APRENDERAPROGRAMAR.COM

IMPORTAR Y USAR CLASES DEL API DE JAVA. EJEMPLO CLASE MATH Y MÉTODO POW. CONSTRUCTORES PRIVADOS. (CU00647B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

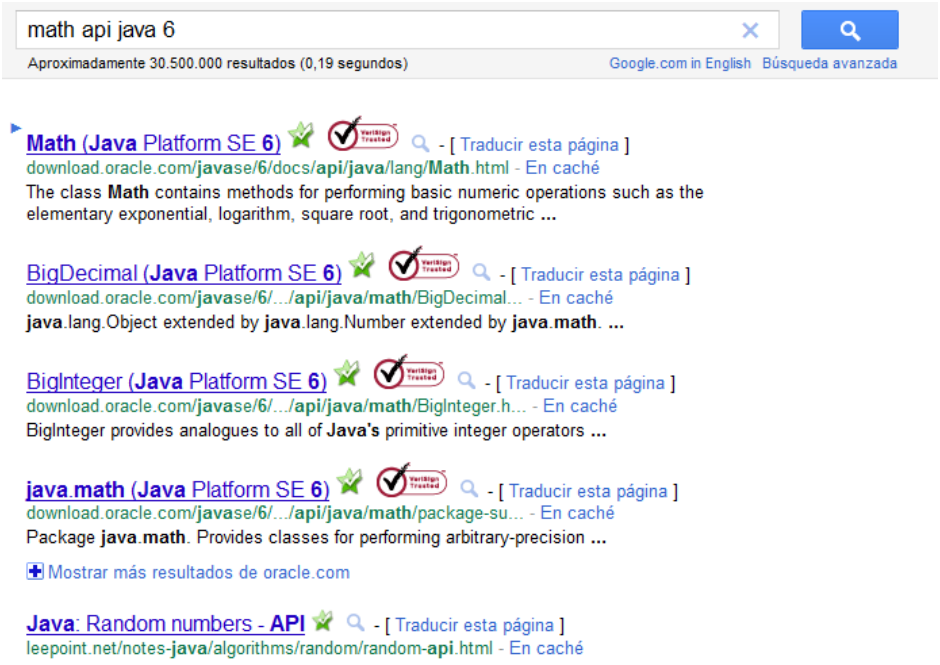
Fecha revisión: 2029

Resumen: Entrega nº47 curso Aprender programación Java desde cero.

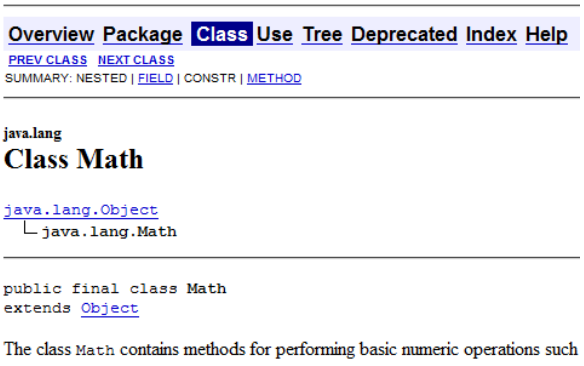
Autor: Alex Rodríguez

IMPORTAR Y USAR CLASES DEL API DE JAVA. EJEMPLO CON LA CLASE MATH.

En el API de Java no solo existen clases. Hay bastantes más cosas como clases abstractas o interfaces, de lo que hablaremos más adelante. De momento vamos a centrarnos en las clases y para ello hemos de prestar atención a aquello que nos aparece en el encabezado de la documentación que consultemos. Realiza en internet una búsqueda con el texto "math api java **X**" donde X será la versión de java que estemos usando.



Nos pueden aparecer diferentes resultados ya que **Math en Java puede hacer referencia a distintas cosas**, por ejemplo al paquete java.math o a la clase Math. Buscaremos el correspondiente a la clase Math y accedemos a él.



Fíjate que en la cabecera de la documentación se indica "Class Math". Si nos indicara otra cosa como Interface Math, Package Math, o cualquier otro texto, no nos encontraríamos frente a la documentación de la clase. Fíjate también en la ruta de la clase dentro del API de Java que aparece expresada como java.lang.Math. Es decir, la clase Math pertenece al paquete java.lang del API de Java. Vamos a buscar ahora el apartado "Method Summary" y dentro de éste el apartado *pow*.

`static double`	**pow**`(double a, double b)` Returns the value of the first argument raised to the power of the second argument.

Aquí nos vamos a fijar en la columna izquierda, que nos indica el tipo devuelto y que resulta ser *double*. En la columna derecha se indica la signatura del método: el método se llama *pow* y requiere dos valores numéricos de tipo *double*. El texto explicativo nos indica que el método devuelve el valor del primer parámetro elevado a la potencia indicada por el segundo parámetro. Escribe e intenta compilar el siguiente código (no nos va a ser posible el compilado):

```
import java.lang.Math; //Importamos la clase Math de la biblioteca del API Java

//Clase que permite elevar un número m a otro número n y obtener un resultado
public class ExponenciadorApiJava {

   //Constructor
   public ExponenciadorApiJava () { //Nada que declarar
}

   public int potenciaApiJava (int m, int n) {
      double a = new Math();
      return a.pow (m, n);
   } //Cierre del método
} //Cierre de la clase
```

En la primera línea de código, antes de la declaración de la clase, hemos incluido una sentencia import y una ruta del API de Java. Con este tipo de sentencias lo que hacemos es indicarle al compilador que para la ejecución del programa cargue en memoria el código contenido en la ruta indicada. Esto es lo que nos permite usar clases del API de Java.

Al tratar de compilar nos salta un error de tipo "Math() has private access in java.lang.Math". Lo cual nos quiere decir que el constructor de la clase Math tiene acceso privado en la clase Math. Hasta ahora hemos visto cómo los constructores los declarábamos siempre de acceso público, pero **en ciertas ocasiones se declaran constructores privados,** lo que significa que no están accesibles y no podemos hacer invocaciones usando la sentencia *new*. Si revisas la documentación del API de la clase Math, nos encontramos con dos tablas: *Field Summary* (campos o atributos de la clase) y *Method Summary* (métodos de la clase). En muchas otras clases nos encontramos con otra tabla denominada *Constructor Summary* que nos indica cómo crear objetos de la clase. Sin embargo, en la clase Math esa tabla falta. El motivo para ello es que dentro del API de Java algunas clases tienen un comportamiento especial, y la

clase Math es una de ellas. Es una clase en la que la gestión de objetos queda a cargo de Java y a nosotros únicamente se nos permite invocar métodos. La sintaxis que define Java en estos casos es del tipo:

```
nombreDeLaClase.nombreDelMétodo (parámetros requeridos);
```

Por ejemplo: *Math.pow (m, n);*

¿Contradice esto el principio de que los métodos se invocan sobre objetos? Podemos responder que sí y que no. Sí porque efectivamente estamos invocando el método usando el nombre de una clase. Y no porque Java, en segundo plano, está utilizando un objeto que crea automáticamente para realizar la gestión de esta invocación. Por tanto la clase Math define un tipo, pero nosotros no vamos a poder crear objetos de ese tipo, de ello se encargará Java. Nosotros veremos la clase Math principalmente como un paquete de código que nos permitirá realizar operaciones matemáticas. Que la clase Math defina un tipo será algo secundario para nosotros. Hablaremos de estos métodos, denominados estáticos, más adelante.

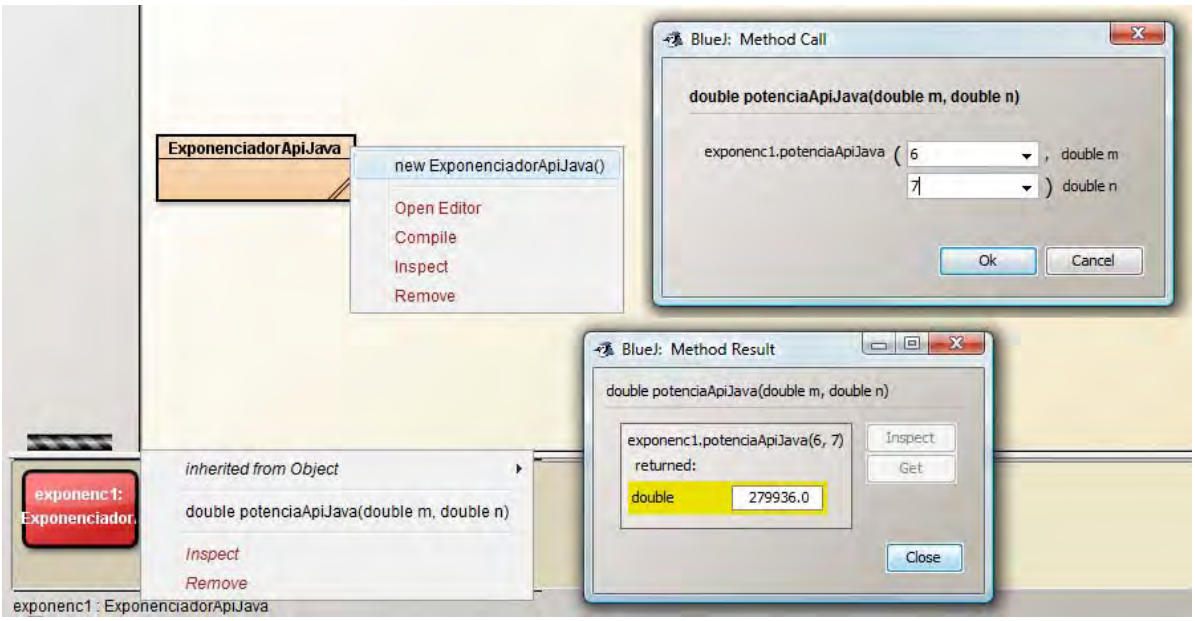
Realicemos un nuevo intento de compilación teniendo en cuenta la forma de invocación para la clase Math:

```
import java.lang.Math;
//Clase que permite elevar un número m a otro número n y obtener un resultado
public class ExponenciadorApiJava {
   public ExponenciadorApiJava () { } //Nada que declarar

   public int potenciaApiJava (int m, int n) {  return Math.pow (m, n);  } //Cierre del método
} //Cierre de la clase
```

Volvemos a obtener un error de compilación del tipo “possible loss of precision found: double required: int”. En este caso el compilador nos indica que el método *pow* devuelve un valor de tipo *double* cuando el método en su signatura indica que se va a devolver un valor de tipo *int* y eso puede ser problemático. El método *potenciaApiJava* lo tenemos definido ahora mismo como un método que devuelve un entero y recibe como parámetros dos enteros. Sin embargo, el método *pow* de la clase Math devuelve un *double* y espera recibir como parámetros dos valores de tipo *double*. Hay varias maneras de resolver esto. Nosotros vamos a optar ahora por ceñirnos a lo que nos indica el API de Java y vamos a cambiar la signatura de nuestro método para que devuelva y reciba valores de los tipos esperados. Intentaremos compilar teniendo en cuenta lo dicho anteriormente:

```
import java.lang.Math;
//Clase que permite elevar un número m a otro número n y obtener un resultado
public class ExponenciadorApiJava {
   public ExponenciadorApiJava () { } //Nada que declarar

   public double potenciaApiJava (double m, double n) {
      return Math.pow (m, n);
   } //Cierre del método
} //Cierre de la clase
```

Crea un objeto de tipo ExponenciadorApiJava e invoca su método para obtener las potencias de 2^3 y de 6^7. Los resultados deben ser los mismos que los obtenidos en los ejemplos anteriores, aunque ahora se nos mostrará un resultado terminado en .0 para indicar que tiene precisión decimal.



Prueba ahora a eliminar la sentencia *import* de la primera línea y a compilar. La compilación es posible. ¿Por qué? Esto se debe a lo que comentamos relativo a que **determinadas clases o paquetes se cargan automáticamente** mientras que otros no se cargan a no ser que se indique específicamente. El paquete java.lang es un paquete que se carga automáticamente. Por ello podemos hacer uso de todas sus clases, como String o Math, sin necesidad de importarlo. Si escribimos la sentencia de importación no habrá mensaje de error, pero tampoco será útil ya que estamos redundando al repetir algo que hace Java automáticamente.

EJERCICIO

Una operación frecuente con valores numéricos puede ser obtener su valor absoluto, es decir, su valor sin signo. Por ejemplo el valor absoluto de -3.22 es 3.22 (hemos eliminado el signo negativo) y el valor absoluto de 7.15 es 7.15 (al ser este número positivo coincide con su valor absoluto). Otra operación frecuente es el cálculo de la raíz cuadrada de un número. Consulta la documentación de la clase Math del api Java para comprobar qué métodos permiten realizar estas tareas. Crea una clase denominada miniCalculadoraEjemplo que tenga dos métodos (basados en el uso de métodos de la clase Math): un método valorAbsoluto que recibe un número de tipo double y devuelva su valor absoluto, y otro método raizCuadrada que reciba un número de tipo double y devuelva su raíz cuadrada. Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00648B

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188