TEMA: HERENCIA EN JAVA

Taller de Programación.

Módulo: Programación Orientada a Objetos

Introducción

- Diferentes tipos de objetos con características y comportamiento común.

Triángulo 	**Círculo** 	**Cuadrado** 
• Lado1 / lado2 / lado3 • color de línea • color de relleno	• radio • color de línea • color de relleno	• lado • color de línea • color de relleno
• Devolver y modificar el valor de cada atributo lado1 / lado2 / lado3 color de línea / color de relleno • Calcular el área • Calcular el perímetro	• Devolver y modificar el valor de cada atributo radio color de línea / color de relleno • Calcular el área • Calcular el perímetro	• Devolver y modificar el valor de cada atributo lado color de línea / color de relleno • Calcular el área • Calcular el perímetro

Inconvenientes hasta ahora. Herencia como solución.

- Esquema de trabajo hasta ahora:
  - Definimos las clases Triángulo, Circulo…
  - Problemas: Replicación de características y comportamiento común.
- Solución → Herencia
  - Permite que la clase ***herede*** características y comportamiento (atributos y métodos) de otra clase (clase padre o superclase). A su vez, la clase define características y comportamiento propio.
  - Ejemplo. Se define **lo común en una clase Figura** y las **clases Triángulo , Círculo y Cuadrado lo heredan.**
  - Ventaja: **reutilización**

Herencia. Ejemplo.

- Diagrama de clases.

Las clases forman una jerarquía.

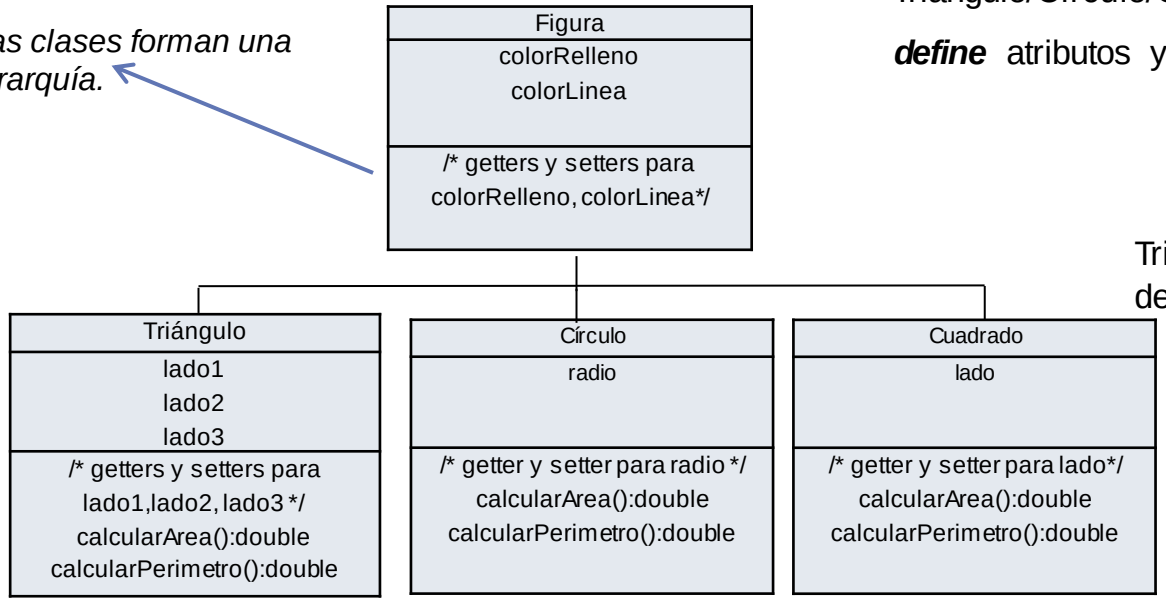


Figura es la **superclase** de Triángulo/Círculo/Cuadrado

define atributos y comportamiento **común**

Triángulo/Círculo/Cuadrado son **subclases** de Figura.

heredan atributos y métodos de Figura

definen atributos y métodos **propios**

definen constructores.

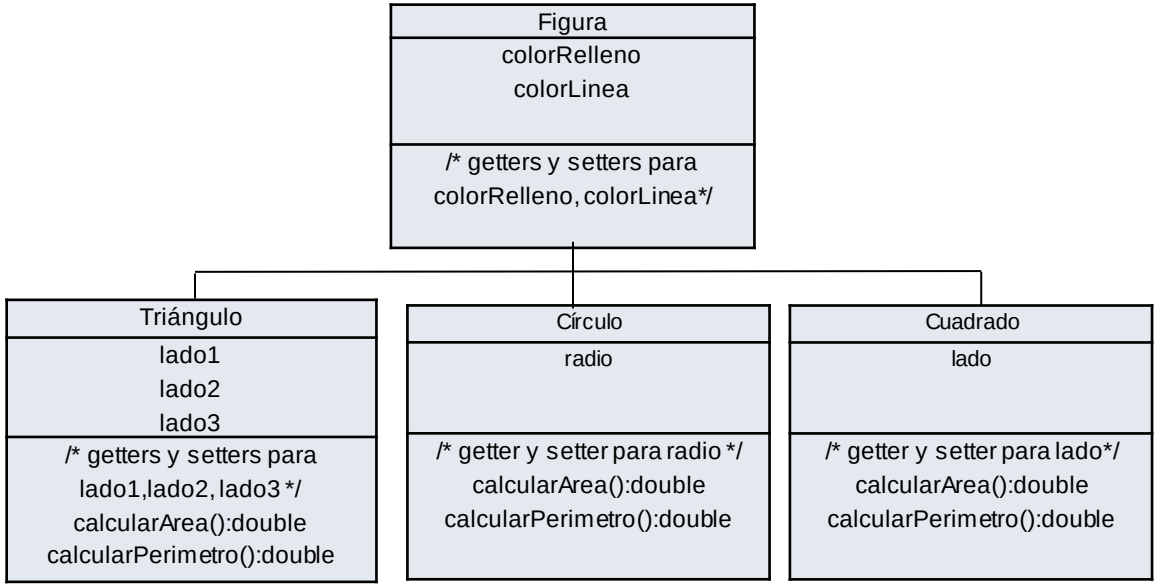
deben *implementar calcularArea() y calcularPerimetro pero de manera diferente*

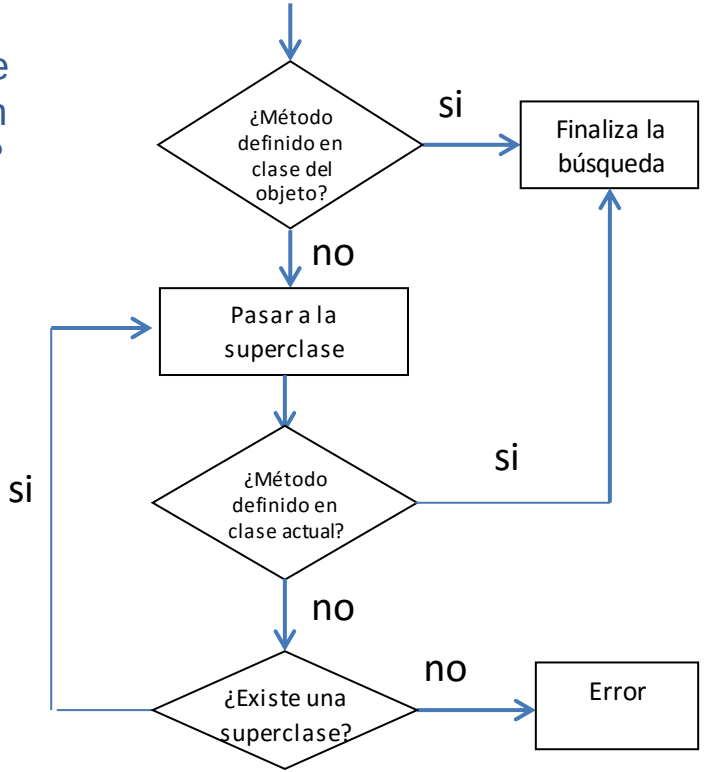
Búsqueda de método en la jerarquía de clases

- Ejemplo … en el main

```
Cuadrado c = new Cuadrado(10,"rojo","negro");
System.out.println(c.calcularArea());
System.out.println(c.getColorRelleno());
```

¿Qué mensajes le puedo enviar a un objeto cuadrado?





Herencia en Java

- Definición de relación de herencia. Palabra clave *extends*.

```
public class NombreSubclase extends NombreSuperclase{
    /* Definir atributos propios */
    /* Definir constructores propios */
    /* Definir métodos propios */
}
```

- La **subclase** *hereda*
  - Atributos declarados en la **superclase**. Como son *privados* son accesibles sólo en métodos de la clase que los declaró. *En la subclase accederlos a través de getters y setters heredados. Ej: getColorRelleno() ó setColorRelleno(#)*
  - Métodos de instancia declarados en la **superclase**.
- La **subclase** *puede declarar*
  - Atributos / métodos / constructores propios.

Clases y métodos abstractos

- **Clase abstracta:** clase de la cual no se crearán objetos. Puede definir métodos sin implementación (***métodos abstractos***) para obligar a las subclases a implementarlos.
  - Ejemplos: la clase **Figura** es *abstracta*. Podría declarar métodos **calcularArea** /**calcularPerimetro** *abstractos*.
  - Declaración de clase abstracta: anteponer ***abstract*** a la palabra class.

```
public abstract class NombreClase {
      /* Definir atributos */
      /* Definir métodos no abstractos (con implementación) */
     /* Definir métodos abstractos  (sin implementación) */
}
```

  - Declaración de método abstracto: sólo se pone el encabezado del método (sin código) anteponiendo ***abstract*** al tipo de retorno.

```
public abstract TipoRetorno nombreMetodo(lista parámetros formales);
```

Ejemplo

Superclase

```
public abstract class Figura{
    private String colorRelleno, colorLinea;

    public String getColorRelleno(){
        return colorRelleno;
    }
    public void setColorRelleno(String unColor){
        colorRelleno = unColor;
    }
    ....
```

MÉTODOS COMUNES

```
    public abstract double calcularArea();
    public abstract double calcularPerimetro();
```

MÉTODOS ABSTRACTOS

```
}
```

Subclase

```
public class Cuadrado extends Figura{
    private double lado;

    /*Constructores*/
    public Cuadrado(double unLado,
                    String unColorR, String unColorL){
        lado=unLado;
        colorRelleno=unColorR;
        colorLinea=unColorL;
    }

    /* Metodos getLado y setLado */
    ....
```

setColorRelleno(unColorR);
setColorLinea(unColorL);

```
    public double calcularPerimetro(){
        return lado*4;
    }
    public double calcularArea(){
        return lado*lado;
    }
```

Implementa

```
}
```