BITWISE COMPLEMENT (~):

In Java, the bitwise complement operator (~) is used to perform bitwise complement or negation on an integer value. It flips all the bits of the operand, changing every 0 to 1 and every 1 to 0.

Example:

```
int number = 42;  // Binary: 00101010
int complement = ~number;  // Binary: 11010101

System.out.println(complement);
```

Output: -43

LOGICAL COMPLEMENT (!):

In Java, the logical complement is an operator called the "logical NOT" operator, represented by the exclamation mark (!). It is used to reverse the logical state of a boolean expression.

The logical NOT operator is a unary operator, which means it operates on a single operand. When applied to a boolean value, it returns the opposite value. If the operand is true, the logical NOT operator returns false, and if the operand is false, it returns true.

Example:

```
boolean isTrue = true;
boolean isFalse = false;

boolean oppositeOfTrue = !isTrue;
boolean oppositeOfFalse = !isFalse;

System.out.println(oppositeOfTrue);
System.out.println(oppositeOfFalse);
```

Output:
```
false
true
```

- HOW TO STORE VALUES EXCEEDING THE ***double*** RANGE?

Typically,double can represent numbers with a precision of about 15 decimal places and a range of approximately ±1.7 × 10^308.
However,to store a value that exceeds the range of a double, we can consider using alternative data types or libraries that support arbitrary-precision arithmetic.

BigDecimal: Many programming languages, such as Java, provide a BigDecimal class that allows precise decimal arithmetic with arbitrary precision. It can handle very large numbers, but it may be slower and less efficient than double for common mathematical operations.

BigInteger: If you need to work with integers that exceed the range of a double, you can use a BigInteger class or library. Similar to BigDecimal, it provides arbitrary-precision arithmetic for integer values.

Specialized libraries: Depending on the programming language you're using, there might be specialized libraries or extensions available for handling large numbers or arbitrary-precision arithmetic. For example, in Python, you could use the decimal module or the sympy library.

- RANGE OF ***long*** AND ***double:***

The ***long*** data type is a 64-bit signed integer type that can store values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (inclusive).

The ***double*** data type is a 64-bit floating-point type and It can represent values from approximately ±4.9e-324 to ±1.8e+308, including positive and negative zero, positive and negative infinity, and NaN (not a number).