## horizontal line





Perform operations on Big numbers in JAVA

# 

# Overview

When the values will be out of range of predefined data types and you need to perform mathematical operations on these values. In this case, JAVA provides a predefined class named “BigDecimal”. This class is present in JAVA math package. We can add this class in our program to write following syntax:

import java.math.BigDecimal;

JAVA includes this predefined class for performing high-precision arithmetic which is very useful in banking or financial domain based applications running on JAVA. This class approximately fit into the same category like the classe named “wrapper” but it has some other very useful methods as well.

Using this class in JAVA, you can do any operation with a big decimal number that you can with an int or float. When you use this class, there is only one difference that you must use method calls in place of mathematical operators. The operations will be slower because of method calls instead of operators. So here, you are exchanging speed for accuracy.

Below Java code explains the concept of converting String to BigDecimal.

# Syntax Code

import java.math.BigDecimal;

public class Syntex

{

    public static void main(String[] args)

    {

        BigDecimal bd=new BigDecimal("1010000000011111111102222222");

        // here bd is a BigDecimal of string “1010000000011111111102222222”

}

}

# Some important methods define in “BigDecimal” class

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Method** | **Decription** |
| **1** | abs() | This method will return a BigDecimal value whose value is the absolute value of this BigDecimal. |
| **2** | add() | This method will return a BigDecimal whose value is equal to the mathemetical addition of two numbers. |
| **3** | compareTo() | This method will compare the BigDecimal with the any specified BigDecimal number. |
| **4** | [divide()](https://www.javatpoint.com/java-math-bigdecimal-divide-method) | This method will return a BigDecimal number whose value is (value / divisor). |
| **5** | divideAndRemainder() | This method will return a two-element BigDecimal array containing the result of divide to Integral value followed by the result of remainder on the two operands. |
| **6** | [equals()](https://www.javatpoint.com/java-bigdecimal-equals-method) | This method is useful for comparing the BigDecimal with the specified object/value for equality. |
| **7** | [hashCode()](https://www.javatpoint.com/java-bigdecimal-hashcode-method) | This method will returns the hash code for the BigDecimal value. |
| **8** | longValue() | This method returns converted this BigDecimal to a long. |
| **9** | max() | This method will return the maximum of this BigDecimal and other specified. |
| **10** | [min()](https://www.javatpoint.com/java-bigdecimal-min-method) | This method will return the minimum of this BigDecimal and other specified. |
| **11** | multiply() | This method will return a BigDecimal whose value is (value × multiplicand). |
| **12** | plus() | This method will return a BigDecimal whose value is (+value). |
| **13** | pow() | This method will return a BigDecimal whose value is equal to the BigDecimal to power any value. |
| **14** | remainder() | This method will return a BigDecimal whose value is (value % divisor). |
| **15** | scale() | This method will return the scale of this BigDecimal value. |
| **16** | signum() | This method will return the signum function of this BigDecimal value. |
| **17** | Subtract() | This method will return a BigDecimal whose value is (value – subtrahend). |
| **18** | toBigInteger() | This method is usful for converting the BigDecimal to a BigInteger. |
| **19** | toEngineeringString() | This method will return a string representation of the specified BigDecimal, by using engineering notation if an exponent is needed. |
| **20** | valueOf() | This method is useful for translating a double into a BigDecimal value. |

# Example 1 : A basic JAVA code using BigDecimal

// This code is for adding two BigDecimal numbers in JAVA

import java.math.BigDecimal;

public class BDE

{

    public static void main(String[] args)

    {

// convert string to BigDecimal

        BigDecimal bd1 =  new BigDecimal("123456789");

        BigDecimal bd2 =  new BigDecimal("123456789");

        BigDecimal bd3 ,bd4, bd5;

        bd3=bd1.add(bd2);

        System.out.println("\n"+"Addition is : "+bd3);

    }

}

**Output**

C:\Users\Adarsh Shukla\Desktop>javac BDE.java

C:\Users\Adarsh Shukla\Desktop>java BDE

Addition is : 246913578

# Example 2 : JAVA program to check a BigDecimal number whether is Prime or not.

import java.math.BigDecimal;

import java.util.Scanner;

public class BigDecimal\_prime

{

    public static void main(String[] args)

    {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Please input a BigDecimal number to check");

        String n = scanner.nextLine();

        BigDecimal bd=new BigDecimal(n);

        int b=0;

        BigDecimal i=new BigDecimal("1");

        BigDecimal j=new BigDecimal("1");

        BigDecimal k=new BigDecimal("0");

        for(;i.compareTo(bd)!=0;i=i.add(j))

        {

            BigDecimal r=bd.remainder(i);

            if(r.compareTo(k)==0)

                b++;

        }

        if(b==1)

            System.out.println("Given number is prime");

        else

            System.out.println("Given number is not prime");

    }

}

**Output**

E:\Adarsh Shukla\CSE\_CSF\Sem 4\OOPs\Codes>javac BigDecimal\_prime.java

E:\Adarsh Shukla\CSE\_CSF\Sem 4\OOPs\Codes>java BigDecimal\_prime

Please input a BigDecimal number to check

123456789123456789111111111112222222222224444444444

Given number is prime