Interfaces

Dr. José Lázaro Martínez Rodríguez

¿Qué es un interfaz?

- Es una clase abstracta "pura"
- Permite al programador establecer la forma de una clase:
  - Nombres de métodos, lista de parámetros, tipos de retorno, pero NO su bloque de instrucciones.
- Una interfaz puede contener atributos, pero estos serán implícitamente estáticos y constantes (no se pueden modificar, desde la clase que implementa esa interfaz)
- Una interfaz proporciona solo la forma pero no la implementación

Interfaces

Las interfaces son un tipo de clase especial que no implementa ninguno de sus métodos (todos son abstractos).

No se pueden instanciar.

Si una clase que hereda de una interface no implementa todos los métodos de esta, deberá ser definida como abstracta.

Interfaces

Las interfaces en Java solucionan en parte la NO existencia de la herencia múltiple.

Una interface no puede implementar ningún método.

Una interface no forma parte de la jerarquía de clases.

Una clase puede implementar n interfaces pero solo puede extender de una sola clase.

Todos los métodos de una interface implícitamente son `public` (no debe utilizarse un modificador de acceso diferente).

Interfaces en Java

- Ejemplo:

```
public interface nombre_interface {
  static final int var = 2000;
    tipo_retorno nombre_metodo ( lista_argumentos ) ;
     . . .
}
```

```
interface InstrumentoMusical {
    void tocar();
    void afinar();
    String tipoInstrumento();
}

class InstrumentoViento implements InstrumentoMusical{
    void tocar() { . . . };
    void afinar() { . . .};
    String tipoInstrumento() {...  ... ... }
}

class InstrumentoViento extends Object implements InstrumentoMusical{
    void tocar() { . . . };
    void afinar() { . . .};
    String tipoInstrumento() {... ... ... }
}

class Guitarra extends InstrumentoViento {
    String tipoInstrumento() {
        return "Guitarra";
    }
}
```

Referencias a interfaces

- No se pueden instanciar, pero sí referenciar
- SI permite múltiples implementaciones dentro de una clase

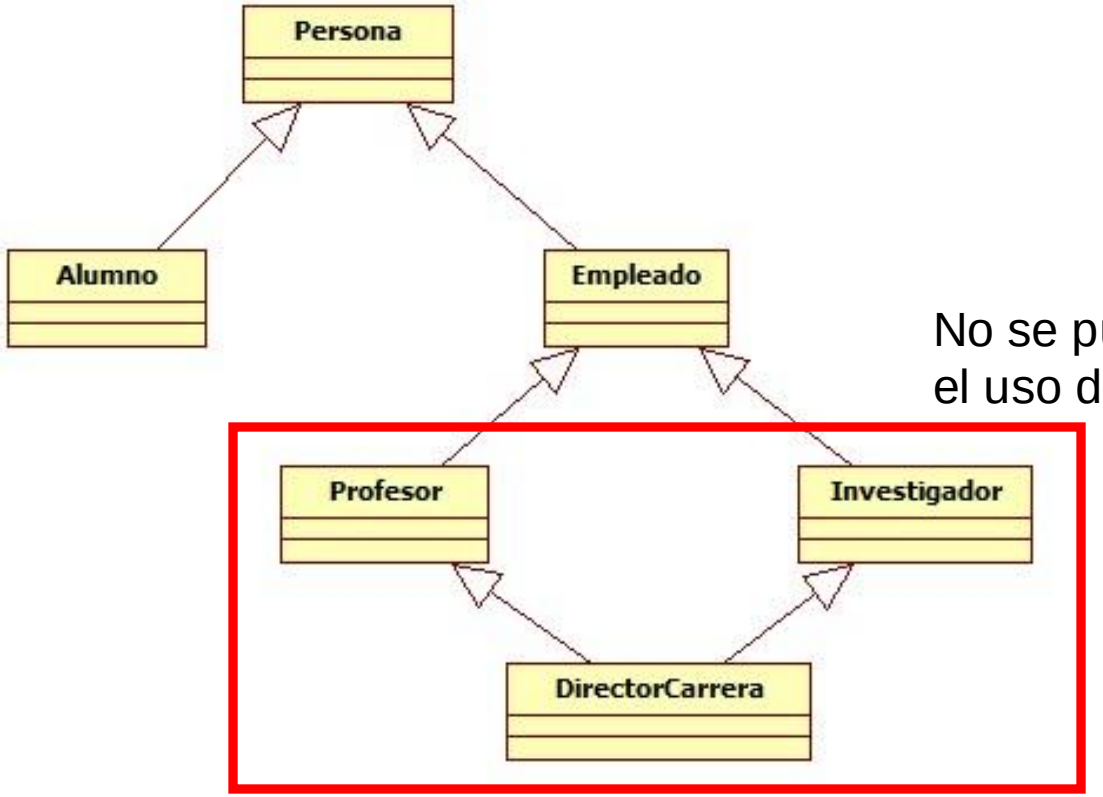
```
InstrumentoMusical instrumento = new Guitarra();
instrumento.play();
System.out.println(instrumento.tipoInstrumento());
```

```
InstrumentoMusical i2 = new InstrumentoMusical(); //ERROR
```

```
interface nombre_interface extends nombre_interface, ... {
    tipo_retorno nombre_metodo ( lista_argumentos ) ;
}
```

En Java

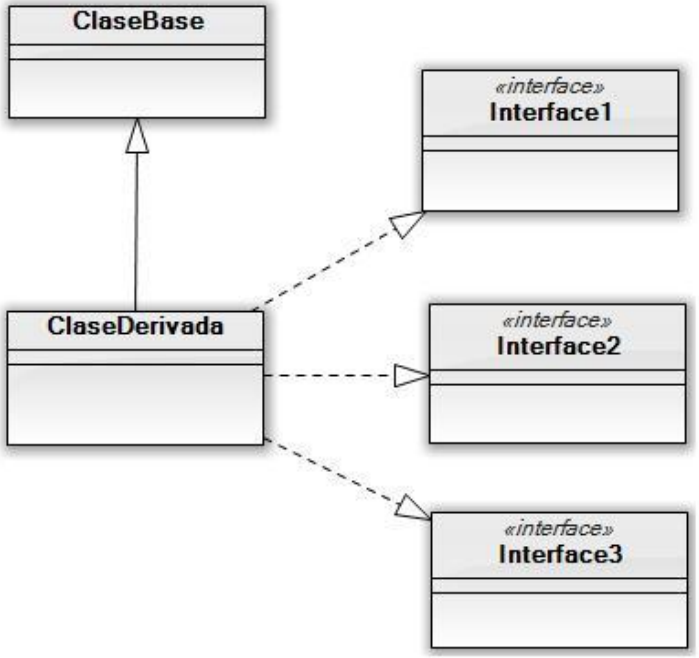
- No existe la herencia múltiple, solo se puede heredar de una sola clase.



No se puede mediante el uso de `extends`

Herencia

- Pero las interfaces proporcionan una alternativa para implementar algo parecido a la herencia múltiple de otros lenguajes. En java una clase solo puede tener una clase base pero puede implementar múltiples interfaces.



¿Cuál es la diferencia entre una interfaz y una clase abstracta?

- Como Java no permite herencia múltiple – una clase sólo puede extender una superclase –
  - esto dificulta que una clase se adecue a más de un comportamiento.
  - Una interfaz, por el contrario, permite que una clase implemente una o más interfaces para resolver el problema de mezclar diversos comportamientos en un mismo tipo de objeto.
- Una clase abstracta ofrece comportamientos comunes a objetos del mismo tipo a través del mecanismo de la herencia.
  - La implementación de una interfaz permite a un objeto comportamientos que no dependen de su jerarquía de clases.

Interfaz y clase abstracta

- Una interfaz se diferencia de una clase abstracta porque una interfaz sólo puede contener constantes y métodos abstractos. Una clase abstracta puede contener métodos concretos, una interfaz no.
- En una interfaz todos los atributos son por defecto **public final static** (constantes) y todos los métodos son **public abstract.** Esto quiere decir que una clase abstracta puede contener atributos variables pero una interfaz no.

Animal Interface

```
public interface Animal
 {
    public void locomotion();
    public void eat();
 }
```

Class Shark

```
public class Shark implements Animal
 {
    public void locomotion()
    {
      System.out.println("I swim.");
    }
    public void eat()
    {
      System.out.println("I hunt for seals.");
    }
 }
```

Class Dog

```
public class Dog implements Animal
 {
   public void locomotion()
    {
      System.out.println("I run on four legs.");
    }
   public void eat()
    {
      System.out.println("I eat kibble.");
    }
 }
```

```
public class  AnimalTest
{
   public static void main(String[] arg)
     {
       Shark shark = new Shark();
       shark.locomotion();
       shark .eat();
       Dog dog = new dog();
       dog.locomotion();
       dog.eat();
     }
}
```

Aplicaciones con interfaces

- Las interfaces juegan un papel fundamental en la creación de aplicaciones Java ya que permiten interactuar a objetos no relacionados entre sí.
- Utilizando interfaces es posible que clases no relacionadas, situadas en distintas jerarquías de clases sin relaciones de herencia, tengan comportamientos comunes.
- **Las interfaces definen un protocolo de comportamiento y proporcionan un formato común para implementarlo en las clases.**

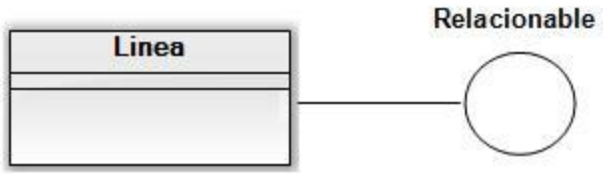
Nomenclatura

- Los nombres de las interfaces suelen acabar en *able* (aunque no es necesario): configurable, arrancable, dibujable, etc.
- Ejemplo de interface en Java:

```
interface Relacionable
//Interface que define relaciones de orden entre objetos.
public interface Relacionable {
    boolean esMayorQue(Object a);
    boolean esMenorQue(Object a);
    boolean esIgualQue(Object a);
}
```

Nomenclatura

- Una clase `Linea` que implementa la interfaz `Relacionable`. En ese caso se dice que la clase **Linea es Relacionable**
- En UML:



- O

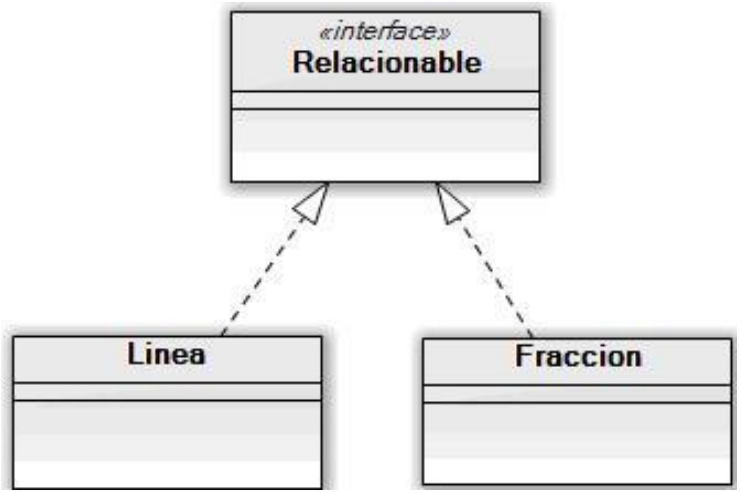
Interfaces y polimorfismo

- La definición de una interface implica una definición de un nuevo tipo de referencia y por ello **se puede usar el nombre de la interface como nombre de tipo**.
- El nombre de una interfaz se puede utilizar en cualquier lugar donde pueda aparecer el nombre de un tipo de datos.
- Si se define una variable cuyo tipo es una interface, se le puede asignar un objeto que sea una instancia de una clase que implementa la interface.

Implementando

- El la figura de ejemplo las clases Linea y Fraccion implementan la interfaz Relacionable.
- Entonces podemos escribir las instrucciones:

```
Relacionable r1 = new Linea(2,2,4,1);
Relacionable r2 = new Fraccion(4,7);
```

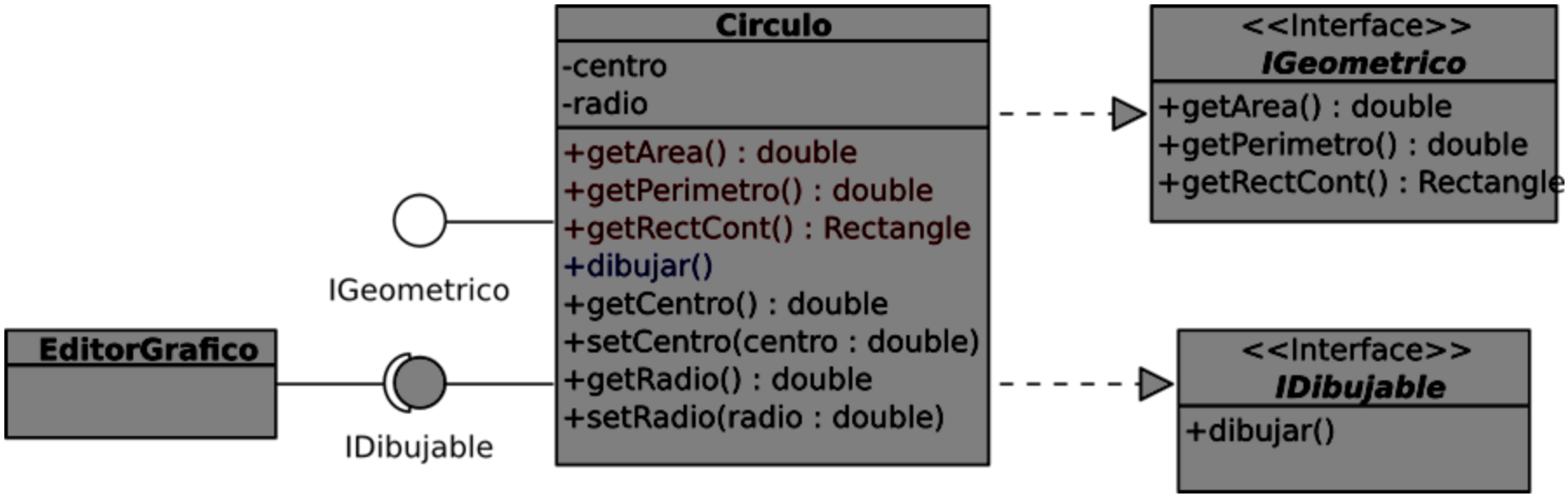


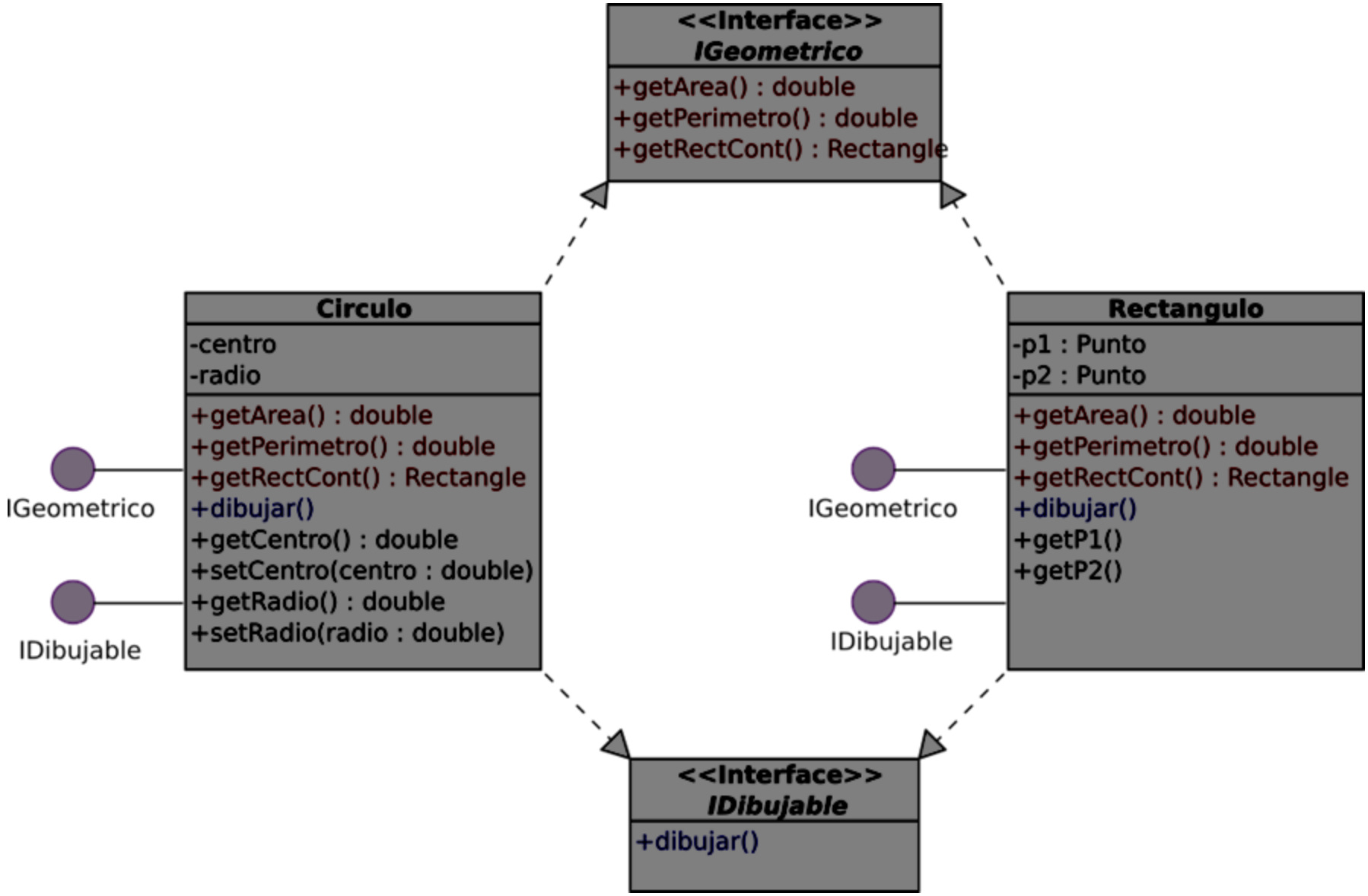
Implementando

- Podemos crear un array de referencias de tipo Relacionable y asignarles referencias a objetos de clases que la implementan.

```
Relacionable [] array = new Relacionable[3];
array[0] = new Linea(2,2,4,1);
array[1] = new Fraccion(4,7);
array[2] = new Linea(14,3,22,1);
for(Relacionable r: array)
            System.out.println(r);
```

Implements de múltiples interfaces




<<Interface>>
IGeometrico
+getArea() : double
+getPerimetro() : double
+getRectCont() : Rectangle
Circulo
-centro
-radio
+getArea() : double
+getPerimetro() : double
+getRectCont() : Rectangle
+dibujar()
+getCentro() : double
+setCentro(centro : double)
+getRadio() : double
+setRadio(radio : double)
IGeometrico
IDibujable
Rectangulo
-p1 : Punto
-p2 : Punto
+getArea() : double
+getPerimetro() : double
+getRectCont() : Rectangle
+dibujar()
+getP1()
+getP2()
IGeometrico
IDibujable
<<Interface>>
IDibujable
+dibujar()

```
public interface IGeometrico {

  public double getArea();

  public double getPerimetro();

  public Rectangle getRectCont();
}
```

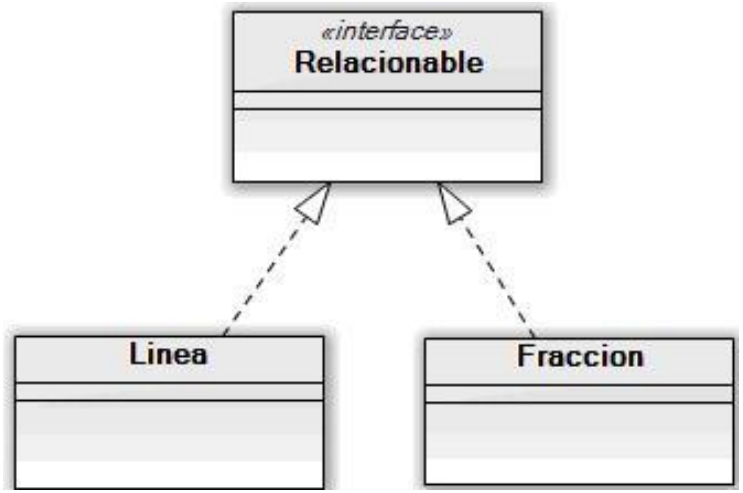
```
public interface IDibujable {
  public void dibujar();
}
```

En Java:

```
import java.awt.Point;
import java.awt.Rectangle;

public class Circulo implements IGeometrico, IDibujable {

  private double centro;
  private double radio;

  public double    getArea()     { /* de IGeometrico */ }
  public double    getPerimetro() { /* de IGeometrico */ }
  public Rectangle getRectCont()  { /* de IGeometrico */ }

  public void dibujar()         { /* de IDibujable */ }

  public Point  getCentro()      { /* de circulo */ }
  public void   setCentro(...)   { /* de circulo */ }

  public double getRadio()       { /* de circulo */ }
  public void   setRadio(...)    { /* de circulo */ }
}
```

Práctica:

- Implementar la interfaz Relacionable y las clases Línea y Fracción.



Ejercicio

- Escriba un programa Java para crear una interfaz Shape con el método getArea(). Cree tres clases Rectángulo, Círculo y Triángulo que implementen la interfaz Shape. Implementar el método getArea() para cada una de las tres clases.