Junta de Andalucía
Consejería de Educación y Deporte

UD 4
Clases y Objetos
–
Módulo de Programación
1º DAW

Autor: Fran Gómez
2025/2026

Objetivos

- **Comprender** los conceptos de Clase y Objeto
- **Crear** clases correctamente **identificando** sus miembros
- **Crear y Utilizar** objetos dentro de los programas.
- **Acceder** a los miembros de los objetos
- **Mejorar** destreza en el uso de Java
- **Manejar** referencias a objetos

Índice de contenidos resumido

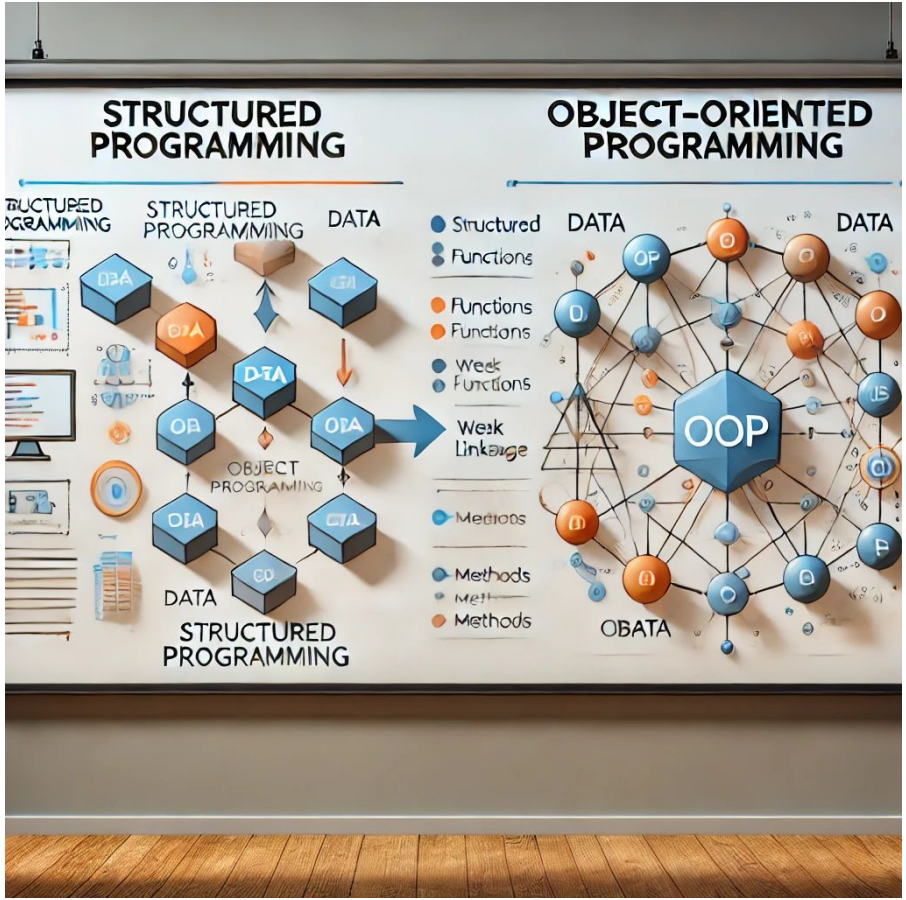
¿Qué se va a evaluar?

	Mod.	RA
RA2 Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos	10%	
a) Se han identificado los fundamentos de la programación orientada a objetos.	1%	10%
b) Se han escrito programas simples.	2%	20%
c) Se han instanciado objetos a partir de clases predefinidas.	1%	10%
h) Se han utilizado constructores.	1%	10%
i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples	1%	10%
RA3 Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje	10%	
g) Se ha comentado y documentado el código.	1%	10%

¿Cómo lo vamos a evaluar?

Instrumento	Técnica
Portfolio: - Se evalúa el trabajo diario - Recurso para estudiar - Útil como CV	- Salir a corregir los ejercicios de clase - Defender los ejercicios necesarios frente al profesor antes del examen.
Proyecto: - Evaluación de todo lo aprendido y más - Aplicación en diferentes contextos - Colaborativo	- Exposición: explicar qué y cómo funciona a la clase, respondiendo a las preguntas.
Examen: - Evaluación individual y sumativa de lo fundamental	- Prueba escrita

Introducción



funciones / datos

```
suma(a,b) {
        return a + b;
}
```

a = 2
b = 3
resultado = 5

La POO es un paradigma de programación que trata de modelar de manera abstracta el mundo real.

Clases: concepto

- Modelar el mundo real
- Plantilla
- Propiedad
  Comportamiento
- Datos + Código
- Atributos y Métodos
- Tipo de dato

Clase Vaca ⇒ Objetos

Clases: sintaxis

[visibilidad] **class** Nombre [**extends** Superclase] [**implements** Interface1, ...] {

// declaraciones de Atributos…

// declaraciones de constructores…

// declaraciones de métodos…

}

```
class NombreClase {
   //definición de la clase
}
```

```
class Persona {
   //definición de Persona
}
```

Atributos


```
class NombreClase {
    tipo atributo1;
    tipo atributo2;
    ...
}
```

```
class Persona {
    String nombre;
    byte edad;
    double estatura;
}
```

Atributos > Inicialización


```
class NombreClase {
    tipo atributo1 = valor;
    ...
}

class Persona {
    String nombre;
    byte edad;
    double estatura;
    final String dni; //una vez asignado no podrá cambiarse
}
```

Clases: Atributos(la información)

- Almacenan los datos o el estado de una clase o sus instancias (objetos)
- Sus nombres suelen ser sustantivos (notación lowerCamelCase)
- Sintaxis: **[visibilidad] [static] [final] tipo atributo1 [= valorInicial];**
- Se declaran al principio
- No hace falta inicializarlos, pero se puede hacer en:
  - Declaración
  - Método
  - Bloque de inicialización
- Visibilidad mejor a private. Siempre son visibles dentro de la clase.
- Atributos de clase: static (pertenecen a la clase, son compartidos por todos los objetos)
- Atributos miembro: pertenecen a cada objeto

Métodos

Funciones que se implementan dentro de una clase

```
public class NombreClase {
    ... //declaración de atributos

    tipo nombreMétodo (parámetros ) {
        cuerpo del método
    }
}
```

Métodos


```
public class Persona {
    String nombre;
    byte edad;
    double estatura;

    void saludar() {
        System.out.println("Hola. Mi nombre es " + nombre);
        System.out.println("Encantando de conocerte");
    }
```

Los nombres de los atributos suelen ser sustantivos y los de los métodos contener un verbo

Métodos

Ejercicio 1 (parte 1)

Crea un paquete llamado ejercicio1

Crea la clase persona correspondiente al modelo dado por el diagrama.

El método cumplirAños debe incrementar en uno la edad

Mientras que crecer debe aumentar la estatura según lo indicado en incremento.

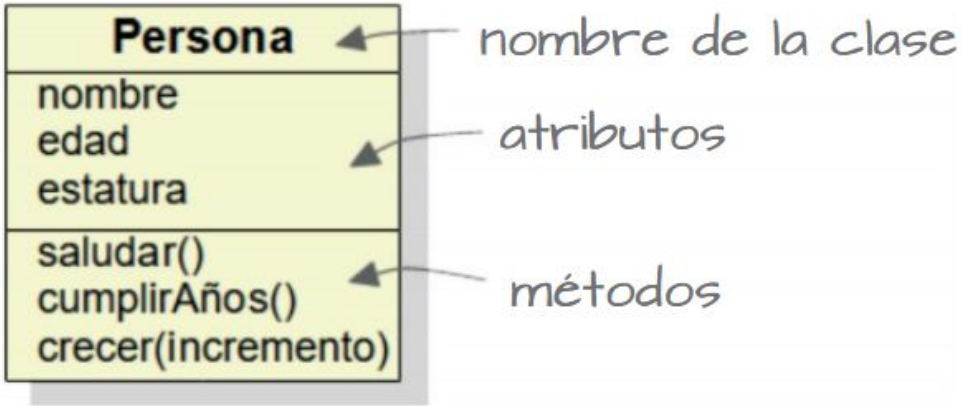


Diagrama de clases

Miembros

A los **atributos** y **métodos** de una clase se les llama **miembros** de la clase.

Cualquier miembro de una clase tiene visibilidad de clase, es decir, puede ser accedido, dentro de la clase.

Ámbito de las variables y atributos


```
class Ambitos {
    int atributo;
    ...
    void metodo() {
        int varLocal;
        ...
        while(...) {
            int varBloque;
            ...
        } //del while
        ...
    } //del método

    ...
} //de la clase
```

ámbito de la clase: podemos utilizar atributo, ~~varLocal~~ y ~~varBloque~~

ámbito del metodo: podemos utilizar atributo, varLocal y ~~varBloque~~

ámbito del while: podemos utilizar atributo, varLocal y varBloque

Ámbito de las variables y atributos

Las variables locales se pueden llamar igual que los atributos

```
public class Ambito {
   int edad; //atributo entero
   void metodo() {
      double edad; //variable local. Oculta al atributo edad (que es entero)
      edad = 8.2; //variable local double, que oculta al atributo de la clase
      ...
   }
}
```

Referencia “this”

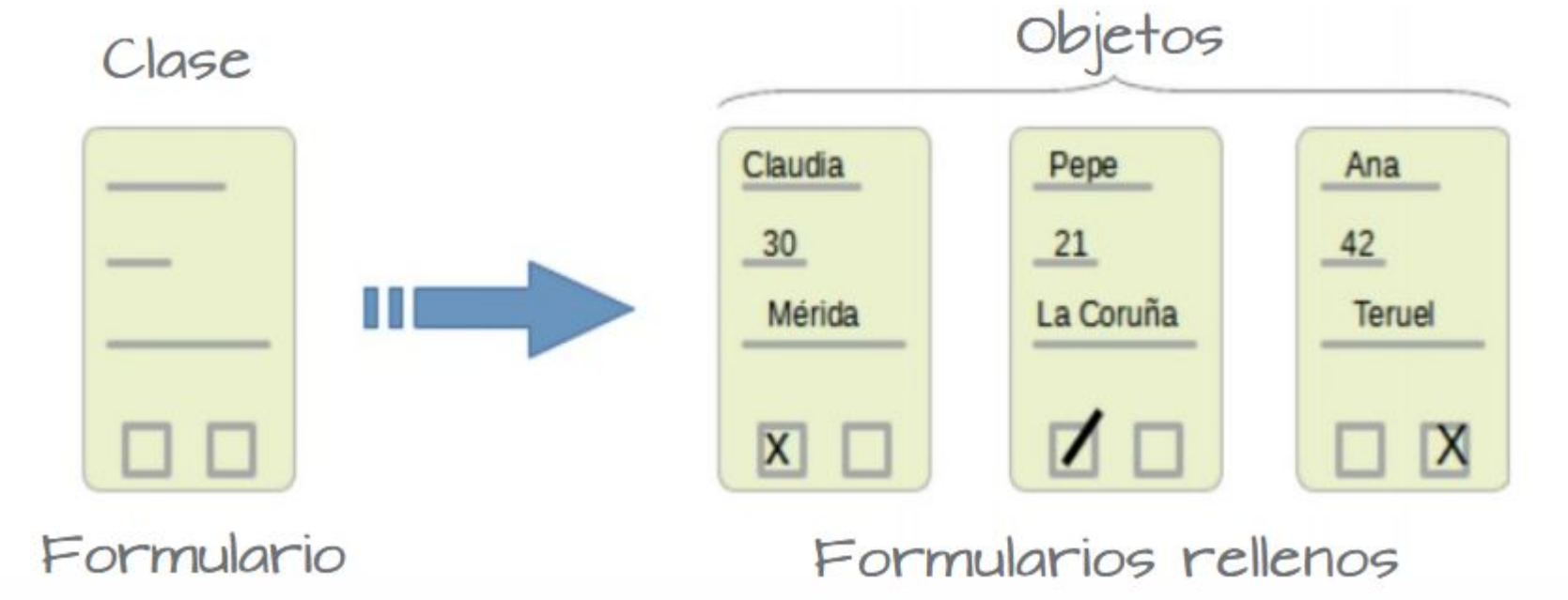
La palabra reservada this hace referencia a la propia clase.

De esta forma podemos acceder a atributos dentro de un método, aunque hayan sido ocultados por variables locales homónimas.

```
public class Ambito {
    int edad; //atributo entero
    void metodo() {
        double edad; //oculta el atributo edad (que es entero)
        edad = 20.0; //variable local, no el atributo
        this.edad = 30; //atributo de la clase
    }
}
```

Objetos

Se llama objetos a las **instancias** de una clase

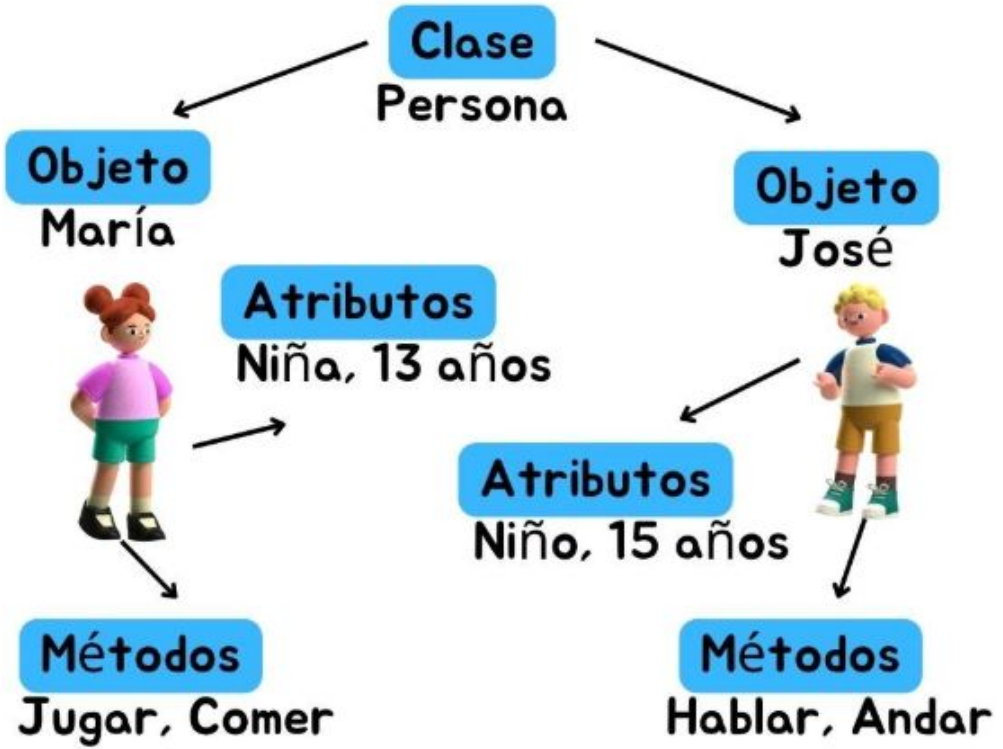


Objetos

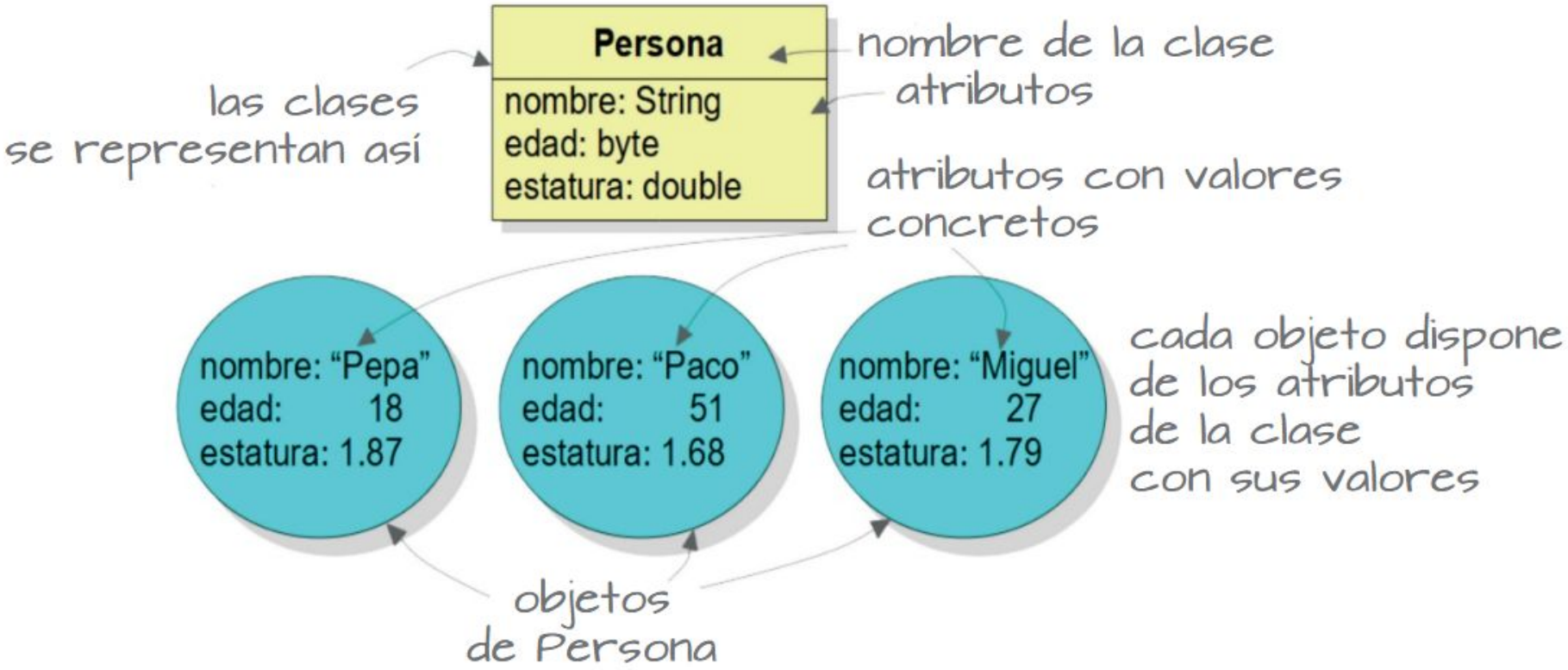
La clase se comporta como un tipo de dato.

Cada objeto es como una variable de ese tipo de dato.

Por tanto, cada objeto tiene sus propios valores para los atributos que han sido definidos mediante su clase.



Objetos


las clases
se representan así
Persona
nombre: String
edad: byte
estatura: double
nombre de la clase
atributos
atributos con valores
concretos
nombre: "Pepa"
edad: 18
estatura: 1.87
nombre: "Paco"
edad: 51
estatura: 1.68
nombre: "Miguel"
edad: 27
estatura: 1.79
cada objeto dispone
de los atributos
de la clase
con sus valores
objetos
de Persona

Referencias

Igual que los arrays, los objetos se acceden mediante referencias a memoria.

De hecho los arrays son objetos también.

Para declarar una variable del tipo de una clase:

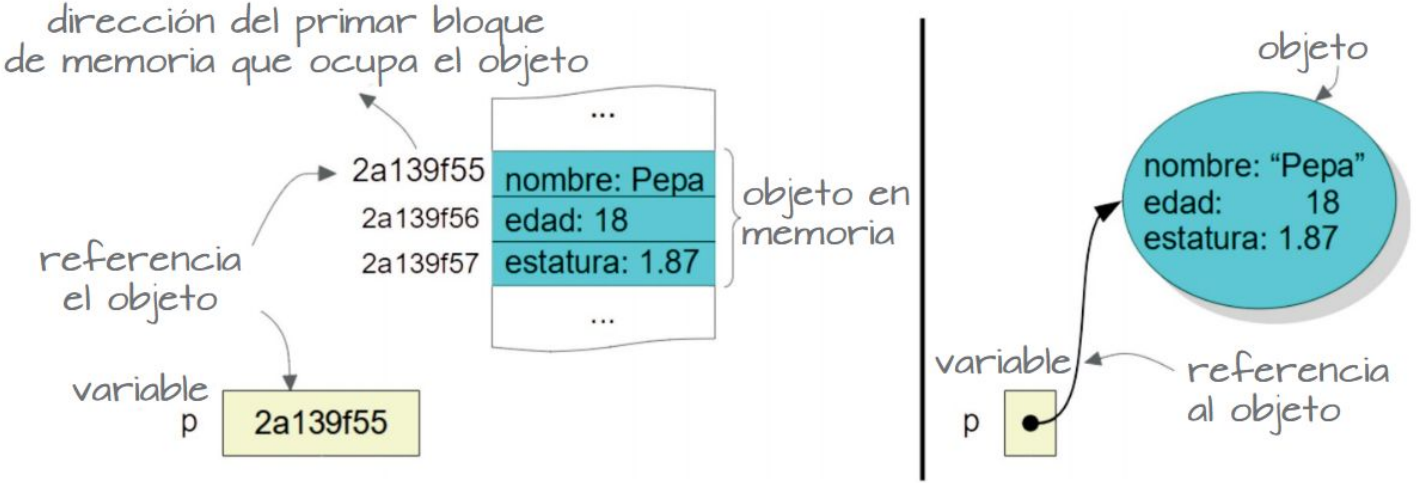
```
Clase nombreVariable;

Persona p; //p es una variable de tipo Persona
```

Creación de objetos

¿Cómo creo un objeto y asigno su referencia a una variable?

Operador "**new**"

```
p = new Persona();
```



Creación de objetos

Ejercicio 1 (parte 2)

Crea una **clase** principal de nombre **Ciudad** mediante la que modelaremos una ciudad ficticia.

Para levantar la ciudad necesitamos crear un **método principal** donde interactuarán las personas.

En el método principal, crea varias instancias de la **clase Persona** para dar vida a las personas, que serán **objetos de tipo Persona**.

Imprime a las personas que creaste para poder mostrarlas. ¿por defecto se imprime la referencia o el contenido del objeto?

Comprueba como se observa este objeto en el depurador

Operador .

Para acceder a los miembros de un objeto se utiliza el punto “.”

```
p = new Persona();
p.nombre = "Pepa";
p.edad = 18;
p.estatura = 1.87;
```

Ejercicio

Ejercicio 1 (parte 3)

Establece valores para los atributos del objeto de tipo Persona que creaste.

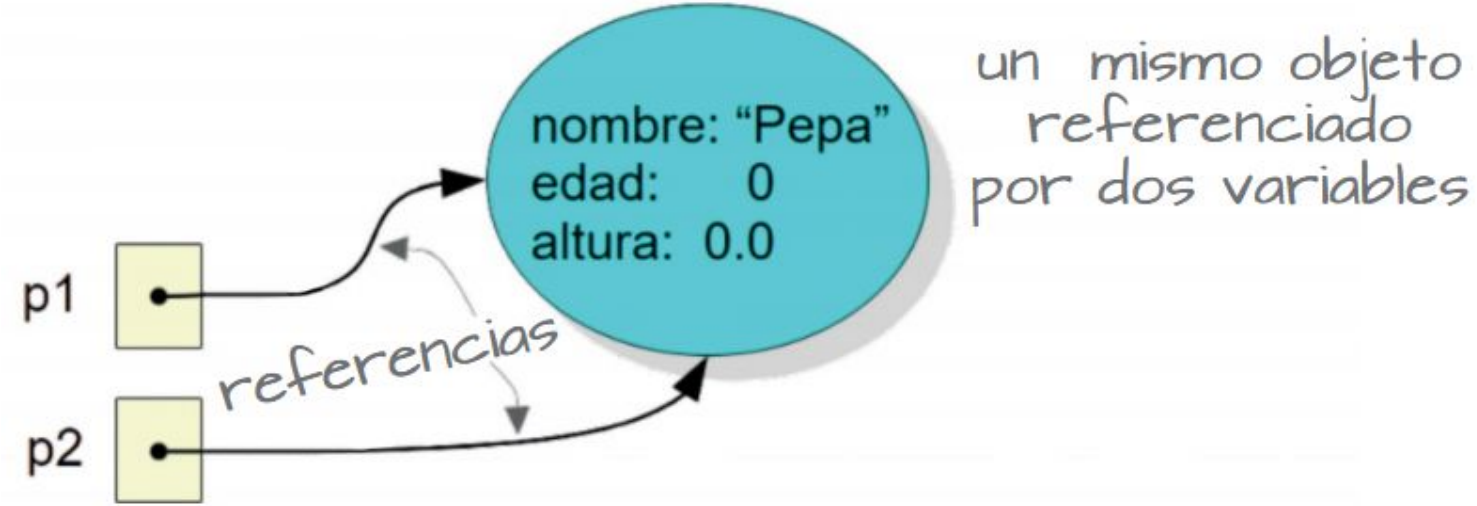
Observa en el depurador cómo se actualizan en tiempo real.

Imprime los atributos de los objetos creados, por ejemplo mediante la siguiente frase:

Hola soy [nombre] tengo [edad] años.

Ejemplo mismo objeto, dos variables


```
Persona p1, p2;
p1 = new Persona(); //p1 referencia al objeto creado
p2 = p1; //asignamos a p2 la referencia contenida en p1
p2.nombre = "Pepa" //es equivalente a utilizar p1.nombre
```



Referencia null

Los tipos objeto se inicializan por defecto a null si no se dice nada.

```
Persona p; //se inicializa por defecto a null
p.nombre //¡error!
```
⇐ NullPointerException

Se puede dar el valor null a una variable tipo objeto en cualquier momento

```
Persona p = new Persona(); //p referencia un objeto
...
p = null; //p no referencia nada
```

Constructores

¿Qué valores toman los atributos de un objeto recién creado?

Ejercicio: Realiza una tabla en el porfolio con los valores por defecto en función de los tipos de datos

Para dar unos valores iniciales distintos a los de por defecto, podemos hacerlo como acabamos de ver:

```
Persona p = new Persona(); //creamos el objeto
p.nombre = "Claudia"; //asignamos valores
p.edad = 8;
p.estatura = 1.20;
```

Constructores

O bien podemos usar constructores:

Los constructores son métodos especiales dentro de la clase que usaremos para crear objetos.

Sintaxis: NombreClase( ) {...}

```
class Vaca {
        // Atributos
        // Constructores
        Vaca() {
                ...
        }
        // Resto de métodos
}
```

Los parámetros que no se inicializan en el constructor, se inicializan a sus valores por defecto según sean de tipos primitivos o no primitivos.

Constructores

```
class Persona {
   ...
   //constructor que asigna valores a todos los atributos
   Persona (String nombre, int edad, double estatura) {
      this.nombre = nombre;
      this.edad = edad;
      this.estatura = estatura;
   }
```

En cualquier parte del código podemos crear objetos usando el operador **new** y el **constructor** que hayamos declarado

```
Persona p = new Persona("Claudia", 8, 1.20); //creamos el objeto
//y lo inicializamos mediante el constructor
```

Tipos de constructores

- Por defecto → Si no declaramos ninguno. En el momento que se declare alguno, el constructor por defecto desaparece.

- Sin parámetros → Persona( )

- Con parámetros (los que queramos)

```
//constructor sobrecargado: solo asigna el nombre
Persona (String nombre) {
   this.nombre = nombre;
   estatura = 1.0; //valor arbitrario para la estatura
   //al no asignar la edad se inicializa por defecto: a 0
}
 Persona b = new Persona("Dolores");
```

Objetos: instancias de una clase

Ejercicio 1 (parte 4)

En nuestra **Ciudad** vamos a crear personas de varias maneras.

1. Usando el constructor por defecto crea 2 personas: pepe y paco. Comprueba que sus atributos tienen los valores por defecto.

2. Crea un constructor con todos los parámetros de Persona. ¿Qué ocurre ahora con el constructor por defecto?

3. Crea un constructor sin parámetros para arreglar el error anterior. Este constructor generará una persona estándar de nombre “anónimo”, edad 18 y estatura 1.70.

4. Crea dos personas más usando el constructor con parámetros.

Ejercicio 1 (parte 5)

5. Crea otro constructor con parámetros que incluya sólo el nombre

6. Ejecuta el programa y depura para ver los resultados.

Constructores > this()

this() se usa para invocar a un constructor → reutilizar constructores

```
class Persona {
   ...
   //constructor que asigna valores a todos los atributos
   Persona (String nombre, int edad, double estatura) {
      this.nombre = nombre;
      this.edad = edad;
      this.estatura = estatura;
   }

   //constructor sobrecargado que solo asigna el nombre
   Persona (String nombre) {
      this(nombre, 0, 1.0); //invoca al primer constructor
      //la edad se pone a 0 y la estatura a 1.0
   }
}
```

¡Sólo se puede poner al principio!

Ejercicio 1 (parte 6)

7. Crea otro constructor que reciba junto a todos los parámetros, otro que sea el apellido, y cree una persona invocando a algún constructor que ya teníamos, pero ademas le concatene el apellido al nombre.

Enumerados

Son variables limitadas a una serie de valores

```
enum DiaDeLaSemana {
    LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO
}
```

```
Scanner sc = new Scanner(System.in);
String dia = sc.nextLine(); //introducimos LUNES
DiaDeLaSemana ingles = DiaDeLaSemana.valueOf(dia);
```

Ejercicio 2

Crea un nuevo paquete llamado ejercicio 2

Crea un enumerado llamado Sexo para añadir el atributo en la clase Persona que indique si una persona es Hombre, Mujer o no binario. Para ello:

Crea una nueva clase para el enumerado.

Añade un nuevo atributo a en la clase Persona del Ejercicio 1. ¿Tendrás que importar la clase que acabas de crear?

¿Qué sexo por defecto tienen las personas creadas en la ciudad? Modifica alguna persona para establecer su sexo.

Añade un nuevo constructor donde además de los atributos que ya teníamos se asigne también el sexo en su creación.

Ejercicio 3

Gestión de Vehículos en un Concesionario

Se desea desarrollar una pequeña aplicación en Java para gestionar los vehículos de un concesionario.

Cada vehículo tendrá unas características básicas y pertenecerá a un tipo concreto (turismo, motocicleta o camión).

El programa debe hacer uso de clases, objetos y enumerados, aplicando correctamente los principios básicos de la programación orientada a objetos.

Recursos y referencias

- Libro que tenéis en la biblioteca y en el departamento
- Diapositivas en Moodle
- Glosario en Moodle
- Ejercicios de refuerzo y de ampliación
- Vuestro portfolio