

**UNIVERSIDAD TECNOLÓGICA DE**

**SAN LUIS RIO COLORADO**

**ACCESO A DATOS Y MÉTODOS**

**MTRO. RODRÍGUEZ NEGRETE MIGUEL ANGEL**

**AUTOR: VICTOR MANUEL GALVAN COVARRUBIAS**

San Luis Rio Colorado, Sonora Junio, 2020

**Sintaxis de un método en Java**

Un método consta de un encabezado y un cuerpo. Para declarar el encabezado de un método, basta con escribir el tipo que retorna, seguido del nombre del método y entre paréntesis la lista de parámetros.

Class Clase1 {

void imprimir() { } **// MÉTODO VACÍO**

int sumar(int a, int b) { **// METODO QUE SUMA NUMEROS**

return a + b;

}

}

**Llamar un objeto**

Se utiliza para utilizar los métodos que hemos declarado.

Clase1 ObjetoEjemplo = new Clase1();

ejemplo.imprimir(); // **LLAMAMOS AL MÉTODO VACÍO**

int resultado1 = ejemplo.sumar(4,6); // **LLAMAMOS AL METODO QUE SUMA NUMEROS**

int resultado2 = ejemplo.sumar(3,3); // **LLAMAMOS AL METODO QUE SUMA NUMEROS**

**Tipos de métodos utilizados en java**

**Método main**

Existe un método especial que permite iniciar la ejecución de un programa java. Este método debe llamarse main, ser estático y público. Además, debe recibir como parámetro un array de Strings, para recibir los argumentos que se le envíen al iniciar la ejecución.

public static void main(String[] args) {

}

**Método público**

Para que un método, no tenga restricciones de acceso, basta anteponer la palabra public a la declaración del método.

Public Saludo(String nombre){

System.out.println(“ Bienvenido: ”+nombre+” Saludos!!!!!”);

}

**Métodos miembro**

Como se ve en el código del ejemplo **Llamar un objeto**, es necesario crear un objeto, antes de acceder a los métodos. Estos métodos, están asociados a los objetos. Por esto, a estos métodos, se los llama miembros del objeto.

**Métodos estáticos**

Existen un segundo tipo de métodos en Java, que están asociados a una clase, en vez de a los objetos producidos por la clase. Para crear un método de este tipo, basta con agregar la palabra static, antes del tipo de retorno del método. Hecho esto, el método quedará asociado a la clase.

public static int MetodoEjemplo(Int num1){

return num1;

}

**Métodos de instancia**

Operan sobre las variables de instancia de los objetos, pero también tienen acceso a las variables de clase. Dentro de un método de instancia, el identificador de una variable de instancia hace referencia al atributo de la instancia concreta que hace la llamada al método (suponiendo que el identificador del atributo no ha sido ocultado por el de un parámetro).

objReferencia.Metodoinstancia(parametros); **// Llamada a un método de instancia**

public void Metodoinstancia( CuentaBancaria origen ) {

this.saldo += origen.saldo;

origen.saldo = 0;

}

**Métodos de clase**

Los métodos de clase pueden trabajar con las variables de clase, pero no pueden acceder a las variables de instancia declaradas dentro de la clase, a no ser que se crea una nueva instancia y se acceda a las variables de instancia a través del nuevo objeto. Los métodos de clase también pueden ser llamados precediéndolos con el identificador de la clase, sin necesidad de utilizar el de una instancia.

Clase.Metodo(parametros); **// Llamada a un método de clase**

public static void Cuentas () {

totalCuentas++;

}

**Ejemplos de métodos con retorno y su función**



**PASO POR VALOR Y PASO POR REFERENCIA**

En Java todos los **parámetros** de los métodos se pasan por valor. Cuando se realiza la llamada a un método, los parámetros formales reservan un espacio en memoria independiente y reciben los valores de los parámetros reales.

Public Car(String modelo,String color,String marca){

// **METODO CON PASO DE PARAMETROS**

}

¿Qué consecuencias tiene el paso por valor de los parámetros?

Cuando el argumento es de tipo primitivo, el paso por valor significa que durante la ejecución del método se reserva un nuevo espacio para el parámetro formal y no se puede modificar el parámetro real durante la ejecución del método.

Cuando el argumento es de tipo referencia el paso por valor significa que no puede modificarse la referencia, pero se pueden realizar llamadas a los métodos del objeto y modificar el valor asignado a las variables miembro accesibles del objeto durante la ejecución del método.

En el siguiente ejemplo el método cambiar utiliza un parámetro de tipo primitivo (a, un valor numérico entero) y un parámetro de tipo referencia (b, un array de valores enteros):

public class Parametros {

public static void main (String [] args ) {

// paso por valor

int num1=10;

int num2=20;

int num3=30;

System.out.println("Antes: " +num1+ " " +num2+" "+num2);

cambiar(n, v);

System.out.println("Después: " +num1+ " " +num2+" "+num3);

}

public static void cambiar (int num1,int num2,int num3) {

//paso por referencia

num1= 50;

num2= 60;

num3= 70;

System.out.println("Dentro: " +num1+ " " +num2+" "+num3);

}

}

El ejemplo anterior origina la siguiente salida por pantalla:

Antes: 10 20 30

Dentro: 50 60 70

Después: 10 60 70

**Modificadores de Acceso**

El primer componente corresponde al modificador de acceso, que puede ser *public* o *private*, éste es opcional, si no ponemos nada, se asume el modificador de acceso por defecto, el segundo componente es el modificador que puede ser *final* o *static* (o ambas), también es opcional.

Un método o función siempre retorna algo, por lo tanto, es obligatorio declararle un tipo (el tercer componente de la sintaxis anterior), puede ser entero (int), booleano (boolean), o cualquiera que consideremos, incluso tipos complejos, luego debemos darle un nombre a dicha función, para poder identificarla y llamarla durante la ejecución, después al interior de paréntesis, podemos poner los argumentos o parámetros. Luego de la definición de la "firma" del método, se define su funcionamiento entre llaves; todo lo que esté dentro de las llaves es parte del cuerpo del método y éste se ejecuta hasta llegar a una instrucción return.

public class Persona {

String nombre;

int edad;

public RegistrarPersona(String nombre, int edad) {

this.nombre = nombre;

this.edad = edad;

}

public void imprimirDatosPersonales() {

System.out.println(" Nombre: "+nombre);

System.out.println(" Edad: "+edad);

}

public final boolean esMayor() {

boolean flag=true;

if (edad >= 18) {

flag=true;

} else{

flag=false;

return flag;

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Private* | *Without modification* | *protected* | *public* |
| Misma clase | **+** | **+** | **+** | **+** |
| Subclase en el mismo paquete |  | **+** | **+** | **+** |
| Clase (no subclase) en el mismo paquete |  | **+** |  | **+** |
| Subclase en otro paquete |  |  | **+** | **+** |
| No subclase en otro paquete |  |  |  | **+** |