¿Cómo podemos trabajar con números decimales en Java?

En JAVA hemos aprendido que un número decimal es un tipo de dato float o double en función del tamaño que queramos tener. Son tipos primitivos, que también existen como clases en Java: Float y Double.

Estas clases tienen una serie de carencias que son cubiertas por la clase java.math.BigDecimal

¿Cómo podemos crear un objeto que sea un BigDecimal?

Podemos usar el constructor de la clase, que crea un objeto a partir de un número o una cadena:

```
// Creamos un bigDecimal a partir de un número entero
BigDecimal x1 = new BigDecimal(7);
// Creamos un bigDecimal a partir de un número decimal
BigDecimal x2 = new BigDecimal(6.9);
// Creamos un bigDecimal a partir de una cadena
BigDecimal x3 = new BigDecimal("13.3");
```

Si queremos obtener un objeto BigDecimal que sea 0, 1 o 10, podemos usar una de las constantes que hay en la clase. Están porque se supone que son números muy utilizados, pero no es obligatorio usarlas:

```
BigDecimal cero = BigDecimal.ZERO;
BigDecimal uno = BigDecimal.ONE;
BigDecimal diez = BigDecimal.TEN;
```

¿Cómo podemos redondear un número?

Para acotar el número de decimales, utilizamos el método setScale(). Hay que indicar el número de decimales que queremos tener y qué método de redondeo queremos usar.

El método devuelve un nuevo BigDecimal redondeado. Veamos ejemplo:

```
BigDecimal numero = new BigDecimal(10.769842);
BigDecimal redondeado = numero.setScale(2, RoundingMode.HALF_DOWN);
```

Los métodos de redondeo los podemos consultar aquí:

https://docs.oracle.com/javase/8/docs/api/java/math/RoundingMode.html

Summary of Rounding Operations Under Different Rounding Modes

	Result of rounding input to one digit with the given rounding mode							
Input Number	UP	DOWN	CEILING	FLOOR	HALF_UP	HALF_DOWN	HALF_EVEN	UNNECESSARY
5.5	6	5	6	5	6	5	6	throw ArithmeticException
2.5	3	2	3	2	3	2	2	throw ArithmeticException
1.6	2	1	2	1	2	2	2	throw ArithmeticException
1.1	2	1	2	1	1	1	1	throw ArithmeticException
1.0	1	1	1	1	1	1	1	1
-1.0	-1	-1	-1	-1	-1	-1	-1	-1
-1.1	-2	-1	-1	-2	-1	-1	-1	throw ArithmeticException
-1.6	-2	-1	-1	-2	-2	-2	-2	throw ArithmeticException
-2.5	-3	-2	-2	-3	-3	-2	-2	throw ArithmeticException
-5.5	-6	-5	-5	-6	-6	-5	-6	throw ArithmeticException

¿Cómo podemos realizar operaciones?

La clase tiene distintos métodos que nos devuelven un nuevo BigDecimal resultado de aplicar operaciones. Las más comunes (sumar, restar, multiplicar y dividir) serían estas:

```
BigDecimal suma = x.add(y);
BigDecimal resta = x.subtract(y);
BigDecimal producto = x.multiply(y);
BigDecimal division = x.divide(y);
```

Si tengo que hacer varias operaciones, se puede llamar a los métodos de forma encadenada. En el siguiente ejemplo, estoy sumando Y a X, luego el resultado lo multiplico por X, y luego al resultado le sumo 10:

```
BigDecimal suma = x.add(y).multiply(x).add(BigDecimal.TEN);
```

IMPORTANTE: Cuando hago una división que puede tener muchos decimales es importante acotar la cantidad de decimales, o podemos tener un error. Se hace del siguiente modo (parecido a como redondeamos):

```
// Cuando llamamos al método divide, además del divisor,
// indicamos número de decimales y técnica de redondeo
BigDecimal division = x.divide(y, 2, RoundingMode.HALF_DOWN);
```

¿Cómo cambiamos de signos?

Métodos para hacer cambios de signos:

```
// Nos devuelve el mismo número en negativo
x = x.negate();
// Nos devuelve el mismo número en positivo
x = x.plus();
// Nos devuelve el mismo número en valor absoluto
x = x.abs();
```

¿Cómo comparamos dos BigDecimal?

Podemos compararlos igual que cualquier otro objeto:

- Método equals() para saber si son iguales
- Método compare() para saber cuál es mayor/menor

Aparte, también tenemos dos métodos más:

```
// Nos devuelve el mayor entre x e y
BigDecimal mayor = x.max(y);
// Nos devuelve el menor entre x e y
BigDecimal menor = x.min(y);
```

¿Cómo puedo obtener un tipo de dato de "toda la vida" a partir de un BigDecimal?

Si quiero convertir de BigDecimal a Double, Float, Integer, etc., puedo usar los métodos que la clase nos proporciona:

```
Integer entero = x.intValue();
Long enteroGrande = x.longValue();
Float decimal = x.floatValue();
Double decimalGrande = x.doubleValue();
```

¿Cómo puedo obtener un String de un BigDecimal?

El método toString() de la clase BigDecimal nos devuelve una representación por defecto del número. No se redondea nunca y se utiliza el punto como separador decimal. Por ejemplo:

```
BigDecimal x = new BigDecimal("123456.7");
System.out.println(x);
// Esto imprime: 123456.7
```

Si queremos obtener una cadena formateada de acuerdo con la representación estándar de nuestro país, podemos usar DecimalFormat. Esta clase nos permite crear un formateador a partir de un patrón (algo similar a lo que hacemos con las fechas). Veamos un ejemplo:

```
BigDecimal x = new BigDecimal("123456.7");
DecimalFormat formato = new DecimalFormat("#,###.00");
System.out.println(formato.format(x));
// Esto imprime: 123.456,70
```

Para construir el patrón, podemos usar:

- # → indica un dígito. Si no se puede rellenar, no aparecerá
- 0 → igual que el anterior, indica un dígito, pero si no se puede rellenar, se mostrará 0
- . → indica la posición del separador decimal
- , → indica la posición del separador de miles
- - → indica la posición del signo negativo

Puedo completar el patrón con lo que yo quiera. Por ejemplo:

```
BigDecimal x = new BigDecimal("123456.7");
DecimalFormat formato = new DecimalFormat("$ #,###.00");
System.out.println(formato.format(x));
// Esto imprime: $ 123.456,70
```