APRENDERAPROGRAMAR.COM

CONCEPTO Y DEFINICIÓN DE CLASE EN JAVA. OBJETOS DEL MUNDO REAL Y ABSTRACTOS. EJEMPLOS. EJERCICIO (CU00644B)

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

Fecha revisión: 2029

Resumen: Entrega nº44 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

LA CLASE VISTA COMO PAQUETE DE CÓDIGO. OBJETOS DEL MUNDO REAL Y ABSTRACTOS.

Cuando hablamos de operadores aritméticos en Java planteamos que aunque en otros lenguajes existe un operador de exponenciación, en Java no es así. Para calcular una potencia dijimos que podíamos hacer varias cosas:

a) Recurrir a multiplicar n veces el término. Por ejemplo min^3 lo podemos calcular como min*min*min. Obviamente esto no es práctico para potencias de exponentes grandes.

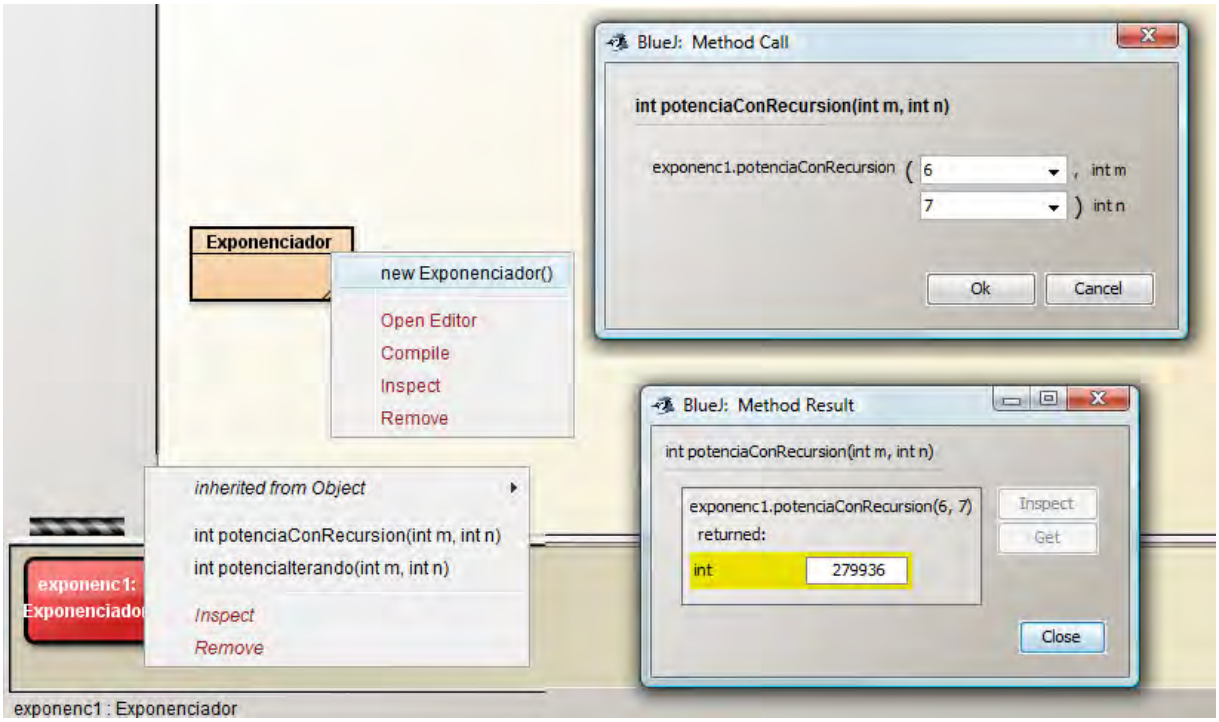
b) Usar un bucle que dé lugar a la repetición de la operación multiplicación n veces, o usar un método que ejecute la operación.

c) Usar herramientas propias del lenguaje que permiten realizar esta operación.

Podemos crear nuestros propios métodos para realizar una exponenciación. Este podría ser el código para ello:

```
//Clase que permite elevar un número entero m a otro número entero n y obtener un resultado
public class Exponenciador {
//Constructor
    public Exponenciador () {
      //Nada que declarar
    }
    //Método 1 para calcular la potencia
    public int potenciaIterando (int m, int n) {
       int resultado = 1;
       for (int i=1; i<=n; i++) {
          resultado = resultado * m;
       }
       return resultado;
    } //Cierre del método
    //Método 2 para calcular la potencia
    public int potenciaConRecursion (int m, int n) {
       if (n==0) { return 1;
       } else  { return m * potenciaConRecursion (m, n-1); }
    } //Cierre del método
} //Cierre de la clase
```

Escribe y compila el código anterior sin preocuparte de si lo entiendes completamente o no. Seguidamente, crea un objeto Exponenciador e invoca sus métodos *potenciaIterando* y *potenciaConRecursion* pasando como parámetros 2 y 3. El resultado en ambos casos debe ser 8, que es el resultado de ejecutar 2^3 = 2*2*2. Haz lo mismo pasando como parámetros 6 y 7. El resultado debe ser 279936, que es el valor que se obtiene al ejecutar 6^7.



Vamos a repasar lo que hemos hecho: hemos definido una clase sin campos, es decir, un objeto de tipo Exponenciador no va a tener atributos. **¿Por qué no tiene atributos?** Simplemente porque no los necesitamos. Esa clase permite crear objetos de tipo Exponenciador sobre los que se pueden invocar dos métodos que al fin y al cabo hacen lo mismo (obtener una potencia) pero de distinta manera. El método iterativo es más fácil de entender: repetimos una multiplicación n veces. El método recursivo es más difícil de entender: se basa en que un método se llame a sí mismo repetidamente cambiando los parámetros de la llamada (n va disminuyendo una unidad en cada llamada) hasta que se llega a un caso "terminal" denominado caso base.

La recursión es una forma de programación que podríamos denominar avanzada, así que no te preocupes ahora de entenderla. Hay programadores con muchos años de experiencia que no la usan directamente porque el seguimiento de procesos recursivos es en general complicado. Ahora simplemente quédate con la idea de que una misma cosa se puede hacer de distintas maneras, y que esas maneras pueden tener distintas propiedades (por ejemplo una ser más rápida que otra, una consumir más memoria que otra, una ser más difícil de seguir que otra, etc.).

Podríamos haber usado otro diseño para lo que hemos hecho: en vez de tener una clase Exponenciador con dos métodos, podríamos haber definido dos clases denominadas ExponenciadorIterativo y ExponenciadorRecursivo, cada una con un método. ¿Es preferible hacerlo de una manera o de otra? La respuesta es que depende. Cuando trabajamos programando tenemos que tomar decisiones de diseño

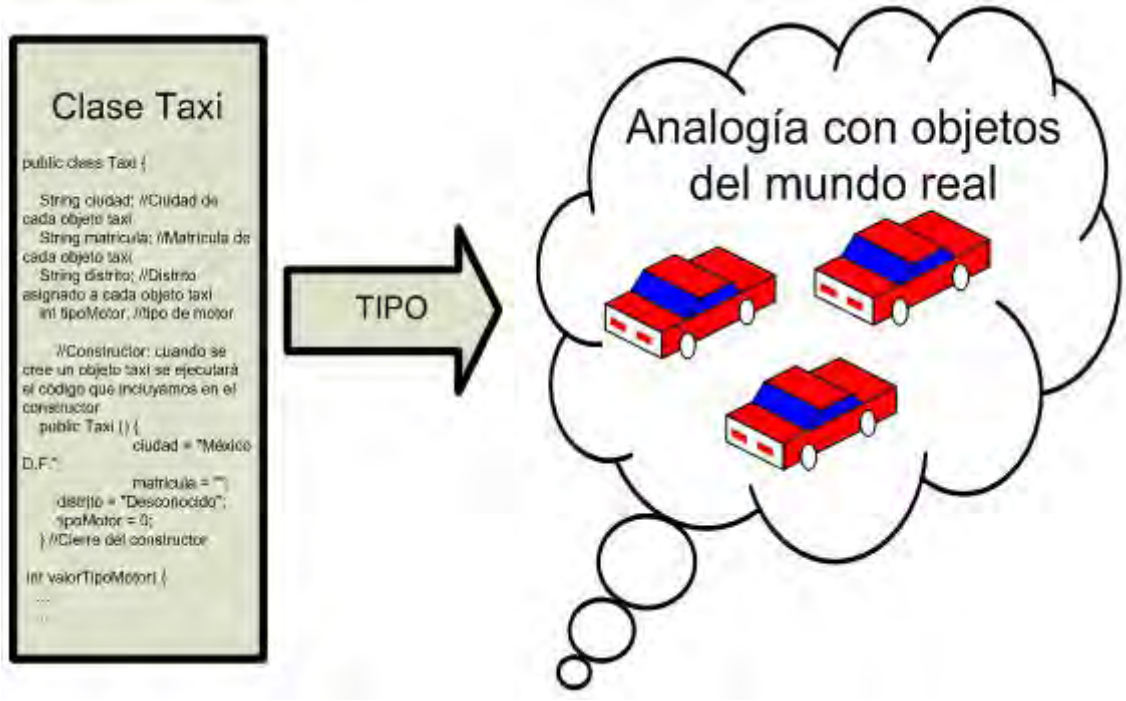
que a veces son más fáciles de tomar y a veces más difíciles. Estas decisiones de diseño pueden tener en algunos casos poca relevancia, pero en otros muchos importantes repercusiones en el futuro del desarrollo de los programas. Conviene analizar las opciones con detenimiento y valorar cuál es la más apropiada. En nuestro caso se trata solo de un ejemplo y elegir una opción u otra nos resulta digamos que indiferente.

Hay algunas cuestiones que merece la pena comentar del código que hemos escrito.

¿Por qué el constructor de esta clase está vacío? Cuando definimos *clase* dijimos que una clase es una abstracción que define un tipo de objeto especificando qué propiedades (atributos) y operaciones disponibles va a tener. Además habíamos dicho que en general un constructor no estará vacío. Llegados a este punto no queda más que buscar otra definición de clase:

Clase: es un paquete o fragmento de código Java que permite crear al menos una instancia (objeto).

En nuestro ejemplo de clase Taxi, el tipo definido por la clase tenía una correspondencia con el mundo real o con objetos que podíamos concebir físicamente.



En nuestro ejemplo de clase Exponenciador, el tipo definido por la clase podemos preguntarnos si tiene alguna analogía en objetos que podamos identificar en el mundo real.

La respuesta es que en este caso no podemos encontrar una analogía. Efectivamente, podemos ir por la calle y encontrarnos un Taxi pero difícilmente iremos por la calle y nos encontraremos un Exponenciador. Aquí radica una dificultad habitual que tienen las personas que están aprendiendo Java. Cuando existe una analogía con el mundo real, el código resulta más fácil de entender. Cuando no la existe, surgen dudas. Dudas por otro lado razonables, porque la terminología Java podemos decir que es un poco confusa: se nos habla de objetos y cuando pensamos en objetos tendemos a pensar en lo que en el día a día se considera un objeto: algo que se puede tocar. Y por derivación suponemos que un objeto en programación orientada a objetos será la representación de un objeto de la vida real. Pero esto no es así: un objeto en Java puede representar un objeto de la vida real, pero también puede ser algo abstracto e intangible. Tendremos que considerarlo un simple espacio de memoria del ordenador con capacidad para realizar procesos. Y esto resulta un poco más difícil de asimilar. La mejor forma de hacerlo es, a nuestro juicio, ir paso a paso creando pequeños programas y descubriendo las posibilidades de Java. Entender bien el paradigma de programación orientada a objetos requiere tiempo.

La definición de objeto que habíamos dado en epígrafes anteriores sigue siendo válida:

Objeto: entidad existente en la memoria del ordenador que tiene unas propiedades (atributos o datos sobre sí mismo almacenados por el objeto) y unas operaciones disponibles específicas (métodos).

La cuestión es que **en algunos casos un objeto no podemos asociarlo a nada en el mundo real.** Además, un objeto en algunas circunstancias puede carecer de atributos, carecer de constructor, o carecer de métodos. Un objeto de tipo Exponenciador según el ejemplo que hemos visto, se correspondería con el caso de objeto sin atributos y sin constructor. En realidad hemos preferido incluir un constructor vacío indicando que no teníamos nada que declarar. Si no lo hubiésemos hecho así,

quien leyera el código podría tener la duda de si nos hemos "olvidado" de incluir el constructor. Este objeto es capaz de ejecutar dos métodos que reciben parámetros y devuelven un resultado, pero internamente no almacena ninguna información como podía ser la matrícula o el distrito de los objetos Taxi.

La clase sigue definiendo un tipo, pero esto es quizás ahora menos relevante. En este caso preferimos verla como un paquete de código que nos permite realizar procesos.

EJERCICIO

Define una clase denominada multiplicadorDieces con un constructor vacío y que contenga un método denominado multiplicarPorDieces que reciba dos parámetros: el primero un número de tipo double y el segundo un número de tipo entero. El método debe devolver el resultado de multiplicar por 10 elevado al segundo número el primer número. Ejemplo: multiplicarPorDieces (2.55, 2) devuelve 2.55*100 = 255. multiplicarPorDieces (3, 5) devuelve 3*100000 = 300000. MultiplicarPorDieces (-0.0563, 3) devuelve -0.563 * 1000 = -56.3. Crea un objeto y comprueba que el método opera correctamente. Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00645B

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