ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Universidade Vigo

TEMA 6: Polimorfismo

Programación II - 2017/2018

Pedro Cuesta Morales, Baltasar García Pérez-Schofield,
Encarnación González Rufino (Dpto. de Informática)

Índice

1. Introducción
2. Enlace tardío
3. Polimorfismo

1. Introducción

Polimorfismo:

"En programación orientada a objetos, el polimorfismo se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos. El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía." (Wikipedia)

Enlace dinámico (enlace tardío):

"Mecanismo por el cual se escoge, en tiempo de ejecución, el método que responderá a un determinado mensaje. Es útil cuando este no puede ser determinado de forma estática, es decir, en tiempo de compilación." (Wikipedia)

2. Enlace tardío

El enlace entre la llamada a un método y el método en sí se produce en tiempo de ejecución.

```
public static void main (String[] args)
{
    Figura f;
    Random rnd = new Random( System.currentTimeMillis() );

    if ( rnd.nextInt( 10 ) > 5 ) {
        f = new Rectangulo( 5, 6 );
    } else {
        f = new Circulo( 3 );
    }

    System.out.println( "Superficie: " + f.calculaArea() );
}
```

¿*calcularArea()* en *Rectangulo* o *Circulo*?

1: http://ideone.com/Fan73X

2. Enlace tardío

En JAVA es posible indicar que un método no utilice enlace dinámico (utilizará obligatoriamente enlace en tiempo de compilación o estático) → *final*

```
class Circulo extends Figura {
    private double radio;

    public Circulo(double r)
    {
        radio = r;
    }

    public final double getRadio()
    {
        return radio;
    }
```

1: http://ideone.com/Fan73X

```
class Rectangulo extends Figura {
    private double lado1;
    private double lado2;

    public Rectangulo(double l1, double l2)
    {
        lado1 = l1;
        lado2 = l2;
    }

    public final double getLado1()
    {
        return lado1;
    }

    public final double getLado2()
    {
        return lado2;
    }
```

3. Polimorfismo

2: http://ideone.com/A2v0iL

Permite tratar con objetos sin conocer exactamente a qué clases pertenecen

Por ejemplo: vector de objetos de una SuperClase (referencias) apuntando a objetos de SubClases, que se recorre invocando a un método sobrecargado

```
public static void main (String[] args) throws java.lang.Exception
{
    Figura[] figuras = new Figura[ 2 ];

    figuras[ 0 ] = new Rectangulo( 5, 6 );
    figuras[ 1 ] = new Circulo( 1 );
    visualizaArea( figuras );
}

public static void visualizaArea(Figura[] figuras)
{
    for (int i = 0; i < figuras.length; ++i) {
        Figura f = figuras[ i ];
        System.out.format( "%s, superficie: %.2f\n", f, f.calculaArea() );
    }
}
```