

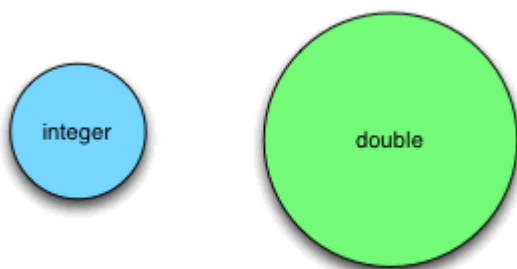
CURSO Java Herencia GRATIS APUNTATE!!

La mayor parte de las veces que trabajamos con números en Java nos es suficientes utilizar los tipos básicos como int y double que nos cubren la mayor parte de nuestras necesidades. De hecho el rango de un tipo double y un integer en Java es el siguiente:

integer: -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807.

double: 1.40129846432481707e-45 to 3.40282346638528860e+38

La realidad es que es suficiente para la “mayor” parte de las operaciones:



Manejando Factoriales (Java BigInteger)

Lamentablemente hay en algunas casuísticas que se nos puede quedar corto. Parece imposible pero a veces ocurre una de las situaciones más sencillas de reproducir es el calculo de un factorial. Supongamos que tenemos el siguiente código:

```

package com.arquitectura;

public class CalcularFactorial {

    public static void main(String[] args) {

        System.out.println(factorial(50));
    }

    static double factorial(double n)
    {
        if (n == 1.0)
            return 1.0;
        else
            return n*factorial(n-1);
    }
}

& amp; amp; amp; nbsp;

```

Este bloque de código nos calculará el factorial de 50 y el resultado será :3.0414093201713376E64 . Que es un número bastante grande. Ahora bien que sucede en el caso del factorial de 200 por ejemplo. La respuesta de Java será curiosa.

Infinity

Nos hemos salido de rango y no podemos calcular el factorial de este número. La solución va a venir de una clase Java especial reservada para esta tipología de operaciones y que se denomina BigInteger. Con ella podremos calcular el factorial de números mucho más grandes vamos a verlo:

```
package com.arquitectura;

import java.math.BigInteger;

public class CalcularFactorial2 {

    public static void main(String[] args) {

        System.out.println(factorial(new BigInteger("1000")));

    }

    static BigInteger factorial(BigInteger n)
    {
        if (n.equals(BigInteger.ZERO))
            return BigInteger.ONE;
        else
            return n.multiply(factorial(n.subtract(BigInteger.ONE)));
    }

}
```

En este caso hemos creado un objeto de tipo BigInteger y hemos usado los métodos que tiene (multiply) para realizar las operaciones que necesitamos. Una vez procesado el resultado será el siguiente:

Como podemos ver un número realmente grande. Así pues siempre podremos usar `int` o `double` pero para cosas que realmente se escapan de lo habitual clases como `BigInteger` o `BigDecimal` pueden venir muy bien ya que aportan el tamaño adecuado.



**CURSO JPA
GRATIS
APUNTATE!!**

StringBuilder

String y Rendimiento

foreach vs Iterator