APRENDERAPROGRAMAR.COM

REDONDEAR A 2 O MÁS DECIMALES EN JAVA. PROBLEMA Y ERRORES DE PRECISIÓN DECIMAL Y REDONDEO. BIGINTEGER, BIGDECIMAL (CU00907C)

Sección: Cursos
Categoría: Lenguaje de programación Java nivel avanzado I

Fecha revisión: 2039

Resumen: Entrega nº7 del curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Manuel Sierra y José Luis Cuenca

CÓMO REDONDEAR A UN NÚMERO DE DECIMALES DESEADO

En la entrega anterior del curso hemos visto métodos de la clase Math útiles para redondeo como round, floor y ceil. Supongamos que partimos de un número con muchos decimales, por ejemplo 8.52522719 ¿Cómo podemos redondearlo al número más próximo con sólo cierto número de decimales? Para esto trabajaremos con dos conceptos: parte entera del número y parte decimal del número.

Separaremos ambas partes para trabajar con ellas y después de obtenidos los resultados volveremos a unir ambas partes. Vamos a explicar cómo proceder usando un ejemplo.

Queremos redondear 8.52522719 a dos decimales. Lo primero que haremos es quedarnos con la parte entera del número usando floor. Para 8.5252719 nos habremos quedado con 8.

Al número inicial le restamos la parte entera obtenida de modo que nos quedamos sólo los decimales (en este ejemplo 0.5252719). Esto lo multiplicamos por 10 tantas veces como decimales queramos obtener. En este ejemplo queremos obtener dos decimales por lo que multiplicamos por 100 obteniendo 52.52719. Ahora tenemos un valor donde la parte decimal que nos interesa (2 cifras) está expresada como un entero. Finalmente aplicamos round para obtener la fracción decimal redondeada al número de cifras que nos interesa. En este caso al aplicar round nos quedamos con 53. Este valor es la parte decimal redondeada.

Finalmente volvemos a convertir a decimal. Para ello a la parte entera inicial que era 8, le sumamos la parte decimal redondeada. En este caso a 8 le sumamos 53/100 obteniendo 8.53 que es el valor inicial redondeado a dos decimales.

En este código mostramos un método que permite recibir un double y redondearlo al número de decimales deseado. Escribe el código y comprueba los resultados.

```
/* Curso java avanzado aprenderaprogramar.com*/
public class RedondeoDecimales {

   public static void main(String[] args) {
      double numero=8.5252719;
      System.out.print("El numero 8.5252719 con 2 decimales queda "+ redondearDecimales(numero, 2));
      System.out.println(". Con 4 decimales queda "+redondearDecimales(numero, 4)+
" y con 6 decimales queda "+redondearDecimales(numero, 6));
      numero=12.5552917;
      System.out.print("El numero 12.5552917 con 2 decimales queda "+ redondearDecimales(numero, 2));
      System.out.println(". Con 4 decimales queda "+redondearDecimales(numero, 4)+
" y con 6 decimales queda "+redondearDecimales(numero, 6));
      numero= -12.5566112345;
      System.out.print("El numero -12.5566112345 con 2 decimales queda "+ redondearDecimales(numero, 2));
```

```
        System.out.println(". Con 4 decimales queda "+redondearDecimales(numero, 4)+
 " y con 6 decimales queda "+redondearDecimales(numero, 6));
    }

    public static double redondearDecimales(double valorInicial, int numeroDecimales) {
        double parteEntera, resultado;
        resultado = valorInicial;
        parteEntera = Math.floor(resultado);
        resultado=(resultado-parteEntera)*Math.pow(10, numeroDecimales);
        resultado=Math.round(resultado);
        resultado=(resultado/Math.pow(10, numeroDecimales))+parteEntera;
        return resultado;
    }
}
```

El resultado esperado es:

El numero 8.5252719 con 2 decimales queda 8.53. Con 4 decimales queda 8.5253 y con 6 decimales 8.525272
El numero 12.5552917 con 2 decimales queda 12.56. Con 4 decimales 12.5553 y con 6 decimales 12.555292
El numero -12.5566112345 con 2 decimales -12.56. Con 4 decimales -12.5566 y con 6 decimales -12.556611

Java provee una clase (BigDecimal) que permite trabajar con alta precisión y especificar distintos modos de redondeo. La explicamos más abajo.

EL PROBLEMA DE LA PRECISIÓN DECIMAL EN LOS CÁLCULOS

Muchas veces esperamos un resultado exacto como 0.5 y sin embargo obtenemos un resultado como 0.499999999. Este problema no sólo afecta a Java, sino a prácticamente todos los lenguajes.

Veamos un ejemplo. Escribe este código y comprueba los resultados:

```
/* Curso java avanzado aprenderaprogramar.com*/
public class multiplicadorDieces1{
    public static void main(String[] args) {
        int resultado = 1;
        for(int i=1; i<=5; i++){ resultado = resultado * 10; }
        System.out.print(2.55*resultado);
    }
}
```

El resultado lógico sería 255000. Sin embargo el programa nos devuelve 254999.99999999997

La razón por la que no está devolviendo el resultado correcto está relacionado con la forma de representar los valores numéricos decimales que tienen los computadores. Se suele pensar que usan números decimales como nosotros pero internamente no funcionan así ya que trabajan en binario.

En este caso nos encontramos con un problema que se presenta con cierta frecuencia: la precisión cuando se trabaja con decimales. Es un problema habitual y hay diferentes maneras de afrontarlo. Vamos a comentar algunas de ellas.

CORRECCIÓN DE LA PRECISIÓN DECIMAL CON MÉTODOS PROPIOS

Podemos desarrollar métodos para solucionar, al menos en el programa que estemos desarrollando, los problemas de precisión decimal. Veamos un ejemplo de ello.

Podemos multiplicar el resultado por 10^x para obtener un entero equivalente que incluya x decimales de interés, y a continuación redondearlo con round. De este modo habremos eliminado toda la parte decimal que no resulta de interés y al mismo tiempo obtenido el valor preciso gracias al redondeo. Dividiendo a continuación por 10^x podemos obtener de nuevo el número exacto.

Ejemplos:

Obtenido	Deseado	Multiplicar	Redondear	Dividir	Resultado
0.034999999999999996	0.035 (3 decimales)	$*10^3$	35	$/10^3$	0.035
254999.99999999997	255000 (0 decimales)	$*10^0$	255000	$/10^0$	255000
0.49999999999999994	0.5 (1 decimal)	$*10^1$	5	$/10^1$	0.5

Escribe este código y comprueba sus resultados:

```
/* Curso java avanzado I aprenderaprogramar.com*/
public class TestPrecision{

   public static void main(String[] args) {
      int angulo = 30;
      double b = Math.toRadians(angulo);
      double seno30=Math.sin(b);
      System.out.println("El seno de 30 sin fijar es " +seno30);
      System.out.println("El seno de 30 fijado es " +fijarNumero(seno30, 1));
      System.out.println("Inicial 0.034999999999999996 fijado es " +fijarNumero(0.034999999999999996, 3));
      System.out.println("Inicial 254999.99999999997 fijado es " +fijarNumero(254999.99999999997, 0));
      System.out.println("Inicial 0.333333333333333 fijado es " +fijarNumero(0.333333333333333, 3));
      System.out.println("Inicial 0.123456789123459999999 fijado es " +fijarNumero(0.123456789123459999999, 14));
   }

   public static double fijarNumero(double numero, int digitos) {
      double resultado;
      resultado = numero * Math.pow(10, digitos);
      resultado = Math.round(resultado);
      resultado = resultado/Math.pow(10, digitos);
      return resultado;
   }
}
```

El resultado esperado es:

```
El seno de 30 sin fijar es 0.49999999999999994
El seno de 30 fijado es 0.5
```

```
Inicial 0.034999999999999996 fijado es 0.035
Inicial 254999.99999999997 fijado es 255000.0
Inicial 0.333333333333333 fijado es 0.333
Inicial 0.12345678912345999999999 fijado es 0.12345678912346
```

Esta forma de trabajar resulta quizás un tanto engorrosa. Por ejemplo podemos no conocer la precisión deseada a priori, o nos puede resultar molesto estar fijando los resultados, o podemos olvidarnos de hacerlo.

CORRECCIÓN DE LA PRECISIÓN DECIMAL CON LA CLASE BIGDECIMAL

La clase java.math.BigDecimal es una clase del api Java para representar números decimales con alta precisión. La clase provee métodos para operaciones aritméticas, redondeos, comparaciones y otras tareas. Esta clase permite controlar de la forma que se desee cómo se realizarán los redondeos. La siguiente tabla resume aspectos relevantes de esta clase:

Aspecto	Ejemplo	Comentarios
Importar la clase	import java.math.BigDecimal;	Importar la clase para hacer uso de ella.
Constructor	a = new BigDecimal ("1.5834186");	Crea una instancia de objeto. Aunque se admite no incluir el número entre comillas, se recomienda hacerlo.
Constructor	Bigdecimal a = BigDecimal .valueOf(numeroDecimal)	Forma alternativa, usar si se quiere convertir un double o float a BigDecimal
Suma entre objetos BigDecimal	a=a.add(b);	Operación suma
Resta entre objetos BigDecimal	a=a.subtract(b);	Operación resta
Multiplicación entre objetos BigDecimal	a=a.multiply(b);	Operación multiplicación
División entre objetos BigDecimal	a=a.divide(b);	Operación división
Importar clase auxiliar para fijar modo de redondeo	import java.math.RoundingMode;	Para definir el modo de redondeo
Establecer número de dígitos de precisión (cantidad decimales) y forma de redondeo	a = a.setScale(1, RoundingMode.DOWN);	En el ejemplo a pasa a valer 1.5 porque se toma 1 decimal y redondeo hacia el cero.

Si no se especifica una forma de redondeo, se trabaja con todos los decimales posibles.

La clase RoundingMode facilita estos modos de redondeo: CEILING, DOWN, FLOOR, HALF_DOWN, HALF_EVEN, HALF_UP, UNNECESSARY, UP

Un aspecto a tener en cuenta es que si creamos un BigDecimal a partir de un valor double o float, podemos trasladar la imprecisión de estos valores al BigDecimal. Por ejemplo el código:

```
double a = 1.0/3.0;
BigDecimal bd = new BigDecimal(a);
System.out.println ("Representación: "+bd);
```

Nos devuelve 0.33333333333333331482961625624739099293947219848632812 5. Esto ocurre porque en esta situación se trata de representar el valor double con tanta precisión como resulta posible.

Para evitar esto se recomienda usar valueOf:

```
BigDecimal bd = BigDecimal.valueOf(1.0/3.0);
System.out.println ("Representación: "+bd);
```

valueOf construye el número BigDecimal a partir de su representación en forma de String según es devuelta por el método toString. De este modo no se requiere tanta precisión como sea posible.

Ejemplo: vamos a conseguir un resultado preciso para el código visto anteriormente.

```
/* Curso java avanzado aprenderaprogramar.com*/
public class multiplicadorDieces1{
   public static void main(String[] args) {
      int resultado = 1;
      for(int i=1; i<=5; i++){ resultado = resultado * 10; }
      System.out.print(2.55*resultado);
   }
}
```

El resultado lógico sería 255000. Sin embargo este código nos devuelve 254999.99999999997

Usaremos ahora BigDecimal para obtener el resultado preciso:

```
/* Curso java avanzado aprenderaprogramar.com*/
import java.math.BigDecimal;
import java.math.RoundingMode;
public class multiplicadorDieces2{
   public static void main(String[] args) {
      int multiplicador=1;
      for(int i=1; i<=5; i++){ multiplicador = multiplicador*10;}
      BigDecimal resultado =
BigDecimal.valueOf(2.55).multiply(BigDecimal.valueOf(multiplicador));
      resultado = resultado.setScale(0, RoundingMode.HALF_DOWN);
      System.out.print("Resultado: "+resultado);
   }
}
```

CLASE BIGINTEGER

La clase java.math.BigInteger es una clase del api Java para representar números enteros muy grandes. La clase provee métodos para operaciones aritméticas entre otros. Los constructores y métodos para BigInteger

La mayor parte de las veces que trabajamos con números en Java nos es suficiente utilizar los tipos primtivos como int y double. Sin embargo en ciertos casos se nos puede quedar corto el rango de estos tipos. Supongamos que tenemos el siguiente código:

```
/* Curso java avanzado aprenderaprogramar.com*/
public class CalcularFactorial {
   public static void main(String[] args) {   System.out.println(factorial(200));    }

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

La respuesta de Java al ejecutarlo será curiosa: Infinity. El problema subyacente es que los computadores no pueden manejar cualquier número infinitamente grande. 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 ya que se trata de una clase especialmente preparada para ello. Veamos un ejemplo. Escribe y ejecuta este código:

```
/* Curso java avanzado aprenderaprogramar.com*/
import java.math.BigInteger;
public class CalcularFactorial2 {
   public static void main(String[] args) {
      System.out.println(factorial(new BigInteger("200")));
   }

   public 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 para realizar las operaciones que necesitamos. Una vez procesado el resultado será el siguiente:

```
788657867364790503552363213932185062295135977687173263294742533244359449963403342920304284011984623904177212138919638830257642790242637105061926624952829931113462857270763317237396988943922445621451664240254033291864131227428294853277524242407573903240321257405579568660226031904170324062351700858796178922222789623703897374720000000000000000000000000000000000000000000000000000
```

EJERCICIO

Crea un programa para el cálculo del importe final a partir del importe inicial y un porcentaje de impuestos. El importe inicial y el porcentaje de impuestos deben solicitarse al usuario. El programa debe mostrar el resultado ateniéndose a estas reglas: debe mostrarse el impuesto como resultado de calcular el importe inicial por el porcentaje dividido entre 100, con precisión de dos decimales y redondeo al entero más próximo o en caso de equidistancia, redondeo al entero mayor. Debe mostrarse el importe final como resultado de sumar el importe inicial con el impuesto, por tanto tendrá precisión de dos decimales.

Ejemplo de resultados a obtener:

Por favor introduzca el importe inicial: 0.70
Por favor introduzca el porcentaje de impuestos: 5
El impuesto a pagar es: 0.04
El importe final es: 0.74

Para comprobar si tu solución es correcta puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00908C

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=58&Itemid=180