Solution factorielle Big JUNIT

```
package fr.afpa.outil2;
import java.math.BigInteger;

import static org.junit.Assert.assertEquals;import static
org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;import static
org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.AfterEach;import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import fr.afpa.outil2.Math;

class MathTest {
@BeforeEach void setUp() throws Exception {}
@AfterEach void tearDown() throws Exception {}

@Test
public void testFactorielleBig1() {
        assertTrue("La factorielleBig de zero vaut 1"
                        ,
Math.factorielleBig(BigInteger.ZERO).equals(BigInteger.ONE));
}
@Test
public void testFactorielleBig2() {
        assertFalse("La factorielleBig de 1 vaut 1"

,!Math.factorielleBig(BigInteger.ONE).equals(BigInteger.ONE)) ;
}
@Test
public void testFactorielleBig3() {
        assertEquals("La factorielleBig de 3 vaut 6"
                        ,Math.factorielleBig(new BigInteger("3")),new
BigInteger("6")) ;
}
@Test
public void testFactorielleBig4() {
        assertTrue("La factorielleBig de 14 vaut 87178291200L"
                        , (Math.factorielleBig(new BigInteger("14"))).equals(new
BigInteger("87178291200")));
}
@Test
public void testFactorielleBig5() {
        assertTrue("La factorielleBig de 20 vaut 2432902008176640000L"
                        , Math.factorielleBig(new BigInteger("20")).equals(new
BigInteger( "2432902008176640000")));
}

@Test  // RAPPEL ,AVANT : La factorielle de 21 ne pouvait pas être calculée"
public void testFactorielleBig6() {
        assertTrue("La factorielleBig de 21 vaut XXXXXXXXXL"
                        , Math.factorielleBig(new BigInteger("21")).equals( new
BigInteger("51090942171709440000")));
}
```

```
//"La factorielleBig de -1 ne peut pas être calculée (hors limite)"
@Test
public void testFactorielleBig12() {
    assertThrows(IllegalArgumentException.class,
            ()->{Math.factorielleBig(new BigInteger("-1"));},
            "La factorielleBig de -1 ne peut pas être calculé (hors limite)" );
}

//"A FINIR, La factorielleBig de 77  peut  être calculée"
//      @Test
//      public void testFactorielleBig13() {
//               assertTrue("La factorielleBig de 77 vaut XXXXXXXXXL",
Math.factorielleBig(new BigInteger("77")).equals( new
BigInteger("1.451830920282858696340707840863le+113")));
//      }

//"La factorielleBig de 9000 ne peut pas être calculé (hors limite)"
@Test
public void testFactorielleBig14() {
    assertThrows(StackOverflowError.class,
            ()->{Math.factorielleBig(new BigInteger("9000"));},
            "La factorielleBig de 9000 ne peut pas être calculé (hors limite)"
);
}
}
```