 + 2$.
The answer is different from Java's output.

VII. IMPORTANCE OF REMAINDER IN COMPUTER APPLICATIONS

As the current age is of information technology, there is literally no area is left where computer applications are not used. As computer programs play vital role in the current era, here I would like to emphasize the use of remainder in everyday programming. Suppose Mrs. ABC wants to compile a cookbook for all her 24 family recipes, and she wants the book to be of 100 pages for recipes only with equal amount of pages for each recipe, then how many pages she has to write for each recipe? A computer program answers this question by using remainder, that is 100 / 24 = Quotient is 4 and the remainder is also 4. Which means she has to compile each recipe in 4 pages and still 4 pages will be left? In this scenario she can add an extra bonus recipe to her readers in her cookbook.
Besides the above example, remainder operation is used in almost every sub domain application of computer science, that is, from Engineering to Commerce, from Creativity to Artificial Intelligence, from Academics to Industry.

VIII. SUGGESTIONS

Following algorithm followed by the code segment may serve as a suggestion for implementing it accordingly to the Euclidean definition:
Algorithm Remainder
[Input numerator and denominator as A and B]
Step 1. Begin
Step 2. Declare R←0, Q←0 [R as remainder, Q as quotient]
Step 3. If A and B both are positive, begin incrementing Q by 1 and multiply B with Q till the result is less than or equal to A.
Step 4. If A is positive and B is negative, begin decrementing Q by 1 and multiply B with Q till the result is less than or equal to A.
Step 5. If A is negative and B is positive, begin decrementing Q by 1 and multiply B with Q till the result is greater than or equal to A.
Step 6. If A is negative and B is negative, begin incrementing Q by 1 and multiply B with Q till the result is less than or equal to A.
Step 7. R ← A – BxQ [According to Euclidean Theorem: A = BxQ + R]
Step 8. Return Q, R
Step 9. End.

Code Segment:

```
public class Program{
   public static void remainder(int a, int b){
      int r=0, q=0, ans=b*q;
      if (a>0 && b>0){
         while(ans <= a){
            ans = b*(++q);
         }
         q--;
      }
      if(a>0 && b<0){
         while(ans <= a){
            ans = b*(--q);
         }
         q++;
      }
      if(a<0 && b>0){
         while(ans >= a){
            ans = b*(--q);
         }
      }
      if(a<0 && b<0){
         while(ans >= a){
            ans = b*(++q);
         }
```



```
        }
        r = a - b*q;
        System.out.println("Quotient is " + q);
        System.out.println("Remainder is " + r);
    }
    public static void main(String[] args) {
        remainder(7, 3);
        remainder(7, -3);
        remainder(-7, 3);
        remainder(-7, -3);
    }
}
```

Output:

Quotient is 2
Remainder is 1

Quotient is -2
Remainder is 1

Quotient is -3
Remainder is 2

Quotient is 3
Remainder is 2

IX. CONCLUSION

The development of computer science and IT is to provide computing support in every field of life thus making it better and easier. In non-IT field the remainder is derived by using Euclidean method. If any field whether it is scientific or everyday computing this difference of implementation of remainder creates confusion and ambiguity, and chances of error which may thus result in fatal consequences. The study done in the above paper is limited to Java programming language's implementation only and provides future scope for testing the Euclidean implementation over other programming languages and programming models for better computing.

REFERENCES

[1] H. Schildt, "Java The Complete Reference Ninth Edition", McGraw-Hill Education, United States of America, pp. 10-22, 2014, ISBN: 978-0-07-180856-9
[2] David M. Burton, "Elementary Number Theory", McGraw-Hill Higher Education, United States of America, pp. 26-32, 2007, ISBN-IO 0-07-305I88-8
[3] Saurabh S. Patel and Sunil N. Kore, "Case Study: Digital Advertisement Board", ISROSET-Journal (IJSRCSE) , Vol.1 , Issue.3 , pp.48-50, May-2013
[4] V. Kapoor, "Data Encryption and Decryption Using Modified RSA Cryptography Based on Multiple Public Keys and 'n'prime Number", Journal (IJSRNSC), Vol.1 , Issue.2 , pp.35-38, May-2013

