JAVA.LANG.STRICTMATH.NEXTAFTER METHOD

Description

The **java.lang.StrictMath.nextAfter*doublestart, doubledirection*** method returns the floating-point number adjacent to the first argument in the direction of the second argument. If both arguments compare as equal the second argument is returned.It include these cases:

- If either argument is a NaN, then NaN is returned.
- If both arguments are signed zeros, *direction* is returned unchanged.
- If start is ±Double.MIN_VALUE and direction has a value such that the result should have a smaller magnitude, then a zero with the same sign as start is returned.
- If start is infinite and direction has a value such that the result should have a smaller magnitude, Double.MAX_VALUE with the same sign as start is returned.
- If start is equal to ±Double.MAX_VALUE and direction has a value such that the result should have a larger magnitude, an infinity with same sign as start is returned.

Declaration

Following is the declaration for **java.lang.StrictMath.nextAfter** method

```
public static double nextAfter(double start, double direction)
```

Parameters

- **start** -- This is the starting floating-point value
- **direction** -- This is the value indicating which of start's neighbors or start should be returned

Return Value

This method returns the floating-point number adjacent to start in the direction of direction.

Exception

- **NA**

Example

The following example shows the usage of java.lang.StrictMath.nextAfter method.

```
package com.tutorialspoint;

import java.lang.*;

public class StrictMathDemo {

   public static void main(String[] args) {

   double d1 = 102.2d, d2 = 0.0d;

   /* returns the floating-point number adjacent to the first argument in the
   direction of the second argument */

   double retval = StrictMath.nextAfter(d1, 9.2d);
   System.out.println("NextAfter = " + retval);

   /* returns the floating-point number adjacent to the first argument in the
   direction of the second argument */
   retval = StrictMath.nextAfter(d2, 9.2d);
```

```
    System.out.println("NextAfter = " + retval);

    // returns 0 if both arguments is zero
    retval = StrictMath.nextAfter(d2, 0.0d);
    System.out.println("NextAfter = " + retval);
  }
}
```

Let us compile and run the above program, this will produce the following result:

```
NextAfter = 102.19999999999999
NextAfter = 4.9E-324
NextAfter = 0.0
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js