Assignment(17-07-2023)-02

Bit-Wise Complement (~) and Big Integer

Submitted by:

Name: Santhosh S

Batch: July-A2

Email: santhoshsanthosh86920@gmail.com

Bit-Wise (~):

Bitwise Complement operator simply means the negation of each bit of the input value.

This operator changes each binary digit of the integer, which means all 0 become 1 and all 1 become 0.

Example:

```
If a  =  12 , The Binary form of 12 = 1100
Bitwise complement operation on (~12) = ~ 1100
                                      =   0011
0011 is equivalent to decimal value -13.
```

Code:

```
class Eg1
{
public static void main (String[] args)
{
int a = 2;
System.out.println("Bit-Wise Complement (~) of a is: "+ ~a);
}
}
```

Output:

```
    Bit-Wise Complement (~) of a is: -13
```

Big Integer:

In Java, **Big Integer** is a class that belongs to the **java.math** package and is used to represent arbitrarily large integers.

It is especially useful when you need to work with numbers that exceed the limits of primitive data types like int, long, etc.,

In Java provides a variety of built-in methods to perform operations and manipulate large integer values.

Here are some of the commonly used methods:

Arithmetic Operations:

add (Big Integer value):

Returns a Big Integer whose value is (this + value).

subtract (Big Integer value):

Returns a Big Integer whose value is (this - value).

multiply (Big Integer value):

Returns a Big Integer whose value is (this * value).

divide (Big Integer value):

Returns a Big Integer whose value is (this / value).

remainder (Big Integer value):

Returns a Big Integer whose value is (this % value).

pow (int exponent):

Returns a Big Integer whose value is (this ^ exponent).

Comparison Operations:

compareTo (Big Integer value):

Compares this Big Integer with another Big Integer. Returns -1 if less, 0 if equal, or 1 if greater.

equals (Object x):

Compares this Big Integer with another object for equality.

Conversion Operations:

toString():

Returns the decimal string representation of this Big Integer.

intValue(), longValue(), floatValue(), doubleValue():

Converts to primitive number types.

Bit Manipulation:

shiftLeft (int n):

Returns a Big Integer whose value is (this << n).

shiftRight (int n):

Returns a Big Integer whose value is (this >> n).

Other Operations:

abs():

Returns the absolute value of this Big Integer.

negate():

Returns the negation of this Big Integer.

gcd (Big Integer val):

Returns the greatest common divisor of this Big Integer and val.

isProbablePrime (int certainty):

Tests if this Big Integer is probably prime.

nextProbablePrime():

Returns the next probable prime after this Big Integer

.

Here's an example that demonstrates some of these methods:

Code:

```
import java.math.BigInteger;
public class BigInteger {
public static void main(String[] args)
{
BigInteger bigNum1 = new BigInteger("123456789012345678901234567890");
BigInteger bigNum2 = new BigInteger("987654321098765432109876543210");

 // Arithmetic operations
BigInteger sum = bigNum1.add(bigNum2);
BigInteger difference = bigNum1.subtract(bigNum2);
BigInteger product = bigNum1.multiply(bigNum2);
BigInteger division = bigNum1.divide(bigNum2);

// Comparison operation
int comparisonResult = bigNum1.compareTo(bigNum2);

// Conversion operation
int intValue = bigNum1.intValue();
double doubleValue = bigNum1.doubleValue();

// Bit manipulation
BigInteger shiftedLeft = bigNum1.shiftLeft(2);

// Display the results
System.out.println("Sum: " + sum);
System.out.println("Difference: " + difference);
System.out.println("Product: " + product);
System.out.println("Division: " + division);
System.out.println("Comparison Result: " + comparisonResult);
System.out.println("intValue: " + intValue);
System.out.println("doubleValue: " + doubleValue);
System.out.println("Shifted Left: " + shiftedLeft);
}
}
```

Output:

```
Sum:   111111111011111111101111111100
Difference:   -864197532097531086208642098320
Product:   121932631137021795147011138493419071010776091147161OO
Division:   0
Comparison Result:   -1
intValue:   1582659436
doubleValue:   1.2345678901234568E29
Shifted Left:   493827156049382765603827156040
```

These are just a few of the methods provided by the BigInteger class to perform operations on large integers in Java.