APRENDERAPROGRAMAR.COM

VARIABLES LOCALES A UN MÉTODO O CONSTRUCTOR EN JAVA. SOBRECARGA DE NOMBRES (CU00638B)

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

Fecha revisión: 2029

Resumen: Entrega nº38 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

VARIABLES LOCALES A UN MÉTODO O CONSTRUCTOR. SOBRECARGA DE NOMBRES.

Una variable que se declara y se usa dentro de un método (o de un constructor) se dice que es una variable local. Su ámbito es sólo el método o constructor y su tiempo de vida es solo el del método, es decir, son **variables temporales que se crean cuando comienza a ejecutarse el método y se destruyen cuando termina de ejecutarse.**

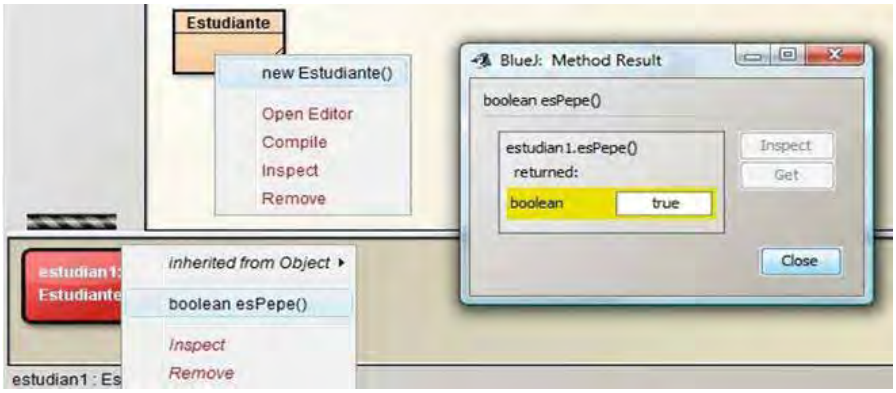
Escribe y compila el siguiente código:

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

   private String nombre; //Campo de los objetos Estudiante

   //Constructor: cuando se cree un objeto Estudiante se ejecutará el código que incluyamos en el constructor
   public Estudiante () {
      nombre = "Pepe";
   } //Cierre del constructor
   //Método que devuelve true si el nombre del objeto tipo Estudiante es Pepe
   public boolean esPepe() {
      boolean seLlamaPepe = false;
      if (nombre == "Pepe") { seLlamaPepe = true; }
      return seLlamaPepe;
   } //Cierre del método
} //Cierre de la clase
```

La variable *seLlamaPepe* es una variable local booleana. Es habitual inicializar las variables locales cuando se las declara, pero no es estrictamente necesario. Sí es obligatorio inicializar las variables en algún momento ya que no se debe considerar que tengan un valor por defecto. Crea un objeto de tipo Estudiante pulsando sobre el icono de la clase con botón derecho y eligiendo la opción new Estudiante(). Sobre el objeto que aparecerá en el banco de objetos, invoca el método esPepe().



Ahora crea otro método y trata de establecer en él la variable *seLlamaPepe* con valor *true*. El compilador lanzará un mensaje de error del tipo "*cannot find symbol – variable seLlamaPepe*". ¿Por qué podemos usar la variable *nombre* en cualquier método mientras que la variable *seLlamaPepe* sólo dentro del método *esPepe()*? Como hemos dicho, el ámbito de una variable declarada en un método es solo ese método. El resto de métodos no conocen la variable. En cambio, un campo de la clase tiene como ámbito toda la clase y lo podemos usar en cualquier lugar del código de la clase. Hay algunas conclusiones interesantes:

1) Podemos usar el mismo nombre de variable local en muchos métodos, puesto que no van a interferir entre ellas.

2) Una declaración de campo siempre la hacemos precedida de *public* o *private*. En las variables locales, estas palabras clave no se usan debido a su carácter temporal.

3) En los métodos tipo función con frecuencia en la sentencia *return* se devuelve como resultado el valor de una variable local que ha sido objeto de cálculo en el método. Tener en cuenta que no se devuelve la variable en sí (que en realidad desaparece cuando termina el método), sino su valor o contenido.

¿Puede un constructor tener variables locales? Sí. Un constructor son una serie de instrucciones. A veces muy sencillas, pero otras veces pueden requerir cálculos o procesos complejos. Por tanto, podemos usar variables locales dentro de ellos declarándolas y usándolas como si se tratara de un método. No tenemos restricciones en cuanto al código que se puede incluir en un constructor.

¿Puede una variable local ser tipo objeto? Sí. Hemos dicho que las clases definen tipos. Por ejemplo podríamos tener una variable local *miTaxi1* declarada como *Taxi miTaxi1;*.

¿Qué ocurre si una variable local tiene el mismo nombre que un campo? Esta situación se daría si tenemos un campo declarado por ejemplo como *String ciudad;* y luego declaramos dentro de un método una variable de ese mismo tipo u otro distinto con el mismo nombre, por ejemplo *boolean ciudad = false;*. En este caso decimos que existe **sobrecarga de nombres**. Podemos tener problemas si no manejamos esta situación adecuadamente. Cuando empleemos en el código el nombre de la variable el compilador no es capaz de adivinar nuestro pensamiento para saber si nos referimos al campo de la clase o a la variable local del método. Este conflicto Java lo resuelve aplicando la regla de prevalencia del ámbito "más local". Es decir, si escribimos un nombre de variable Java usa la variable "más local" disponible. Java tiene prevista la solución para poder usar simultáneamente campos y variables locales con el mismo nombre, mediante el uso de la palabra clave *this*. Esto lo explicaremos más adelante. Reflexionemos ahora sobre los tipos o formas de variables que hemos visto hasta el momento. Se resumen en el siguiente esquema:

		ÁMBITO	DURACIÓN
Tipos de variables	Campos (atributos de objeto)	Toda la clase	Indefinida (vida del objeto)
	Parámetros de un método o constructor	Local solo el método o constructor	Temporal
	Variables locales de un método o constructor	Local solo el método o constructor	Temporal

EJERCICIO

Considera estás desarrollando un programa Java donde necesitas trabajar con objetos de tipo Motor (que representa el motor de una bomba para mover fluidos). Define una clase Motor considerando los siguientes atributos de clase: tipoBomba (int), tipoFluido (String), combustible (String). Define un constructor asignando unos valores de defecto a los atributos y los métodos para poder establecer y obtener los valores de los atributos. Crea un método tipo función que devuelva un booleano (true o false) denominado dimeSiMotorEsParaAgua() donde se cree una variable local booleana motorEsParaAgua de forma que si el tipo de motor tiene valor 1 tomará valor true y si no lo es tomará valor false. El método debe devolver la la variable local booleana motorEsParaAgua.

Compila el código para comprobar que no presenta errores, crea un objeto, usa sus métodos y comprueba que se obtienen resultados correctos. Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00639B

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