# Exception : Non-terminating decimal expansion

BigDecimal is used in financial application because accurate results is required.

Exception in thread “main” java.lang.ArithmeticException: Non-terminating decimal expansion; no exact representable decimal result.

This happens due to a couple of reasons:  
1. The BigDecimal by default always tries to return the exact result of an operation.  
2. Due to this, certain division operations like 1 divided by 3, the exact quotient will have an infinitely long decimal expansion. This will cause the division operation to fail and throw the error as described above.

This is also described in the BigDecimal Java Doc quoted below:  
*“In the case of divide, the exact quotient could have an infinitely long decimal expansion; for example, 1 divided by 3. If the quotient has a nonterminating decimal expansion and the operation is specified to return an exact result, an ArithmeticException is thrown. Otherwise, the exact result of the division is returned, as done for other operations.”*

To fix the issue, we need to provide a scale (i.e. the precision) for the quotient and also the rounding mode to the [BigDecimal.divide()](http://java.sun.com/j2se/1.5.0/docs/api/java/math/BigDecimal.html" \l "divide(java.math.BigDecimal, int, java.math.RoundingMode)). In the sample below, I have used a scale of 2 and the rounding mode as [RoundingMode.HALF\_UP](http://java.sun.com/j2se/1.5.0/docs/api/java/math/RoundingMode.html" \l "HALF_UP). You can increase the scale if you want a greater precision.

package com.jm.client;

import java.math.BigDecimal;  
import java.math.RoundingMode;

public class TestBigDecimal {

public static void main(String[] args) {

String returnVal = TestBigDecimal.divide("1", "5");  
System.out.println("Test #1: returnVal = " + returnVal);  
returnVal = TestBigDecimal.divide("1", "2");  
System.out.println("Test #2: returnVal = " + returnVal);  
// Test(#3) will now work as we have provided a scale  
// and a rounding mode to the divide() method  
returnVal = TestBigDecimal.divide("1", "3");  
System.out.println("Test #3: returnVal = " + returnVal);  
}

//Divide val1 by val2 and return the result as String.  
public static String divide(String val1, String val2) {

BigDecimal v1 = new BigDecimal(val1);

BigDecimal v2 = new BigDecimal(val2);  
return v1.divide(v2, 2, RoundingMode.HALF\_UP).toPlainString();  
}

}

The output of the above program now is as desired:

*Test #1: returnVal = 0.20  
Test #2: returnVal = 0.50  
Test #3: returnVal = 0.33*

# [Truncate a BigDecimal without rounding](http://stackoverflow.com/questions/24272849/how-to-truncate-a-bigdecimal-without-rounding)

Use either [RoundingMode.DOWN](http://docs.oracle.com/javase/7/docs/api/java/math/RoundingMode.html" \l "DOWN) or [RoundingMode.FLOOR](http://docs.oracle.com/javase/7/docs/api/java/math/RoundingMode.html" \l "FLOOR).

# JSRs

are Java Specification Requests, basically change requests for the Java language, libraries and other components.

It's all part of the Java Community Process, whereby interested parties can put forward their ideas for enhancements and (hopefully) have them taken up and acted upon. The process is detailed [here](http://jcp.org/en/procedures/overview).

For example, the Bluetooth one you mention is tracked [here](http://jcp.org/en/jsr/detail?id=82) and the definitive list is maintained [here](http://jcp.org/en/jsr/all).