



UNIVERSIDAD DE ALMERÍA

# Grado en Ingeniería Informática

## Introducción a la Programación

### 2021-2022



# Tema 3. Clases y Objetos

- Introducción
- Clases y objetos
- Métodos básicos
- Paquetes
- Representación de clases
- Paso de objetos a métodos
- Composición
- Herencia
- Clases Abstractas e Interfaces
- Excepciones

# Tema 3. Clases y Objetos

## Introducción

### Principios de la orientación a objetos

Todos los programas de computadora constan de dos elementos: **código y datos**.

Sin embargo, un programa se puede organizar conceptualmente alrededor de su código o de sus datos.

En los lenguajes procedimentales, orientados a procesos o **estructurados** funcionan como “**un código que actúa sobre datos**”.

**El paradigma orientado a objetos** organiza el programa alrededor de los datos (es decir, objetos) y se puede caracterizar por el **control de los datos sobre el código**.

# Tema 3. Clases y Objetos

## Introducción

La idea principal de la **POO** es un conjunto de objetos que interactúan entre sí y que están organizados en **clases**. Sus principios fundamentales son:

- ❑ **Abstracción:** Consiste en olvidarnos de los detalles, constituyendo así un mecanismo fundamental para permitir la comprensión de sistemas complejos.

- ❑ **Encapsulamiento u ocultación de la información:** Consiste en la combinación de los datos y las operaciones que se pueden ejecutar con esos datos, impidiendo usos indebidos, ya que el acceso a los datos se hará a través de las operaciones que tengamos definidas (métodos.) La base del encapsulamiento o de la ocultación de la información en Java es la **clase**.

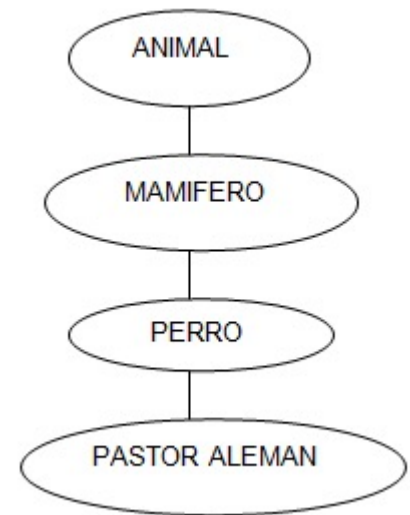
# Tema 3. Clases y Objetos

## Introducción

❑ **Herencia:** Consiste en la capacidad para crear nuevas clases que llamaremos descendientes o derivadas y que se construyen sobre otras ya existentes, que denominamos antecesoras, superclase o clase base, permitiendo que estas últimas (superclase) le transmitan sus propiedades a las derivadas.

Un pastor alemán tiene las características de los perros, que a su vez tiene la de los mamíferos (o es un mamífero), y a su vez estos las de los animales.

Se puede establecer una jerarquía:



# Tema 3. Clases y Objetos

## Introducción

❑ **Polimorfismo:** Consiste en que un mismo mensaje pueda actuar sobre diferentes tipos de objetos y comportarse de modo distinto.

Recordar que la **POO** se inspira en una *abstracción* del mundo real, en la que los objetos se clasifican en grupos o clases. Y podemos definir cada *grupo* o *clase* mediante las *propiedades* y *comportamientos* que presentan todos sus miembros. Una *propiedad* es un dato que conocemos de cada miembro del grupo, mientras que un *comportamiento* es algo que puede hacer dicho miembro. Dicho esto, es posible definir una clase mediante un conjunto de propiedades y comportamientos, y para ello utilizaremos en Java la palabra reservada **class**

# Tema 3. Clases y Objetos

## Clases y objetos

Los dos conceptos más importantes de la POO son los de **clase** y **objeto**.

Un **objeto** en su acepción o significado más amplio es cualquier cosa tanto tangible como intangible que podamos imaginar. En un programa escrito en el estilo orientado a objetos tendremos una serie de objetos interaccionando entre sí.

Para que una computadora sea capaz de crear un objeto es necesario proporcionarle una definición (también llamada plantilla o molde) y eso es lo que denominamos **clase**. Una clase es una plantilla implementada en software que describe un conjunto de objetos con **atributos/propiedades** y **comportamiento** similares.

Una **instancia u objeto** de una clase es una representación concreta y específica de una clase y que reside en la memoria del ordenador.

# Tema 3. Clases y Objetos

## Clases y objetos

### ☐ Atributos/Propiedades

Los atributos/propiedades son las **características individuales** que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. Los atributos se guardan en **variables** denominadas **de instancia**, y cada objeto particular puede tener valores distintos para estas variables.

Las variables de instancia también denominados **datos miembro**, son declaradas en la clase, pero sus valores son fijados y cambiados en el objeto.

Además de las variables de instancia hay **variables de clase**, las cuales se aplican a la clase y a todas sus instancias. Por ejemplo, el número de ruedas de un automóvil es el mismo cuatro, para todos los automóviles.



# Tema 3. Clases y Objetos

## Clases y objetos

### ☐ Comportamiento

El comportamiento de los objetos de una clase se implementa mediante **funciones miembro** o **métodos**. Un método es un conjunto de instrucciones que realizan una determinada **tarea** y son similares a las funciones de los lenguajes estructurados.

Del mismo modo que hay variables de instancia y de clase, también hay **métodos de instancia** y **métodos de clase**. En el primer caso, un objeto llama a un método para realizar una determinada tarea, en el segundo, el método se llama desde la propia clase.

# Tema 3. Clases y Objetos

## Clases y objetos

Cuando se escriben los programas orientados a objetos, primero es necesario definir las clases. Cuando el programa está en ejecución, se crean los objetos de esas clases, que van a llevar a cabo tareas. Para indicar a una clase u objeto que realice una tarea es necesario enviarle un **mensaje**.

Por ejemplo, a la clase cuenta, le puedo **mandar un mensaje** "nuevo" para crear una instancia u objeto de esa clase. Igualmente, a un objeto de la clase cuenta le puedo **enviar el mensaje** "depositar" para hacer un depósito de 100 euros. Es decir, enviamos mensajes para que realicen tareas.

# Tema 3. Clases y Objetos

## Clases y objetos

Para que la clase u objeto entienda o procese el mensaje, debe de estar programado, es decir, debe de haber una secuencia de instrucciones que se relacionen con ese **mensaje**, precisamente esa secuencia de instrucciones es lo que se conoce como **método** y existen métodos definidos para una clase (**métodos de clase**) y métodos definidos para objetos (**método de instancia**.) Por supuesto, en ambos casos, pueden necesitar el paso de parámetros o argumentos.

### Diferencias

**Método de instancia:** Cuando afecte a una instancia u objeto individual.  
**Método de clase** (estáticos - **static**): Se definirá cuando tengamos una tarea que afecte a todas las instancias (objetos) de la clase.

# Tema 3. Clases y Objetos

## Clases y objetos

De manera análoga a los métodos, que hemos definido de clase y de instancia, podemos definir **atributos** de clase y atributos de instancia. Un **atributo estático o de clase** es aquel del que no existe una copia en cada objeto. Todos los objetos de una misma clase comparten su valor. Un atributo estático se declara mediante la palabra reservada `static`

**Métodos estáticos** son aquellos que no requieren de ningún objeto para ejecutarse y no pueden utilizar ningún atributo que no sea estático.

### Diferencias

**Atributos de instancia:** Contienen información propia de cada objeto.

**Atributos de clase (estáticos - `static`):** Contiene información compartida por todas las instancias (objetos) de la clase.

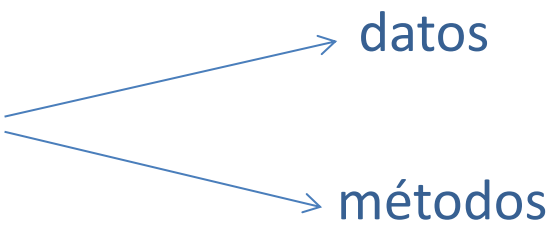
# Tema 3. Clases y Objetos

## Clases y objetos

### Declaración de una clase

Para crear una clase se utiliza la palabra reservada **class** y a continuación el nombre de la clase. La definición de la clase se pone entre las llaves de apertura y cierre. El nombre de la clase empieza por letra mayúscula.

```
<modificador> class <nombre_clase> {  
    <declaración de miembros de clase>  
}
```



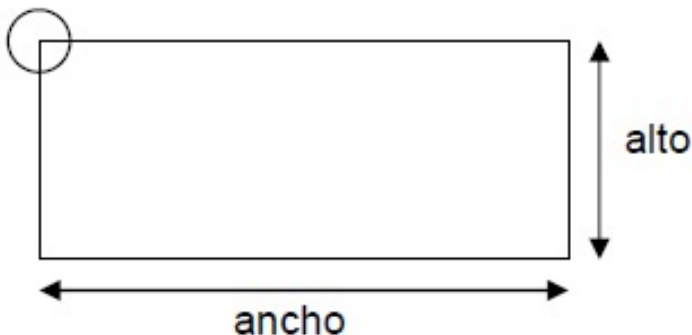
# Tema 3. Clases y Objetos

## ➡ Clases y objetos

**Ejemplo 1:** Crearemos una clase denominada Rectangulo, que describa las características comunes a esas figuras planas.

### Atributos/Propiedades

1. Origen del rectángulo: el origen es la posición de la esquina superior izquierda del rectángulo.
2. Dimensiones del rectángulo: ancho y alto.



```
public class Rectangulo {  
    private int x;  
    private int y;  
    private int ancho;  
    private int alto;  
}
```

Atributos

# Tema 3. Clases y Objetos

## Clases y objetos

### Comportamiento

Vamos a añadir operaciones a la clase Rectángulo. Nos interesa:

1. Calcular el área.
2. Desplazar el origen.
3. Saber si contiene en su interior un punto determinado del plano.

### Método para el cálculo del área:

No es necesario a ese método pasarle ni el ancho ni el alto del rectángulo, porque son directamente accesibles (esto es, ya están declarados).

```
public int calcularArea() {  
    return ancho * alto;  
}
```

# Tema 3. Clases y Objetos

## Clases y objetos

**Método que desplaza horizontalmente dx y verticalmente dy:**

Queremos que desplace al rectángulo horizontalmente una distancia de dx y verticalmente una distancia de dy. Para ello hacemos lo siguiente:

```
public void desplazar(int dx, int dy) {  
    x += dx;  
    y += dy;  
}
```

**Método que determina si un punto está o no en el interior de un rectángulo:**

Necesitamos conocer las coordenadas del punto:  $x_1$ ,  $y_1$ .

Para que el punto  $(x_1, y_1)$  esté dentro del rectángulo debe cumplir:

$x_1 > x$  and  $x_1 < x + \text{ancho}$

$y_1 > y$  and  $y_1 < y + \text{alto}$



# Tema 3. Clases y Objetos

## Clases y objetos

```
public boolean estaDentro(int x1, int y1) {  
    if (x1 > x && x1 < x + ancho && y1 > y && y1 < y + alto)  
        return true;  
    else  
        return false;  
}
```

# Tema 3. Clases y Objetos


## Clases y objetos

**Métodos.** Como ya sabemos, son el comportamiento u operaciones que pueden realizar los objetos de una clase. Y a nivel de implementación en Java son *funciones miembro* de la clase.

**Ámbito de las variables y atributos.** El ámbito de una variable define en qué lugar puede utilizarse, y coincide con el bloque de código en el que se declara dicha variable. El ámbito puede contener otros ámbitos, formando una estructura jerárquica. Las variables declaradas en bloques de estructuras de control se les denomina *variables de bloque*. A las variables declaradas en una función o método se le conocen como *variables locales*. Y a cualquier variable declarada en una clase se le denomina *atributo* o variable miembro, y podrán ser utilizada en cualquier lugar de la misma.

# Tema 3. Clases y Objetos

## Métodos básicos

-  ☐ Constructores
- ☐ Métodos de acceso y modificadores
- ☐ toString
- ☐ equals
- ☐ compareTo

# Tema 3. Clases y Objetos

## Métodos básicos

Algunos métodos son comunes a todas las clases. En esta sección los estudiaremos

## ☐ Constructores

Son métodos que controlan cómo se **crea** (asigna memoria) e **inicializa** (los atributos con determinados valores) un objeto. Tienen el mismo nombre que la clase. Gracias a la sobrecarga, se pueden definir diversos constructores, como por ejemplo el constructor por defecto, el constructor copia, etc.

Cuando se hace una llamada al constructor, se **crea** un objeto que se sitúa en memoria e **inicializa** los atributos o datos miembros declarados en la clase.

# Tema 3. Clases y Objetos

## ➤ □ Constructores

Características de los constructores:

- Debe tener el mismo nombre que la clase a la que pertenece.
- En una clase puede haber varios constructores con el mismo nombre y con distinta cantidad o tipos de argumentos. Es decir, permite sobrecarga.
- Un constructor no puede devolver ningún valor, incluyendo el **void**.
- Un constructor debería declararse público (**public**), para que pueda ser invocado desde cualquier parte donde se desee crear un objeto de su clase.
- Es el primer método que se ejecuta y lo hace de forma automática.

# Tema 3. Clases y Objetos

## ➤ □ Constructores

Se suele hablar de 2 tipos de constructores:

- **Constructores por defecto:** son aquellos que no tienen argumentos, es decir, no se les pasa ningún parámetro. Los atributos del objeto son iniciados con los valores predeterminados por el sistema.
- **Constructores generales o explícitos:** son aquellos a los que les pasamos parámetros. Un ejemplo de este tipo de constructores tenemos al **constructor copia**, que se le pasa un objeto y construye otro objeto exactamente igual al original que se le pasa como parámetro. Es decir, el **constructor copia** es un constructor que recibe por parámetro un objeto del mismo tipo de la clase, asignando atributo por atributo al nuevo objeto generado.

# Tema 3. Clases y Objetos

## ➤ □ Constructores

### Constructor por defecto

```
public Rectangulo() {  
    super();  
    x = 0;  
    y = 0;  
    ancho = 0;  
    alto = 0;  
}
```

### Constructor general

```
public Rectangulo(int x, int y, int ancho, int alto) {  
    super();  
    this.x = x;  
    this.y = y;  
    this.ancho = ancho;  
    this.alto = alto;  
}
```

Cuando los nombres de los parámetros coinciden con los nombres o identificadores de los atributos se usa **this** para distinguir los atributos del parámetro (variable). La palabra reservada **this** permite utilizar un atributo cuando ha sido ocultado por una variable local. En el ámbito de la clase, **this** se interpreta como la *propia clase*. El constructor es tan importante que, si el programador no prepara ninguno, el compilador crea uno por defecto, inicializando las variables de los tipos primitivos a su valor por defecto, y los `String` y las demás referencias a objetos a **null**.

# Tema 3. Clases y Objetos

## ➤ □ Constructores

Un constructor de una clase puede llamar a otro constructor previamente declarado, siempre y cuando esté declarado en la misma clase, y se hace usando de nuevo la palabra reservada **this** (constructor genérico **this()**), en este contexto la palabra **this** sólo puede aparecer en la primera sentencia.

```
public Rectangulo(int x, int y) {  
    this(x, y, 0, 0);  
}
```

Llamada al constructor

```
public Rectangulo(int x, int y, int ancho, int alto) {  
    super();  
    this.x = x;  
    this.y = y;  
    this.ancho = ancho;  
    this.alto = alto;  
}
```



# Tema 3. Clases y Objetos

## ➤ □ Constructores

**Constructor copia** (construir un objeto en base a otro objeto de la misma clase).

```
public Rectangulo(Rectangulo rect) {  
    super();  
    this.x = rect.x;  
    this.y = rect.y;  
    this.ancho = rect.ancho;  
    this.alto = rect.alto;  
}
```

Recuerda que el **constructor copia** es un constructor que recibe por parámetro un objeto del mismo tipo de la clase, asignando atributo por atributo al nuevo objeto generado

# Tema 3. Clases y Objetos

Para usar un objeto en un programa se debe de declarar, crear y enviar mensajes

## Declaración de un objeto

```
<nombre de clase> <nombre de objeto>;
```

Ejemplo: Rectangulo rect;

## Creación de un objeto

```
<nombre de objeto> = new <nombre de clase> (<argumentos>)
```

Ejemplo: rect = **new** Rectangulo(1, 1, 1, 1);

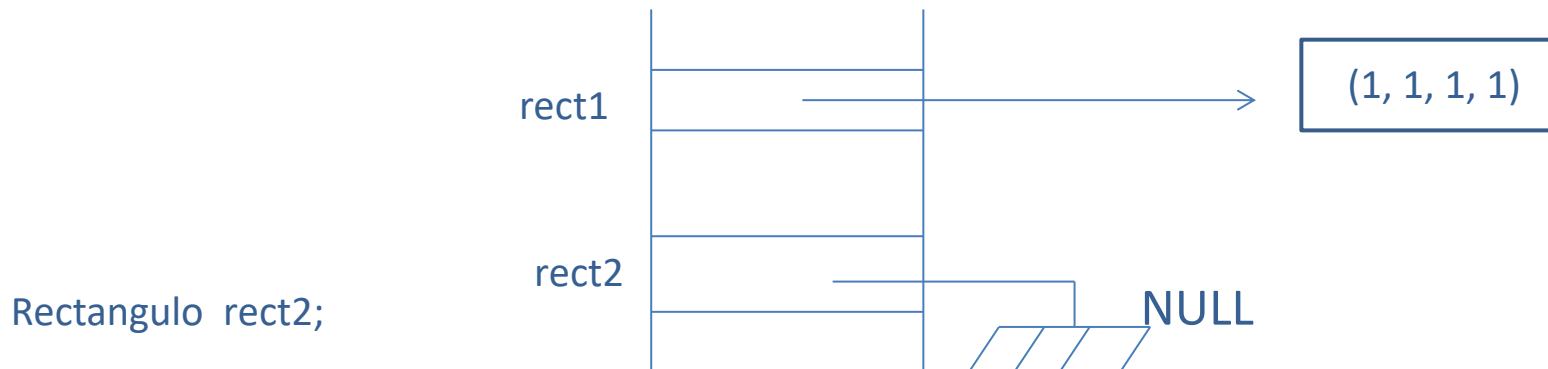
# Tema 3. Clases y Objetos

## Declaración + creación de un objeto

`<nombre de clase> <nombre de objeto> = new <nombre de clase> (<argumentos>)`

Ejemplo: `Rectangulo rect1= new Rectangulo(1, 1, 1, 1);`

rect1 tiene la dirección de memoria donde se almacena el objeto y se dice que rect1 apunta al objeto o referencia al objeto.



`Rectangulo rect2;`

# Tema 3. Clases y Objetos

## Envío de mensajes a un objeto

Una vez creado un objeto, para llevar a cabo una determinada tarea, por ejemplo, obtener su área, es necesario enviarle un mensaje. La sintaxis de **envío de mensajes** es:

`<nombre de objeto> . <nombre del método> (<parámetros>)`

Por ejemplo:

```
int medidaArea1 = rect1.calcularArea();  
rect1.desplazar(10, 20);  
boolean esta = rect1.estaDentro(10, 10);
```

# Tema 3. Clases y Objetos

## ➤ □ Métodos de acceso y modificadores

Los atributos de una clase se declaran normalmente como privados (`private`). En consecuencia, los métodos que no pertenecen a la clase no pueden acceder directamente a ellos. En ocasiones nos gustaría examinar el valor de un atributo e incluso cambiarlo. Una posibilidad es declarar los atributos como públicos (`public`). Sin embargo, esta elección viola el principio de ocultamiento de información. Para evitarlo proporcionaremos métodos para examinar y cambiar los atributos.

Un método que examina, pero no cambia el estado de un objeto es un **método de acceso**.

Un método que cambia el estado de un objeto es un **método modificador**

# Tema 3. Clases y Objetos

## ➤ □ Métodos de acceso y modificadores

Se conocen como **Getters** (de acceso) y **Setters** (modificadores)

```
public int getAncho() {  
    return ancho;  
}  
  
public void setAncho(int ancho) {  
    this.ancho = ancho;  
}
```

Ejemplo de uso: `int anchura = rect1.getAncho();`  
`rect1.setAncho(10);`

# Tema 3. Clases y Objetos

## ➤ □ toString

Es un método que devuelve un **String** (cadena de caracteres) mostrando el estado del objeto, es decir sus atributos.

```
public String toString() {  
    return "Rectangulo x=" + x + ", y=" + y + ", ancho=" + ancho  
        + ", alto=" + alto;  
}
```

Ejemplo de uso: `System.out.println(rect1.toString());`

# Tema 3. Clases y Objetos

## ➤ ☐ equals

Es un método que se utiliza para comprobar si dos referencias describen el mismo valor (compara si dos objetos son iguales o no, atributo por atributo).

```
public boolean equals(Object obj) {  
    Rectangulo otro = (Rectangulo) obj;  
    return x == otro.x && y == otro.y && ancho == otro.ancho  
        && alto == otro.alto;  
}
```

Ejemplo de uso:

```
if (rect1.equals(rect2)) → parámetro explícito  
    System.out.println("Mismos rectangulos");  
else  
    System.out.println("Distintos rectangulos");
```

*parámetro implícito* ←



# Tema 3. Clases y Objetos

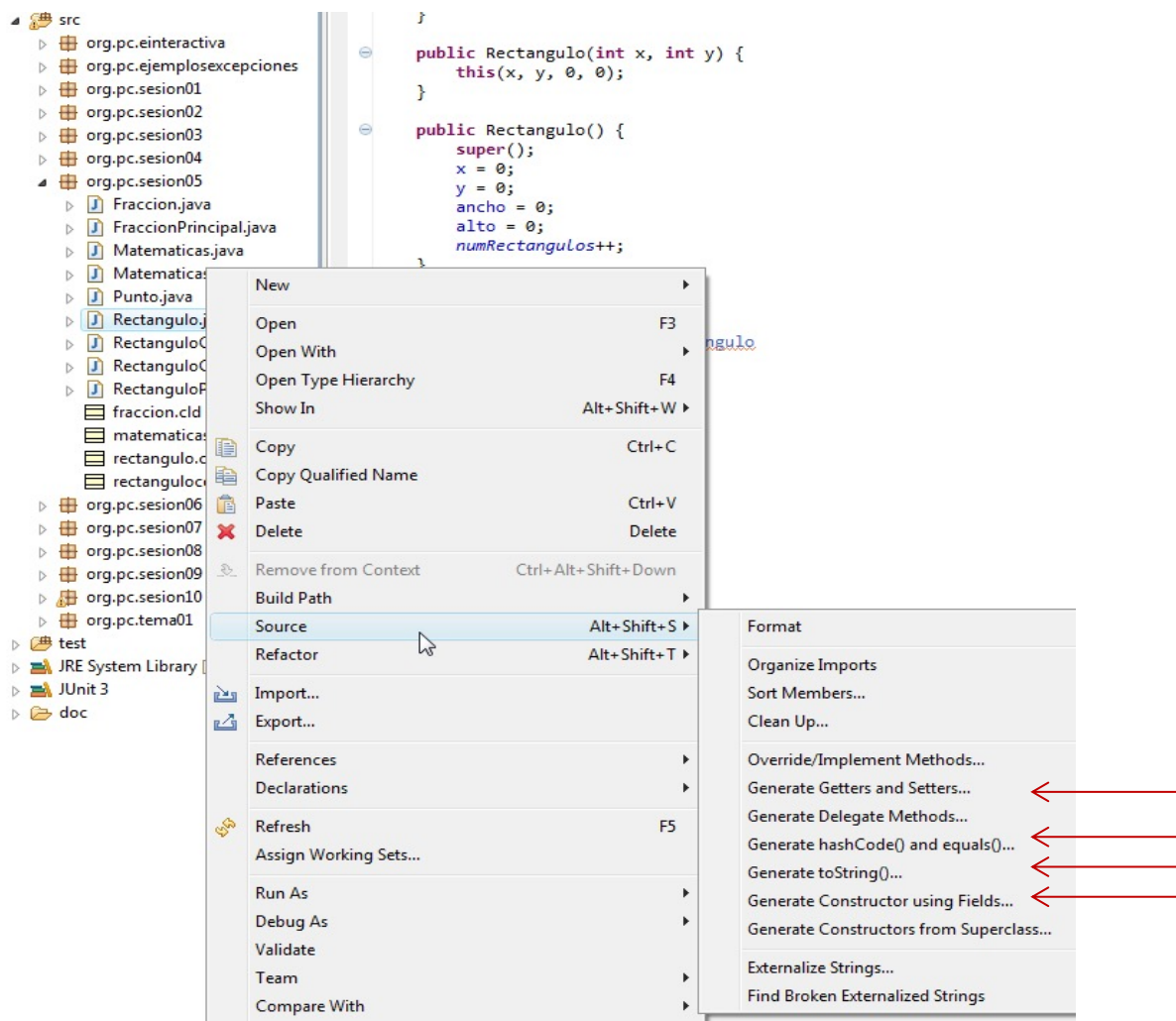
## ➤ ☐ compareTo

Es un método que se utiliza para comparar, por algún criterio, dos objetos. Devuelve **0** si los objetos son iguales, **1** si el primer objeto es mayor y **-1** si el primer objeto es menor.

```
public int compareTo(Object obj) {  
    Rectangulo otro = (Rectangulo) obj;  
    if (calcularArea() == otro.calcularArea())  
        return 0;  
    else if (calcularArea() < otro.calcularArea())  
        return -1;  
    else  
        return 1;  
}
```

Ejemplo de uso: **if (rect1.compareTo(rect2) == 0)**  
    *System.out.println("Rectangulos iguales");*  
**else if (rect1.compareTo(rect2) == -1)**  
    *System.out.println("El primer rectángulo es menor que el segundo");*  
**else**  
    *System.out.println("El primer rectángulo es mayor que el segundo");*

Eclipse da la facilidad de escribir de manera automática algunos de estos métodos.



```
1 package org.ip.tema03;
2
3 public class Rectangulo {
4     private int x;
5     private int y;
6     private int ancho;
7     private int alto;
8     private static int numRectangulos = 0;
9
10    public Rectangulo(int x, int y, int ancho, int alto) {
11        super();
12        this.x = x;
13        this.y = y;
14        this.ancho = ancho;
15        this.alto = alto;
16        numRectangulos++;
17    }
18
19    public Rectangulo(int x, int y) {
20        this(x, y, 0, 0);
21    }
22
23    public Rectangulo(Rectangulo rect) {
24        super();
25        this.x = rect.x;
26        this.y = rect.y;
27        this.ancho = rect.ancho;
28        this.alto = rect.alto;
29        numRectangulos++;
30    }
31
```

```
32- public int getX() {
33     return x;
34 }
35- public void setX(int x) {
36     this.x = x;
37 }
38- public int getY() {
39     return y;
40 }
41- public void setY(int y) {
42     this.y = y;
43 }
44- public int getAncho() {
45     return ancho;
46 }
47- public void setAncho(int ancho) {
48     this.ancho = ancho;
49 }
50- public int getAlto() {
51     return alto;
52 }
53- public void setAlto(int alto) {
54     this.alto = alto;
55 }
56- public static int getNumRectangulos() {
57     return numRectangulos;
58 }
59
60- @Override
61 public String toString() {
62     return "Rectangulo [x=" + x + ", y=" + y + ", ancho=" + ancho
63         + ", alto=" + alto + "]";
64 }
65
```

```
66- @Override
67 public boolean equals(Object obj) {
68     if (this == obj)
69         return true;
70     if (obj == null)
71         return false;
72     if (getClass() != obj.getClass())
73         return false;
74     Rectangulo otro = (Rectangulo) obj;
75     return x == otro.x && y == otro.y && ancho == otro.ancho
76         && alto == otro.alto;
77 }
78
79- /**
80  * Metodo que compara dos rectangulos, devolviendo 0 si son iguales,
81  * -1 si el primero es menor y 1 si el primero es mayor
82  * @param obj
83  * @return un entero 0, -1, 1
84  */
85- public int compareTo(Object obj) {
86     Rectangulo otro = (Rectangulo) obj;
87
88     if (calcularArea() == otro.calcularArea())
89         return 0;
90     else if (calcularArea() < otro.calcularArea())
91         return -1;
92     else
93         return 1;
94 }
95
96- public void desplazar(int dx, int dy) {
97     x = x + dx;
98     y = y + dy;
99 }
```

```
100
101 public boolean estaDentro(int x1, int y1) {
102     if (x1 > x && x1 < x + ancho && y1 > y && y1 < y + alto)
103         return true;
104     else
105         return false;
106 }
107
108 public int calcularArea() {
109     return ancho * alto;
110 }
111 }
112
```



```

1 package org.ip.tema03;
2
3 public class RectanguloPrincipal {
4
5     public static void main(String[] args) {
6         Rectangulo rect1 = new Rectangulo(10, 15, 20, 5);
7         Rectangulo rect2 = new Rectangulo(7, 15, 7, 5);
8         Rectangulo rect3 = new Rectangulo(10, 15, 20, 20);
9
10        int altura = rect1.getAlto();
11        if (altura < 10)
12            System.out.println("Rectangulo 1 pequeno");
13        System.out.println("Rectangulo 1 " + rect1.toString());
14        if (rect1.equals(rect2))
15            System.out.println("Mismos rectangulos 1 y 2");
16        else
17            System.out.println("Distintos rectangulos 1 y 2");
18        if (rect1.equals(rect3))
19            System.out.println("Mismos rectangulos 1 y 3");
20        else
21            System.out.println("Distintos rectangulos 1 y 3");
22        System.out.println("El numero de rectangulos creados es "
23            + Rectangulo.getNumRectangulos());
24        rect1.desplazar(7, 9);
25        System.out.println("Rectangulo 1 " + rect1.toString());
26        if (rect1.compareTo(rect3) == 0)
27            System.out.println("Rectangulos 1 y 3 con igual area");
28        else if (rect1.compareTo(rect3) == -1)
29            System.out.println("Rectangulo 1 tiene menor area que 3");
30        else
31            System.out.println("Rectangulo 1 tiene mayor area que 3");
32        if (rect1.estaDentro(20, 30))
33            System.out.println("El punto (20, 30) esta dentro del rectangulo 1");
34        else
35            System.out.println("El punto (20, 30) no esta dentro del rectangulo 1");
36    }
37 }

```

## Salida

```

Rectangulo 1 pequeno
Rectangulo 1 Rectangulo [x=10, y=15, ancho=20, alto=5]
Distintos rectangulos 1 y 2
Distintos rectangulos 1 y 3
El numero de rectangulos creados es 3
Rectangulo 1 Rectangulo [x=17, y=24, ancho=20, alto=5]
Rectangulo 1 tiene menor area que 3
El punto (20, 30) no esta dentro del rectangulo 1

```



**¡MUCHAS GRACIAS!**



UNIVERSIDAD DE ALMERÍA

