BigInteger Class in Java

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

For example 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 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 which has Integer.MAX\_VALUE number of bits in it which should be sufficient to store mostly all large values.

Below is an example Java program that uses BigInteger to compute Factorial.

|  |
| --- |
| // Java program to find large factorials using BigInteger  import java.math.BigInteger;  import java.util.Scanner;    public class Example  {      // Returns Factorial of N      static BigInteger factorial(int N)      {          // Initialize result          BigInteger f = new BigInteger("1"); // Or BigInteger.ONE            // Multiply f with 2, 3, ...N          for (int i = 2; i <= N; i++)              f = f.multiply(BigInteger.valueOf(i));            return f;      }        // Driver method      public static void main(String args[]) throws Exception      {          int N = 20;          System.out.println(factorial(N));      }  } |

Run on IDE

Output:

2432902008176640000

If we have to write above program in C++, that would be too large and complex, we can look at[Factorail of Large Number](http://www.geeksforgeeks.org/factorial-large-number/).  
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.

**Declaration**

int a, b;

BigInteger A, B;

**Initialization:**

a = 54;

b = 23;

A = BigInteger.valueOf(54);

B = BigInteger.valueOf(37);

And for Integers available as string you can initialize them as:

A = new BigInteger(“54”);

B = new BigInteger(“123456789123456789”);

Some constant are also defined in BigInteger class for ease of initialization :

A = BigInteger.ONE;

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

// and BigInteger.TEN

Mathematical operations:

int c = a + b;

BigInteger C = A.add(B);

Other similar function are subtract() , multiply(), divide(), remainder()  
But all these function take BigInteger as their argument so if we want these 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(BigIntger.valueOf(val));

***Extraction of value from BigInteger:***

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