

# NumberParser (Java)

DOI [10.5281/zenodo.1028916](https://doi.org/10.5281/zenodo.1028916)

[Last release](#) -- [Test program](#)

[Main page](#) ([versión en español](#))

## Introduction

The *NumberParser* package provides a common framework to deal with all the Java numeric types. It relies on the following four classes (NumberX):

- *Number* only supports the *double* type.
- *NumberD* can support any numeric type via *Object*.
- *NumberO* can support different numeric types simultaneously.
- *NumberP* can parse numbers from strings.

```
//1.23 (double).
Number number = new Number(1.23);

//123 (int).
NumberD numberD = new NumberD(123);

//1.23 (double). Others: 1 (int) and '?' (char).
NumberO numberO = new NumberO
(
    1.23, new ArrayList()
    {{
        add(NumericTypes.Integer);
        add(NumericTypes.Character);
    }}
);

//1 (long).
NumberP numberP = new NumberP
(
    "1.23", new ParseConfig(NumericTypes.Long)
);
```

## Common Features

All the NumberX classes have various characteristics in common.

- Defined according to the fields *getValue()* (*double* or *Object*) and *getBaseTenExponent()* (*int*). All of them support ranges beyond  $[-1, 1] \cdot 10^{2147483647}$ .
- Static (*NumberD.Addition(numberD1, numberD2)*) and non-static (*numberD1.greaterThan(numberD2)*) support for the main arithmetic and comparison operations.
- Errors managed internally and no exceptions thrown.

- Numerous instantiating alternatives. Implicitly convertible between each other and to related types.

```
//12.3*10^456 (double).
Number number = new Number(12.3, 456);

//123 (int).
NumberD numberD =
(
    new NumberD(123).lessThan(new NumberD(new Number(456))) ?
    //123 (int)
    new NumberD(123.456, NumericTypes.Integer) :
    //123.456 (double)
    new NumberD(123.456)
);

//Error (ErrorTypesNumber.InvalidOperation) provoked when dividing by zero.
NumberO numberO = NumberO.Division
(
    new NumberO
    (
        123.0, OtherTypes.IntegerTypes
    )
    , new NumberO(0)
);

//1.2340000000000e+308*10^5373 (double).
NumberP numberP = new NumberP("1234e5678");
```

## Math2 Class

This class includes all the NumberParser mathematical functionalities.

### Custom Functionalities

- *RoundExact/TruncateExact* can deal with multiple rounding/truncating scenarios not supported by the native methods.
- *GetPolynomialFit/ApplyPolynomialFit* allow to deal with second degree polynomial fits.
- *Factorial* calculates the factorial of any integer number up to 100000.

```
//123000 (double).
Number number = Math2.RoundExact
(
    new Number(123456.789), 3, RoundType.AlwaysToZero,
    RoundSeparator.BeforeDecimalSeparator
);

//30 (double).
NumberD numberD = Math2.ApplyPolynomialFit
(
    Math2.GetPolynomialFit
    (
        new NumberD[]
        {
```

```

        new NumberD(1), new NumberD(2), new NumberD(4)
    },
    new NumberD[]
    {
        new NumberD(10), new NumberD(20), new NumberD(40)
    }
)
, new NumberD(3)
);

//3628800 (int).
NumberD numberD = Math2.Factorial(new NumberD(10));

```

## Native Methods

*Math2* also includes *NumberD*-adapted versions of a big number of *Math* and *.NET System.Math* methods.

It also includes *PowDecimal*\ *SqrtDecimal* which allow to unrestrictedly use *NumberX* variables with *Math.pow*\ *Math.sqrt*. Note that this Java version doesn't rely on the original C# custom implementation (detailed explanations in [varocarbas.com Project 10](http://varocarbas.com/Project_10)) because of only making sense within the .NET conditions (i.e., high-precision *decimal* type not natively supported by the in-built methods).

```

//1.582502898380e+14 (double).
Number number = Math2.PowDecimal
(
    new Number(123.45), 6.789101112131415161718
);

//4.8158362157911885 (double).
NumberD numberD = Math2.Log(new NumberD(123.45));

```

## Further Code Samples

The [test application](#) includes a relevant number of descriptive code samples.

## Authorship & Copyright

I, Alvaro Carballo Garcia (varocarbas), am the sole author of each single bit of this code.

Equivalently to what happens with all my other online contributions, this code can be considered public domain. For more information about my copyright/authorship attribution ideas, visit the corresponding pages of my sites:

- <https://customsolvers.com/copyright/>  
ES: [https://customsolvers.com/copyright\\_es/](https://customsolvers.com/copyright_es/)
- <http://varocarbas.com/copyright/>  
ES: [http://varocarbas.com/copyright\\_es/](http://varocarbas.com/copyright_es/)