Uso de Métodos Java

Este capítulo presenta la información que usted necesita para agregar métodos a una clase. También explica cómo usar métodos constructores para instanciar objetos.

Objetivos

- Definir el comportamiento de una clase a través de métodos.
- Declarar un método como público para obtener encapsulamiento.
- Pasar argumentos a un método.
- Describir la signatura, o firma, de un método.
- Explicar cómo funciona la sobrecarga de métodos.
- Utilizar un constructor para instanciar un objeto.

Métodos - Panorama General

Cuando se ejecuta un programa, se ejecuta una serie de instrucciones para alcanzar un cierto objetivo. Los métodos, término Java para las operaciones, realzan este concepto al permitir separar las instrucciones en bloques de código que se pueden ejecutar de forma independiente. Los bloques de código dentro de un mismo método tienen un orden obligado (el primer bloque se ejecuta primero) mientras que los métodos individuales pueden ser invocados en el momento que se considere apropiado.

Métodos - Panorama General

- Un método puede invocar a otro, lo que significa que el primer método hace una pausa y el segundo comienza. Cuando el segundo termina, el primero retoma en el punto en que había realizado la pausa.
- No hay límites en cuanto a la cantidad de llamadas a otros métodos que un método puede hacer o respecto a la cantidad de niveles que los métodos pueden llamarse entre sí. Un método puede incluso llamarse a sí mismo y a esta técnica se la conoce como *recursión*.
- Todos los programas utilizan métodos. Las aplicaciones comienzan con un método llamado main . Este es responsable de la invocación de otros métodos y de la coordinación del flujo del resto del programa.

¿Por qué son necesarios los métodos?

Si main() es un método que se necesita en cualquier aplicación y además un método puede contener cualquier secuencia de instrucciones, ¿usted necesita otros métodos?
La respuesta es "sí", por las razones que siguen:

- Los métodos hacen a los programas más fáciles de leer y de modificar.
- Los métodos hacen que el desarrollo y mantenimiento sea más rápido.
- Los métodos son fundamentales para que el software sea reutilizable.
- Los métodos evitan la duplicación de código.
- Los métodos son necesarios para modificar variables privadas (debido a la encapsulación).

Declaración de Métodos - Sintaxis

En un programa Java, los métodos se escriben dentro del cuerpo de las clases. Las tareas que estos ejecutan deberían ser relevantes para el propósito de la clase.

La estructura de un método es la siguiente:

[modificadores]- tipo de retorno – identificador de método - ([parámetros]) - {cuerpo_del_método}

Declaración de Métodos - Sintaxis

- **modificadores** - Varias palabras clave de Java se pueden usar para modificar la forma en que los métodos se almacenan o se ejecutan. public y static son modificadores del método main(). Si usted no escribe ningún modificador, se utilizará el modificador por defecto .
  Todas las variables son privadas, debido a la encapsulación. Por lo tanto, usted necesitará crear métodos públicos que puedan modificar estas variables.
- **tipo_de_retorno** - Algunos métodos se utilizan para calcular un valor o requerir un objeto para guardar en una variable. Dichos métodos deben devolver ese valor al lugar desde donde son invocados. El tipo de retorno de un método determina de qué manera el método será utilizado por otros. Los métodos que no devuelven un valor deben tener void como tipo_de_retorno. El valor que devuelve un método debe ser del tipo declarado. Solamente se puede devolver un valor.

Declaración de Métodos - Sintaxis

- **identificador_de_método** - Es el nombre que se usa para invocar al método. Los identificadores de método deberían ser formas verbales significativas que permitan inferir su propósito a partir del contexto en que se usan.
- **parámetros** - (Opcional) Los parámetros permiten pasar variables a un método de manera que la tarea que lleva a cabo pueda trabajar sobre datos específicos cada vez que es utilizado y de esta forma hacer genéricas a las variables. Usted puede pasar un número ilimitado de parámetros, los cuales pueden ser de cualquier tipo primitivo o referencia. Cada parámetro debe definirse con un tipo y un identificador:
  (tipo - identificador_variable)- Ejemplo: (int myInt)
  Si se pasa más de un argumento, se debe separar cada par por una coma, como en el siguiente ejemplo:
  (int myInt, long myLong)
- **cuerpo_del_método** - El cuerpo del método es el bloque de código entre llaves que contiene la secuencia de instrucciones que ejecutan la tarea.

Declaración de Métodos - Ejemplo

```
1  public abstract class Person
2  {
3      private String firstName;
4      private String lastName;
5      private String initial;
6      private int ID;
7      private Address homeAddress;

8      public String getName(){return firstName;}
9      public void setFirstName(String fName){}
10     public void setLastName(String lName){}
11     public void setInitial(String init){}
12     public int getID(){return ID;}
13     public void setID(String id){}
14     public Address getAddress(){return homeAddress;}
15     public void setAddress(Address addr){}
16 }
```

Invocación de Métodos

Para invocar un método, utilice la siguiente sintaxis. Asegúrese de que la invocación se encuentra dentro del cuerpo de algún método de clase.

object_reference.method_identifier([arguments])

Invocación de Métodos - Ejemplo

```
1  class VoidMethodInvocation
2  {
3    public static void main (String args[])
4    {
5      ClassOne one = new ClassOne ();
6      one.returnNothing();
7    }
8  }
9  class ClassOne
10 {
11   public void returnNothing()
12  {
13     //lo que el método hace
14   }
15  }
```

Invocación de Métodos - Tipos de Métodos

Métodos de Objeto

La forma estándar de declarar un método obliga a que este sea precedido con la referencia al objeto y el operador punto ("."). (los métodos static, están exentos de esto).

Cuando se invoca un método de un objeto, se debe preceder el nombre del método por la variable referencia

Invocación de Métodos - Tipos de Métodos

Métodos static

Hasta este momento en este módulo, todos los métodos han sido métodos de objetos estándar. Estos requieren que el objeto del método sea instanciado antes de que el método se ejecute. Los métodos estáticos pueden ser ejecutados sin previa instanciación de un objeto. Un método estático, indicado por el modificador static en su declaración, debería estar precedido por el nombre de su clase y el operador punto ("."). El modificador static debe ser usado con utilidades genéricas o con métodos que podrían ser llamados en cualquier momento, sin necesidad de instanciar algún objeto de la clase.

El método main siempre debe ser estático. El intérprete de la JVM debe poder ejecutar main sin haber instanciado la clase en la cual este fue escrito.

Invocación de Métodos - Tipos de Métodos

El ejemplo a continuación usa la clase Math para obtener un número aleatorio sin crear ningún objeto de la clase Math.

```
1  class StaticExample
2
3  {
4     public static void main (string args[])
5     {
6       double rand = Math.random();
7       System.out.println("A random number: " + rand);
8     }
9  }
```

La referencia “this”

Cuando un método de un objeto invoca otro método del mismo objeto, no es necesario poner una referencia delante. En el ejemplo, method1 invoca a method2 (línea 5) sin referencia a objeto. Por lo tanto, se asume que method2 es parte de la clase Example.

Tal sintaxis es permitida porque cuando el compilador encuentra un método sin referencia verifica la clase local en búsqueda de un método que coincida. Si lo encuentra, entonces agrega la referencia this a la invocación del método.

this es una palabra clave que significa "**referencia al mismo objeto**".

La referencia “this” - Ejemplo

```
1  class Example
2  {
3      void method1()
4      {
5          this.method2();
6      }
7      void method2()
8      {
9          //whatever method2 does
10     }
11 }
```

Pasaje de argumentos

Si especifica argumentos cuando declara un método, entonces debe dar valores para esos argumentos cuando invoca el método. Los argumentos se deben declarar de la misma forma que en la declaración del método: entre los paréntesis del método, como pares tipo-variable, tal como se muestra a continuación:

nombre_método (tipo identificador_variable)

Ejemplo: myMethod (int myInt)

Si hay varios argumentos, separe cada par con comas:

Ejemplo: myMethod (int myInt, long myLong)

Sobrecarga de métodos

- La sobrecarga de métodos significa que dos o más métodos en la misma clase y con el mismo nombre tienen diferentes argumentos

- A la combinación del nombre del método y sus argumentos se le denomina la **"signatura del método"** (o firma del método).

Sobrecarga de métodos

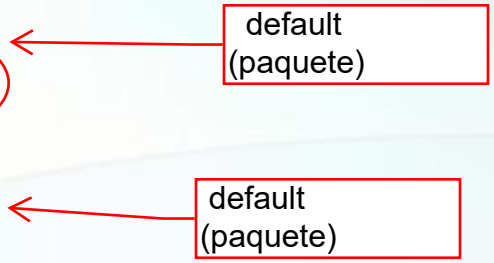
Definición:

En el lenguaje de programación Java, pueden existir varios métodos en una clase con el mismo nombre, pero con diferentes argumentos. Esto se denomina ***sobrecarga de métodos*** y es posible debido a que los métodos se identifican no sólo por su nombre sino también por los argumentos que le son pasados. La combinación de nombre y lista de parámetros de un método se conoce como la ***signatura del método*** (o firma del método)

Sobrecarga de métodos - Ejemplo

```
1  class Example
2  {
3      public static void main (String args[])
4      {
5          StoreVal store = new StoreVal();
6          store.setValue(9);
7          int i = store.getValuePlus(2);
8          double d = store.getValuePlus(3.6);
9      }
10 }
```

Sobrecarga de métodos - Ejemplo




```
11 class StoreVal
12 {
13    int value;
14    void setValue(int val)
15    {
16        value = val;
17    }
18    int getValuePlus(int more)
19    {
20        return (value + more);
21    }
22    double getValuePlus(double more)
23    {
24        return (value + more);
25    } }
```

Constructores

- Los constructores son métodos que permite tener mayor control cuando se inicializa un objeto.

- Los constructores siguen todas las reglas de los métodos para su formación excepto:

  -No tienen un tipo de retorno

  -El Nombre de un constructor es siempre el mismo que el del nombre de la clase.

Constructores

Definición:

En el proceso de instanciación, la palabra clave new automáticamente llama a un constructor cada vez que se crea un objeto. Los constructores son métodos especiales que se pueden utilizar para tener mayor control sobre el proceso de instanciación. Los constructores permiten especificar, al momento de instanciación de un objeto, qué variables se inicializan y qué valores se asignan a ellas. Si no se especifica ninguna información de inicialización cuando se crea un objeto, todas las variables se inicializan a sus valores por omisión (por ejemplo null, para variables String).

Constructores

Aunque los constructores son "especiales", siguen siendo métodos y cumplen con todas las reglas de estos, excepto:

- Los constructores no tienen tipo de retorno.
- El nombre de un constructor es siempre el mismo que el nombre de la clase.
- Usted necesita hacer dos cosas para utilizar un constructor: crearlo dentro de la declaración de clase e instanciar la clase utilizando la palabra reservada new, que invocará al constructor.

Cómo Escribir un Constructor

Para escribir un constructor para una clase, el único requisito es incluir el nombre del constructor dentro del bloque de la clase. Se puede incluir una declaración dentro del bloque de código de los parámetros para inicializar variables de esa clase con ciertos valores.

```
declaración_clase
{
nombre_constructor([argumentos])
{
[sentencias_inicialización_variables];
}
}
```

Cómo Escribir un Constructor - Ejemplo

El ejemplo crea un constructor que da nombre a un artículo ordenado por un cliente, en este caso una camisa (shirt).
El constructor especifica que ese artículo es una camisa Oxford.
Esto es, el constructor asigna "Oxford Shirt“ a la variable type del objeto Shirt.

```
1  class Shirt
2  {
3      String type;
4      // Shirt() es el constructor
5      Shirt()
6      {
7          type = "Oxford Shirt";
8      }
9  }
```

Utilización de un Constructor

Para crear un objeto, utilice la siguiente sintaxis:

nombre_clase variable_referencia = new constructor([arguments])

El siguiente ejemplo usa el constructor definido en el ejemplo anterior para instanciar una variable referencia, myShirt, que es una referencia a un objeto Shirt: **Shirt myShirt = new Shirt();**

El ejemplo que sigue utiliza un constructor diferente Shirt(String differentType), que tiene un argumento String. Como en el ejemplo anterior, en este ejemplo se instancia la variable referencia myShirt, que es una referencia a un objeto Shirt, pero además se pasa un String al parámetro differentType para clasificarlo como "Dress Shirt": **Shirt myShirt = new Shirt("Dress Shirt");**

Sobrecarga de Constructores

- Al igual que los otros métodos, los constructores pueden sobrecargarse. Esto brinda una gran variedad de formas para crear objetos de una misma clase. En el ejemplo hay dos constructores: Shirt() y Shirt(String differentType).

- Esta sobrecarga se permite debido a que los dos métodos tienen diferentes listas de parámetros y diferentes signaturas. La clase Shirt vista anteriormente puede ahora instanciarse utilizando dos constructores. Este ejemplo crea un nuevo objeto Shirt usando el primer constructor, Shirt().

Sobrecarga de Constructores

```
class Wardrobe
 {
  // uso del constructor por defecto de la clase Shirt
 Shirt myShirt = new Shirt();
 }
```

El siguiente ejemplo crea un nuevo objeto Shirt con el tipo "Dress Shirt", utilizando el segundo constructor, Shirt(String differentType).

```
class Wardrobe
{
Shirt myShirt = new Shirt("Dress Shirt");
}
```

El constructor por defecto

Toda clase debe tener al menos un constructor. Si el compilador del lenguaje de programación Java encuentra una clase que no tiene constructor, insertará uno. Este es un constructor muy simple que se agrega sólo a los efectos de satisfacer las restricciones sintácticas del compilador. Se lo denomina el constructor por defecto.

Considere la siguiente clase Sweater:

```
class Sweater
{
String name;
}
```

El constructor por defecto

El compilador agregará el constructor por defecto, tal como si la clase hubiera sido declarada como se presenta en el siguiente ejemplo:

```
1  class Sweater
2  {
3      String name;
4
5      // the default constructor
6      public Sweater() { }
9  }
```

Estructura del código

Instrucciones y Bloques de Código

Los métodos se escriben en series de instrucciones, siendo cada instrucción una acción pequeña y lógica que aporta en la dirección de la acción del método.

En el lenguaje de programación Java las instrucciones siempre terminan con un punto y coma (;), Los límites de una estructura de código (una clase o método, por ejemplo) se marcan con llaves ({}). Las llaves se requieren para delimitar clases y métodos. Los grupos de instrucciones encerrados entre tales llaves se denominan bloques de código. Alguno de los comandos Java usados para controlar el flujo de un programa de una sentencia a otra pueden ser realzados utilizando tales bloques de código.

Estructura del código

Instrucciones y Bloques de Código

Los métodos y las variables de una clase se deben especificar dentro de la declaración de la clase (dentro de las llaves para esa declaración).

Un bloque de código es sintácticamente equivalente a una instrucción simple. Usted puede escribir un bloque de código en todo lugar que pueda escribir una instrucción.

Estructura del código

Espacios

Espacio es el término usado para describir a aquellos caracteres dentro del código fuente que no tienen influencia en el programa más que para mejorar su apariencia. Los espacios no afectan para nada el programa, los ejemplos siguientes son equivalentes:

Ejemplo 1:

```
1. {
2. i=6+9;
3. j=i-3;
4. }
```

Estructura del código

Ejemplo 2:

```
1. {
2. i = 6 + 9;
3.
4. j = i - 3;
5. }
```

Puesto que Java es un lenguaje de programación de formato libre, se puede escribir todas las instrucciones de un programa en una sola línea o expandir una instrucción sobre varias líneas. Sin embargo, ambas opciones hacen al código difícil de leer e interpretar y por lo tanto de mantener y reutilizar. Utilice espacios consistentemente para alinear las instrucciones dentro de su bloque de código, separar instrucciones en líneas, posicionar las llaves, etc. y de esta forma hacer su código fácil de leer y mantener.

Resumen del Capítulo

Los siguientes términos brindan un resumen de los principales conceptos de este módulo. ¿Puede usted dar breves definiciones (tres a cinco palabras) de dichos términos?

- Método
- Recursión
- main
- Parámetros
- this
- Sobrecarga
- Constructor
- static

.Bibliografía .

- Migración a la Programación OO
  con tecnología Java
  SL-210 – Guía del estudiante