Universidad Mayor de San Simón
Facultad de Ciencias y Tecnología

Objetos y Clases

Corina Flores Villarroel

Contenido

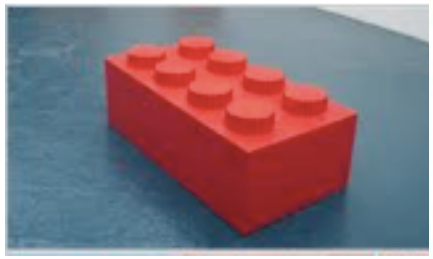
- Conceptos Básicos
- Objetos y clases

Programación Orientada a Objetos

- La Programación Orientada a Objetos, es un **paradigma** o modelo que utiliza objetos como la base de construcción. Al decir, que es un paradigma nos referimos a una serie de pautas y un estilo para que sigan los programadores.

Programación Orientada a Objetos

- La POO, se basa en cómo en el mundo real los objetos estan compuestos por otros objetos más pequeños, los cuales se pueden combinar.
-



Programación Orientada a Objetos

Principios:

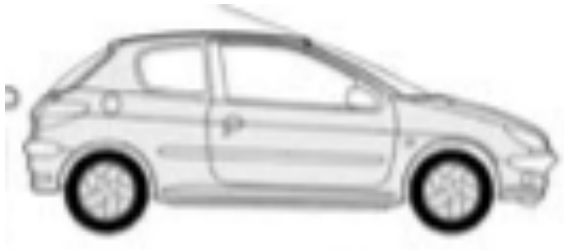
1. Abstracción, es suprimir y ocultar algunos detalles de un proceso o de un elemento, para resaltar algunos aspectos, detalles o estructuras.

La abstracción es la forma en que nuestra mente modela la realidad, formando los objetos. Por eso se crea objetos en los programas que simulen los comportamientos de los objetos de un mundo real.

Programación Orientada a Objetos

Principios:

1. Abstracción, ejemplo …



Cuando pensamos en un automóvil, no nos preocupamos de sus componentes más pequeños: el asiento, el motor, el tipo de llantas, los espejos, etc., sino, que lo vemos como una entidad u objeto, **esto es abstraer!!!** … con tal que podamos interactuar con él para satisfacer nuestra necesidad de desplazarnos por ejemplo.

Conceptos Básicos

- Programa: Conjunto de **instrucciones**, comandos, órdenes y procedimientos escritos en un **lenguaje de programación** para que una computadora pueda realizar una tarea determinada de manera exacta y rápida.

Un programa debe caracterizarse por ser:

- Correcto
- Completo
- Eficiente

Objetos

¿Qué es un objeto?

Es una entidad en la que se basa la POO.

Se denomina también como un individuo particular, o una instancia de un clase.

OBJETO

Representación específica y concreta de **UN AUTO** a partir de ese grupo de autos

Objetos

Aplicando el concepto de Abstracción a la comprensión de una cosa (objeto), en función de sus características y comportamientos

Continuando con el ej.

Características

- Color
- Marca
- Modelo
- Placa

Comportamiento

- getColor()
- mostrarKilometraje()
- Acelerar()

Objetos

Los objetos son entidades que combinan:

- **estado** (atributo),

- **comportamiento** (m

- **identidad**

auto1

ESTADO, se refiere a la asignación de valores concretos (dato
Ej.

COMPORTAMIENTO, esta definido por los metodos con que puede operar dicho objeto, ie, que operaciones se puede realizar con él.
Ej. mostrarKilometraje()

IDENTIDAD,propiedad de un objeto que lo diferencia del resto
Ej. auto1

Clases

¿Qué es una clase?
Define una **categoría de objetos** que tienen:

Clase

Características (ATRIBUTOS)

- Color
- Marca
- Modelo
- Placa

SIMILARES (comunes)

Comportamiento (METODOS)

- getColor()
- mostrarKilometraje()
- Acelerar()

Clases

¿Qué es una clase?

Una clase es una agrupación o colección de objetos que comparten características y comportamiento comunes.

La **Clase** representa una abstracción, la esencia que comparten los objetos.

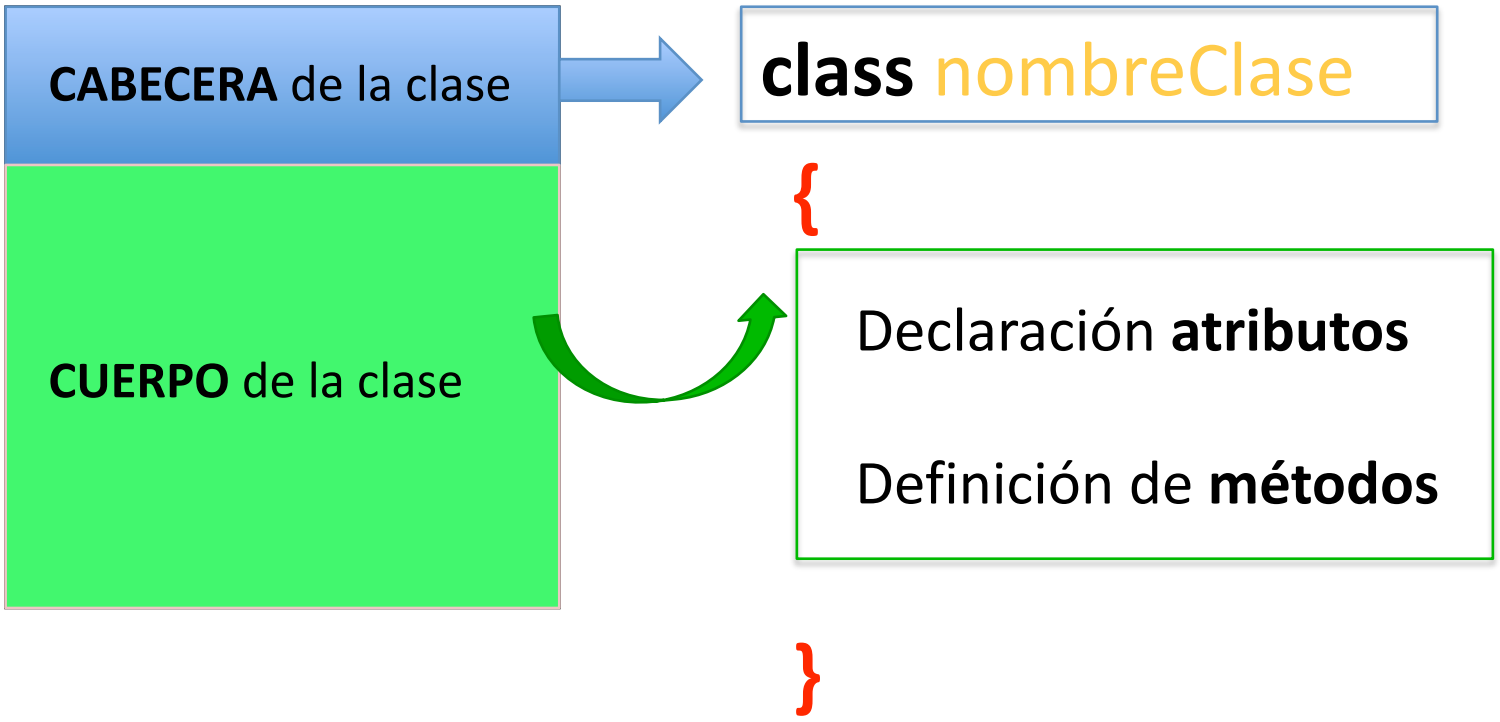
* Un objeto es un ejemplo de una clase.
* Un objeto no es una clase, y una clase no es un objeto

Las clases muestran:

- **visión externa** de comportamiento (**interface**)
- **visión interna** (**implementación**)

Representación de Clases

Una clase se representa en un **modelo** (representación gráfica) como se ve en la figura:



Declaración de Atributos

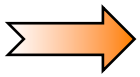
En la declaración se distinguen 3 elementos :

- El nombre del **tipo de dato** o tipo de valores que puede tomar un atributo.
- El **nombre** del atributo, denominado también *identificador o variable*.
- El separador **punto y coma** (;) que marca el fin de una declaración

Declaración de Atributos

```
class Auto{
    // Declaración de atributos
        String  color ;
        String  marca ;
        int     modelo ;
        String  placa ;

// Declaración y definición de métodos
}
```

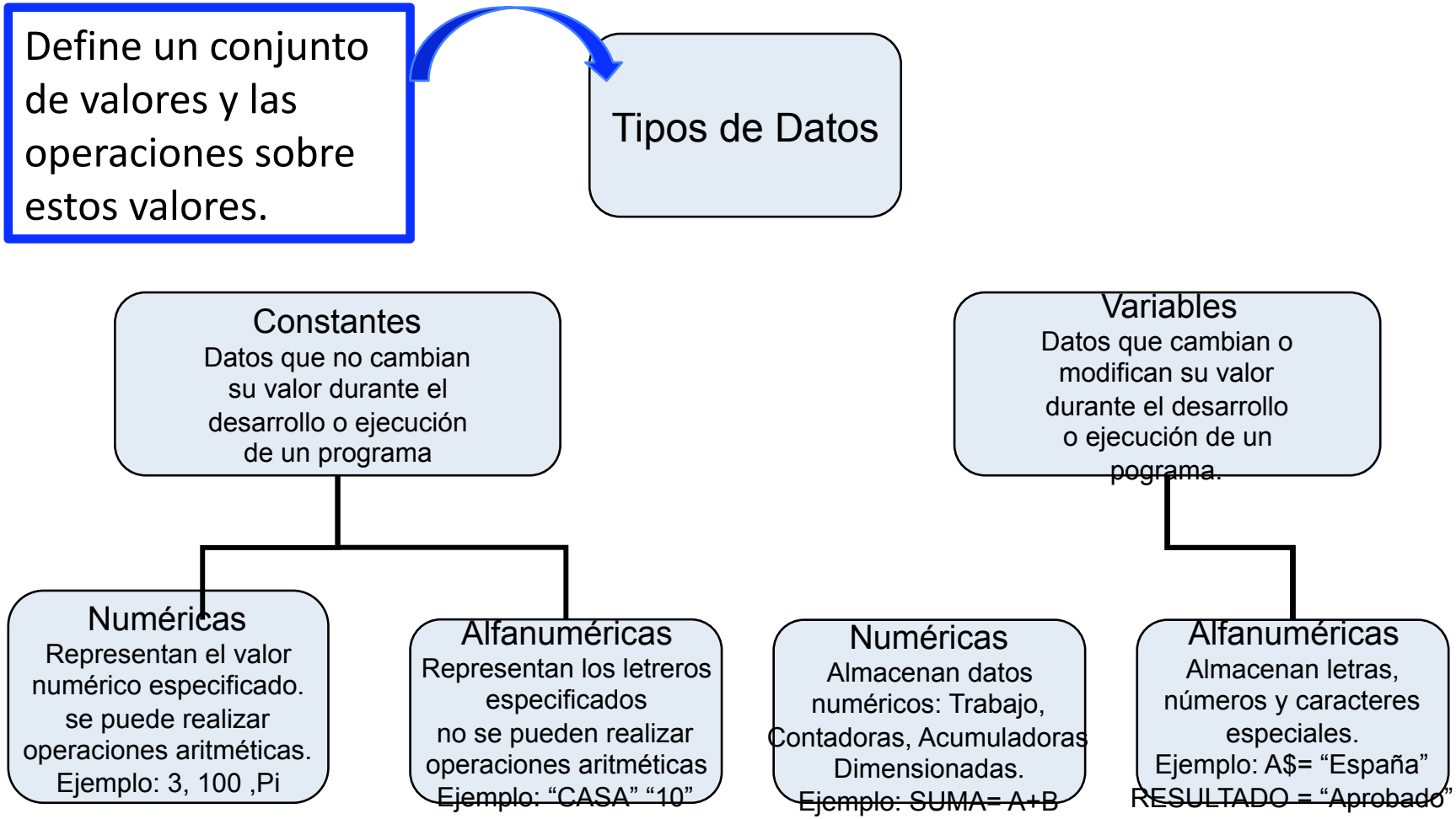


Separadores

Tipo Dato **Identificador**

Tipo de Datos

Define un conjunto de valores y las operaciones sobre estos valores.



Tipo de Dato: Constante

- Datos que no cambian su valor durante el desarrollo o ejecución de un programa

```
double Valor_de_Pi ;

Valor_de_Pi = 3.141592 ;
```

Tipo de Dato: Variable

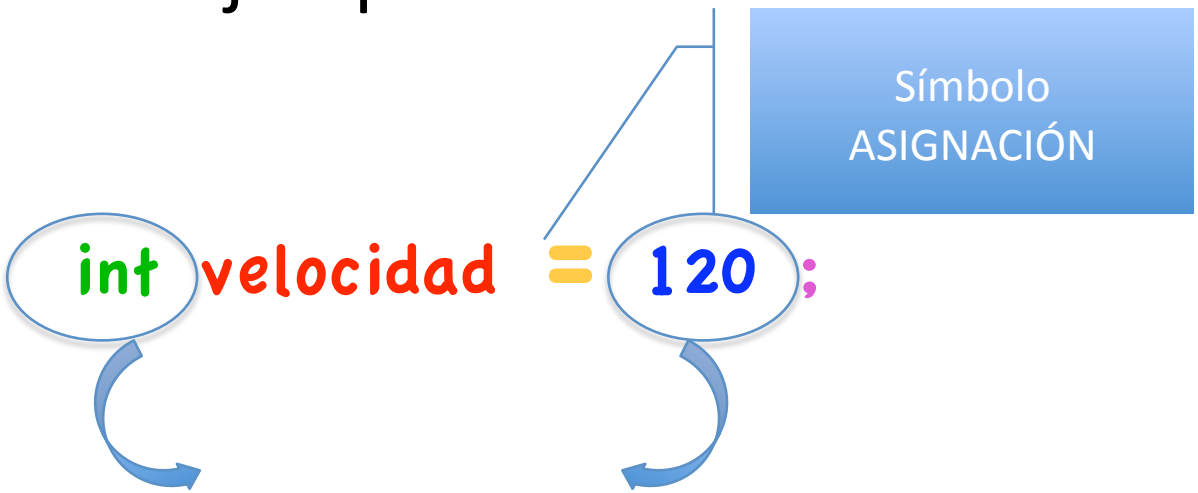
- Datos que cambian o modifican su valor durante el desarrollo o ejecución de un programa.

```
int  velocidad ;
     velocidad =  120 ;
```

Estado de un objeto

Tipo de Objeto: Asignación

- Las variables o llamadas también como contenedoras tienen la capacidad de recibir valores de acuerdo al tipo de dato.
- Del ejemplo …

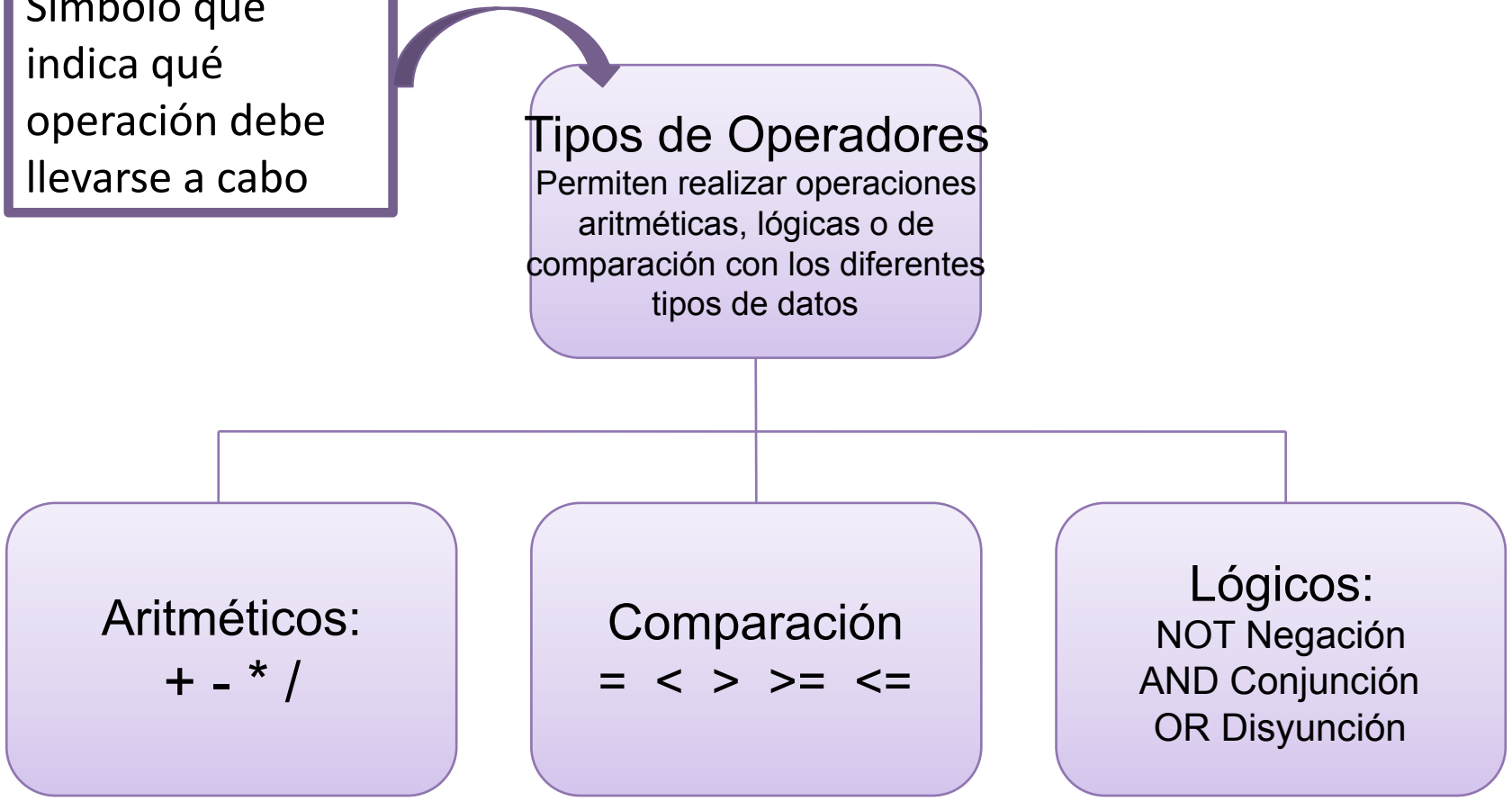


Tipo de Objeto: Asignación

TipoDato	NombVariable	SimbAsignación	Valor	
int	velocidad	=	120	;

Operadores

Símbolo que indica qué operación debe llevarse a cabo



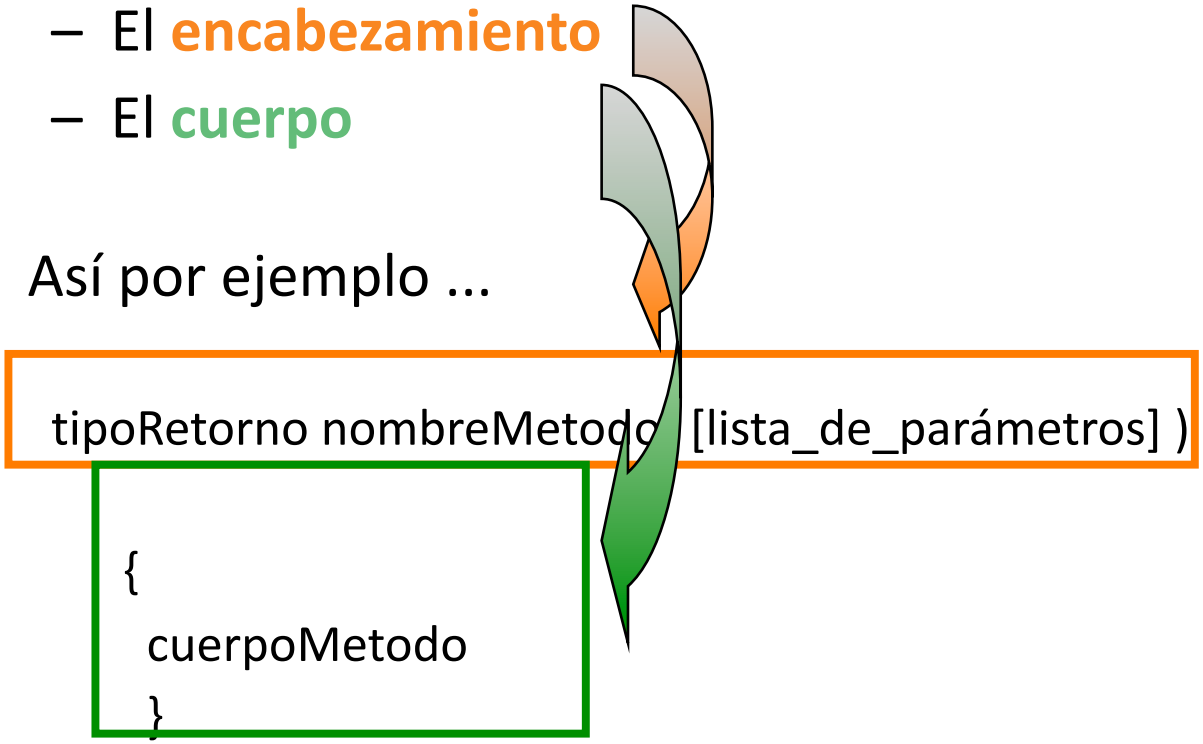
Métodos

- Definen el comportamiento de una clase
- Es la agrupación de un conjunto de instrucciones que resuelven una tarea particular.
- Una **invocación** (llamada) a un método activa ese comportamiento sobre un determinado objeto.

Métodos

- La definición de un método tiene dos partes:
  - El **encabezamiento**
  - El **cuerpo**

- Así por ejemplo ...



Métodos

- Encabezamiento de un método
  - Contiene a su vez tres elementos:
    1. **Tipo de resultado o retorno**
    2. **Nombre del método**
    3. La **lista de parámetros** que se encuentran entre **()**

Ejemplo:


```
public String getColor()
{
   // put your code here
   return color;
}
```

Métodos

- El ***tipo de retorno*** de un método dependerá de:
- Si después de procesar los datos, al método no se pide que devuelva un valor, el tipo de retorno será **void.**
- En cambio, si se pide por ejemplo, calcular el salario líquido de una persona, el tipo de retorno estará en función al resultado que se produzca del cálculo de salario, es decir: el tipo de retorno puede ser **int** o **double**
- Para devolver un valor desde un método, se utiliza la palabra clave **return**. Como última instrucción en el cuerpo del método.

Métodos

Nombre del método

- El nombre del método puede ser cualquier identificador legal en Java.
- Un identificador legal en Java es aquel que comprende una serie de caracteres. Así:

calculaSalario()
imprimeListas()
buscaUnEmpleado()

- No puede ser sin embargo, el mismo identificador que una palabra reservada o ser igual al nombre de una variable que aparece en su cuerpo del método.

Métodos

Parámetro o argumento

- Es una variable que es recibida por un método. Un parámetro influye en el comportamiento o el resultado de la ejecución del método.

Una **variable** es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato

Métodos

PARAMETROS

ACTUALES, permiten **enviar** valores al método. Se encuentran generalmente en la **llamada al método**.

FORMALES, permiten **recibir** valores para usar dentro el método. Se encuentran en la **definición del método.**

Métodos

Parámetros **formales**

```
public void setKilometraje(int kilom)
  {
     kilometraje = kilom;
  }
```

Parámetros **actuales**

```
auto1.setKilometraje(45) ;
```

Llamada a método

Existen dos formas de invocar: **interna** y **externa**

- Para llamar a un método interno, se requiere:

  nombreMetodo (lista de parámetros)

Un nombre relacionado con la tarea que realizará el método.

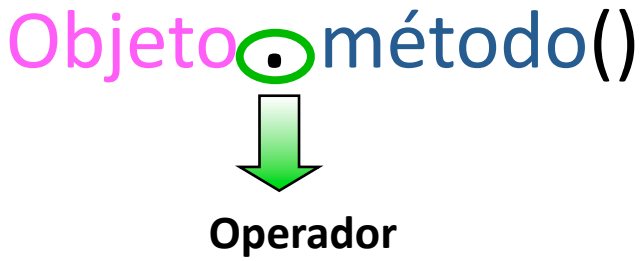
Todos los valores necesarios para ejecutar la tarea, sin especificar sus tipos

Así:

```
setValor(double respuesta) ;
```

Llamada a método

- La llamada o invocación a un **método externo** se realiza con el fin de que un objeto pueda hacer uso de los servicios que presta el método.
- Para realizar la invocación se utiliza la siguiente notación:



Objeto, es el nombre del objeto o la instancia de una clase.

Método, es el nombre del método que pertenece a la clase

Paso de parámetros

- Al hecho de que los **parámetros actuales** envian valores que los **parámetros formales** de un método reciben se denomina: **PASO DE PARAMETROS**.

- Esto sucede en el momento en el que se invoca al método. Por ej.:

auto1.setKilometraje(45);

void **setKilometraje**(int kilom)

Paso de parámetros

Es importante que:

Tanto los parametros actuales y formales de un método,deben coincidir en :

- Número
- Tipo de datos con los que se declaran
- El orden en el que aparecen en la lista

El constructor

Permite la **creación de objetos** que pertenecen a una clase.

- Tiene el mismo nombre de la clase
- No tiene un tipo de retorno
- Siempre es el primero que se escribe después de la declaración de atributos.
- Siempre es de acceso público
- Asigna valores iniciales a los atributos de la clase.

El constructor

```
public class Auto
{
   // Declaración de atributos
   private String color;
   private String marca;
   private int modelo;
   private String placa;
```

```
public Auto()
      {
            // Constructor 1
            color      = "azul";
            marca      = "ford sport";
            modelo  = 2010;
            placa      = "777CBA";
      }
```

```
public Auto(String color, String marca, int mod, String pl)
      {
            // Constructor 2
            this.color  = color;
            this.marca  = marca;
            modelo      = mod;
            placa       = pl;
      }
```

Creación de objetos

- Recordemos que … podemos crear diferentes objetos de una clase, tantos como necesitemos!!!

- Para crear un objeto o una instancia de una clase
  - Empleamos la palabra reservada **new**
  - Declarar el **identificador** que representará al objeto con el tipo de dato que hace referencia a la clase.
  - Después invocamos al **constructor** de la clase.

Creación de objetos

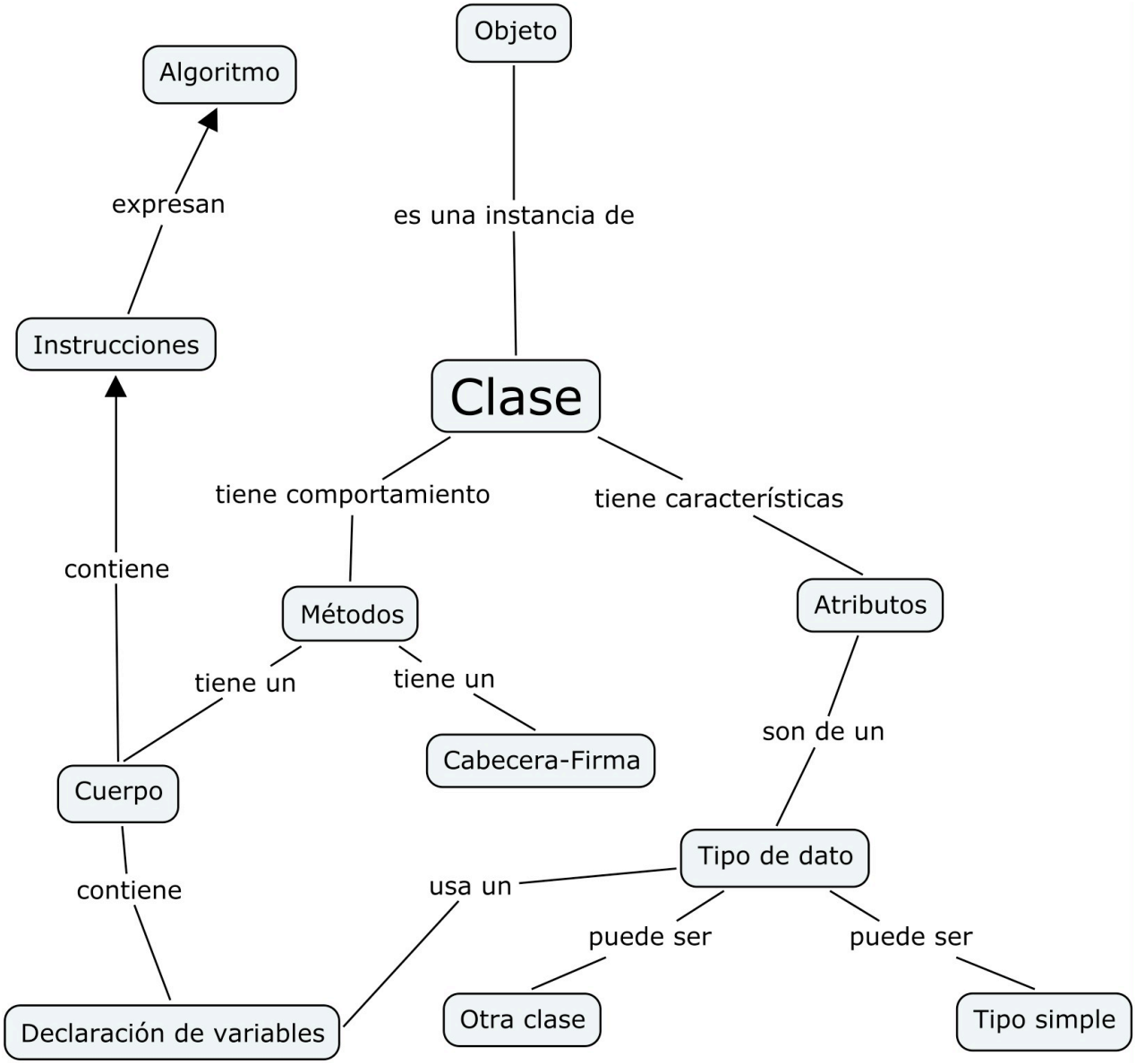
- Así por el ejemplo : Creamos un objeto de nombre **auto1**

a.) Declaramos el objeto

Auto auto1;

b.) Definimos el objeto

Auto auto1 = new Auto(“amarillo”, “Nissan”, 2006, “241LPZ”) ;


Objeto
Algoritmo
expresan
es una instancia de
Instrucciones
Clase
tiene comportamiento
tiene características
contiene
Métodos
Atributos
tiene un
tiene un
son de un
Cabecera-Firma
Cuerpo
Tipo de dato
contiene
usa un
puede ser
puede ser
Declaración de variables
Otra clase
Tipo simple