Universidad Europea

# 4

# Objetos, atributos y métodos


Ve más allá

# CONTENIDOS

- **Unidad 3: Arrays lineales**
  1. Arrays en Java
  2. Operaciones con Arrays
  3. Foreach y envoltorio Arrays
  4. **Objetos, atributos y métodos**






Universidad Europea

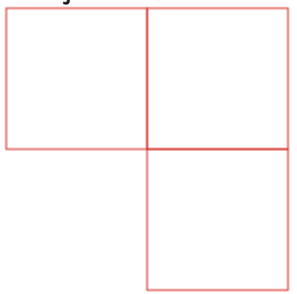
# INDICE

- Objetos, atributos y métodos
  1. Introducción
  2. Java es Orientado a Objetos
  3. Clases y Objetos
  4. Atributos
  5. Métodos
  6. Parámetros



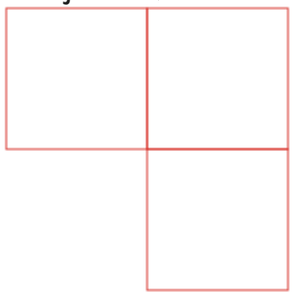


Universidad Europea



# Introducción

- Como primera aproximación a la Programación Orientada a Objetos, en este apartado vamos a hablar de los objetos, los atributos y los métodos.
- Todo lo visto hasta ahora se puede considerar programación estructurada, en decir, crear programas con diferentes sentencias de control, combinadas con otras secuenciales.
- El siguiente paso en la evolución de los lenguajes de alto nivel, es la creación de funciones que pueden ser llamadas desde diferentes lugares y que sirven para evitar repetir código y hacer nuestros programas más fáciles de entender y mantener.
- Por último, con la llegada de la Programación Orientada a Objetos, se crea el concepto de clase, así como sus dos principales componentes: atributos y métodos (las funciones anteriormente comentadas)

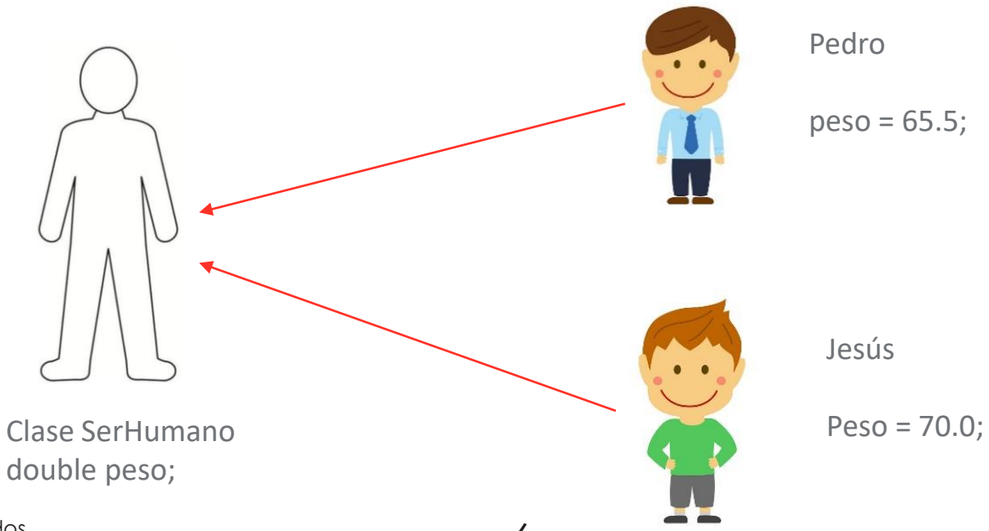


# Java es Orientado a Objetos

- Hemos podido comprobar que en Java no se puede hacer nada sin crear una clase. Esto lo descubrimos con nuestro primer programa **HolaMundo**, en el que tuvimos que crear una clase y un método. El método **main**, que hicimos en aquella ocasión, es un método especial que es el que ejecuta cuando pulsamos sobre el “run” en nuestro entorno de desarrollo.
- En una clase, se suelen incluir atributos: variables globales que pueden ser utilizadas desde cualquier lugar dentro de la clase y métodos, que es donde se programa.
- Hay otros lenguajes de programación, como C++ que pueden ejecutarse utilizando POO o no, pero en Java es obligatorio que haya, al menos, una clase.

# Clases y Objetos

- Las clases son como moldes y cada vez que hacemos un new se crea un objeto a partir de dicho molde. Se podría decir que las clases son como tipos y los objetos son ejemplos concretos de esos tipos. Por ejemplo, se puede crear la clase **SerHumano** y los objetos **Pedro** o **Jesús**. Ambos objetos comparten estructura (atributos y métodos) y se diferencian en los valores que tienen sus atributos. Por ejemplo, SerHumano podría tener el atributo peso de tipo double, que en Pedro valga 65,5 y en Jesús 70,0.

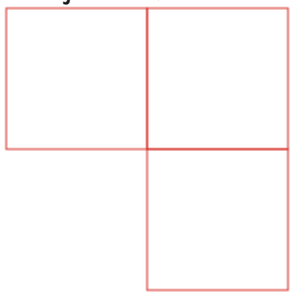


# Ejemplo

- Como ejemplo de todo lo anterior, vamos a ver cómo sería el programa de HolaMundo en su versión Orientado a Objetos.

```
1 public class HolaMundo {
2         private String saludo;
3
4         public String getSaludo() {
5                 return saludo;
6         }
7
8         public void setSaludo(String saludo) {
9                 this.saludo = saludo;
10        }
11
12        public static void main(String[] args) {
13                HolaMundo ejemplo = new HolaMundo();
14                ejemplo.setSaludo("HolaMundo");
15                System.out.println(ejemplo.getSaludo());
16        }
17
18 }
```

- En el ejemplo anterior, se crea un objeto de la clase HolaMundo al que le llamamos “ejemplo”. El operador que utilizamos para crear el objeto es el **new**. A partir de ese momento, ya se pueden utilizar los métodos de dio objeto, que son getSaludo y setSaludo. El método setSaludo sirve para darle valor al atributo “saludo”, mientras que el getSaludo sirve para recuperar el valor de dicho atributo.

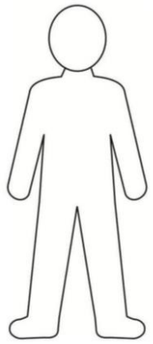


# Atributos

- Son variables que, en lugar de declararse dentro de un método, se declaran a nivel de clase. Son variables que pueden ser utilizadas por cualquier método de la clase. En el ejemplo del HolaMundo, el atributo se llama saludo y es de tipo String.

- Los atributos pueden ir acompañados de una palabra clave que determina su visibilidad fuera de la clase donde se define. En el ejemplo, al ser “private” quiere decir que no es visible desde un código que se encuentre fuera de la propia clase donde ha sido definido. Lo más habitual es que los atributos sean de este tipo.

# Atributos



```
1
2  public class SerHumano {
3
4      private double peso;
5      private double altura;
6
7      public SerHumano(double peso, double altura) {
8          this.peso = peso;
9          this.altura = altura;
10     }
11     public double getPeso() {
12         return peso;
13     }
14     public void setPeso(double peso) {
15         this.peso = peso;
16     }
17     public double getAltura() {
18         return altura;
19     }
20     public void setAltura(double altura) {
21         this.altura = altura;
22     }
23 }
```

atributos

Pedro

peso = 65.5;
altura = 170;

```
2  public class Main {
3
4      public static void main(String[] args) {
5
6          SerHumano Pedro = new SerHumano(65.5, 170);
7          System.out.println("Pedro pesa: " + Pedro.getPeso());
8          System.out.println("Pedro mide: " + Pedro.getAltura());
9
10     }
11
12 }
```

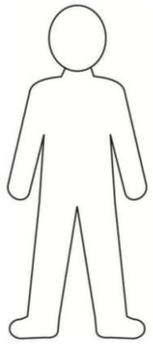
# Métodos

- Siempre programamos dentro de un método, por lo que debemos acostumbrarnos a crear métodos para poner nuestras instrucciones. Lo hemos hecho desde un principio en el main, pero lo que estamos viendo con la Programación Orientada a Objetos, es que también podemos crear otros métodos para realizar parte de nuestra tarea. Los métodos del ejemplo son de tipo “public” lo que quiere decir que se pueden utilizar fuera de la clase. La mayoría de los métodos suelen ser públicos para poder acceder a las funcionalidades del objeto.
- En Java hay dos tipos de métodos según su función:
  - **Lectores**: Son aquellos que devuelven un valor. En su interior se encuentra la instrucción “return” y a continuación el valor que devuelven. En el código de ejemplo, el método getSaludo() es de este tipo, por eso a la izquierda del nombre del método se pone el tipo de dato que devuelve (String). Este método devuelve el valor que tenga el atributo saludo.
  - **Modificadores**: Son aquellos métodos que modifican el valor de un atributo. En el código de ejemplo, el método setSaludo(String saludo) es de este tipo. Como no devuelve nada a la izquierda de su nombre se pone “void” (vacío) para indicar que no tiene “return” en su interior.

# Parámetros

- Los parámetros son los datos que reciben los métodos para poder realizar su tarea. Podemos decir que un método es un pequeño programa que tiene su entrada (parámetros) y su salida (return o valor de retorno).
- Todos los métodos se utilizan incluyendo los paréntesis tras el nombre del método. Podemos encontrarnos con dos casos:
  - Sin parámetros: En este caso los paréntesis no tienen nada en su interior. En nuestro ejemplo el método getSaludo() no tiene parámetros.
  - Con parámetros: Cuando un método necesita un dato para trabajar con él, puede obtenerlo a través de un parámetro de entrada, que se declara como si fuera una variable local de método. No se definen en el cuerpo del método como una variable local, sino entre los paréntesis tras el nombre del método. La forma de declararlo es igual que cualquier variable, es decir, primero el tipo y luego nombre del parámetro. En nuestro caso el método setSaludo tiene un parámetro de tipo String, que sirve para darle valor al atributo saludo de la clase HolaMundo.

# Métodos



Pedro

peso = 65.5;
altura = 170;

parámetros

```
1
2  public class SerHumano {
3
4      private double peso;
5      private double altura;
6
7      public SerHumano(double peso, double altura) {
8          this.peso = peso;
9          this.altura = altura;
10     }
11     public double getPeso() {
12         return peso;
13     }
14     public void setPeso(double peso) {
15         this.peso = peso;
16     }
17     public double getAltura() {
18         return altura;
19     }
20     public void setAltura(double altura) {
21         this.altura = altura;
22     }
23 }
```

Método lector

Método modificador

```
2  public class Main {
3
4      public static void main(String[] args) {
5
6              SerHumano Pedro = new SerHumano(65.5, 170);
7              System.out.println("Pedro pesa: " + Pedro.getPeso());
8              System.out.println("Pedro mide: " + Pedro.getAltura());
9
10     }
11
12 }
```

# Para practicar

En el siguiente vídeo tutorial verás gran parte de lo que hemos visto de forma teórica. Ve realizando lo que dice el tutorial y practica.

Video





Universidad Europea

# Conclusiones

- En esta unidad hemos conocido los Arrays en Java. La mayoría de los lenguajes de programación tienen esta estructura, muy útil cuando necesitamos almacenar varios valores del mismo tipo.
- Hemos visto su estructura, cómo se declaran y las características más importantes.
- Los Arrays se utilizan habitualmente para hacer alguna operación con sus elementos, como recorrerlos, imprimirlos, sumarlos o buscar algo dentro de su contenido.
- También hemos conocido una versión alternativa del bucle for, muy útil cuando queremos recorrer los elementos de una colección de datos. Además, también le hemos echado un vistazo al envoltorio Arrays, donde hay muchas herramientas útiles para trabajar con ellos.
- Por último, hemos creado nuestro primer objeto, trabajando con atributos y métodos. Todo esto como una primera aproximación a la Programación Orientada a Objetos (POO), ya que lo ampliaremos cuando llegue la unidad dedicada a la POO.

# APUNTE...

Universidad Europea

# GRACIAS

Pedro J. Camacho

Universidadeuropea.com

Ve más allá