



UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Curso: Programación Orientada a Objetos

Sesión 2:

**Constructores Sobre escritura de
métodos**



UNIVERSIDAD TECNOLÓGICA DEL PERÚ
Facultad de Ingeniería

Curso: Programación Orientada a Objetos

Indicador de Logro

Al finalizar la unidad, el estudiante aplica los conceptos básicos de la programación orientada a objetos, el concepto de herencia, relaciones entre clases, en la solución de problemas usando Java.



UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Curso: Programación Orientada a Objetos

Importancia

La **utilidad del encapsulamiento** va por la **facilidad para manejar la complejidad**, ya que tendremos a las **Clases como cajas negras** donde **sólo se conoce el comportamiento pero no los detalles** internos, y esto es conveniente porque **únicamente deberíamos de conocer qué hace la Clase** pero **no** será necesario saber **cómo lo hace**.

Encapsulamiento en POO

¿Qué es el encapsulamiento?

Se denomina encapsulamiento al **ocultamiento** del estado, es decir, **de los datos miembro, de un objeto** de manera que **sólo se puede cambiar mediante las operaciones** definidas para ese objeto.

Cada objeto está aislado del exterior, **el aislamiento protege a los datos asociados a un objeto contra su modificación por quien no tenga derecho a acceder a ellos.**

De esta forma **el usuario de la clase puede obviar la implementación de los métodos y propiedades para concentrarse sólo en cómo usarlos**. Por otro lado se evita que el usuario pueda cambiar su estado de maneras imprevistas e incontroladas.

Ejemplo01: Encapsulamiento en POO

```
public class Alumno{  
    private String nombre;  
    private int edad; //edad que tiene un alumno en el momento que finaliza su carrera.  
    public void setNombre(String nombre){this.nombre=nombre;}  
    public String getNombre(){return nombre;}  
    public void setEdad(int edad){this.edad=edad;}  
    public int getEdad(){return edad;}  
  
    public Alumno(){  
        setEdad(0);  
        setNombre("anónimo");  
    }  
    public Alumno(String nombre, int edad){  
        setEdad(edad);  
        setNombre(nombre);  
    }  
    public void imprimir(){  
        System.out.print(getNombre()+" a sus "+getEdad()+" años ha concluido la carrera de Ing. Sistemas" )  
    }  
}
```

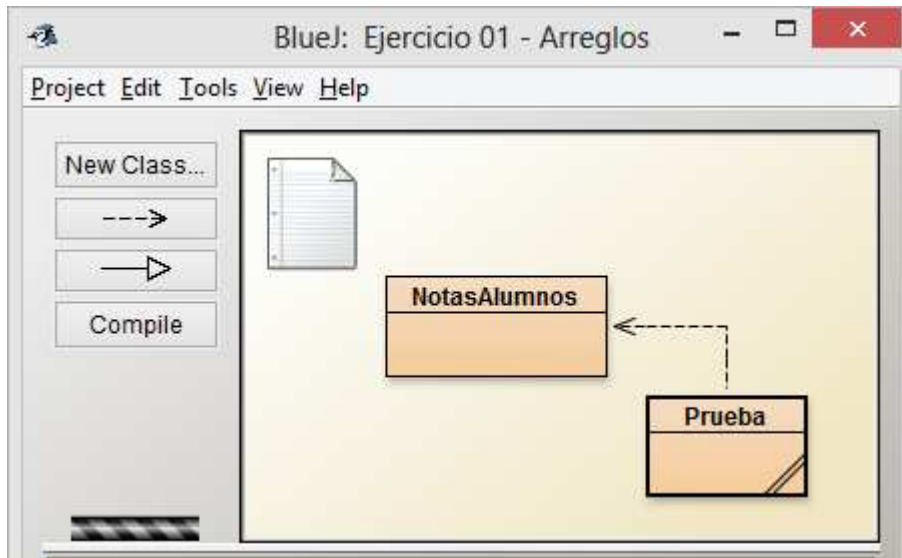
Ejemplo02: Encapsulamiento en POO

Escribir un programa que solicite la carga de un valor positivo “n” y nos muestre todos los números desde 1 hasta el valor ingresado (de uno en uno). *Ejemplo:* Si ingresamos 30 se debe mostrar en pantalla los números del 1 al 30. **Usar arreglos.**

Ejemplo03: Encapsulamiento en POO

Desarrollar un programa que lea por teclado las notas de “n” alumnos de una clase, calcule la nota media del grupo, muestre las notas superiores a la media. El valor de “n” se lee por teclado.

El Diagrama de Clases en BlueJ sería:



La salida del programa sería:

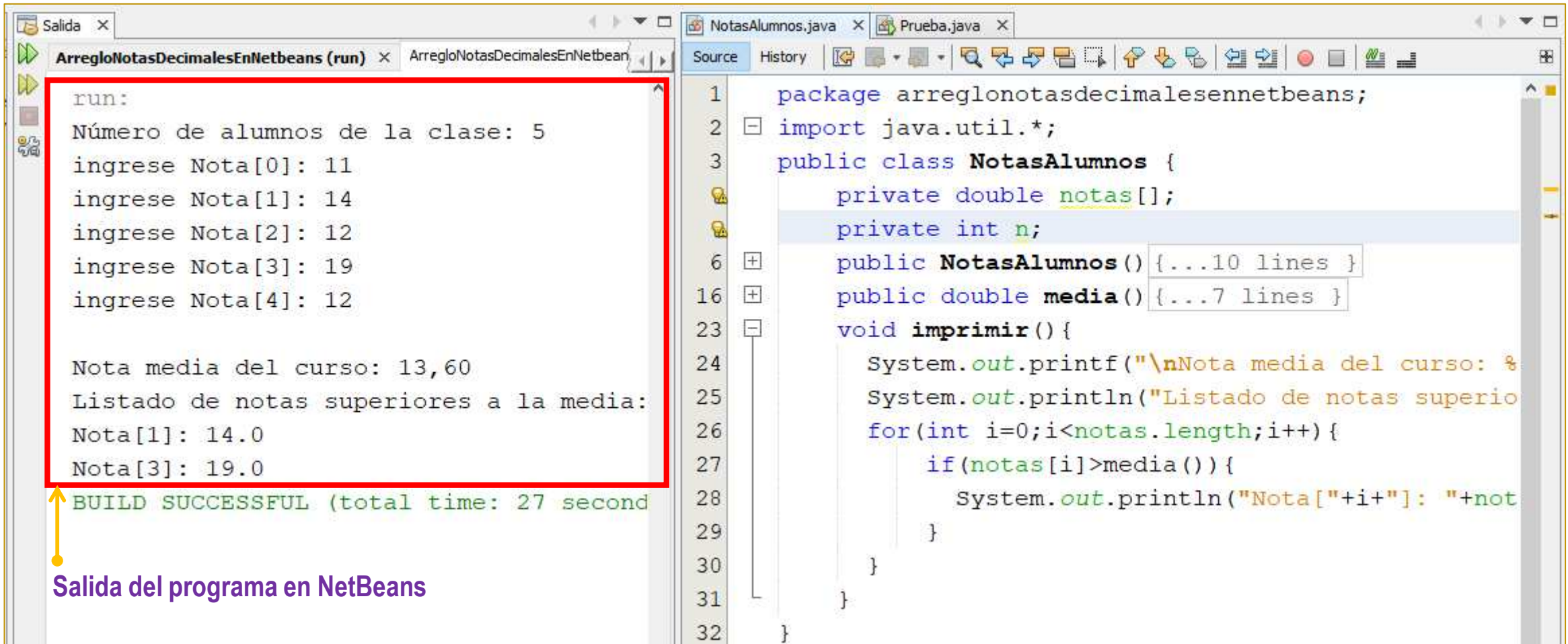
```
BlueJ: Ventana de Terminal - Arreglos de Notas Decimales en BlueJ
Opciones
Número de alumnos de la clase: 5
Ingrese Nota[0]: 11
Ingrese Nota[1]: 14
Ingrese Nota[2]: 12
Ingrese Nota[3]: 19
Ingrese Nota[4]: 12

Nota media del curso: 13.60
Listado de notas superiores a la media:
Nota[1]: 14.0
Nota[3]: 19.0

Can only enter input while your programming is running
```

Ejemplo03: Encapsulamiento en POO

Desarrollar un programa que lea por teclado las notas de “n” alumnos de una clase, calcule la nota media del grupo, muestre las notas superiores a la media. El valor de “n” se lee por teclado.



The screenshot displays the NetBeans IDE interface. On the left, the 'Salida' (Output) window shows the program's execution. On the right, the 'Source' window shows the Java code for 'NotasAlumnos.java'.

Salida (Output) Window:

```
run:
Número de alumnos de la clase: 5
ingrese Nota[0]: 11
ingrese Nota[1]: 14
ingrese Nota[2]: 12
ingrese Nota[3]: 19
ingrese Nota[4]: 12

Nota media del curso: 13,60
Listado de notas superiores a la media:
Nota[1]: 14.0
Nota[3]: 19.0
BUILD SUCCESSFUL (total time: 27 second)
```

Source Window (NotasAlumnos.java):

```
1 package arreglonotasdecimalesennetbeans;
2 import java.util.*;
3 public class NotasAlumnos {
4     private double notas[];
5     private int n;
6     public NotasAlumnos() {...10 lines }
16    public double media() {...7 lines }
23    void imprimir() {
24        System.out.printf("\nNota media del curso: %
25        System.out.println("Listado de notas superio
26        for(int i=0;i<notas.length;i++){
27            if(notas[i]>media()){
28                System.out.println("Nota["+i+"]: "+not
29            }
30        }
31    }
32 }
```

An arrow points from the text "Salida del programa en NetBeans" to the output window.

Sobrecarga de métodos en Java

Una clase **puede tener más de un constructor** siempre y cuando no coincidan los parámetros en orden de tipología. También se puede aplicar la sobrecarga a los métodos:

```
public class Alumno {  
    private String codigo;  
    private String nombre;  
    private double nota1;  
    private double nota2;  
  
    public Alumno(String codigo,String nombre,double nota1,double nota2){  
        this.codigo=codigo;  
        this.nombre=nombre;  
        this.nota1=nota1;  
        this.nota2=nota2;  
    }  
  
    public Alumno(String codigo,String nombre){  
        this.codigo=codigo;  
        this.nombre=nombre;  
        this.nota1=0.0;  
        this.nota2=0.0;  
    }  
  
    public Alumno(){  
    }  
}
```

Sobrecarga de métodos en Java

Una clase **puede tener más de un constructor** siempre y cuando no coincidan los parámetros en orden de tipología. **También se puede aplicar la sobrecarga a los métodos:**

```
package ejemplo_sobrecarga_clase_perro;
public class Perro{
    private String nombre;
    private int edad;
    public Perro(String nombre, int edad){
    public void setNombre(String nombre){
    public String getNombre(){
    public void setEdad(int edad){
    public int getEdad(){
    public void cambiar(String no
    public void cambiar(int edad)
    public void cambiar(String nombre, int edad){
    public void imprimir(){
}
```

CIERRE



¿Qué hemos aprendido hoy?