**BigInteger Class in Java**

BigInteger class is used for the mathematical operation which involves very big integer calculations that are outside the limit of all available primitive data types.

In this way, BigInteger class is very handy to use because of its large method library and it is also used a lot in competitive programming.   
Now below is given a list of simple statements in primitive arithmetic and its analogous statement in terms of BigInteger objects.

**Example:**

int a, b;

BigInteger A, B;

Initialization is asfollows**:**

a = 54;

b = 23;

A = BigInteger.valueOf(54);

B = BigInteger.valueOf(37);

And for Integers available as strings you can initialize them as follows:

A = new BigInteger(“54”);

B = new BigInteger(“123456789123456789”);

Some constants are also defined in BigInteger class for ease of initialization as follows:

A = BigInteger.ONE;

// Other than this, available constant are BigInteger.ZERO

// and BigInteger.TEN

Mathematical operations are as follows:

int c = a + b;

BigInteger C = A.add(B);

Other similar functions are subtract(), multiply(), divide(), remainder(), but all these functions take BigInteger as their argument so if we want this operation with integers or string convert them to BigInteger before passing them to functions as shown below:

String str = “123456789”;

BigInteger C = A.add(new BigInteger(str));

int val = 123456789;

BigInteger C = A.add(BigInteger.valueOf(val));

Extraction of value from BigInteger is as follows:

int x = A.intValue(); // value should be in limit of int x

long y = A.longValue(); // value should be in limit of long y

String z = A.toString();

Comparison

if (a < b) {} // For primitive int

if (A.compareTo(B) < 0) {} // For BigInteger

Actually compareTo returns -1(less than), 0(Equal), 1(greater than) according to values. For equality we can also use:

if (A.equals(B)) {} // A is equal to B

### **Methods of BigInteger Class**

| **Method** | **Action Performed** |
| --- | --- |
| add(BigInteger val) | Returns a BigInteger whose value is (this + val). |
| [**abs()**](https://www.geeksforgeeks.org/biginteger-abs-method-in-java/) | Returns a BigInteger whose value is the absolute value of this BigInteger. |
| [and(BigInteger val)](https://www.geeksforgeeks.org/biginteger-and-method-in-java/) | Returns a BigInteger whose value is (this & val). |
| [andNot(BigInteger val](https://www.geeksforgeeks.org/biginteger-andnot-method-in-java/)) | Returns a BigInteger whose value is (this & ~val). |
| [bitCount()](https://www.geeksforgeeks.org/biginteger-bitcount-method-in-java/) | Returns the number of bits in the two’s complement representation of this BigInteger that differ from its sign bit. |
| [bitLength()](https://www.geeksforgeeks.org/biginteger-bitlength-method-in-java/) | Returns the number of bits in the minimal two’s-complement representation of this BigInteger, excluding a sign bit. |
| [byteValueExact()](https://www.geeksforgeeks.org/java-8-biginteger-bytevalueexact-method-with-examples/) | Converts this BigInteger to a byte, checking for lost information. |
| [clearBit(int n)](https://www.geeksforgeeks.org/biginteger-clearbit-method-in-java/) | Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit cleared. |
| [compareTo(BigInteger val)](https://www.geeksforgeeks.org/biginteger-compareto-method-in-java/) | Compares this BigInteger with the specified BigInteger. |
| divide(BigInteger val) | Returns a BigInteger whose value is (this / val). |
| [divideAndRemainder(BigInteger val)](https://www.geeksforgeeks.org/java-8-biginteger-divideandremainder-method-with-examples/) | Returns an array of two BigIntegers containing (this / val) followed by (this % val). |
| [doubleValue()](https://www.geeksforgeeks.org/biginteger-doublevalue-method-in-java/) | Converts this BigInteger to a double. |
| [equals(Object x)](https://www.geeksforgeeks.org/biginteger-equals-method-in-java/) | Compares this BigInteger with the specified Object for equality. |
| [flipBit(int n)](https://www.geeksforgeeks.org/biginteger-flipbit-method-in-java/) | Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit flipped. |
| [floatValue()](https://www.geeksforgeeks.org/biginteger-floatvalue-method-in-java/) | Converts this BigInteger to a float. |
| gcd(BigInteger val) | Returns a BigInteger whose value is the greatest common divisor of abs(this) and abs(val). |
| [getLowestSetBit()](https://www.geeksforgeeks.org/biginteger-getlowestsetbit-method-in-java/) | Returns the index of the rightmost (lowest-order) one bit in this BigInteger (the number of zero bits to the right of the rightmost one bit). |
| hashCode() | Returns the hash code for this BigInteger. |
| [intValue()](https://www.geeksforgeeks.org/biginteger-intvalue-method-in-java/) | Converts this BigInteger to an int. |
| [intValueExact()](https://www.geeksforgeeks.org/biginteger-intvalueexact-method-in-java-with-examples/) | Converts this BigInteger to an int, checking for lost information. |
| isProbablePrime(int certainty) | Returns true if this BigInteger is probably prime, false if it’s definitely composite. |
| [longValue()](https://www.geeksforgeeks.org/biginteger-longvalue-method-in-java/) | Converts this BigInteger to a long. |
| [longValueExact()](https://www.geeksforgeeks.org/java-8-biginteger-longvalueexact-method-with-examples/) | Converts this BigInteger to a long, checking for lost information. |
| [max(BigInteger val)](https://www.geeksforgeeks.org/biginteger-max-and-min-methods-in-java/) | Returns the maximum of this BigInteger and val. |
| [min(BigInteger val](https://www.geeksforgeeks.org/biginteger-max-and-min-methods-in-java/)) | Returns the minimum of this BigInteger and val. |
| [mod(BigInteger m](https://www.geeksforgeeks.org/biginteger-mod-method-in-java/) | Returns a BigInteger whose value is (this mod m). |
| [modInverse(BigInteger m)](https://www.geeksforgeeks.org/java-math-biginteger-modinverse-method-in-java/) | Returns a BigInteger whose value is (this-1 mod m). |
| [modPow(BigInteger exponent, BigInteger m](https://www.geeksforgeeks.org/biginteger-modpow-method-in-java/) | Returns a BigInteger whose value is (thisexponent mod m). |
| multiply(BigInteger val) | Returns a BigInteger whose value is (this \* val). |
| [negate()](https://www.geeksforgeeks.org/biginteger-negate-method-in-java/) | Returns a BigInteger whose value is (-this). |
| nextProbablePrime() | Returns the first integer greater than this BigInteger that is probably prime. |
| [not()](https://www.geeksforgeeks.org/biginteger-not-method-in-java/) | Returns a BigInteger whose value is (~this). |
| [or(BigInteger val)](https://www.geeksforgeeks.org/biginteger-or-method-in-java/) | Returns a BigInteger whose value is (this | val). |
| [pow(int exponent)](https://www.geeksforgeeks.org/biginteger-pow-method-in-java/) | Returns a BigInteger whose value is (thisexponent). |
| [probablePrime(int bitLength, Random rnd)](https://www.geeksforgeeks.org/java-math-biginteger-probableprime-method-in-java/) | Returns a positive BigInteger that is probably prime, with the specified bitLength. |
| [remainder(BigInteger val)](https://www.geeksforgeeks.org/biginteger-remainder-method-in-java/) | Returns a BigInteger whose value is (this % val). |
| [setBit(int n)](https://www.geeksforgeeks.org/biginteger-setbit-method-in-java/) | Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit set. |
| [shiftLeft(int n)](https://www.geeksforgeeks.org/biginteger-shiftleft-method-in-java/) | Returns a BigInteger whose value is (this << n). |
| [shiftRight(int n)](https://www.geeksforgeeks.org/biginteger-shiftright-method-in-java/) | Returns a BigInteger whose value is (this >> n). |
| [shortValueExact()](https://www.geeksforgeeks.org/java-8-biginteger-shortvalueexact-method-with-examples/) | Converts this BigInteger to a short, checking for lost information. |
| [signum()](https://www.geeksforgeeks.org/biginteger-signum-method-in-java/) | Returns the signum function of this BigInteger. |
| [sqrt()](https://www.geeksforgeeks.org/biginteger-sqrt-method-in-java/) | Returns the integer square root of this BigInteger. |
| sqrtAndRemainder() | Returns an array of two BigIntegers containing the integer square root s of this and its remainder this – s\*s, respectively. |
| subtract(BigInteger val) | Returns a BigInteger whose value is (this – val). |
| [testBit(int n)](https://www.geeksforgeeks.org/biginteger-testbit-method-in-java/) | Returns true if and only if the designated bit is set. |
| [toByteArray()](https://www.geeksforgeeks.org/biginteger-tobytearray-method-in-java/) | Returns a byte array containing the two’s-complement representation of this BigInteger. |
| [toString()](https://www.geeksforgeeks.org/biginteger-tostring-method-in-java/) | Returns the decimal String representation of this BigInteger. |
| [toString(int radix)](https://www.geeksforgeeks.org/biginteger-tostring-method-in-java/) | Returns the string representation of this BigInteger in the given radix. |
| [valueOf(long val)](https://www.geeksforgeeks.org/biginteger-valueof-method-in-java/) | Returns a BigInteger whose value is equal to that of the specified long. |
| [xor(BigInteger val)](https://www.geeksforgeeks.org/biginteger-xor-method-in-java/) | Returns a BigInteger whose value is (this ^ val). |

**Illustration:**

Factorial of 100 contains 158 digits in it so we can’t store it in any primitive data type available. We can store as large an Integer as we want in it. There is no theoretical limit on the upper bound of the range because memory is allocated dynamically but practically as memory is limited you can store a number that has Integer.MAX\_VALUE number of bits in it which should be sufficient to store mostly all large values.