


Métodos Avanzados de Programación Científica y Computación

M^a Luisa Díez Platas

Tema 1. Introducción a la programación orientada a objetos

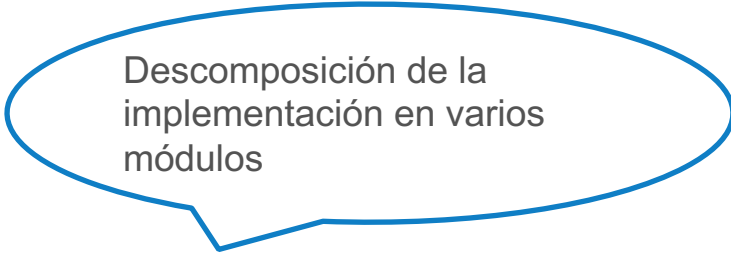
¿Cómo estudiar este tema?

IDEAS CLAVE	LO + RECOMENDADO	+ INFORMACIÓN	TEST
<p>¿Cómo estudiar este tema?</p> <p>Introducción a la programación orientada a objetos</p> <p>Diseño de clases</p> <p>Introducción a UML para el modelado de los problemas</p> <p>Introducción a la eficiencia y complejidad de un algoritmo</p>	<p>No dejes de leer...</p> <p>Conceptos de programación orientada a objetos</p> <p>No dejes de ver...</p> <p>TV Ventajas e inconvenientes de la POO</p> <p>TV Introducción a la POO</p> <p>TV Clases, objetos y métodos en Java</p> <p>TV Diagrama de clases UML</p>	<p>A fondo</p> <p>Conceptos básicos de programación orientada a objetos</p> <p>Ejemplos de diseño de clases y objetos</p> <p>Diseñar y programar, todo es empezar</p> <p>Aproximación a la programación orientada a objetos moderna</p> <p>Aproximación al pensamiento orientado a objetos</p> <p>Lenguajes de programación orientada a objetos</p> <p>Recursos externos</p> <p>Eclipse</p>	

- ¿Qué es una clase y como se diseña?
- Diferencia entre clase y objeto.
- Modelado gráfico con UML(*Unified Modelling Language*).
- Eficiencia y complejidad.

Complejidad del software

- La complejidad del dominio del problema
- La gestión del proceso de desarrollo



Descomposición de la
implementación en varios
módulos

¿Los datos?

Orientación a objetos

Conjunto de disciplinas (ingeniería) que desarrollan y modelan software que facilitando la construcción de sistemas complejos a partir de **componentes**. Estos componentes se denominan clases y son en esencia **tipos abstractos de datos** (TAD) con características propias de la OO.

TAD

1. Definición de un tipo visible y utilizable en el exterior del módulo .
2. Ocultar la estructura interna del tipo que no necesite manejarse directamente.
3. Proporcionar todas las primitivas necesarias para manejar las variables que se definan del tipo.

Orientación a objetos

- Nuevo modo de pensar y diseñar aplicaciones → nuevo paradigma
- Lenguajes que soporten el paradigma
- Diseño modular
- Reutilización

Lenguajes de POO

1. ADA
2. C++
3. Java
4. Smalltalk
5. Eiffel
6. Ruby
7. Python → multiparadigma
8. C#
9. Delphi
10. PHP....

Lenguajes de modelado
OO: UML

Modelado orientado a objetos

abstracción

Separar las características esenciales de un elemento o entidad de las no esenciales → descripción de una entidad del mundo real

encapsulamiento

Asegura que el contenido de la información de un objeto está oculta al mundo exterior.

modularidad

Subdividir una aplicación en partes más pequeñas (llamadas *módulos*), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

Débil acoplamiento
Fuerte cohesión

“dividir un programa en módulos que se pueden compilar por separado, pero que tienen conexiones con otros módulos” (Liskov).

jerarquía

La jerarquía es una propiedad que permite una ordenación de las abstracciones

generalización-especialización(herencia)
agregación

Modelado orientado a objetos

polimorfismo

Propiedad, que indica la posibilidad de que una entidad tome muchas formas. Un método distintos comportamientos

ligadura dinámica.

otras propiedades

Concurrencia, Genericidad, Persistencia y Manejo de Excepciones.

Programación orientado a objetos (POO). Conceptos fundamentales

“La POO es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa un ejemplar de una clase y cuyas clases son miembros de una jerarquía de clases unidas mediante relaciones de herencia.”

(Booch)

- Un programa OO consiste en una serie de mensajes...
 1. Mensaje de creación de objetos
 2. Intercambio de mensajes entre objetos
 3. Eliminación de objetos

POO. Conceptos fundamentales

Objeto. Entidad que contiene los atributos que describen el estado de un objeto del mundo real y las acciones que se asocian con el objeto del mundo real.

OBJETO = DATOS + OPERACIONES

Clase. Descripción de un conjunto de objetos. Consta de métodos y datos(**atributos**) que resumen las características comunes de un conjunto de objetos. Muestra el comportamiento general de un grupo de objetos.

Visibilidad

Métodos. Comportamiento que soporta el objeto

Getters y setters

Constructores

Otros

Nombres → clases

Adjetivos y complementos del verbo → atributos

Verbos → métodos

POO en Java.

Clase.

private: únicamente se puede ver este atributo dentro de la clase.

public: Se puede acceder a él desde cualquier clase del programa.

protected: Accesible desde la propia clase y sus subclases

```
class Miclase{
    //definicion de atributos
    [visibilidad] [tipo] nombre;
    //definicion de métodos
    Miclase(parametros){//implementacion}
    [visibilidad] [tipo retorno] nombre(parámetros){//implementacion}

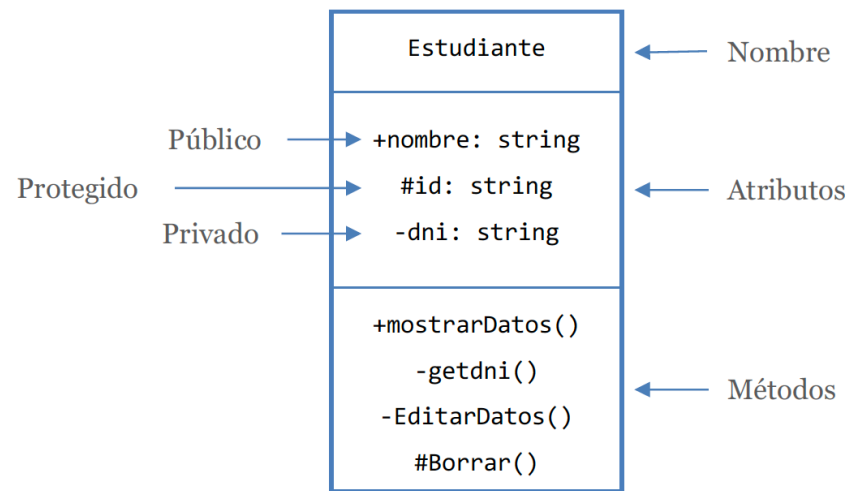
}
```

```
class Persona{
    //definicion de atributos
    private int edad;
    //definicion de métodos
    public Persona(){edad=0;}
    public Persona(int ed;){edad=ed;}
    public getEdad(){return edad;}
    public setEdad(int ed){edad=ed;}

}
```

Modelado con UML

Lenguaje para la especificación, visualización y construcción de artefactos de los sistemas de software.



Eficiencia y complejidad de un algoritmo

Eficiencia. Medida del uso de los recursos computacionales requeridos por la ejecución en función del tamaño de las entradas

Complejidad. Medida del número de operaciones elementales en la peor de las ejecuciones en tamaño.

```
package Metodosdebusqueda;
public class Burbujas {
    public void OrdenarBurbujas(int[] arreglo)
    {
        int aux;
        boolean cambios=false;
        while(true)
        {
            cambios=false;
            for(int k=1;k<arreglo.length;k++){
                if(arreglo[k]<arreglo[k-1]){
                    auxiliar= arreglo[k];
                    arreglo[k]=arreglo[k-1];
                    arreglo[k-1]=auxiliar;
                    cambios=true;
                }
            }
            if (cambios==false)
                break;
        }
    }
}
```

Método burbuja (BubbleSort)

Orden de complejidad

Eficiencia y complejidad de un algoritmo

```
package Metodosdebusqueda;
public class Burbujas {
    public void OrdenarBurbujas(int[] arreglo)
    {
        int aux;
        boolean cambios=false;
        while(true)
        {
            cambios=false;
            for(int k=1;k<arreglo.length;k++){
                if(arreglo[k]<arreglo[k-1]){
                    auxiliar= arreglo[k];
                    arreglo[k]=arreglo[k-1];
                    arreglo[k-1]=auxiliar;
                    cambios=true;
                }
            }
            if (cambios==false)
                break;
        }
    }
}
```

Método burbuja (BubbleSort)

$O(n^2)$

```
package Metodosdebusqueda;
public class QuickSort {
    public void OrdenarQuickSort(int[] arreglo)
    {
        arreglo = quicksort1(arreglo);
    }
    public int [] quicksort1(int numeros[])
    {
        return quicksort2(numeros,0,numeros.length-1);
    }
    public int[] quicksort2(int numeros[], int izq, int der)
    {
        if(izq>=der)
            return numeros;
        int i=izq, d=der;
        if(izq!=der)
        {
            int pivote;
            int aux;
            pivote = izq;
            while(izq!=der){
                while(numeros[der]>=numeros[pivote] && izq<der)
                    der--;
                while(numeros[izq]<numeros[pivote] && izq<der)
                    izq++;
                if(der!=izq)
                {
                    aux = numeros[der];
                    numeros[der]=numeros[izq];
                    numeros[izq]=aux;
                }
            }
            if(izq==der){
                quicksort2(numeros,i,izq-1);
                quicksort2(numeros,izq+1,d);
            }
        }
        else
            return numeros;
        return numeros;
    }
}
```

$O(n^2)$

$O(n \log n)$

promedio

UNIVERSIDAD
INTERNACIONAL
DE LA RIOJA

unir

www.unir.net