APRENDERAPROGRAMAR.COM

PASO DE OBJETOS COMO PARÁMETROS A UN MÉTODO O CONSTRUCTOR JAVA. DIFERENCIAS ENTRE OBJETO Y TIPO PRIMITIVO. (CU00642B)

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

Fecha revisión: 2029

Resumen: Entrega nº42 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

PASO DE OBJETOS COMO PARÁMETROS A UN MÉTODO O CONSTRUCTOR EN JAVA

Hasta ahora habíamos visto como un método o constructor puede recibir parámetros de los tipos de datos primitivos en Java, como int, boolean, etc. e incluso de tipos envoltorio como Integer. Vamos a ver que también se pueden recibir otro tipo de parámetros. Partimos de esta definición de clase Taxi, escríbela en tu editor:


```
/* Esta clase representa un taxi ejemplo - aprenderaprogramar.com */

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 1: constructor sin parámetros
   public Taxi () {
      ciudad = "México D.F.";      matricula = "";       distrito = "Desconocido";       tipoMotor = 0;
   } //Cierre del constructor

   //Constructor 2: constructor con parámetros
   public Taxi (String valorMatricula, String valorDistrito, int valorTipoMotor) {
   ciudad = "México D.F.";    matricula = valorMatricula;   distrito = valorDistrito; tipoMotor = valorTipoMotor;
   } //Cierre del constructor

   //Método para establecer la matrícula de un taxi
   public void setMatricula (String valorMatricula) { matricula = valorMatricula;     } //Cierre del método
   //Método para establecer el distrito de un taxi
   public void setDistrito (String valorDistrito) { distrito = "Distrito " + valorDistrito;    } //Cierre del método
   //Método para establecer el tipo de motor de un taxi
   public void setTipoMotor (int valorTipoMotor) {  tipoMotor = valorTipoMotor;     } //Cierre del método

   //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
```

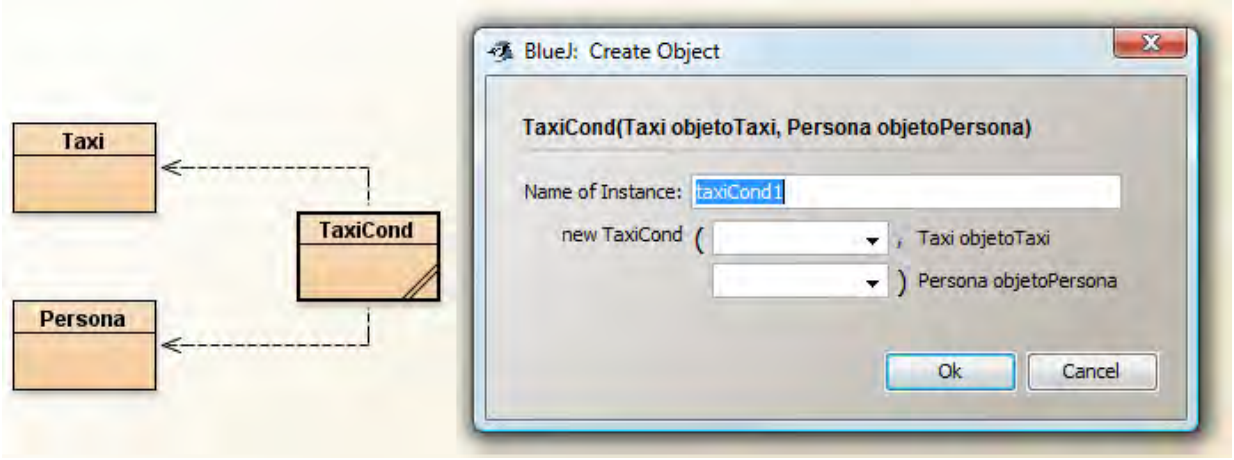
Recupera el código de la clase Persona que usamos anteriormente y crea la clase Persona. Modifica el código de la clase TaxiCond que usamos anteriormente de forma que el constructor pase a ser este:

```
    //Constructor
    public TaxiCond (Taxi objetoTaxi, Persona objetoPersona) {
        //Creamos un objeto Taxi con los mismos datos del Taxi recibido como parámetro
        vehiculoTaxi = new Taxi (objetoTaxi.getMatricula(), objetoTaxi.getDistrito(), objetoTaxi.getTipoMotor() );
        //Creamos un objeto Persona con los mismos datos de la Persona recibidos como parámetro
       conductorTaxi = new Persona (objetoPersona.getNombre() );    }
```

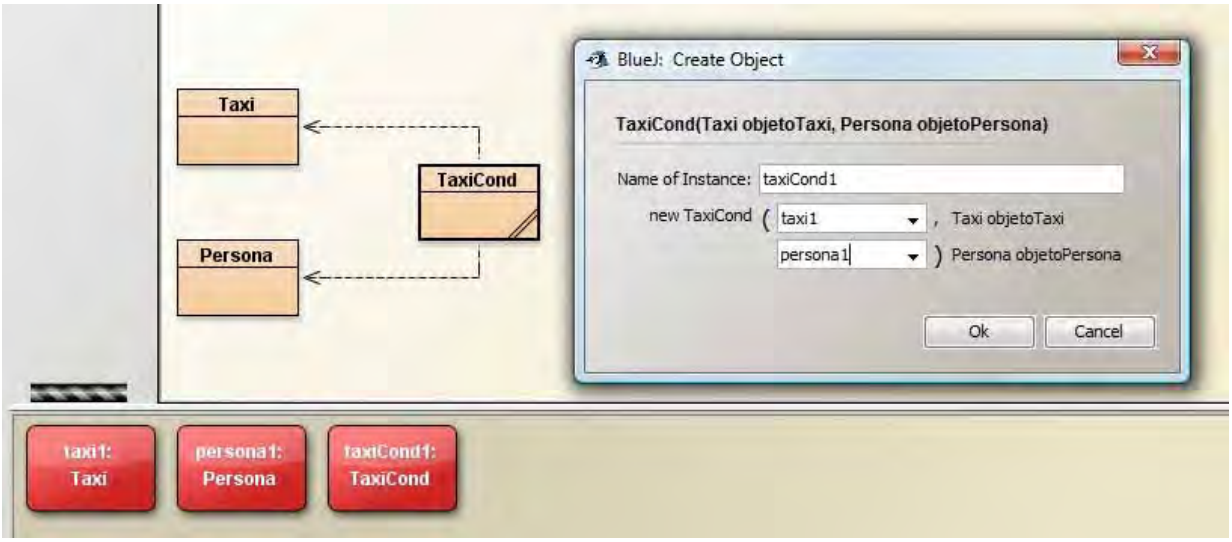
Ahora debes tener tres clases: Taxi, Persona y TaxiCond. La clase Taxi representa un taxi, la clase Persona representa una persona y la clase TaxiCond representa un taxi con conductor.

¿Por qué hemos usado una expresión como *vehiculoTaxi = new Taxi (objetoTaxi.getMatricula(), objetoTaxi.getDistrito(), objetoTaxi.getTipoMotor());* en vez de simplemente *vehiculoTaxi = objetoTaxi;* ? La respuesta hay que buscarla en algo que tendremos que analizar más ampliamente un poco más adelante: **un objeto es algo distinto a un tipo primitivo y no podemos aplicarle la lógica de los tipos primitivos**. El hecho de crear nuevos objetos responde a que no queremos modificar los objetos que se pasan como parámetro. Esto entra dentro de la lógica y propiedades de los objetos que iremos estudiando poco a poco. De momento, simplemente utilizaremos ahora este código sin pararnos a pensar demasiado en él.

Nos encontramos frente a un constructor que nos requiere como parámetros objetos complejos y no tipos primitivos ni objetos simples de tipo String. Intenta ahora crear un objeto de tipo TaxiCond pulsando sobre el icono de la clase y con botón derecho eligiendo la opción new TaxiCond.



Aparecerá una ventana que nos pide además del nombre de la instancia (objeto que vamos a crear), que indiquemos los dos objetos necesarios para crear el nuevo objeto. Nosotros no disponemos de esos dos objetos, así que vamos a cerrar esta ventana y a crear primero un objeto Taxi y un objeto Persona, que nos van a ser necesarios para crear el objeto TaxiCond. Luego, crea el objeto TaxiCond introduciendo los nombres de los objetos Taxi y Persona creados previamente en las casillas correspondientes (también resulta válido hacer click sobre los iconos de los objetos).



Con este ejemplo hemos comprobado que un constructor (o un método) nos puede requerir como parámetros uno o varios objetos y que para pasar los mismos escribimos sus nombres.

EJERCICIO

Define tres clases: Casa, SalonCasa y CocinaCasa. La clase SalonCasa debe tener como atributos numeroDeTelevisores (int) y tipoSalon (String) y disponer de un constructor que los inicialice a 0 y "desconocido". La clase CocinaCasa debe tener como atributos esIndependiente (boolean) y numeroDeFuegos (int) y un constructor que los inicialice a false y 0. La clase Casa tendrá los siguientes atributos de clase: superficie (double), direccion (String), salonCasa (tipo SalonCasa) y cocina (tipo CocinaCasa). Define un constructor para la clase Casa que establezca a unos valores de defecto los atributos simples y que cree nuevos objetos si se trata de atributos objeto. Define otro constructor que reciba como parámetros la superficie, dirección y un objeto de tipo SalonCasa y otro de tipo CocinaCasa. Compila el código para comprobar que no presenta errores, y crea un objeto de tipo Casa usando el constructor que recibe parámetros. Ten en cuenta que antes tendrás que haber creado los objetos de tipo SalonCasa y CocinaCasa para poder pasárselos al constructor. Comprueba que el objeto Casa 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: CU00643B

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