POO

PROGRAMACIÓN ORIENTADA
A OBJETOS

INTRODUCCIÓN

- La POO (o OOP de sus siglas en inglés) **es un paradigma de programación** y puede aplicarse a cualquier lenguaje.
- Disponible en la mayoría de los lenguajes tradicionales:
  - C se ha convertido en C++ y luego en JAVA
  - Pascal en Delphi
  - VB 6.0 incorporaba parte de la POO y actualemnte **VB.NET** se encuentra enteramente orientado a objetos

PARADIGMAS DE PROGRAMACIÓN

- "*Un **paradigma de programación** indica un método de realizar cómputos y la manera en que se deben estructurar y organizar las tareas que debe llevar a cabo un programa* "
- se asocian a un determinado **estilo de programación.**
- Los lenguajes de programación suelen implementar, a menudo de forma parcial, **varios** paradigmas.
- Algunos Paradigmas: POO, Estructurado, Funcional, Lógico

PARADIGMA ORIENTACIÓN A OBJETOS

- Facilidad de diseño y relación con el mundo real.
- Reutilizar piezas de código (mas allá del copy/paste)
- Ofrece mayor dominio sobre el programa liberándonos aún más de su control.
- Encapsulamiento (Ocultar el estado de los objetos)

¿QUÉ NO ES LA POO?

- No es un un lenguaje.
- De hecho las técnicas de POO pueden utilizarse en cualquier lenguaje conocido.
- La POO no suplanta otro paradigma de programación → Lo complementa

PROGRAMACIÓN ORIENTADA AL OBJETO

- Concéptos fundamentales:
  - **Clase:** especificación de un conjunto de elementos
  - **Objeto:** Elemento autónomo y con una funcionalidad concreta. Instancias concretas de una clase.
  - **Encapsulación**
  - **Herencia**
  - **Polimorfismo**

DEFINICIÓN DE CLASE

- Plantilla para definir elementos (Objetos)
- Describen a un objeto con determinados atributos y un comportamiento.
- Pueden estar directamente relacionada con otras clases.
- Ejemplo: Un vaso puede tener muchas formas y colores, pero comparten determinadas características comunes, y sirve para una determinada función.

DEFINICIÓN DE CLASE

- Si bien una maceta es parecida a un vaso, en cuanto a sus características, no así en su funcionalidad.
- Cada objeto pertenece a una determinada clase.

DEFINICIÓN DE CLASE

- Una clase puede tener distintas características.
- En el caso del vaso puede tener una determinada forma, color, material o un dibujo que lo diferencie de otro.
- A estas características llamaremos ***atributos***.

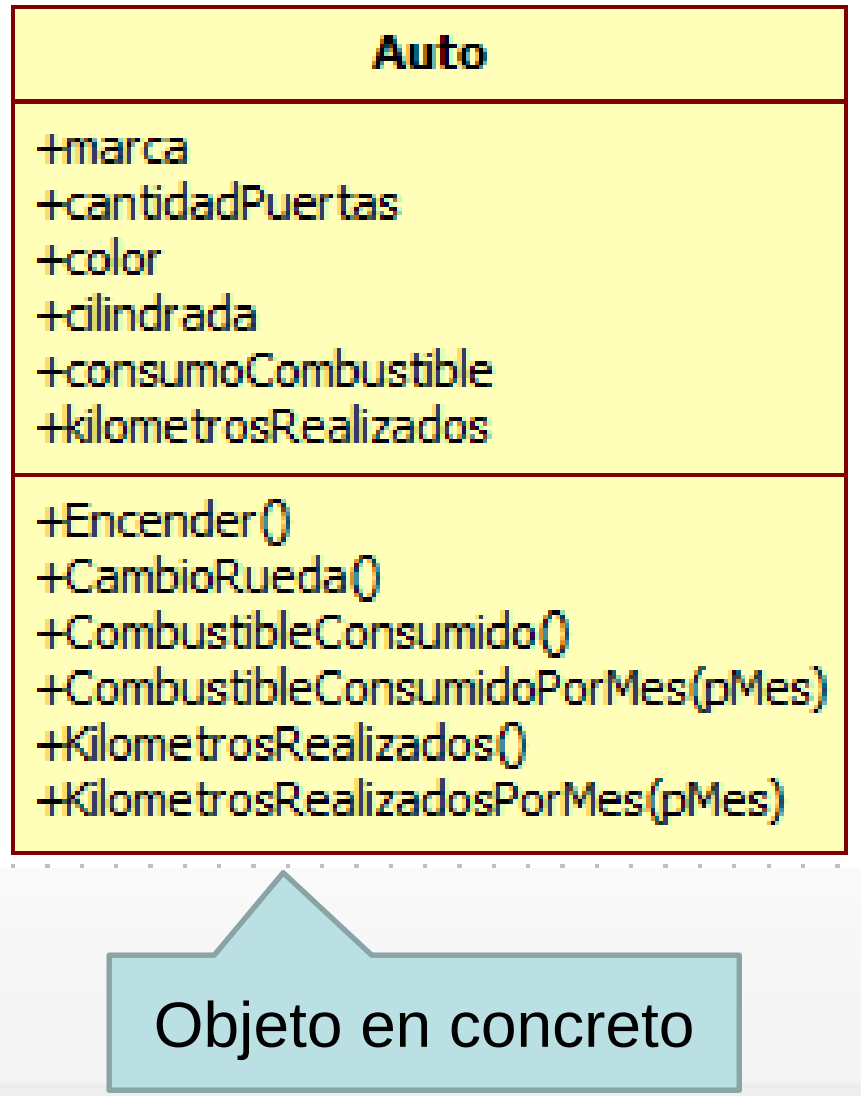
DEFINICIÓN DE OBJETO

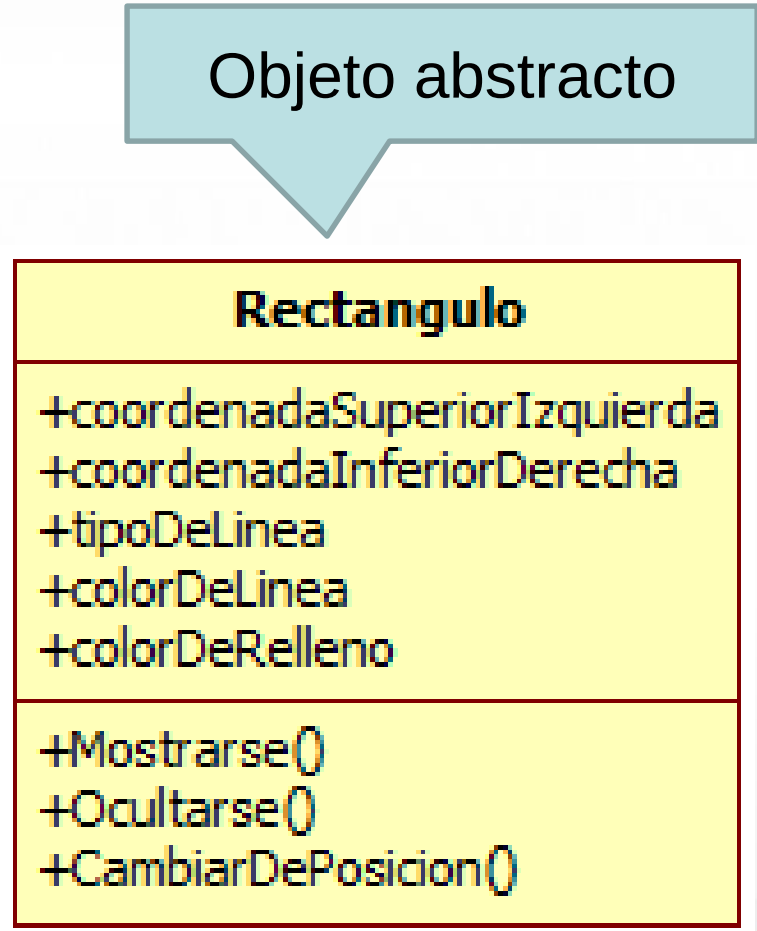
- Es un **conjunto de datos y métodos.**
- Tienen un comportamiento y un estado
- Son instancias de una clase.
- Interactúan mediante mensajes.

DEFINICIÓN DE OBJETO

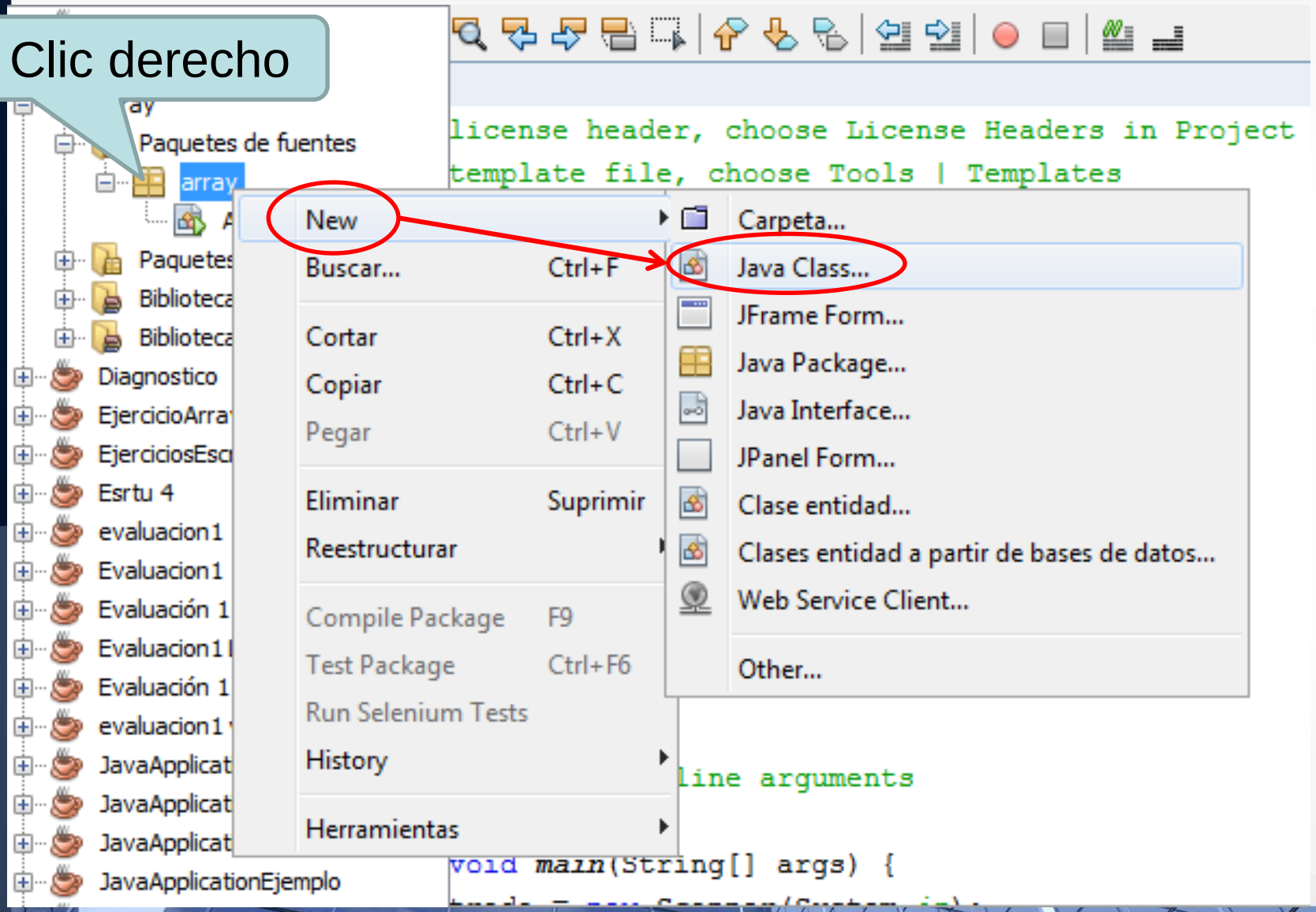
- **Datos (o propiedades):** lo que antes hemos llamado características o atributos.
- **Métodos:** comportamientos que pueden realizar.
- **IMPORTANTE:**
  - En POO, no se pueden desligar los datos de los métodos de un objeto.
  - Un objeto no debe contener datos o métodos que no le correspondan, **sólo los suyos**

EJEMPLOS DE OBJETO

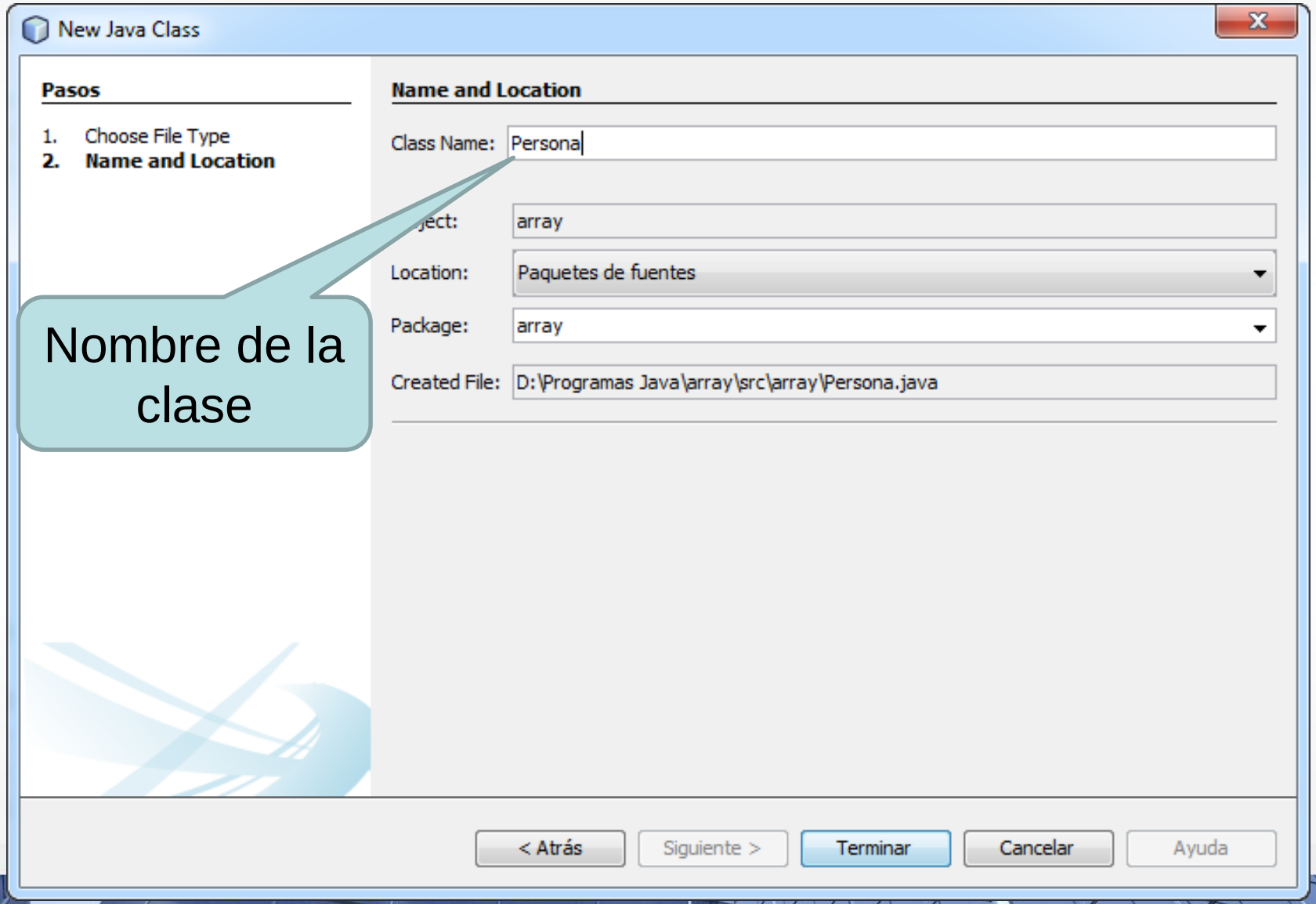




CREAR UNA CLASE EN JAVA



CREAR UNA CLASE EN JAVA



CLASE CREADA EN JAVA

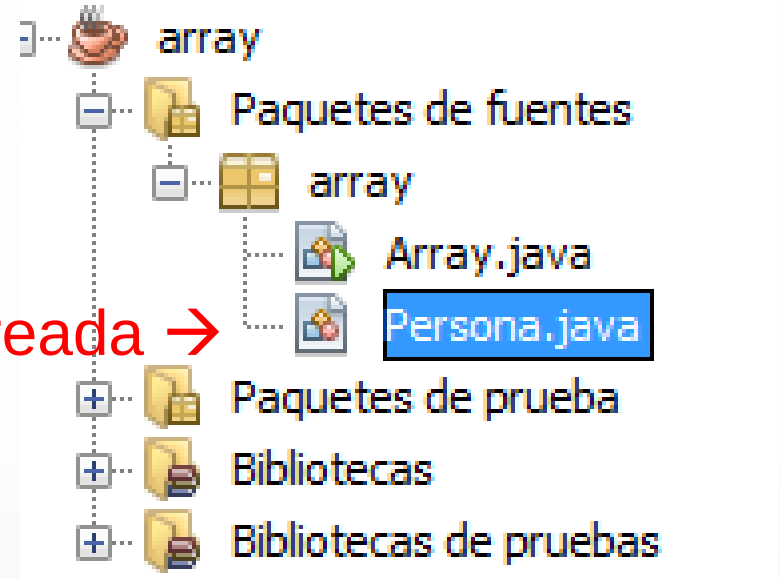
```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package array;
7
8  /**
9   *
10  * @author Rafael
11  */
12 public class Persona {
13
14 }
15
```

← Nombre del paquete donde se encuentra la clase

← Declaración de la clase

VISTA PROYECTO

Vista de la solución del proyecto una vez creada la clase



Clase creada →

DEFINICIÓN DE ENCAPSULACIÓN

- Una clase debe ser una estructura **cerrada**, no se debe poder acceder a ella si no es a través de los métodos definidos para ella.
- Por lo tanto todos los datos de una clase son **privados** y se accede a ellos mediante métodos públicos.
- Cada dato o atributo de una clase deberá tener un método **accesor** y/u otro **modificador**.

DEFINICIÓN DE ENCAPSULACIÓN

- **Ejemplo:** Observemos un Auto como un objeto.
  - ¿Cómo conocemos la temperatura del motor?
  - El Auto, dentro de sus atributos, tiene un ***termostato***.
  - En el tablero tenemos un **indicador** que se conecta al termostato, y de esa forma indica la temperatura.
  - El termostato está **oculto**, sólo el fabricante del auto sabe donde está.
  - Lo que el conductor ve es indicador que es el "**método**" por el cual puede conocer la temperatura del motor.

ENCAPSULACIÓN EN VB.NET

- En JAVA podemos **y debemos** aplicar el concépto de encapsulación de la siguiente forma.
- Definir atributos:

```
public class Persona {
    private int cedula;
    private String nombre;
    private String apellido;
    private String direccion;

}
```

ENCAPSULACIÓN

- Para los métodos accesores y modificadores debemos definir un función **get** para obtener un valor y una procedimiento **set** para asignar un valor para cada propiedad de la clase.
- Ejemplo: para la propiedad **cedula** existirá
  - La función **getCedula()**
  - El procedimiento **setCedula(int cedula)**;

METODOS ACCESORES Y MODIFICADORES (get y set)

```java
public class Persona {
    private int cedula;
    private String nombre;
    private String apellido;
    private String direccion;

    public int getCedula() {
        return cedula;
    }

    public void setCedula(int cedula) {
        this.cedula = cedula;
    }
```

```java
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
```

CONSTRUCTORES

- Para poder utilizar un objeto, previamente hemos de crearlo; esto es lo que hacemos mediante el ***constructor*** *de la clase.*
- El constructor es un método especial que se llama automáticamente cuando se crea un objeto.
- El constructor o los constructores son notados de una forma especial:

```
public Persona(){

}
```

Mismo nombre que la clase

INSTANCIAR UNA CALSE Y CREAR UN OBJETO

Para instanciar una clase, en Java se utiliza el operador new.

```
Persona person = new Persona()
```

- Le estamos diciendo al método constructor que nos devuelva un nuevo objeto Persona, y que se lo asigne a la variable **person**.

CREAR OBJETOS CON SUS DATOS

- Ahora queremos crear una persona concreta que se llame Juan Castillo, con sus correspondientes datos:

```
Persona person = new Persona();

person.setCedula(24567890);
person.setNombre("Juan");
person.setApellido("Castillo");
person.setDireccion("Artigas 462");
```

Observar como se utiliza el método modificador **set**, para establecer un valor

CREAR CONSTRUCTORES

- Habitualmente, los constructores de clase se crean de tal modo que podamos hacer las dos cosas a la vez: crear el objeto y dar valores a sus datos, veamos cómo:

```
Persona per = new Persona(24567890, "Juan", "Castillo", "Artigas 462");
```

Para esto se creará un nuevo constructor en la clase Persona:

```
public Persona(){

}

public Persona(int pCed, String pNomb, String pApe, String pDir){
    cedula = pCed;
    nombre = pNomb;
    apellido = pApe;
    direccion = pDir;
}
```

VARIOS CONSTRUCTORES

- Normalmente las clases tiene más de un constructor, de esta forma podemos crear objetos e inicializarlos de distintas formas. Así, podemos tener un constructor de la clase *Persona* que recibe solo la edad, otro la edad y la estatura, otro la edad, la estatura y el color de ojos, etc.
- El número y tipo de constructores solo depende de nuestras necesidades.