

Ejercicio U6_B1_E1: Observa como netbeans me señala el siguiente código como obsoleto pues tacha los new Integer()

```
class App {
    public static void main(String[] args) {
        Integer z = new Integer(44);
        z=new Integer(z.intValue()+1);
        System.out.println(z);
    }
}
```



Para meter un int en un Integer hay dos formas equivalentes:

- `z=Integer.valueOf(z.intValue()+1);`
- `z= new Integer(z.intValue()+1);`

En el API nos indica que valueOf puede ser más eficiente

Ejercicio U6_B1_E2:

Son dos formas de pasar un String a un valor numérico real. La diferencia es simplemente que valueOf() devuelve un Double, es decir, un objeto y parseDouble() devuelve un double, es decir, un tipo primitivo.

<code>static double</code>	 devuelve tipo primitivo double	<code>parseDouble(String s)</code> Returns a new double initialized to the value represented by the string argument.
<code>short</code>		<code>shortValue()</code> Returns the value of this Double as a short (by casting to short).
<code>static String</code>		<code>toHexString(double d)</code> Returns a hexadecimal string representation of the double argument.
<code>String</code>		<code>toString()</code> Returns a string representation of this Double object.
<code>static String</code>		<code>toString(double d)</code> Returns a string representation of the double argument.
<code>static Double</code>		<code>valueOf(double d)</code> Returns a Double instance representing the specified double value.
<code>static Double</code>	 devuelve objeto Double	<code>valueOf(String s)</code> Returns a Double object holding the double value represented by the string argument.

Según el contexto usaremos uno u otro. Cuando la rapidez y eficiencia es importante, se usará parseDouble(), si lo que necesitamos por la lógica del programa es un objeto en lugar de un tipo primitivo, entonces valueOf(). En nuestros ejemplos sencillos, es indiferente usar uno u otro.

Ejercicio U6_B1_E3:

`d=Double.valueOf(s).doubleValue()` -> Si utilizo una variable intermedia
`Double x = Double.valueOf(s);`

```
d=x.doubleValue();
```

`d=Double.valueOf(s)` -> Si utilizo una variable intermedia

```
Double x = Double.valueOf(s);
```

```
d=x; //AQUÍ HAY AUTOBOXING (unboxing concretamente)
```

y observamos que ciertamente `d=x.doubleValue();` y `d=x;` son equivalentes por el mecanismo de autoboxing