APRENDERAPROGRAMAR.COM

CLASE MATH API JAVA Y FUNCIONES MATEMÁTICAS (TRIGONOMÉTRICAS, LOGARÍTMICAS, ETC.) ROUND, FLOOR, CEIL. (CU00906C)

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

Fecha revisión: 2039

Resumen: Entrega nº6 del curso "Lenguaje de programación Java Nivel Avanzado I".

Autor: Manuel Sierra y José Luis Cuenca

FUNCIONES CLASE MATH JAVA

En cuanto a las funciones matemáticas en Java, las funciones disponibles vienen definidas en la clase Math. Hay muchas funciones disponibles. Se puede consultar la lista completa en la documentación oficial del API de Java (según versión de Java en uso, por ejemplo para la versión 8 ver http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html).

A continuación, mostraremos las funciones más importantes y ejemplos de uso:

Funciones Matemáticas	Significado	Ejemplo de uso	Resultado
abs	Valor absoluto	int x = Math.abs(-2);	x = 2;
atan	Arcotangente	double x = Math.atan(1);	x = 0.78539816339744;
sin	Seno	double x = Math.sin(0.5);	x = 0.4794255386042;
cos	Coseno	double x = Math.cos(0.5);	x = 0.87758256189037;
tan	Tangente	double x = Math.tan(0.5);	x = 0.54630248984379;
exp	Exponenciación neperiana	double x = Math.exp(1);	x = 2.71828182845904;
log	Logaritmo neperiano	double x = Math.log(2.7172);	x = 0.99960193833500;
pow	Potencia	double x = Math.pow(2,3);	x = 8.0;
round	Redondeo	double x = Math.round(2.5);	x = 3;
random	Número aleatorio	double x = Math.ramdom();	x = 0.20614522323378;
floor	Devuelve el entero menor	double x = Math.floor(2.5);	x = 2.0;
ceil	Devuelve el entero mayor	double x = Math.ceil(2.5);	x = 3.0

Destacar que las funciones matemáticas, al pertenecer a la clase Math, se invocan siempre de la siguiente manera: Math.funcion(argumentos).

Las funciones relacionadas con ángulos (**atan, cos, sin, tan, etc.)** trabajan en radianes. Por tanto, para operar con grados, tendremos que realizar la conversión oportuna. La propia clase Math facilita los

métodos toRadians para transformar grados sexagesimales en radianes y toDegrees para transformar radianes en grados sexagesimales, aunque las conversiones pueden no ser totalmente precisas. Por ejemplo cos(toRadians(90.0)) debería devolver 0, pero es probable que devuelva un valor aproximadamente cero pero no exactamente cero debido a que la precisión decimal no es absoluta.

La función ***random***,permite generar números aleatorios en el rango]0,1[. Por tanto el 0 y el 1 están excluidos.

La función exponenciación neperiana o exponenciación de e, matemáticamente significa $\mathbf{e^x}$, que en Java sería **Math.exp(x)**,donde **x** es un número real y la base es la constante neperiana **e** = 2.7172...

La función logaritmo neperiano, matemáticamente significa **Ln x**, que en Java correspondería a la expresión **Math.log(x).**

La función potencia, matemáticamente significa $\mathbf{base^{exponente}}$**,** que en Java se convertiría en **Math.pow(base,exponente),**donde base y exponente son números reales, por lo tanto, si queremos obtener la raíz cubica de 2,la instrucción sería Math.pow(2,0.333).

No hay una función directa para obtener la parte entera de un número real, pero para estos casos, se puede obtener de la siguiente manera:

int x = (int)(8.7); --> x = 8;

int x = (int)(-8.7); --> x = -8;

Aclarar que obtener la parte entera es distinto a redondear.

Si vas a trabajar con constantes físicas o matemáticas, te resultará de interés la instrucción final para la declaración de constantes. La ventaja de declarar una constante en vez de una variable, consiste en que la constante no puede variar en el transcurso del programa. Por tanto, se impide que por error pueda tener un valor no válido en un momento dado. Las constantes facilitan la documentación del programa y lo hacen fácil de modificar. Una declaración tipo de constante podría ser la siguiente: final double pi = 3.14159265358979;

Sin embargo, el propio Java tiene una constante propia para definir la constante matemática PI:

Math.PI;

El siguiente programa, es un ejemplo uso de la constante del número PI en la conversión de un angulo sexagesimal a radianes.

```
/* Ejemplo de clase java usando la constante PI de la clase Math – aprenderaprogramar.com */
public class Programa {
    public static void main(String args[]) {
        double sexagesimal = 30;
        double radianes = Math.PI/180 * sexagesimal;
        System.out.println("Angulo en radianes : "+radianes);
    }
}
```

DIFERENCIA ENTRE ROUND, CEIL Y FLOOR

Las funciones round, ceil y floor se usan para obtener un entero próximo a un número decimal y tienen similitudes, de hecho en algunos casos devuelven el mismo resultado. Sin embargo también tienen diferencias que es interesante conocer. Estas tres funciones se aplican sobre valores numéricos decimales y retornan un valor numérico que en el caso de round es un entero long, mientras que en el caso de floor y ceil retornan un valor de tipo double coincidente o equivalente con un entero.

El método round redondea siempre al entero más próximo, por ejemplo 2.6 redondea a 3 mientras que -2.6 redondea a -3. Si el decimal está exactamente entre dos valores se redondea al entero superior más próximo (por ejemplo 2.5 redondea a 3 y -2.5 redondea a -2).

El método floor diremos que devuelve el entero menor, por ejemplo 2.9 quedaría en 2.0 y -2.9 quedaría en -3.0. También 2.1 quedaría en 2.0 y -2.1 quedaría en -3.0.

El método ceil diremos que devuelve el entero mayor, por ejemplo 2.9 quedaría en 3.0 y -2.9 quedaría en -2.0. También 2.1 quedaría a 3.0 y -2.1 quedaría en -2.0.

En cada programa deberemos determinar qué método es el adecuado para obtener los resultados deseados. Por ejemplo, si tenemos que redondear cantidades de dinero parece más lógico utilizar round. En cambio, si estamos trabajando con edades de una persona en años utilizando decimales, podremos razonar de otra manera. La edad de una persona es un valor positivo (no es posible que tome valores negativos). Una persona decimos que tiene x años mientras no cumpla x+1 años, de modo que en todo el periodo intermedio decimos que tiene x años. Por ejemplo, si cumplo 35 años el 4 de febrero de 2096, desde el 4 de febrero de 2096 hasta el 3 de febrero de 2097 diré que tengo 35 años. Pero en un programa que trabaje con decimales, en el punto intermedio entre estas dos fechas tendría 35.50 años. Si quiero obtener el valor que representa la edad a partir del valor decimal, lo lógico será utilizar el método floor porque nos devolverá 35 para todos los valores decimales entre 35 y 36, tal y como expresamos en el lenguaje natural las edades. En este caso tanto round como ceil no ofrecerían un resultado adecuado a nuestros intereses.

Escribe este código y comprueba sus resultados:

```
/* Curso java avanzado I aprenderaprogramar.com*/
public class Redondeo {
   double x=0;
   public static void main(String[] args) {

      System.out.println("x = Math.round(2.6);  dará como resultado : "+ Math.round(2.6));
      System.out.println("x = Math.round(-2.6); dará como resultado : "+ Math.round(-2.6));
      System.out.println("x = Math.round(2.4);  dará como resultado : "+ Math.round(2.4));
      System.out.println("x = Math.round(-2.4); dará como resultado : "+ Math.round(-2.4));
      System.out.println("x = Math.round(2.5);  dará como resultado : "+ Math.round(2.5));
      System.out.println("x = Math.round(-2.5); dará como resultado : "+ Math.round(-2.5)+"\n");

      System.out.println("x = Math.ceil(2.6);   dará como resultado : "+ Math.ceil(2.6));
      System.out.println("x = Math.ceil(-2.6);  dará como resultado : "+ Math.ceil(-2.6));
      System.out.println("x = Math.ceil(2.4);   dará como resultado : "+ Math.ceil(2.4));
      System.out.println("x = Math.ceil(-2.4);  dará como resultado : "+ Math.ceil(-2.4));
      System.out.println("x = Math.ceil(2.5);   dará como resultado : "+ Math.ceil(2.5));
      System.out.println("x = Math.ceil(-2.5);  dará como resultado : "+ Math.ceil(-2.5)+"'\n");
```

```
        System.out.println("x = Math.floor(2.6);  dará como resultado : "+ Math.floor(2.6));
        System.out.println("x = Math.floor(-2.6); dará como resultado : "+ Math.floor(-2.6));
        System.out.println("x = Math.floor(2.4);  dará como resultado : "+ Math.floor(2.4));
        System.out.println("x = Math.floor(-2.4); dará como resultado : "+ Math.floor(-2.4));
        System.out.println("x = Math.floor(2.5);  dará como resultado : "+ Math.floor(2.5));
        System.out.println("x = Math.floor(-2.5); dará como resultado : "+ Math.floor(-2.5));
    }
}
```

Los resultados que nos devuelve este programa los resumimos en la siguiente tabla:

Valor inicial	2.6	-2.6	2.4	-2.4	2.5	-2.5
Resultado con round	3	-3	2	-2	3	-2
Resultado con ceil	3.0	-2.0	3.0	-2.0	3.0	-2.0
Resultado con floor	2.0	-3.0	2.0	-3.0	2.0	-3.0

EJERCICIO

Supón que un partido de fútbol tiene una duración de 90 minutos. El minuto 1 se considera que abarca desde los 0 segundos hasta los 59 segundos. El minuto 2 abarca desde los 60 segundos hasta los 119 segundos. Así sucesivamente hasta el último minuto, que es el minuto 90 y abarca desde los 5340 segundos hasta los 5400 segundos. Crea un programa que pida al usuario el número de segundos transcurridos y mediante el uso de alguna función de redondeo (floor, ceil ó round) de la clase Math muestre el minuto en que nos encontramos. El programa debe pedir valores de segundos hasta que el usuario decida terminar. Ejemplo de ejecución:

Introduzca valor de segundos: 32
El minuto es el 1
¿Otro valor (s/n)? s
Introduzca valor de segundos: 2595
El minuto es el 44
¿Otro valor (s/n)? n

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

Próxima entrega: CU00907C

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