Clases abstractas e interfaces

Clases abstractas

Una clase abstracta…

es una clase que no se puede instanciar

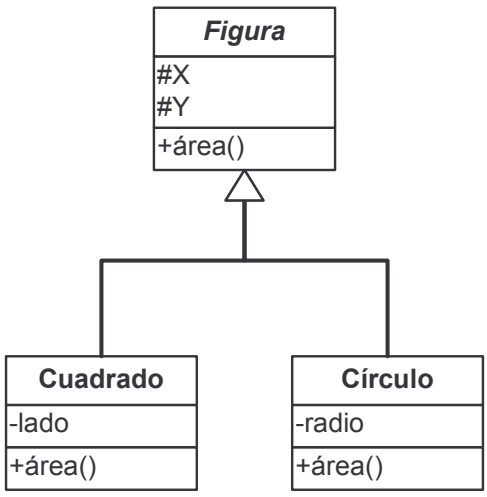
se usa únicamente para definir subclases

¿Cuándo es una clase abstracta?

En cuanto uno de sus métodos no tiene implementación (en Java, el método abstracto se etiqueta con la palabra reservada `abstract`).

¿Cuándo se utilizan clases abstractas?

Cuando deseamos definir una abstracción que englobe objetos de distintos tipos y queremos hacer uso del polimorfismo.

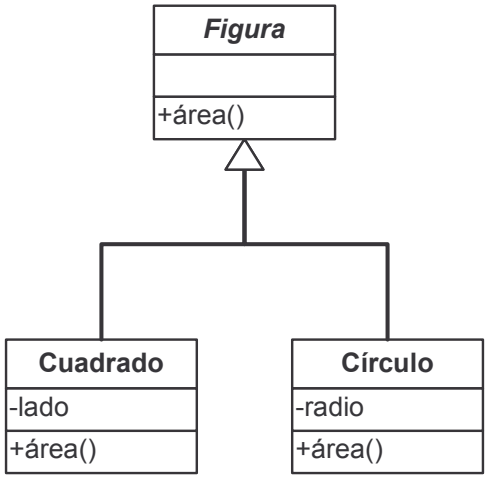


`Figura` es una clase abstracta (nombre en cursiva en UML) porque no tiene sentido calcular su área, pero sí la de un cuadrado o un círculo. Si una subclase de Figura no redefine `area()`, deberá declararse también como clase abstracta.

```
public abstract class Figura
{
  protected double x;
  protected double y;

  public Figura (double x, double y)
  {
    this.x = x;
    this.y = y;
  }

  public abstract double area ();
}


public class Circulo extends Figura
{
  private double radio;

  public Circulo (double x, double y, double radio)
  {
    super(x,y);
    this.radio = radio;
  }

  public double area ()
  {
    return Math.PI*radio*radio;
  }
}


public class Cuadrado extends Figura
{
  private double lado;

  public Cuadrado (double x, double y, double lado)
  {
    super(x,y);
    this.lado = lado;
  }

  public double area ()
  {
    return lado*lado;
  }
}
```

Una interfaz es una clase completamente abstracta (una clase sin implementación)

En el ejemplo anterior, si no estuviésemos interesados en conocer la posición de una `Figura`, podríamos eliminar por completo su implementación y convertir `Figura` en una interfaz:




```
public interface Figura
{
  public double area ();
}
```

- En Java, las interfaces se declaran con la palabra reservada **`interface`** de manera similar a como se declaran las clases abstractas.

- En la declaración de una interfaz, lo único que puede aparecer son declaraciones de métodos (su nombre y signatura, sin su implementación) y definiciones de constantes simbólicas.

- Una interfaz no encapsula datos, sólo define cuáles son los métodos que han de implementar los objetos de aquellas clases que implementen la interfaz.

```
public class Circulo implements Figura
{
  private double radio;

  public Circulo (double radio)
  {
    this.radio = radio;
  }

  public double area ()
  {
    return Math.PI*radio*radio;
  }
}


public class Cuadrado implements Figura
{
  private double lado;

  public Cuadrado (double lado)
  {
    this.lado = lado;
  }

  public double area ()
  {
    return lado*lado;
  }
}
```

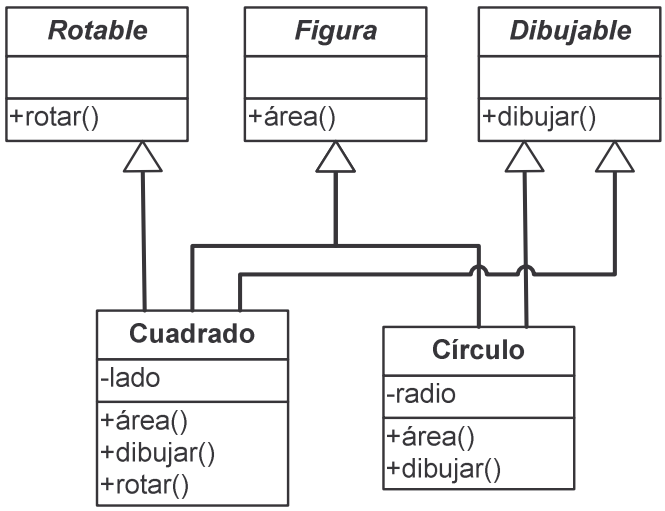
- En Java, para indicar que una clase implementa una interfaz se utiliza la palabra reservada **`implements`**.

- La clase debe entonces implementar **todos** los métodos definidos por la interfaz o declararse, a su vez, como una clase abstracta (lo que no suele ser especialmente útil):

```
abstract class SinArea implements Figura
{
}
```

Herencia múltiple de interfaces

Una clase puede implementar varios interfaces simultáneamente, pese a que, en Java, una clase sólo puede heredar de otra clase (herencia simple de implementación, múltiple de interfaces).




```
public abstract class Figura
{
  public abstract double area ();
}

public interface Dibujable
{
  public void dibujar ();
}

public interface Rotable
{
  public void rotar (double grados);
}


public class Circulo extends Figura
                    implements Dibujable
…

public class Cuadrado extends Figura
                     implements Dibujable, Rotable
…
```