APRENDERAPROGRAMAR.COM

CÓMO CREAR CONSTRUCTORES E INICIALIZAR OBJETOS EN JAVA. EJERCICIO EJEMPLO RESUELTO. (CU00639B)

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

Fecha revisión: 2029

Resumen: Entrega nº39 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

CÓMO CREAR CONSTRUCTORES EN JAVA. EJERCICIOS EJEMPLOS RESUELTOS.

Los constructores de una clase son **fragmentos de código que sirven para inicializar un objeto** a un estado determinado. Una clase puede carecer de constructor, pero esto no es lo más habitual. Normalmente todas nuestras clases llevarán constructor. En un constructor es frecuente usar un esquema de este tipo:


```
public MismoNombreQueLaClase (tipo parámetro1, tipo parámetro2 …, tipo parámetro n ) {
    campo1 = valor o parámetro;
    campo2 = valor o parámetro;
    .
    .
    .
    campo n = valor o parámetro;
}
```

Los constructores tienen el mismo nombre que la clase en la que son definidos y nunca tienen tipo de retorno, ni especificado ni void. Tenemos aquí un aspecto que nos permite diferenciar constructores de métodos: un constructor nunca tiene tipo de retorno mientras que un método siempre lo tiene. Es recomendable que en un constructor se inicialicen **todos** los atributos de la clase aunque su valor vaya a ser nulo o vacío. Si un atributo se quiere inicializar a cero (valores numéricos) siempre lo declararemos específicamente: nombreAtributo = 0;. Si un atributo se quiere inicializar a contenido nulo (atributos que son objetos) siempre lo declararemos específicamente: nombreAtributo = null;. Si un atributo tipo texto se quiere inicializar vacío siempre lo declararemos específicamente: nombreAtributo = “”;. El motivo para actuar de esta manera es que declarando los atributos como nulos o vacíos, dejamos claro que esa es nuestra decisión como programadores. Si dejamos de incluir uno o varios campos en el constructor puede quedar la duda de si hemos olvidado inicializar ese campo o inducir a pensar que trabajamos con malas prácticas de programación.

> ***La inicialización de campos y variables es un proceso muy importante. Su mala definición es fuente de problemas en el desarrollo de programas. Como regla de buena programación, cuando crees campos o variables, procede de forma inmediata a definir su inicialización.***

Un constructor puede:

a) Carecer de parámetros: que no sea necesario pasarle un parámetro o varios al objeto para inicializarse. Un constructor sin parámetros se denomina “constructor general”.

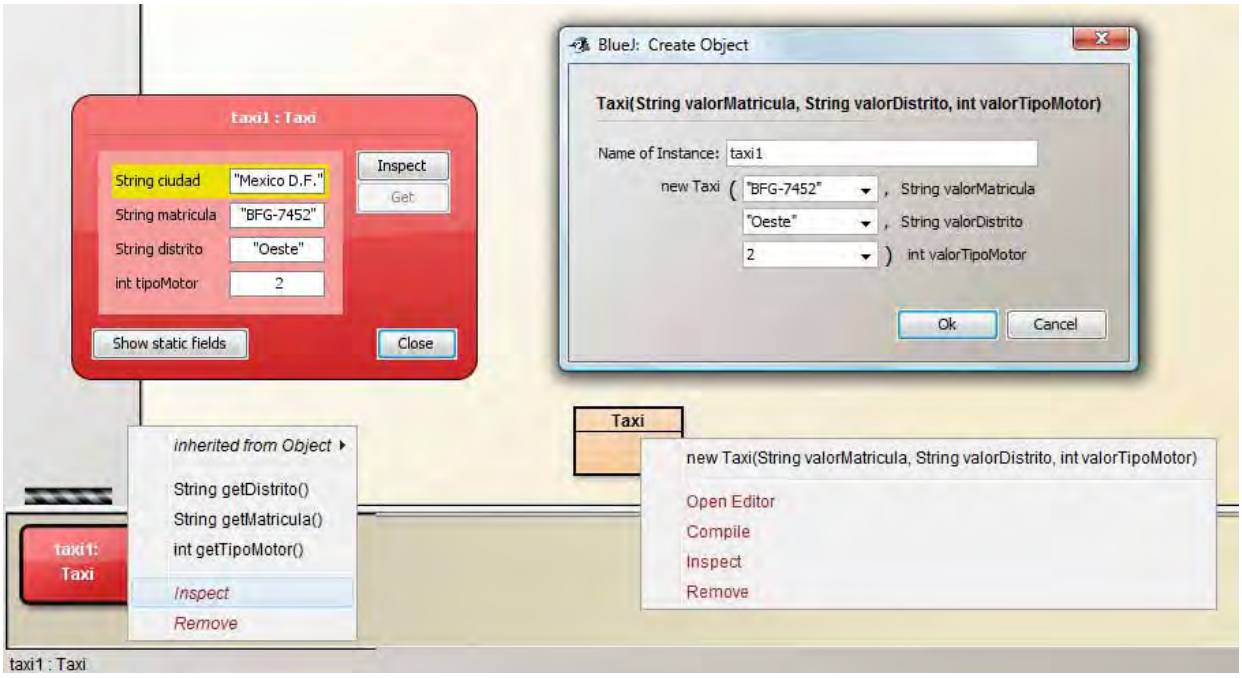
b) Carecer de contenido. Por ejemplo, *public Taxi () { }* podría ser un constructor, vacío. En general un constructor no estará vacío, pero en algunos casos particulares puede estarlo. Si el constructor carece de contenido los campos se inicializan con valor nulo o, si son tipos definidos en otra clase, como se haya definido en el constructor de la otra clase. Excepto en casos controlados, evitaremos que existan constructores vacíos.

Si un constructor tiene parámetros, el funcionamiento es análogo al que ya hemos visto para métodos. Cuando vayamos a crear el objeto con BlueJ, se nos pedirá además del nombre que va a tener el objeto, el valor o contenido de los parámetros requeridos. Un parámetro con frecuencia sirve para inicializar el objeto como hemos visto, y en ese caso el objeto tendrá el valor pasado como parámetro como atributo "para siempre", a no ser que lo cambiemos por otra vía como puede ser un método modificador del atributo. No obstante, en algunos casos los parámetros que recibe un constructor no se incorporarán directamente como atributos del objeto sino que servirán para realizar operaciones de diversa índole. Escribe y compila el siguiente código:

```
public class Taxi { //El nombre de la clase

   private String ciudad; //Ciudad de cada objeto taxi
   private String matricula; //Matrícula de cada objeto taxi
   private String distrito; //Distrito asignado a cada objeto taxi
   private int tipoMotor; //Tipo de motor asignado a cada objeto taxi. 0 = desconocido, 1 = gasolina, 2 = diesel

   //Constructor: cuando se cree un objeto taxi se ejecutará el código que incluyamos en el constructor
   public Taxi (String valorMatricula, String valorDistrito, int valorTipoMotor) {
      ciudad = "México D.F.";
      matricula = valorMatricula;
      distrito = valorDistrito;
      tipoMotor = valorTipoMotor;
   } //Cierre del constructor

    //Método para obtener la matrícula del objeto taxi
   public String getMatricula () { return matricula; } //Cierre del método

   //Método para obtener el distrito del objeto taxi
   public String getDistrito () { return distrito; } //Cierre del método

   //Método para obtener el tipo de motor del objeto taxi
   public int getTipoMotor () { return tipoMotor; } //Cierre del método

} //Cierre de la clase
```

Este código es similar al que vimos en epígrafes anteriores. La diferencia radica en que ahora en vez de tener un constructor que establece una forma fija de inicializar el objeto, **la inicialización depende de los parámetros que le lleguen al constructor.** Además, hemos eliminado los métodos para establecer el valor de los atributos. Ahora éstos solo se pueden consultar mediante los métodos get. Pulsa con botón derecho sobre el icono de la clase y verás como la opción *new Taxi* incluye ahora los parámetros dentro de los paréntesis. Escoge esta opción y establece unos valores como matrícula "BFG-7452", distrito

"Oeste" y tipo de motor 2. Luego, con el botón derecho sobre el icono del objeto, elige la opción *Inspect* para ver su estado.



Que un constructor lleve o no parámetros y cuáles tendremos que elegirlo para cada clase que programemos. En nuestro ejemplo hemos decidido que aunque la clase tiene cuatro campos, el constructor lleve solo tres parámetros e inicializar el campo restante con un valor fijo. Un constructor con parámetros es adecuado si tiene poco sentido inicializar los objetos vacíos o siempre con el mismo contenido para uno o varios campos. No obstante, **siempre hay posibilidad de darle contenido a los atributos a posteriori si incluimos métodos "setters".** El hacerlo de una forma u otra dependerá del caso concreto al que nos enfrentemos.

El esquema que hemos visto supone que en general vamos a realizar una declaración de campo en cabecera de la clase, por ejemplo *String ciudad;*, y posteriormente inicializar esa variable en el constructor, por ejemplo *ciudad = "México D.F.";*. ¿Sería posible hacer una declaración en cabecera de clase del tipo *String ciudad = "México D.F.";*? La respuesta es que sí. El campo quedaría inicializado en cabecera, pero esto en general debe ser considerado una mala práctica de programación y contraproducente dentro de la lógica de la programación orientada a objetos. Por tanto de momento trataremos de evitar incluir código de ese tipo en nuestras clases y procederemos siempre a inicializar en los constructores.

EJERCICIO

Define una clase Bombero considerando los siguientes atributos de clase: nombre (String), apellidos (String), edad (int), casado (boolean), especialista (boolean). Define un constructor que reciba los parámetros necesarios para la inicialización y los métodos para poder establecer y obtener los valores de los atributos. Compila el código para comprobar que no presenta errores, crea un objeto y comprueba que se inicializa correctamente consultando el valor de sus atributos después de haber creado el objeto. Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00640B

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