

# Programación

## UD 7: Programación Orientada a Objetos

# Programación Orientada a Objetos

---

## Introducción

La programación Orientada a objetos se define como un paradigma de la programación, una manera de programar específica, donde se organiza el código en **unidades denominadas clases, de las cuales se crean objetos** que se relacionan entre sí para conseguir los objetivos de las aplicaciones.

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

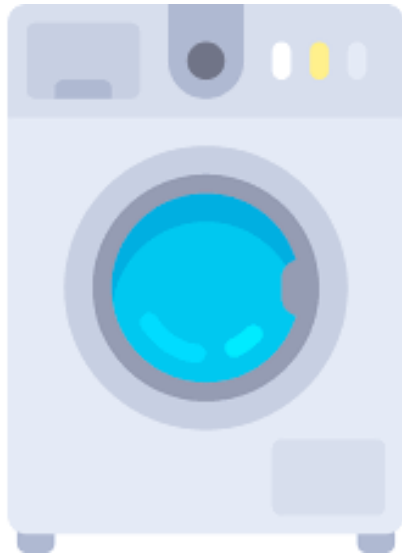
Al programar orientado a objetos tenemos que aprender a pensar cómo resolver los **problemas de una manera distinta a como se realizaba anteriormente, en la programación estructurada.**

# Programación Orientada a Objetos

---

## ¿Qué es una clase?

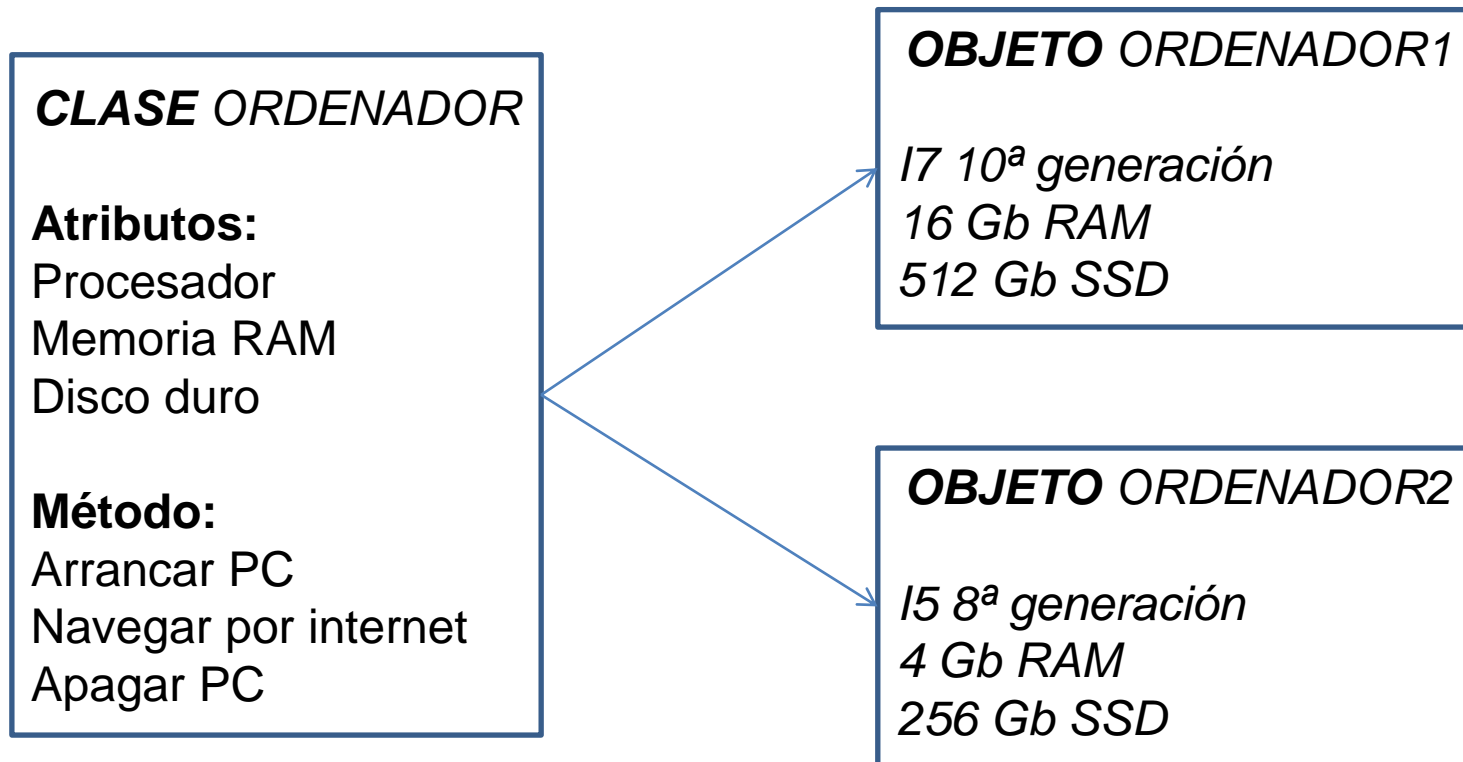
Una clase es una entidad lógica que **define a nivel teórico** las propiedades (**atributos**) y el comportamiento (**métodos**) de un objeto de la vida real.



# Programación Orientada a Objetos

## ¿Y qué es un objeto?

Un objeto es la **instancia de una clase**. Dicho de otra forma, utiliza la plantilla que le ofrece la clase para poder concretar esas características.

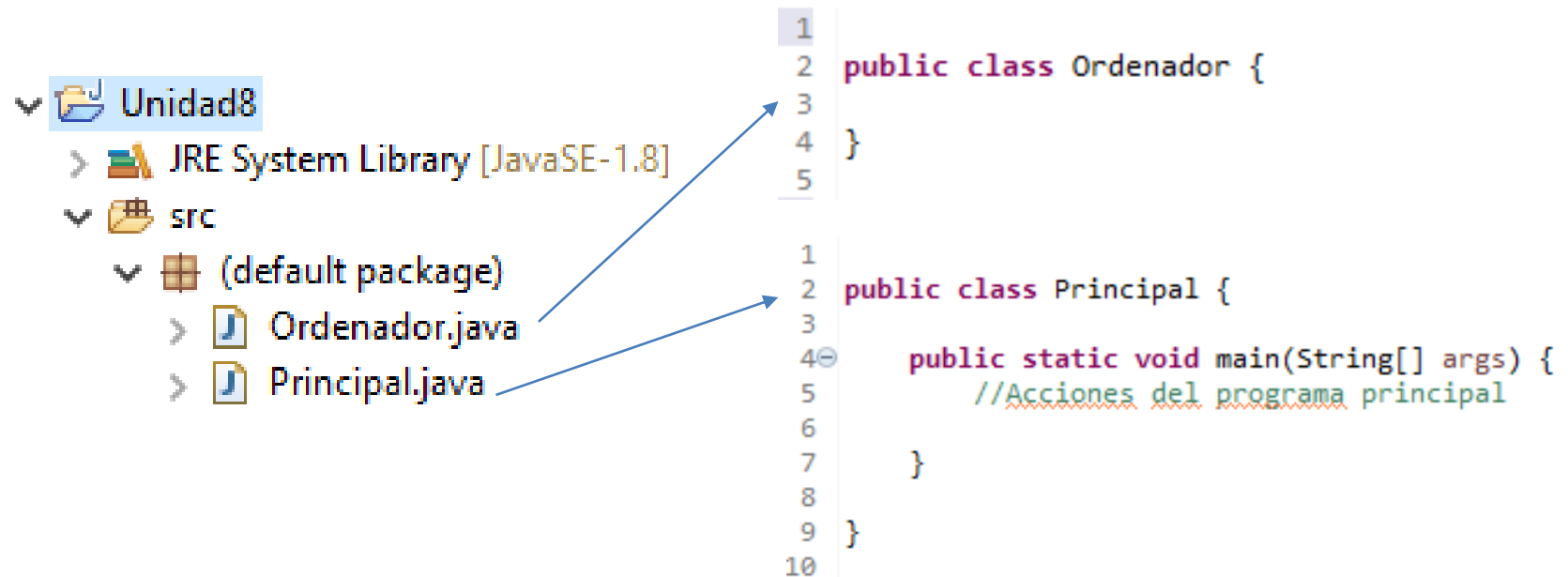


# Programación Orientada a Objetos

## ¿Cómo gestionamos las clases en Eclipse?

La forma correcta de proceder será crear, **por proyecto**:

- 1 clase principal que contendrá el método ***main()***, que se encargará de ejecutar las acciones del programa principal.
- 1 clase por cada una de las clases que sean necesarias



# Programación Orientada a Objetos

---

## Contenido de una clase en Java

```
public class Ordenador {  
  
    //Atributos del Ordenador:  
    // - Procesador  
    // - Memoria RAM  
    // - Disco duro  
  
    //1 Metodo constructor  
  
    //Resto métodos que realizarán acciones:  
    // - Encender ordenador  
    // - Navegar por internet  
    // - Apagar ordenador  
  
}
```

# Programación Orientada a Objetos

## Contenido de una clase en Java - Atributos

```
public class Ordenador {  
  
    //Atributos del Ordenador:  
    // - Procesador  
    // - Memoria RAM  
    // - Disco duro  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //1 Metodo constructor  
  
    //Resto métodos que realizarán acciones:  
    // - Encender ordenador  
    // - Navegar por internet  
    // - Apagar ordenador  
  
}
```

LOS ATRIBUTOS LOS DEFINIMOS PRECEDIDOS DE LA PALABRA **"PRIVATE"**, LUEGO VEREMOS QUÉ SIGNIFICA...

# Programación Orientada a Objetos

## Contenido de una clase en Java – Métodos

```
public class Ordenador {  
  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //1 Metodo constructor  
  
    //Resto métodos que realizarán acciones:  
    // - Encender ordenador  
    // - Navegar por internet  
    // - Apagar ordenador  
    public void encender_ordenador() {  
        System.out.println("Encendemos el ordenador con procesador: " + procesador + "y " +  
            " memoria: " + ram + "Gb y disco de " + disco + "Gb.");  
    }  
    public void navegar_internet() {  
        System.out.println("Navego por internet");  
    }  
    public void apagar_ordenador() {  
        System.out.println("Apago el ordenador");  
    }  
}
```

LOS METODOS LOS DEFINIMOS PRECEDIDOS DE LA PALABRA “**PUBLIC**”, LUEGO VEREMOS QUÉ SIGNIFICA...



# Programación Orientada a Objetos

## Contenido de una clase – Método constructor

Vale, ya hemos definido una clase, pero... ¿cómo lo instanciamos en un objeto?, lo hacemos en la clase **Principal** a través de la palabra **new** que será la que acabará llamando al **método constructor**.

```
public class Principal {  
    public static void main(String[] args) {  
        //Acciones del programa principal  
        Ordenador ordenador1 = new Ordenador(/*¿Y qué le pasamos aquí?*/);  
    }  
}
```

TIPO DE DATO: LA  
CLASE DEFINIDA

NOMBRE DE  
LA VARIABLE

Con new  
instanciamos  
la clase en un  
objeto

Parámetros de  
entrada para la  
creación del objeto:  
**llama al método  
constructor**

# Programación Orientada a Objetos

---

## Contenido de una clase – Método constructor

Un método constructor es un método especial de una clase que se llama automáticamente siempre que se declara un objeto de esa clase. Su función es inicializar el objeto **y se ha de llamar igual que la clase.**

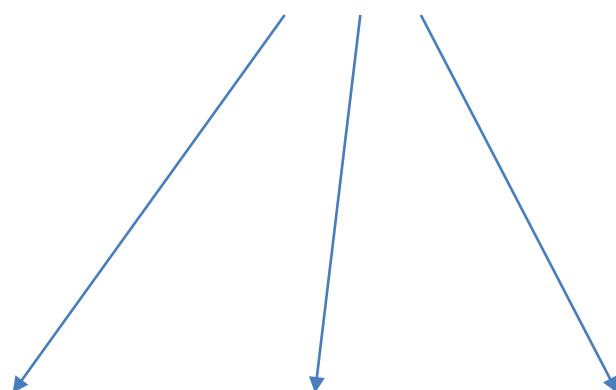
```
public class Ordenador {  
  
    //Atributos del Ordenador:  
    // - Procesador  
    // - Memoria RAM  
    // - Disco duro  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //1 Metodo constructor  
    public Ordenador(String proc_entrada, int ram_entrada, int disco_entrada) {  
        procesador = proc_entrada;  
        ram = ram_entrada;  
        disco = disco_entrada;  
    }  
}
```

# Programación Orientada a Objetos

## Contenido de una clase – Método constructor

Dentro de la clase Principal, llamamos al método constructor concretando unos parámetros para la creación del objeto:

```
public class Principal {  
    public static void main(String[] args) {  
        //Creamos un objeto: ordenador I7 de 16Gb de RAM y 512Gb de disco  
        Ordenador ordenador1 = new Ordenador("I7",16,512);  
    }  
}  
  
public class Ordenador {  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //1 Metodo constructor  
    public Ordenador(String proc_entrada, int ram_entrada, int disco_entrada) {  
        procesador = proc_entrada;  
        ram = ram_entrada;  
        disco = disco_entrada;  
    }  
}
```



# Programación Orientada a Objetos

Una vez creado el objeto, llamemos a un método...

Para llamar a un método de un objeto se hará con la nomenclatura:

<nombre del objeto>.<nombre del método>

Vamos a llamar desde **Principal** al método “encender\_ordenador()” del objeto que acabamos de crear que se llama ordenador1 y que ya declaramos anteriormente:

```
public class Principal {  
  
    public static void main(String[] args) {  
        //Creamos un objeto: ordenador I7 de 16Gb de RAM y 512Gb de disco)  
        Ordenador ordenador1 = new Ordenador("I7",16,512);  
        ordenador1.encender_ordenador();  
    }  
}
```

```
public void encender_ordenador() {  
    System.out.println("Encendemos el ordenador con procesador: " + procesador + " y " +  
        " memoria: " + ram + "Gb y disco de " + disco + "Gb.");  
}
```

# Programación Orientada a Objetos

Una vez creado el objeto, llamemos a un método...

El resultado de esta ejecución será...

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         //Creamos un objeto: ordenador I7 de 16Gb de RAM y 512Gb de disco)
6         Ordenador ordenador1 = new Ordenador("I7",16,512);
7         ordenador1.encender_ordenador();
8     }
9
10 }
```

## **OBJETO ORDENADOR1**

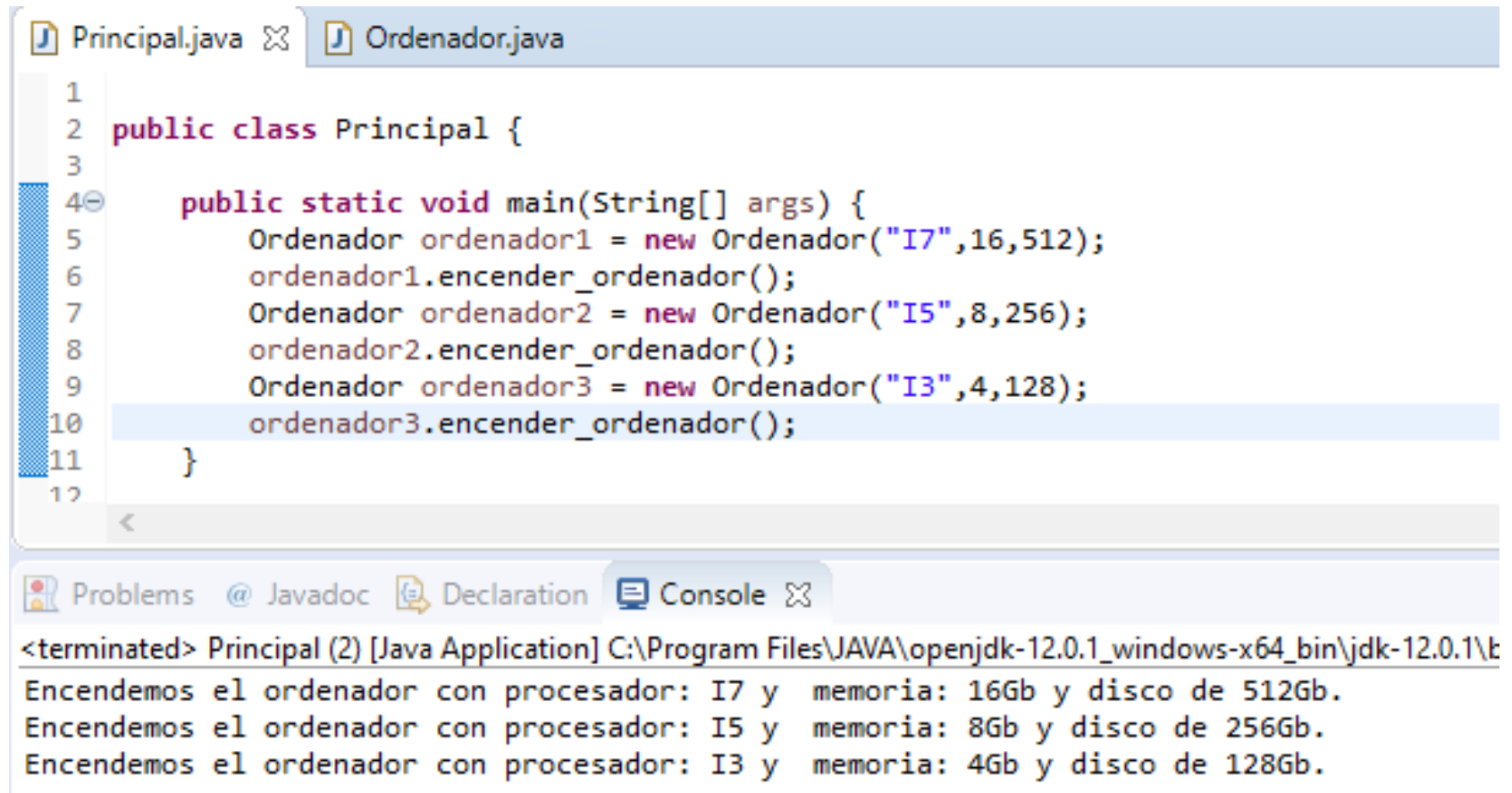
*I7 10ª generación  
16 Gb RAM  
512 Gb SSD*

Problems @ Javadoc Declaration Console X

<terminated> Principal (2) [Java Application] C:\Program Files\JAVA\openjdk-12.0.1\_windows-x64\_bin\jdk-12.0.1\bin\java.exe  
Encendemos el ordenador con procesador: I7 y memoria: 16Gb y disco de 512Gb.

# Programación Orientada a Objetos

Creemos y encendamos varios ordenadores..



The screenshot shows an IDE with two tabs: 'Principal.java' and 'Ordenador.java'. The 'Principal.java' tab is active, displaying the following code:

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         Ordenador ordenador1 = new Ordenador("I7",16,512);
6         ordenador1.encender_ordenador();
7         Ordenador ordenador2 = new Ordenador("I5",8,256);
8         ordenador2.encender_ordenador();
9         Ordenador ordenador3 = new Ordenador("I3",4,128);
10        ordenador3.encender_ordenador();
11    }
12
```

Below the code editor, the 'Console' tab is active, showing the output of the program:

```
<terminated> Principal (2) [Java Application] C:\Program Files\JAVA\openjdk-12.0.1_windows-x64_bin\jdk-12.0.1\
Encendemos el ordenador con procesador: I7 y memoria: 16Gb y disco de 512Gb.
Encendemos el ordenador con procesador: I5 y memoria: 8Gb y disco de 256Gb.
Encendemos el ordenador con procesador: I3 y memoria: 4Gb y disco de 128Gb.
```

# Programación Orientada a Objetos

¿Y qué era eso de “private” y “public”?

Son **modificadores de visibilidad**, de forma que:

- Los elementos con modificador **“private”** sólo son visibles por la propia clase “Ordenador”, es decir, desde el “Principal” y desde otra clase del proyecto NO SE VEN.
- Los elementos con modificador **“public”** son visibles por cualquier clase del proyecto, por lo que desde el “Principal” y desde cualquier otra clase del proyecto SI SE VEN.

```
public class Ordenador {  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //1 Metodo constructor  
    public Ordenador(String proc_entrada, int ram_entrada, int disco_entrada) {  
        procesador = proc_entrada;  
        ram = ram_entrada;  
        disco = disco_entrada;  
    }  
  
    public void encender_ordenador() {  
        System.out.println("Encendemos el ordenador con procesador: " + procesador + " y " +  
            " memoria: " + ram + "Gb y disco de " + disco + "Gb.");  
    }  
}
```

# Programación Orientada a Objetos

---

¿Y si desde “Principal” quisiéramos acceder a un atributo “private”?

El atributo **private** NO es visible por el resto de clases, por lo que Java nos da un error:

```
public class Principal {  
    public static void main(String[] args) {  
        Ordenador ordenador1 = new Ordenador("I7",16,512);  
        System.out.println(ordenador1.procesador);  
    }  
}
```

The field Ordenador.procesador is not visible

Lo normal en cualquier lenguaje Orientado a objetos es que..

- **Los atributos sean privados (Private)** de forma que sólo sean accesibles por la clase que los contiene y NO por el resto.
- **Los métodos sean públicos (Public)** de forma que cualquier método sea accesible por cualquier clase.



# Programación Orientada a Objetos

---

## Getters y Setters

Como hemos visto anteriormente, los atributos NO son accesibles directamente, de forma que si queremos acceder a ellos, tiene que ser a través de métodos públicos.

Los métodos públicos que se encargan de leer y de escribir sobre un atributo de un objeto son denominados comúnmente “getters” y “setters” respectivamente.

```
public class Ordenador {  
  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //getter del atributo procesador  
    public String get_procesador() {  
        return procesador;  
    }  
  
    //setter del atributo procesador  
    public void set_procesador(String proc_entrada) {  
        procesador = proc_entrada;  
    }  
}
```

# Programación Orientada a Objetos

## Getters y Setters (ejemplo de ejecución)

```
1
2 public class Principal {
3
4     public static void main(String[] args) {
5         //Concretamos la clase ordenador en un objeto llamado Ordenador1
6         Ordenador ordenador1 = new Ordenador("I7",16,512);
7
8         //Utilizamos el getter para guardarnos el valor de un atributo
9         String miprocesador = ordenador1.get_procesador(); //Guardamos "I7"
10
11        //Modificamos el valor de ese atributo
12        miprocesador = miprocesador + " 10ª generación";
13
14        //Utilizamos el setter para cambiar el valor del atributo
15        ordenador1.set_procesador(miprocesador);
16
17        //Imprimimos el valor modificado del atributo "procesador"
18        System.out.println(ordenador1.get_procesador()); //Imprime "I7 10ª generación"
19    }
20
21 }
```

Problems @ Javadoc Declaration Console

<terminated> Principal (2) [Java Application] C:\Program Files\JAVA\openjdk-12.0.1\_windows-x64\_bin\jdk-12.0.1\bin\java  
I7 10ª generación

# Programación Orientada a Objetos

## Palabra reservada “this”

La palabra reservada “this” se antepone a un atributo o a un método para especificar que estamos hablando de un dato de ESE objeto.

Imaginemos que, en el constructor de **Ordenador**, quisiéramos que los parámetros de entrada se llamaran igual que los atributos de la clase, p. ej.:

```
public class Ordenador {  
  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //1 Metodo constructor  
    public Ordenador(String procesador, int ram, int disco) {  
        procesador = procesador;  
        ram = ram;  
        disco = disco;  
    }  
}
```

The assignment to variable procesador has no effect

JAVA NO sabe que estamos haciendo, porque esas asignaciones no le llevan a ningún sitio!. En su lugar deberíamos hacer lo siguiente:


# Programación Orientada a Objetos

---

## Palabra reservada “this”

Para no confundir a Java entre los datos del objeto y el resto haríamos:

```
public class Ordenador {  
  
    private String procesador;  
    private int ram;  
    private int disco;  
  
    //1 Metodo constructor  
    public Ordenador(String procesador, int ram, int disco) {  
        this.procesador = procesador;  
        this.ram = ram;  
        this.disco = disco;  
    }  
}
```



# Programación Orientada a Objetos

## Atributos y métodos static

Son aquellos datos que NO pertenecen a los objetos, sino a la clase. Por ejemplo, imaginemos que queremos llevar la cuenta de los ordenadores que estamos creando, haríamos lo siguiente:

```
public class Ordenador {  
  
    private String procesador;  
    private int ram;  
    private int disco;  
    static int numordenadores=0;  
  
    //1 Metodo constructor  
    public Ordenador(String procesador, int ram, int disco) {  
        this.procesador = procesador;  
        this.ram = ram;  
        this.disco = disco;  
        numordenadores++;  
    }  
}
```

Cada vez que llamamos al constructor, se incrementa en 1 el contador “numcontadores” que está **en la clase** “Ordenador”.

# Programación Orientada a Objetos

## Atributos y métodos static

Creamos 3 objetos de la clase **Ordenador** en **Principal**, y por cada uno, incrementamos el valor “numordenadores”

```
public class Principal {  
    public static void main(String[] args) {  
        //Concretamos la clase ordenador en un objeto llamado Ordenador1  
        Ordenador ordenador1 = new Ordenador("I7",16,512);  
        //Concretamos la clase ordenador en un objeto llamado Ordenador2  
        Ordenador ordenador2 = new Ordenador("I5",8,256);  
        //Concretamos la clase ordenador en un objeto llamado Ordenador3  
        Ordenador ordenador3 = new Ordenador("I3",4,128);  
  
        System.out.println("De momento hemos creado " + Ordenador.numordenadores + " ordenadores.");  
    }  
}
```

<terminated> Principal (2) [Java Application] C:\Pro:  
De momento hemos creado 3 ordenadores.

Para acceder a un dato de tipo “static”, utilizamos el nombre de la clase en lugar del nombre del objeto, ya que el dato **pertenece a la clase** y NO al objeto

# Programación Orientada a Objetos

## Estructuras de datos dinámicas de objetos

A partir de ahora, los ArrayList, Hashmap, Stack y Queues que creemos, ya no están limitadas por las clases Integer o String, sino que podemos hacerlas de nuestros propios objetos del siguiente modo:

```
public class Principal {  
    public static void main(String[] args) {  
        ArrayList<Ordenador> lista_orde = new ArrayList<Ordenador>();  
        Ordenador orde1 = new Ordenador("I7",16,512);  
        Ordenador orde2 = new Ordenador("I5",8,256);  
        Ordenador orde;  
        lista_orde.add(orde1);  
        lista_orde.add(orde2);  
        Iterator iter = lista_orde.iterator();  
        while (iter.hasNext()) {  
            orde = (Ordenador)iter.next();  
            orde.encender_ordenador();  
        }  
    }  
}
```

Como **iterator** devuelve un tipo "Object", lo casteamos a la clase esperada, en este caso, la clase "Ordenador".

```
<terminated> Principal (2) [Java Application] C:\Program Files\JAVA\openjdk-12.0.1_windows-x64_bin\j  
Encendemos el ordenador con procesador: I7 y memoria: 16Gb y disco de 512Gb.  
Encendemos el ordenador con procesador: I5 y memoria: 8Gb y disco de 256Gb.
```

# Programación Orientada a Objetos

---

## Actividades

1-Replica el ejercicio contenido en esta presentación con la clase “Ordenador”.

2-Repite el ejercicio con la clase “Lavadora” que tenga los atributos y los métodos que consideres necesario. Luego, en el programa principal, crea lavadoras de varias marcas y ¡ponlas a centrifugar!.

