

JAVA BÁSICO

Herencia y polimorfismo

Contenido

2

- ☐ Herencia
- ☐ Polimorfismo

Interfaz

3

- La interacción de los objetos con el mundo exterior se realiza a través de los métodos públicos que ofrece.
- A éste conjunto de métodos se le llama la interfaz del objeto.
- Por ejemplo, el conjunto de botones que están al frente de la televisión, son la interfaz entre los componentes electrónicos internos de la televisión y el mundo exterior. Cada botón ofrece una funcionalidad, por ejemplo, el botón de encendido, prende y apaga la televisión.
- La forma común de una interfaz es un conjunto de métodos sin implementación.

Ejemplo

4

- Imaginemos que nos solicitan un sistema para hacer cálculos sobre figuras geométricas.
- El sistema no necesita dibujar, solamente debe implementar operaciones matemáticas sobre las figuras.
- Las primeras operaciones serán el cálculo del perímetro y el área.
- Para asegurar que todas las figuras geométricas ofrezcan las mismas operaciones, todas las clases deben implementar la interfaz IFigura.

Ejemplo

5


```
public interface IFigura {  
  
    public double area ();  
    public double perimetro ();  
  
}
```

- ❑ Implementar una interfaz, permite a una clase formalizar el comportamiento que promete proveer.
- ❑ Cuando una clase implementa una interfaz debe dar cuerpo a todos los métodos definidos en ella.
- ❑ Se pueden implementar cualquier cantidad de interfaces.
- ❑ Para implementar una interfaz se usa la palabra reservada **implements**

Ejemplo

6

```
public class Rectangulo implements IFigura {  
  
    double ladoA;  
    double ladoB;  
  
    public Rectangulo (double A, double B){  
        ladoA = A;  
        ladoB = B;  
    }  
    @Override  
    public double area() {  
        return ladoA*ladoB;  
    }  
}
```




Método de la
interfaz, ya
implementado

Ejemplo

7

```
public class Rectangulo implements IFigura {  
  
    double ladoA;  
    double ladoB;  
  
    public Rectangulo (double A, double B){  
        ladoA = A;  
        ladoB = B;  
    }  
    @Override  
    public double area() {  
        return ladoA*ladoB;  
    }  
    @Override  
    public double perimetro() {  
        return 2*ladoA+2*ladoB;  
    }  
}
```



Métodos de la
interfaz, ya
implementados

8

Herencia

Herencia

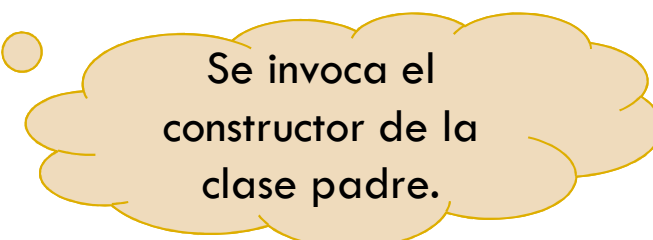
9

- La herencia es la capacidad de las clases de Java de utilizar el comportamiento y los atributos de una clase definida previamente.
- La clase que hereda el comportamiento y los atributos de otra clase, puede agregar funcionalidad y atributos propios, de tal manera que se diferencie de la clase padre.
- Permite la reutilización de código.
- Solo se puede heredar de una clase.
- Para heredar de una clase se utiliza la palabra reservada **extends**

Ejemplo Herencia

10

```
public class Cuadrado extends Rectangulo {  
  
    public Cuadrado (double lado){  
        super (lado, lado);  
    }  
}
```



Se invoca el constructor de la clase padre.

```
public static void main (String [] args){  
    Cuadrado c= new Cuadrado (3);  
    System.out.println ("P="+c.perimetro ());  
    System.out.println ("A="+c.area ());  
    System.out.println (c);  
}
```

Clase abstracta

11

- Una clase abstracta, es una clase que puede tener métodos sin implementar.
- No se pueden crear objetos directamente de una clase abstracta. Se debe crear una clase que herede a esta clase para que implemente los métodos que no tienen cuerpo.
- Para definir una clase abstracta se usa la palabra reservada **abstract**.

```
public abstract class Poligono{

    String nombre;
    int numLados;

    public Poligono (int lados, String n){
        numLados = lados;
        nombre = n;
    }

    public String toString (){
        return nombre;
    }

    public abstract double semiPerimetro ();
    public abstract boolean esEquilatero ();
}
```

```

public class Circulo extends Poligono implements IFigura{
    double radio;

    public Circulo (double radio, String nombre){
        super (0, nombre);
        this.radio = radio;
    }

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

    @Override
    public double perimetro() { return 2*Math.PI*radio; }

    @Override
    public double semiPerimetro () { return perimetro () /2; }

    @Override
    public boolean esEquilatero () {
        return true;
    }
}

```

```

public class Rectangulo extends Poligono implements IFigura {
    double ladoA;
    double ladoB;
    public Rectangulo (double A, double B, String nombre){
        super (4, nombre);
        ladoA = A;
        ladoB = B;
    }
    @Override
    public double area() { return ladoA*ladoB; }
    @Override
    public double perimetro() { return 2*ladoA+2*ladoB; }
    @Override
    public double semiPerimetro (){ return perimetro () /2;
}

    @Override
    public boolean esEquilatero (){
        if (ladoA == ladoB)
            return true;
        else
            return false;
    }
}

```

Código para la clase Main

15

```
public class FigurasGeometricas {  
  
    public static void main(String[] args) {  
        Rectangulo rectangulo = new Rectangulo (2,  
                                                    3,  
                                                    "Rectangulo");  
        Circulo circulo = new Circulo (5, "Circulo");  
  
        System.out.print (rectangulo+" Area = ");  
        System.out.println (rectangulo.area());  
        System.out.print (circulo+" Area = ");  
        System.out.println (circulo.area());  
    }  
}
```

Ejemplo Herencia

16

```
public class Cuadrado extends Rectangulo {  
  
    public Cuadrado (double lado){  
        super (lado, lado, "Cuadrado");  
    }  
}
```


17

Polimorfismo

Polimorfismo

18

- Polimorfismo significa "muchas formas".
- Es la capacidad de los objetos de Java para comportarse como sus padres, o como sus interfaces.
- Es de gran utilidad para manejar conjuntos de objetos de diferentes clases pero que heredan de la misma o implementan la misma interfaz, como si fueran todos del mismo tipo.

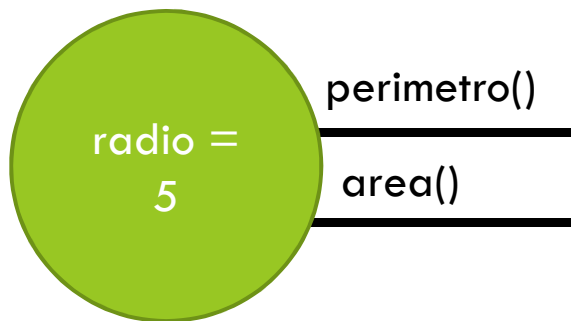
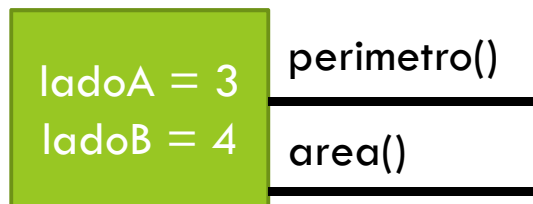
Ejemplo polimorfismo

19

```
public static void main(String[] args) {  
    Rectangulo rectangulo = new Rectangulo (2,  
                                            3,  
                                            "Rectangulo");  
    Circulo circulo = new Circulo (5, "Circulo");  
    Cuadrado cuadrado = new Cuadrado (9);  
  
    IFigura [] figuras = new IFigura[3];  
    figuras[0] = rectangulo;  
    figuras[1] = circulo;  
    figuras[2] = cuadrado;  
  
    for (int i = 0; i<figuras.length; i++){  
        System.out.println ("Perímetro de "+figuras[i]+"  
                             = "+figuras[i].perimetro());  
    }  
}
```

Representación Gráfica

20

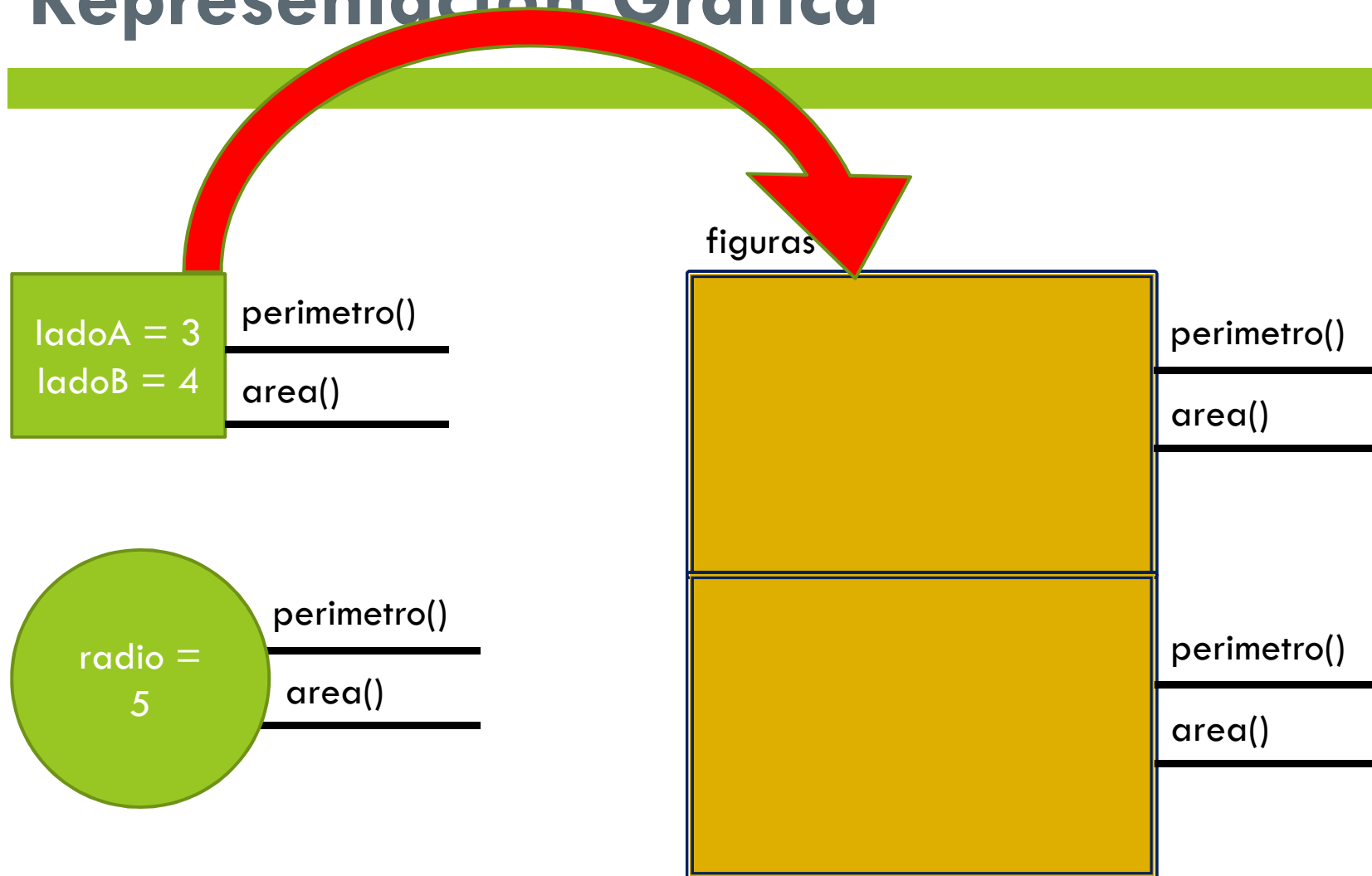


figuras



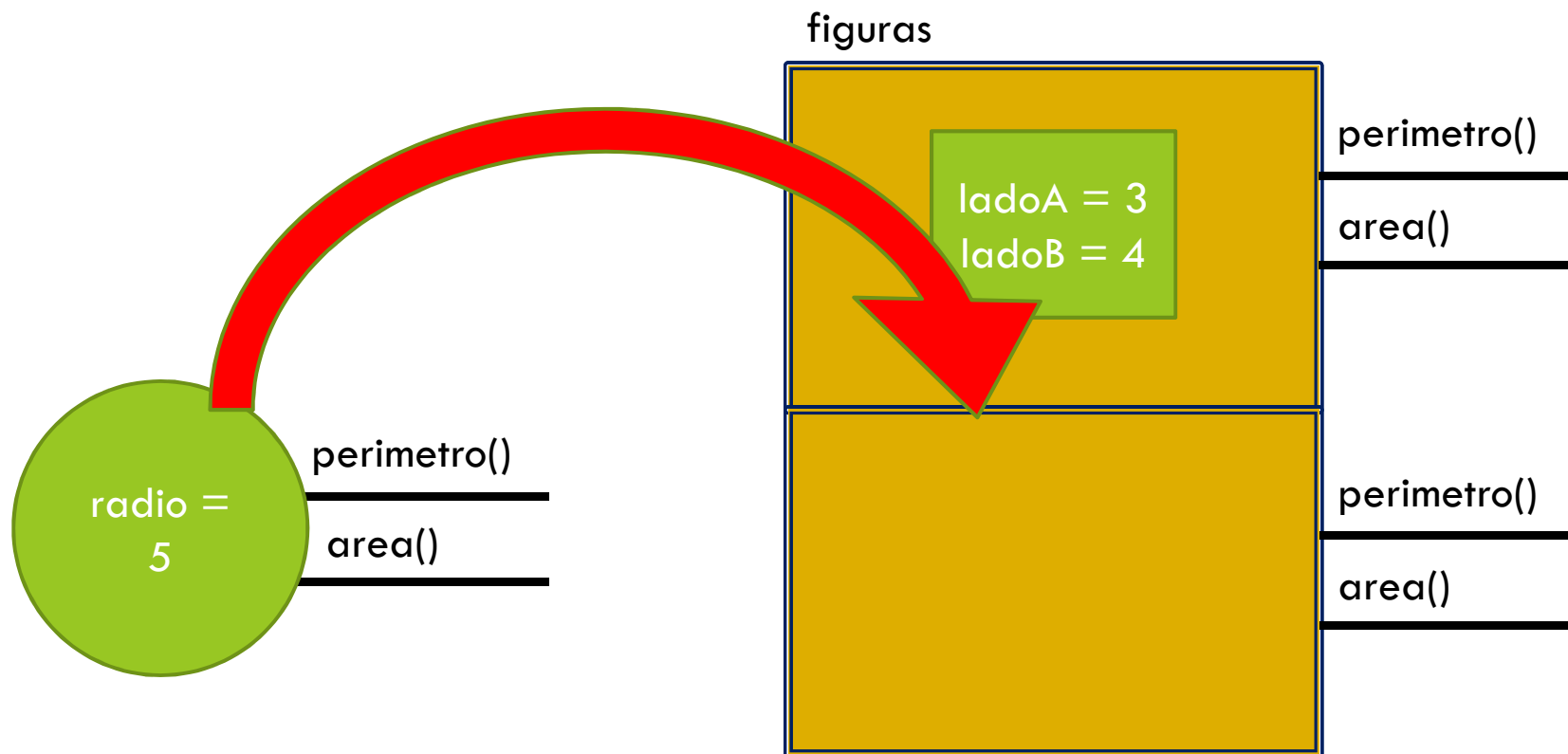
Representación Gráfica

21



Representación Gráfica

22



Representación Gráfica

23

