

Métodos Set, Get y Constructores

Material de clase elaborado por Sandra Victoria Hurtado Gil

1. Tipos de métodos

Una clase puede tener diferentes tipos de métodos, de acuerdo con lo que se desee realizar con los objetos de esta clase. Aunque los métodos pueden ser muy diferentes, existen algunos métodos especiales que son muy comunes en las clases, como los de consulta o modificación de atributos. Esto permite tener la siguiente clasificación de los métodos:

Tipo de Método	Utilidad
De acceso o consulta – “get”	Para consultar el valor de un atributo.
De modificación – “set”	Para definir o modificar el valor de un atributo.
Constructor	Permite asignar recursos iniciales que necesiten los objetos, en el momento de su construcción o creación.
Destructor	Permite liberar los recursos que tenga un objeto, en el momento de su destrucción.
<i>main</i>	Permite que la clase sea ejecutada por la máquina virtual de Java.
Otros	Todos los demás métodos de una clase, que permiten prestar diferentes servicios.

En este capítulo se explicarán los métodos “get”, “set” y los constructores. En el caso de los destructores, como estos métodos no son muy usados en Java, no se trabajarán en este material.

2. Métodos “get”: Acceso o consulta de un atributo

Se tiene la clase Cuenta, representada en el siguiente diagrama:

Cuenta
número: String saldo: double tipo: String
consultarSaldo(): double consignar(cantidad: double): void retirar(cantidad: double): boolean


Es posible consignar y retirar dinero de los objetos cuenta, pero ¿cómo se hace para consultar la información, por ejemplo, para saber el tipo de cuenta? Para esto se usará un método de acceso o consulta.

Es muy común tener métodos que permitan consultar u obtener el valor de un atributo, como por ejemplo el método “consultarSaldo” de la clase Cuenta. Sin embargo, para facilitar el desarrollo

de los programas en Java, se ha establecido una convención internacional para dar un nombre estándar a los métodos que consultan el valor de **un** atributo.

El nombre de estos métodos debe comenzar con la palabra “**get**”, seguida del nombre del atributo.

Por ejemplo, para la clase Cuenta, el método para consultar el saldo quedaría:



Cuenta
número: String saldo: double tipo: String
getSaldo(): double consignar(cantidad: double): void retirar(cantidad: double): boolean

Un método “get” no recibe parámetros, solo debe retornar el valor del atributo, como se presenta a continuación:

```
double getSaldo() {  
    return saldo;  
}
```

Cuando se utiliza un método “get” por lo general se tiene una variable para recibir el valor que se está consultando, y así poder usarlo, o también se puede incluir en expresiones. Por ejemplo, en el siguiente código se crea una cuenta, se consigna un valor y luego se muestra el saldo:

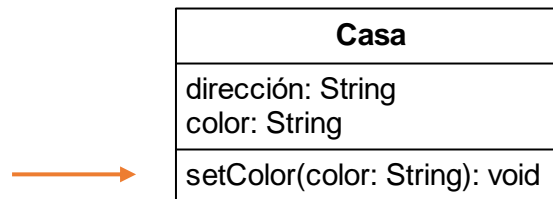
```
/**  
 * Usa un objeto Cuenta para consultar el saldo  
 * @version 1.0  
 */  
public class Bancaria  
{  
    public static void main(String arg[])  
    {  
        Cuenta cuenta1 = new Cuenta();  
        cuenta1.consignar(100000);  
        double saldoCuenta = cuenta1.getSaldo();  
        System.out.println("El saldo es: "+ saldoCuenta);  
    }  
}
```

3. Métodos “set”: Modificación de un Atributo

De manera similar a los métodos que consultan el valor de un atributo, también se tienen métodos para asignar, modificar o mutar el valor de **un** atributo.

El estándar internacional para los nombres de los métodos que asignan o modifican el valor de un atributo es: comenzar con la palabra “**set**”, seguida del nombre del atributo.

Por ejemplo, se tiene la clase Casa, que tiene la dirección y el color. En cualquier momento se puede cambiar el color de la casa y para representar esto se tendrá el método `set`, como se ilustra en el diagrama:



Un método “set” no retorna nada (*void*), pero sí recibe como parámetro el nuevo valor que se desea dar al atributo.

```
void setColor(String color) {
    this.color = color;
}
```

De esta manera, se puede cambiar el color de la casa, usando el método “`setColor()`”. Por ejemplo, en el siguiente código se crea una casa que se pinta de blanco.

```
/**
 * Se crea un objeto casa y se le da color blanco
 * @version 1.0
 */
public class PintaCasa
{
    public static void main(String arg[])
    {
        Casa miCasita = new Casa();
        miCasita.setColor("blanco");
    }
}
```

4. Constructor

Suponga que se tiene la siguiente clase en Java, que representa objetos Cajero (como un cajero electrónico):

Cajero
cantidadDinero: int estado: char
getEstado(): char getCantidadDinero(): int retirarDinero(cantidad: int): boolean recargarDinero(cantidad: int): void

En esta clase, ¿cuáles serían los valores iniciales para los atributos “cantidadDinero” y “estado”? Si se crea un objeto Cajero y se pregunta por el estado, ¿Qué mostraría?

Aunque Java da unos valores iniciales a estos atributos, tal vez no corresponden a lo que se desea tener. Se desea que el cajero comience con cantidad de dinero cero y con el estado ‘i’, que representa inactivo.


Sería ideal poder darle valores iniciales adecuados a los atributos desde que se construye el objeto, sin tener que usar métodos *set*.

Para poder dar valores iniciales a los objetos desde que se crean se usa un método especial, llamado “**constructor**”. Este método ayuda a construir o crear adecuadamente los objetos asignándoles los recursos que necesitan, entre ellos los valores de los atributos.

El método constructor es bien especial porque tiene algunas diferencias con los demás métodos:

- No tiene tipo de retorno (ni siquiera se escribe *void*)
- El nombre es **exactamente** el mismo de la clase. Si la clase comienza con mayúscula, este método también.
- Se utiliza llamado al operador *new* (no se llama con el operador punto como los demás métodos).

Por ejemplo, para definir un constructor para la clase Cajero, en el diagrama quedaría:

Cajero
cantidadDinero: int estado: char
 Cajero() getEstado(): char getCantidadDinero(): int retirarDinero(cantidad: int): boolean recargarDinero(cantidad: int): void

El código del constructor es:

```

/**
 * Constructor de los objetos Cajero, que comienzan con cero en
 * su cantidad de dinero y estado 'i' (inactivo)
 */
Cajero() {
    this.cantidadDinero = 0;
    this.estado = 'i';
}

```

Los constructores se ejecutan cuando se crea un nuevo objeto (**con la instrucción *new***):

```
Cajero cajero1 = new Cajero();
```

Aquí se está usando el constructor

Ahora veamos otro ejemplo. Se tiene la clase Cuenta, representada en el siguiente diagrama:


Cuenta
número: String saldo: double tipo: String
getSaldo(): double getTipo(): String consignar(cantidad: double): void retirar(cantidad: double): boolean

Se definirá un constructor para esta clase. Sin embargo, a diferencia de la clase Cajero, donde cada objeto de esta clase empezaba con cantidad cero y estado inactivo, aquí no tiene mucho sentido que todos los objetos Cuenta que se creen sean del mismo tipo o tengan el mismo número ni saldo.

Al momento de crear el objeto Cuenta se debería definir cuál es el número que tendrá, cuál es el tipo de cuenta y cuál será el saldo inicial –que es el dinero con el cual la persona puede abrir la cuenta en el banco.

Se definirá, por lo tanto, un constructor con tres parámetros: un *String* que corresponde al número de la cuenta, un *double* para el saldo inicial y un *String* para el tipo de cuenta, que puede ser “corriente” o “ahorros”.

El diagrama queda:



Cuenta
número: String saldo: double tipo: String
Cuenta(número: String, saldoInicial: double, tipo: String) getSaldo(): double getTipo(): String consignar(cantidad: double): void retirar(cantidad: double): boolean

A continuación, se mostrará el código de este constructor:

```
/**
 * Constructor de objetos Cuenta, con un saldo inicial.
 * @param numero    el número de la cuenta
 * @param saldoInicial  saldo inicial (en pesos) de la cuenta
 * @param tipo    el tipo de cuenta: "ahorros" o "corriente"
 */
Cuenta(String numero, double saldoInicial, String tipo)
{
    this.numero = numero;
    this.saldo = saldoInicial;
    this.tipo = tipo;
}
```

En este caso, cuando se desee construir un objeto Cuenta (con **new**), será necesario enviar los tres valores requeridos como parámetros:

```
Cuenta cuenta1 = new Cuenta("897-000", 20000, "corriente");
```

Ya NO es posible crear un objeto cuenta sin enviar parámetros, como se hacía en capítulos anteriores:

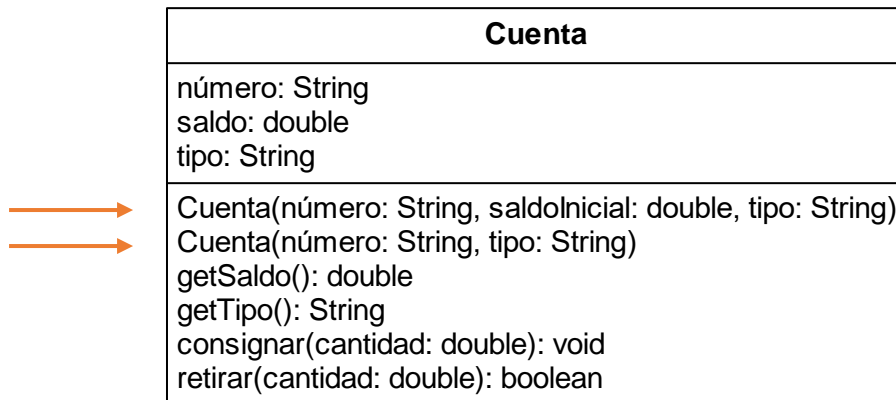
```
Cuenta cuentaNueva = new Cuenta(); //ERROR:
// Deben enviarse parámetros al constructor
```

Varios métodos constructores

Algo interesante es que una clase puede tener cero, uno o más constructores, lo que significa que se tiene más de una forma de crear los objetos.

Por ejemplo, ¿qué pasa si una persona desea abrir su cuenta bancaria sin tener dinero para el saldo inicial? En este caso no es adecuado crear el objeto con un saldo, sino que se debe crear la cuenta con saldo cero.

Pero como esto no aplica para todas las cuentas lo mejor es tener dos constructores: uno que no necesite el saldo (comenzaría con saldo cero) y otro que pida el saldo (para cuando sea diferente de cero). El diagrama de clases para Cuenta, con los dos constructores, es:



Constructor por defecto

Incluso cuando no se tienen constructores en una clase, es posible crear objetos de esa clase (con *new*). Esto es posible porque Java, cuando una clase no tiene constructor, proporciona un **constructor por defecto**.

Este es un constructor que no recibe parámetros y que da unos valores iniciales a los atributos del objeto, así:

- Los atributos numéricos los inicializa en cero.
- Los atributos *boolean* los inicializa en *false*.
- Los atributos *char* los inicializa en el carácter nulo.
- Los atributos que sean de tipos referenciados los inicializa en *null*.

Sin embargo, si se escribe algún constructor en la clase, Java ya **no** proporciona el constructor por defecto y se debe usar el constructor que se escribió para crear los objetos.

Por ejemplo:

Carro
placa: String modelo: int
acelerar(): void frenar(): void



Moto
placa: String modelo: int
Moto(placa: String, modelo: int) acelerar(): void frenar(): void



<p>Para construir un objeto Carro se puede usar el constructor por defecto:</p> <pre>Carro carro1 = new Carro();</pre>	<p>Para construir un objeto Moto NO se puede usar el constructor por defecto:</p> <p>Moto moto1 = new Moto();</p> <p>Se debe usar el constructor definido en la clase:</p> <pre>Moto moto1 = new Moto("qwe-234", 2011);</pre>
<p>El objeto creado queda con los valores que define Java:</p> <p>carro1</p> <pre>placa= null modelo=0</pre>	<p>El objeto creado queda con los valores que se enviaron en el constructor:</p> <p>moto1</p> <pre>placa= "qwe-234" modelo=2011</pre>