

Bloque 2

Introducción al diseño de tipos

Objetos, interfaces, clases

Fundamentos de Programación
Departamento de Lenguajes y Sistemas Informáticos



2023/24

Clases

- Estructura de un proyecto Java:

- ▶ **N**ombreDelProyecto

- ▶ *src (carpeta que por defecto crea eclipse para los programas fuentes)*

- p**aquetes

- ▶ **E**numerados

- ▶ **I**nterfaces

- ▶ **C**lases

- ▶ **a**tributos (almacenan las propiedades)
 - ▶ **m**étodos (consultan o actualizan las propiedades)
 - ▶ **v**ariables (permiten almacenar valores para las operaciones)
 - ▶ **C**ONSTANTES



2023/24

Clases: Definición de un nuevo tipo

- Se pueden definir nuevos tipos de objetos a los que ya están disponibles en las librerías contenidas en JRE, mediante la implementación de **clases** (Ejemplo: *Persona*)
- En una **clase** se describen:
 - Los *atributos*. Nos indica qué información se manejan de un objeto: *apellidos, fecha nacimiento, teléfono, sexo..*
 - Los *métodos*, que permiten: crear los objetos, consultar y/o modificar el valor de sus atributos, comparar dos objetos, ordenarlos entre sí: *new Persona(..), getApellidos(), setTeléfono(786112233), getSexo()...*
- Al *conjunto de los valores* que tiene los atributos de un objeto en un momento dado se le llama *Estado*



2023/24

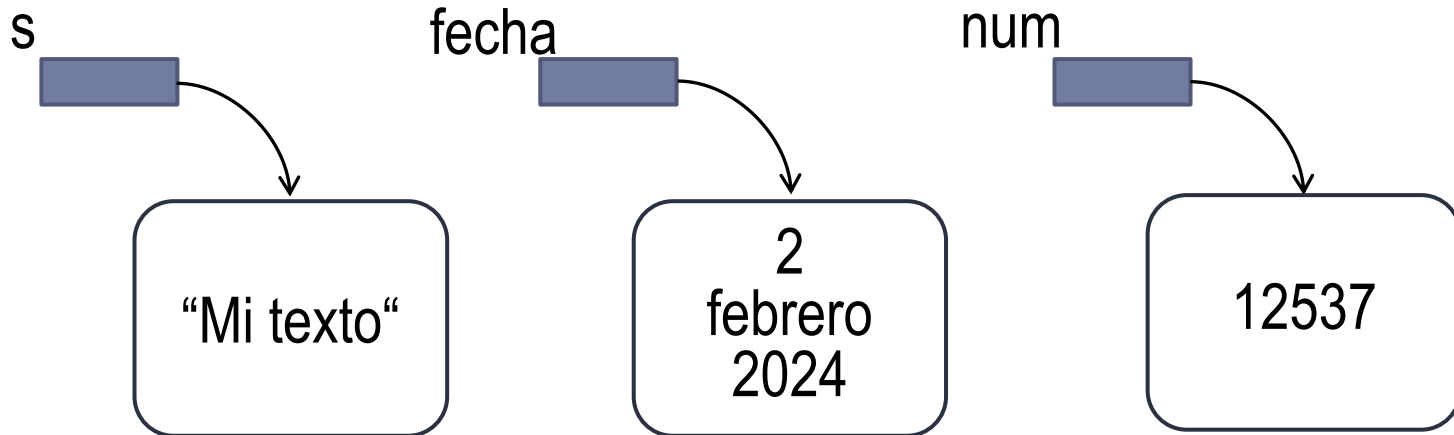
Objetos

- Un **objeto** es la existencia (una instancia) de un tipo de objeto. *Es decir: hay un espacio reservado en la memoria para guardar valores de los atributos.*

```
String s = "Mi texto";
```

```
LocalDate fecha = LocalDate.now();
```

```
Integer num = 12439;
```





2023/24

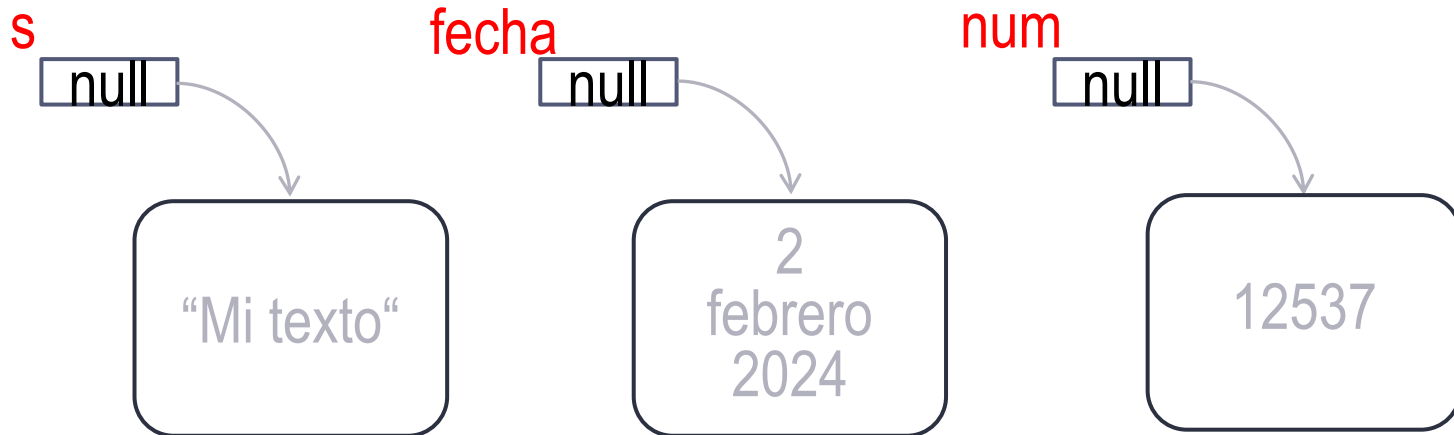
Objetos. Construcción paso a paso

- *Declaración de atributos:*

String **s** = "Mi texto";

LocalDate **fecha** = LocalDate.now();

Integer **num** = 12439;





2023/24

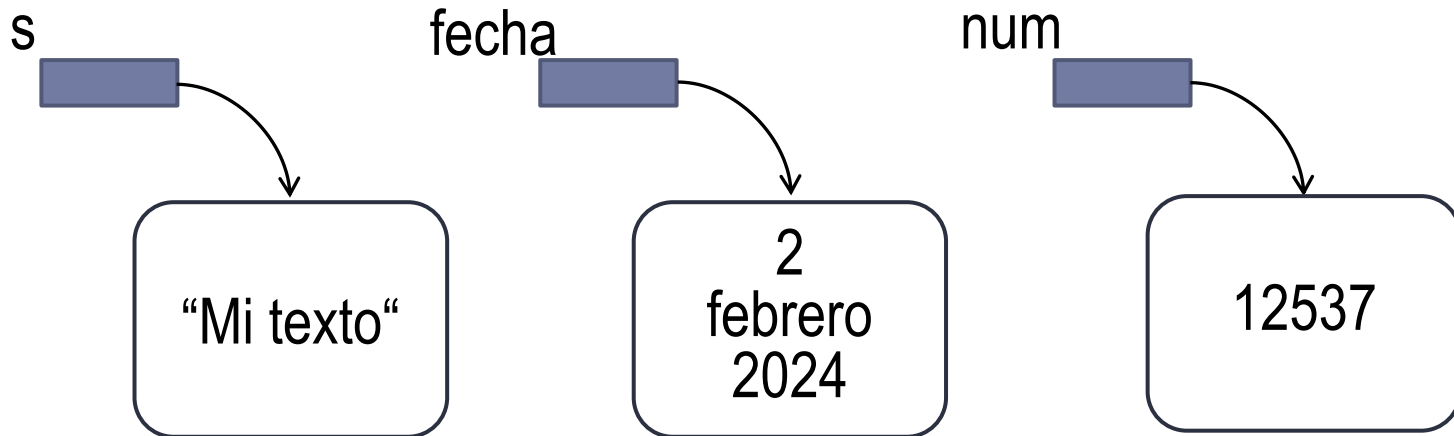
Objetos. Construcción paso a paso

- **Inicializando los valores** (poniendo al objeto en un **estado**).

```
String s = "Mi texto";
```

```
LocalDate fecha = LocalDate.now();
```

```
Integer num = 12439;
```





Objetos

2023/24

La creación de objetos se realiza de dos formas

- Mediante el uso de un **método constructor**, usando la palabra reservada **new**. Un método constructor se denomina con el mismo nombre que la clase):

```
Persona p1 = new Persona("Gómez García", ...);  
Automóvil auto = new Automóvil ("Mercedes", "9234MGB",...);
```

Métodos constructores

- Invocando a algún método estático que devuelva el objeto:

```
LocalDate fecha = LocalDate.now();  
LocalDate fecha = LocalDate.(1492,10,12);
```

Métodos estáticos

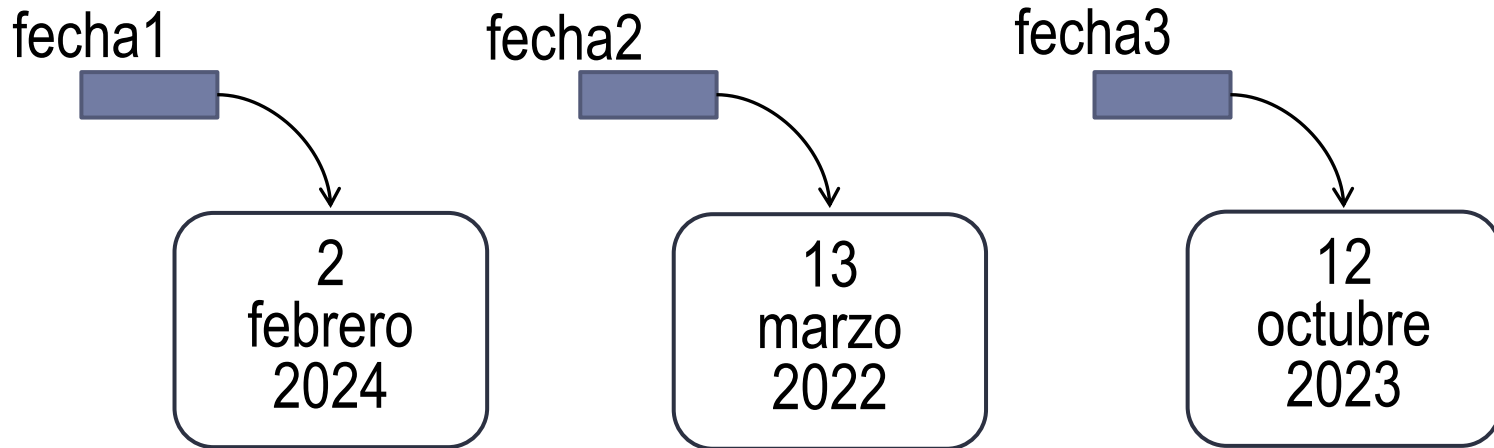
El que se use un constructor o un método estático que devuelva un objeto depende de cómo se decida la implementación.



2023/24

Objetos

- Los objetos tienen:
 - Identidad
 - Propiedades
 - Funcionalidades
 - Estado





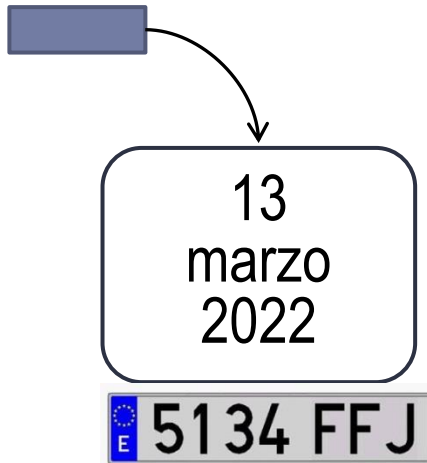
2023/24

Objetos

- Los objetos tienen:
 - **Identidad**
 - Propiedades
 - Funcionalidades
 - Estado

Cuando se construye un objeto se le asigna una dirección de memoria que lo identifica (en ejemplo, la matrícula).

fecha1



```
LocalDate fecha1 = LocalDate.of(2022,3,13);
```



2023/24

Objetos

- Los objetos tienen:
 - **Identidad vs Igualdad**
 - Propiedades
 - Funcionalidades
 - Estado

Ambos objetos son del mismo tipo,
y representan la misma fecha, pero son
objetos independientes.

*Por tanto, fecha1 y fecha2
son iguales (tienen el mismo estado), pero no
son idénticos (tienen distintas direcciones)*

```
LocalDate fecha1 = LocalDate.of(2022,3,13);  
LocalDate fecha2 = LocalDate.of(2022,3,13);
```

fecha1



5134 FFJ

fecha2



6673 FBX

≠



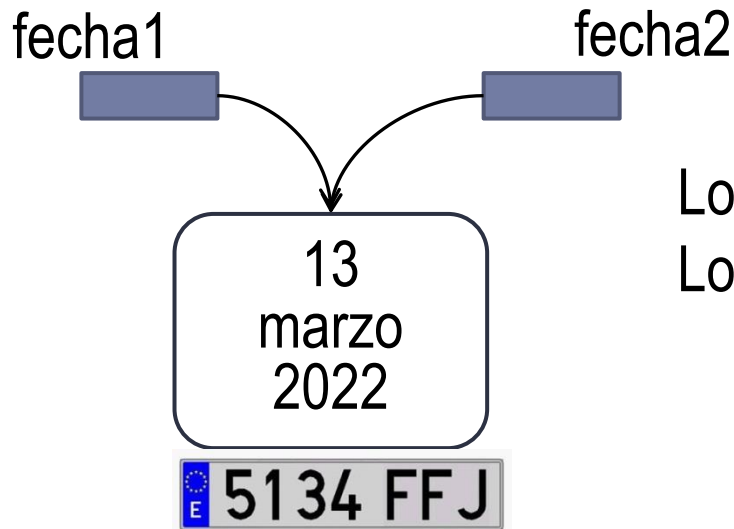
2023/24

Objetos

- Los objetos tienen:

- **Identidad vs Igualdad**
- Propiedades
- Funcionalidades
- Estado

Ambos objetos son el mismo. Por tanto, fecha1 y fecha2 son objetos idénticos y, en consecuencia, también son iguales (tienen el mismo estado)



```
LocalDate fecha1 = LocalDate.of(2022,3,13);  
LocalDate fecha2 = fecha1
```



2023/24

Objetos

- Los objetos tienen:
 - Identidad
 - **Propiedades**
 - Funcionalidades
 - Estado

Existen dos tipos de propiedades:

- **Básicas** que coinciden con los atributos
- **Derivadas**, se obtienen a partir de las básicas.

En este ejemplo tenemos tres atributos. *día, mes, año*

fecha1



Tipo: LocalDate





2023/24

Objetos

- Los objetos tienen:
 - Identidad
 - Propiedades
 - **Funcionalidades**
 - Estado

fecha1



13
marzo
2022



Cada objeto dispone de funcionalidades (métodos)

Los métodos pueden devolver datos sobre el estado o modificar el objeto.

- *getDayOfMonth()* → tipo *int*
 - *getMonthValue()* → tipo *int*
 - *getYear()* → tipo *int*
 - *isLeapYear()* → tipo *boolean*
- Devuelven propiedades básicas
- Devuelve propiedad derivada

Tipo: `LocalDate`

$\text{año} \% 400 == 0 \parallel$
 $(\text{año} \% 4 == 0 \ \&\& \ \text{año} \% 100 != 0)$



2022/23

Objetos

Cada objeto tiene su estado

- Los objetos tienen:
 - Identidad
 - Propiedades
 - Funcionalidades
 - Estado

- `fecha1.getDayOfMonth()` → 13
- `fecha1.getMonthValue()` → 3
- `fecha1.getYear()` → 2022
- `fecha1.isLeapYear()` → false
- `fecha2.getDayOfMonth()` → 28
- `fecha2.getMonthValue()` → 5
- `fecha2.getYear()` → 2024
- `fecha2.isLeapYear()` → true

fecha1



13
marzo
2022



fecha2



28
mayo
2024



Tipo: LocalDate



Repaso

2023/24

- ¿Qué cuatro cosas definen a un objeto?
- ¿Qué significa que dos objetos son idénticos?
- Si dos objetos son independientes, pero tienen el mismo estado, ¿serán idénticos?



2023/24

Clases: Definición de un nuevo tipo

- Gráficamente una clase puede ser algo así:

Cabecera o definición

Atributos (*propiedades básicas*)

Métodos:

- Constructores
- Consultores (get)
- Modificadores (set)
- Representación textual (toString)
- Criterio de igualdad (equals)
- Criterio de ordenación (compareTo)
- Otros métodos



2023/24

Diseño e implementación de tipos

Implementación de la/s **clase/s**:

- Cabecera
- **Atributos**
- Métodos
 - Constructores
 - Consultores (get)
 - Modificadores (set)
 - Representación textual (toString)
 - Otros métodos



2023/24

Diseño e implementación de tipos

Implementación de la/s **clase/s**:

Cabecera / **Atributos** / **Método/s Constructor/es** / **Otros Métodos**

```
public class NombreDeClase ...{  
    private tipo nombreAtributo-1;  
    private tipo nombreAtributo-2; } atributos  
    ...  
    public nombreDeClase (tipo nombreParámetro1,...) {  
        this.nombreAtributo1=nombreParámetro1;  
        ...  
    }  
    public tipo nombreMétodo (...) {  
        ...  
    }  
    ...  
}  
//fin de la clase
```

Métodos
Constructor/es

Otros
métodos



2023/24

Ejercicio. Tipo Punto

Se trata de implementar una clase para definir el tipo *Punto* que represente lo que conocemos por un punto en el plano euclídeo de dos dimensiones. Paso que hay que seguir:

1. Construir un proyecto de tipo java: *T01_FiguraGeométricas*
2. Crear los paquetes *geometría* y *geometria.test* dentro del proyecto (en la carpeta *src* que se crea automáticamente por Eclipse)
3. En el paquete *geometría* se crea la clase *Punto.java*
4. En el paquete *geometria.test* se crea la clase *TestPunto.java* que debe incluir el método *main()*
5. Implementemos los 4 puntos anteriores en nuestro entorno Eclipse-Java



2023/24

Ejercicio. Tipo Punto



Ahora *hay que pensar* para encontrar que atributos y métodos tendrá nuestro tipo Punto



2023/24

Ejercicio. Tipo Punto

Atributos:

- Parece lógico que, si queremos anotar en un papel, en el que se han dibujado unas coordenadas cartesianas, pidamos el valor de abscisa (la x) y el valor de la ordenada (la y). Por ello la clase Punto deberá tener dos atributos para “anotar” esa coordenada: uno que denominaremos “ x ” y otro que denominaremos “ y ” (*también le podríamos denominar, por ejemplo, **abscisa** y **ordenada**, pero es más largo de escribir*)
- Nos queda otra cosa que pensar: ¿Nuestros puntos admitirán coordenadas con decimales?. Digamos que sí. Entonces, ¿**de qué tipo serán x e y .**?



2023/24

Ejercicio. Tipo Punto

Métodos (*qué operaciones deseamos para nuestro tipo Punto*):

- Un *constructor* que reciba las dos propiedades básicas.
- Dos métodos *consultores*: Uno para cada propiedad básica.
- Dos métodos *modificadores*. Uno para cada propiedad básica.
- La *representación textual* del objeto
- Un método *distancia* que recibiendo un punto como parámetro devuelva la distancia al objeto que invoca a dicho método.
- Añadiremos un *constructor* sin parámetros para crear un objeto que sea el origen de coordenadas.

¡Hala! pues a Eclipse a definir los dos atributos y los métodos



2023/24

Ejercicio. Tipo Punto

Test (*¡tendremos que probar que hemos implementado bien el tipo!*):

Dentro del método *main* que estará en la clase **TestPunto** haremos lo siguiente:

1. Construiremos un punto “p1” con los valores (1, 2) y un segundo punto “p2” con los valores (4.5, 6.11)
2. Visualizaremos los dos puntos.
3. Modificaremos la abscisa de p2 con el valor 4 y la ordenada con el valor 6.
4. Visualizaremos otra vez los dos puntos.
5. Visualizaremos la distancia de p1 a p2 y la de p2 a p1.
6. Por último, construiremos un punto “origen” con el constructor por defecto (el que no tiene parámetros)
7. Visualizaremos el origen
8. Visualizaremos la distancia de p1 al origen de coordenadas



2023/24

Ejercicio. Tipo Circunferencia

Se trata de implementar el tipo *Circunferencia* que represente una circunferencia en el plano euclídeo dentro del proyecto *T01_FigurasGeométricas*. En los paquetes *geometría* y *geometria.test*, crearemos:

1. En el paquete *geometría* la clase *Circunferencia.java*
2. En el paquete *geometria.test* la clase *TestCircunferencia.java* que deberá incluir el método *main()*



2023/24

Ejercicio. Tipo Circunferencia

Atributos:

- Parece lógico que, los atributos sean el *centro* y el *radio* de la circunferencia
- ¿De qué tipo será el *centro*?. Nuestro proyecto ya dispone del tipo *Punto*, ¡aprovechémoslo!: el centro será de tipo *Punto*.
- ¿Parece lógico que el *radio* admita magnitudes con decimales?. Digamos que sí. Entonces radio deberá ser de tipo *Float* o *Double*. Escojamos *Double*.



2023/24

Ejercicio. Tipo Circunferencia

Métodos (qué operaciones deseamos para nuestro tipo Circunferencia):

- Un *constructor* que reciba las dos propiedades básicas.
- Dos métodos *consultores*: Uno para cada propiedad básica.
- Dos métodos *modificadores*. Uno para cada propiedad básica.
- La *representación textual* del objeto: Como la de Punto seguida de un espacio, una R: y el valor del radio. Por ejemplo: (1,0,2,0) R:2.5
- Un método *longitud* que devuelva la longitud de la circunferencia. Utiliza **Math.PI** para obtener el valor de π
- Un método *área* que devuelva el área del círculo interior a la circunferencia. Puedes multiplicar el radio por sí mismo o usar **Math.pow**(base, exponente) para calcular la potencia del radio elevado al cuadrado.

¡Hala! pues a Eclipse a definir los atributos y los métodos



2023/24

Ejercicio. Tipo Circunferencia

Test (*¡tendremos que probar que hemos implementado bien el tipo!*):

Dentro del método *main* que estará en la clase **TestCircunferencia** haremos lo siguiente:

1. Construiremos un punto “*centro*” con los valores (1, 2)
2. Construiremos una circunferencia “*miCircunferencia*” usando como parámetros del constructor el *centro* y un *radio* con una longitud de 2.5.
3. Visualizaremos *miCircunferencia*.
4. Visualizaremos la longitud de *miCircunferencia*.
5. Visualizaremos el área del círculo interior a *miCircunferencia*.

Si has llegado hasta aquí sin copiar de otros ni de otras.

¡Enhorabuena! vas por buen camino