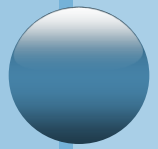
Curso de Java

Introducción al Paradigma Orientado a Objetos

Prof. Ing. Guido Acosta




Contenido de la clase

-Conceptos fundamentales

-Definición de clase

-Encapsulamiento

-Operadores . (punto) y new

Orientación a objetos

¿Cómo se diseña una casa?

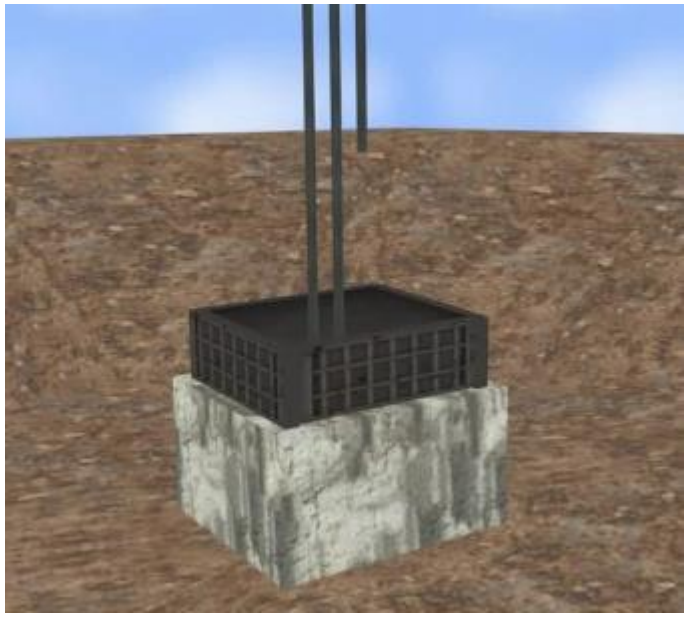
Orientación a objetos

¿Cómo se diseña una casa?

1. **Planos**

Orientación a objetos

¿Cómo se diseña una casa?
2. Cimiento



Orientación a objetos

¿Cómo se diseña una casa?
3. Pilares

Orientación a objetos

Reutilización de código

- **Operaciones sobre cadena**
- **Operaciones sobre fecha**
- **Conexiones a BD**

¿Nosotros escribimos código para realizar las tareas mencionadas arriba?


Programación Orientación a objetos

- Es un paradigma de programación que usa objetos y sus interacciones.
- Está basada en varias técnicas, incluyendo:
  - Herencia
  - Abstracción
  - Polimorfismo
  - Encapsulamiento


Conceptos básicos - POO

- Los objetos son entidades que combinan:
  - Estado (atributos)
  - Comportamiento (métodos)
  - Identidad


El estado del objeto

- **Está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).**


El comportamiento del objeto

- **Está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.**


POO vs Programación Estructurada

- La POO difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida.
- La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan.
- En la POO, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

Ejemplo de objetos

Características
- Nombre
- Especie
- Color
- Edad

Acciones
- Comer
- Dormir
- Correr

Características
- Marca
- Color
- Tipo
- Precio
- No. de Puertas
- Tipo Combustible
- Cilindros
- Transmisión

Acciones
- Encender
- Avanzar
- Retroceder
- Detener
- Apagar

Conceptos fundamentales

Transformar el mundo real en código

¿Qué es una clase?

Al igual que un diseñador crea prototipos de dispositivos que podrían utilizarse en repetidas ocasiones para construir los dispositivos reales, una clase es un prototipo de software que puede utilizarse para instanciar (es decir crear) muchos objetos iguales.

Conceptos fundamentales

Sintaxis para generar una clase

```
<modificador>class<nombre_clase>
{
        <declaración_atributo>
        <declaración_constructor>
        <declaración_método>
}
```

```
public class Vehiculo
{
        private double cargaMax;
        public Vehiculo()
        {
                cargaMax = 0;
        }
        public void SetCargaMax (int valor)
        {
                cargaMax= valor;
        }
}
```


Conceptos fundamentales

¿Qué contiene una clase?

Los miembros de una clase son un conjunto de elementos que definen a los objetos (atributos ó propiedades), así como los comportamientos o funciones (métodos) que maneja el objeto.

Clase

- Persona

Objetos

- Patricia
- Juan
- Pedro
- Martina

¿Qué atributos podría tener la clase Persona?
¿Y qué métodos?

Conceptos fundamentales

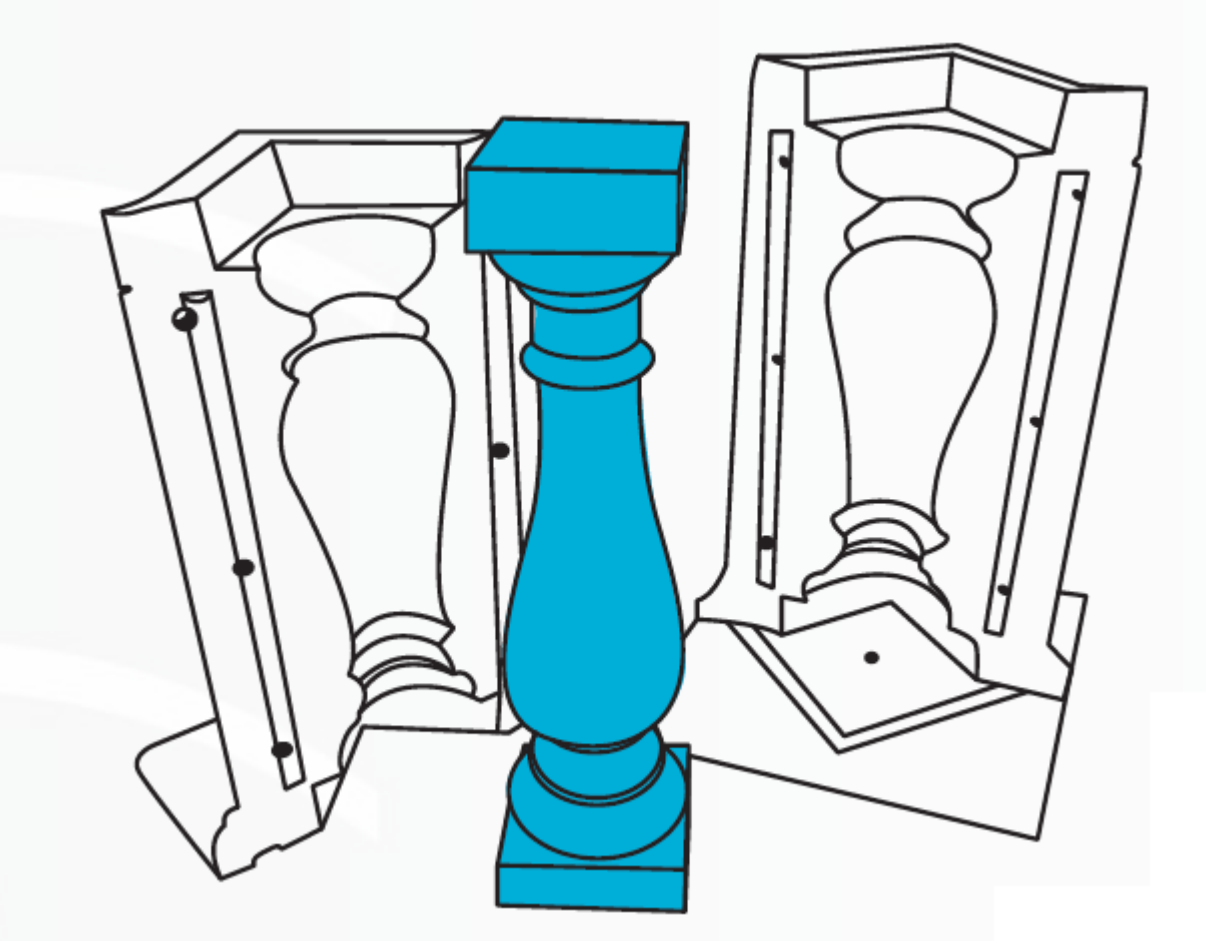
Definición de atributos

```
<modificador><tipo><nombre> [ = <valor_inicial>];
```

```
public class  Persona{
        private String nombre;
        private String apellido;
        private int edad;
        private int peso;
        private int estatura;

}
```

Conceptos fundamentales

Analogía de una clase y un objeto



Conceptos fundamentales

Definición de métodos

<modificador><tipo_retorno><nombre> (<argumentos>) { <sentencia> }

```
public class  Persona{
        private String nombre;
        private String apellido;
        private int edad;
        private int peso;
        private int estatura;

        public double indiceMasaCorporal(){
                // IMC = peso[kg]/estatura[m2]
                return  peso / (altura * altura);
        }
}
```


Conceptos fundamentales

¿Qué es el constructor de una clase?

- El constructor es un miembro que sirve para inicializar la instancia de una clase.
- El constructor es invocado cuando se usa el operador new.
- El constructor es parecido a los métodos, pero no tiene un tipo de retorno y su nombre es el mismo que el de la clase y también puede o no recibir parámetro como los métodos.

Conceptos fundamentales

¿Qué es el constructor de una clase?

```
public class  Persona{
        private String nombre;
        private String apellido;
        private int edad;
        private int peso;
        private int estatura;

        public  Persona(String nombre, String apellido){
            this.nombre = nombre;
            this.apellido = apellido;
        }
}
```

El operador new

El operador new nos permite crear objetos en Java.

Declaración
Figura unaFigura;

Creación
unaFigura = new Figura();

Obs.: unaFigura es una instancia de la clase Figura.

El operador . (punto)

El operador . nos permite acceder a los distintos miembros de una clase.

objeto.miembro

El operador . (punto)

El operador . nos permite acceder a los distintos miembros de una clase.

objeto.miembro

Acceso a los atributos

objeto.atributo

Acceso a los métodos

objeto.método(lista de parámetros)


El operador . (punto)

El operador . nos permite acceder a los distintos miembros de una clase.

objeto.miembro

Acceso a los atributos

objeto.atributo

Acceso a los métodos

objeto.método(lista de parámetros)

Ejemplo

unaFigura = new Figura();

unaFigura.color
unaFigura.cambiarColor(“verde”)


Conceptos fundamentales

Ejercicio

Escribir un programa que lea los datos de 3 personas y muestra el índice de masa corporal de cada persona. De cada persona necesitamos almacenar su nombre y su edad. Al momento de la impresión se deberán imprimir el nombre de la persona, su edad y su índice de masa corporal.

Orientación a objetos

-Todo es un objeto

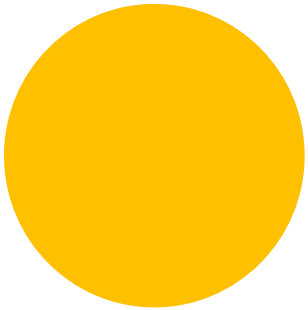
-Las clases definen el comportamiento de un conjunto de objetos.

-Un objeto es un caso particular de una clase. Es decir una instancia.

-Cada objeto tiene un estado. (Una memoria que contiene los datos)

-Los objetos se comunican entre sí mediante mensajes

Orientación a objetos



Orientación a objetos

-Todo es un objeto

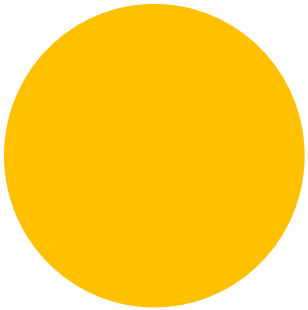
-Las clases definen el comportamiento de un conjunto de objetos.

-Un objeto es un caso particular de una clase. Es decir una instancia.

-Cada objeto tiene un estado. (Una memoria que contiene los datos)

-Los objetos se comunican entre sí mediante mensajes.

Orientación a objetos

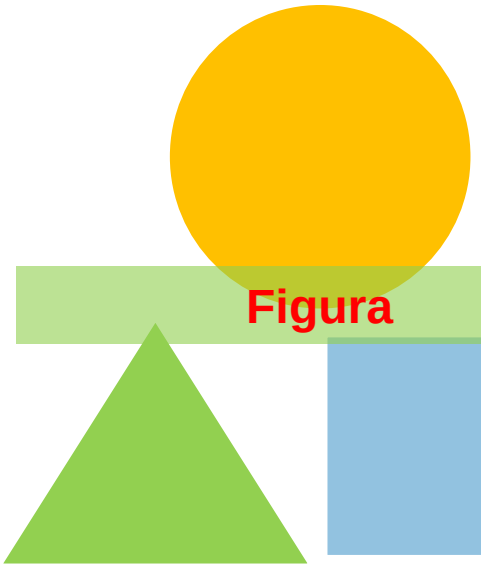


Orientación a objetos

Electrodoméstico

Vehículo

Animal



Figura

Orientación a objetos

-Todo es un objeto

-Las clases definen el comportamiento de un conjunto de objetos.

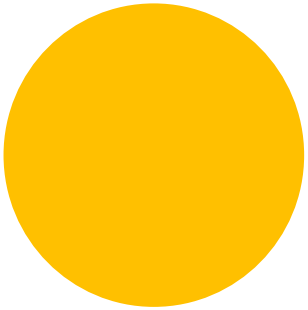
-Un objeto es un caso particular de una clase. Es decir una instancia.

-Cada objeto tiene un estado. (Una memoria que contiene los datos)

-Los objetos se comunican entre sí mediante mensajes.

Orientación a objetos

Figura



círculo: Objeto de la clase Figura

triángulo: Objeto de la clase Figura

cuadrado: Objeto de la clase Figura


Orientación a objetos

-Todo es un objeto

-Las clases definen el comportamiento de un conjunto de objetos.

-Un objeto es un caso particular de una clase. Es decir una instancia.

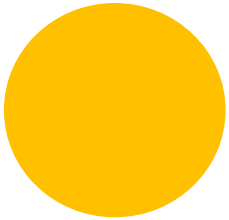
-Cada objeto tiene un estado. (Una memoria que contiene los datos)

-Los objetos se comunican entre sí mediante mensajes.

Orientación a objetos

Figura

Color



círculo: color naranjado

círculo: color celeste

círculo: color gris

Orientación a objetos

-Todo es un objeto

-Las clases definen el comportamiento de un conjunto de objetos.

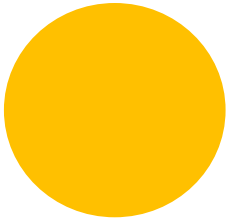
-Un objeto es un caso particular de una clase. Es decir una instancia.

-Cada objeto tiene un estado. (Una memoria que contiene los datos)

-Los objetos se comunican entre sí mediante mensajes.

Orientación a objetos

Figura
Color
cambiarColor



círculo: color naranjado

círculo: color celeste

círculo: color gris

Orientación a objetos

Los objetos encapsulan:

-Datos
-Comportamiento

Figura
Color Tamaño Posición
cambiarColor cambiarTamaño cambiarPosicion

Orientación a objetos

Los objetos encapsulan:

-Datos
-Comportamiento

	Figura
Atributos	Color Tamaño Posición
Métodos	cambiarColor cambiarTamaño cambiarPosicion

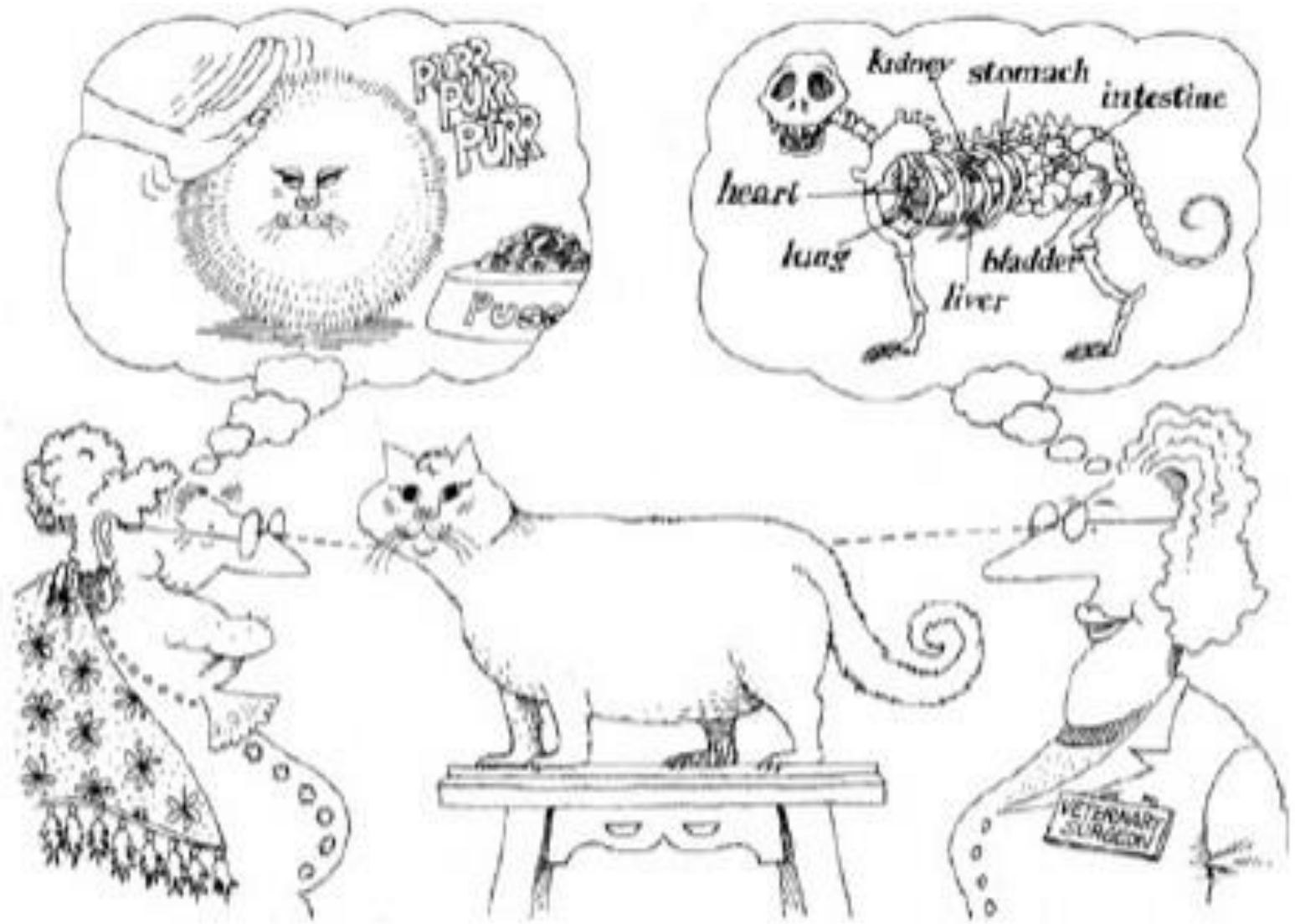
Abstracción

¿Qué características del objeto tomamos al momento del diseño?

Abstracción

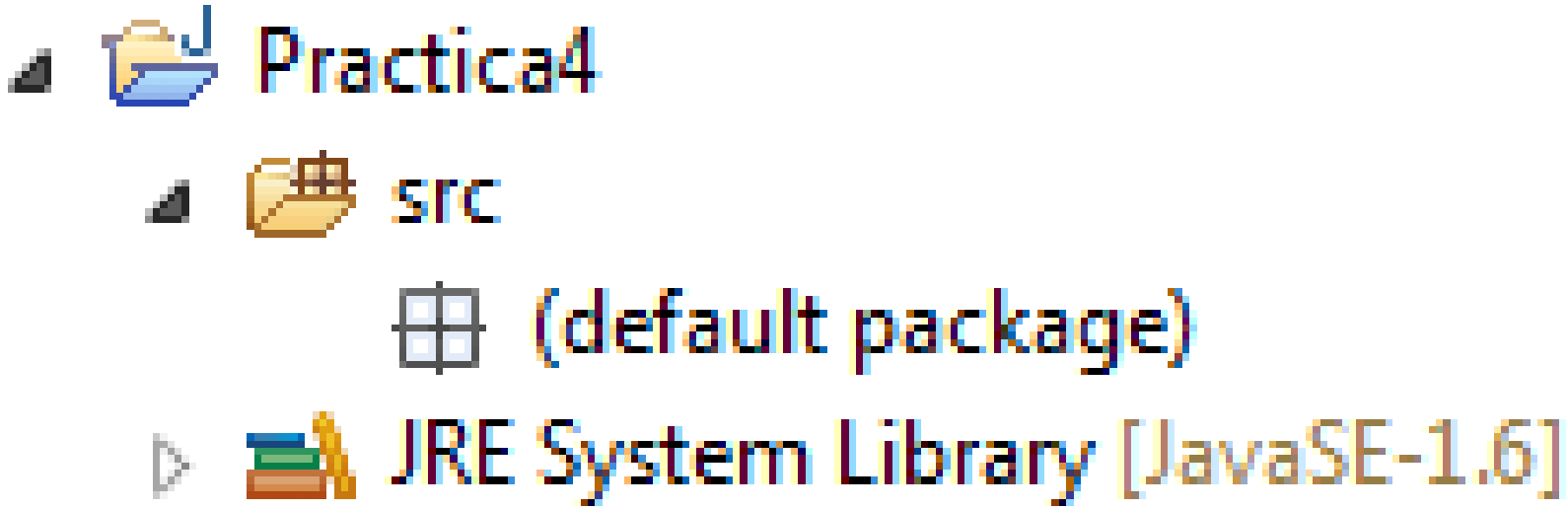
"Separar por medio de una operación intelectual las cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción"

Abstracción



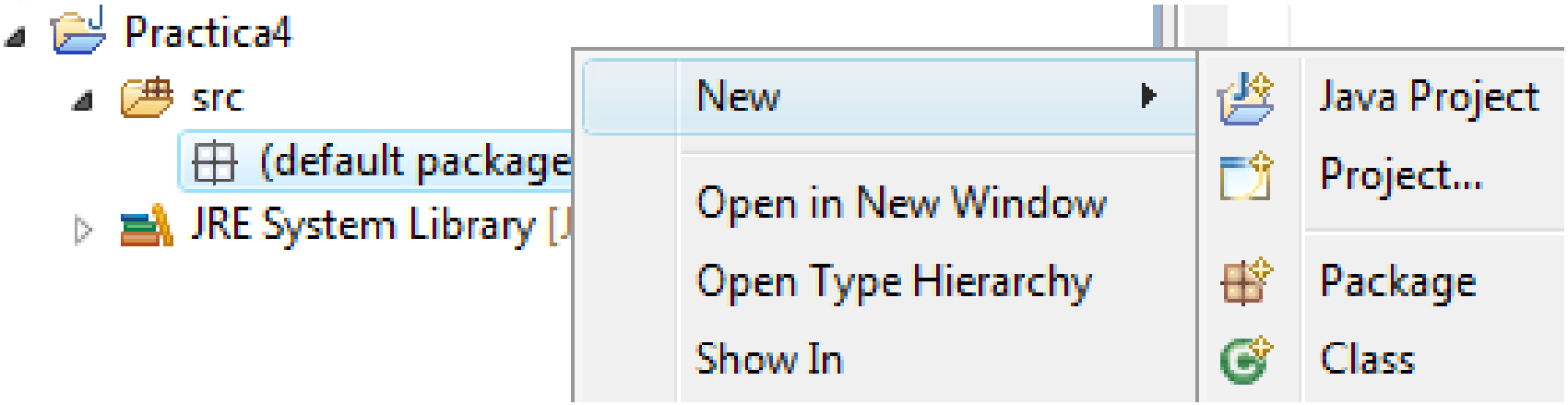
Definición de una clase

Proyecto



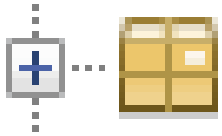
Definición de una clase

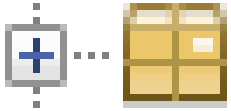
Paquete

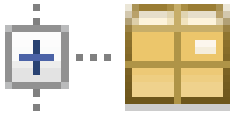


Definición de una clase

Paquete

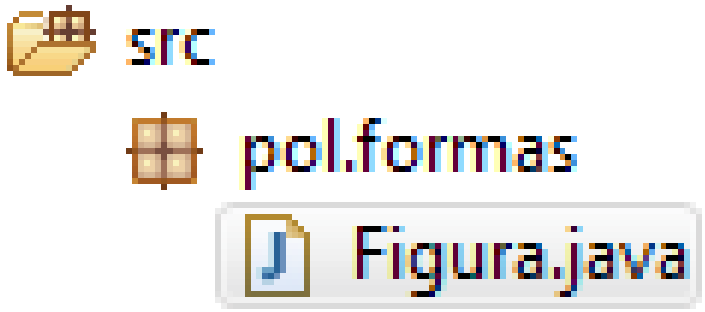






Definición de una clase

Paquete



Definición de una clase

Clase

```
package pol.formas;

public class Figura {

}
```

Definición de una clase

Atributos

```
package pol.formas;

public class Figura {
      String color;
      int tamaño;

}
```

Definición de una clase

Métodos

```
package pol.formas;

public class Figura {
	String color;
	int tamaño;

	void cambiarColor(String nuevoColor){
		this.color = nuevoColor;
	}

}
```

Programación Orientada a Objetos

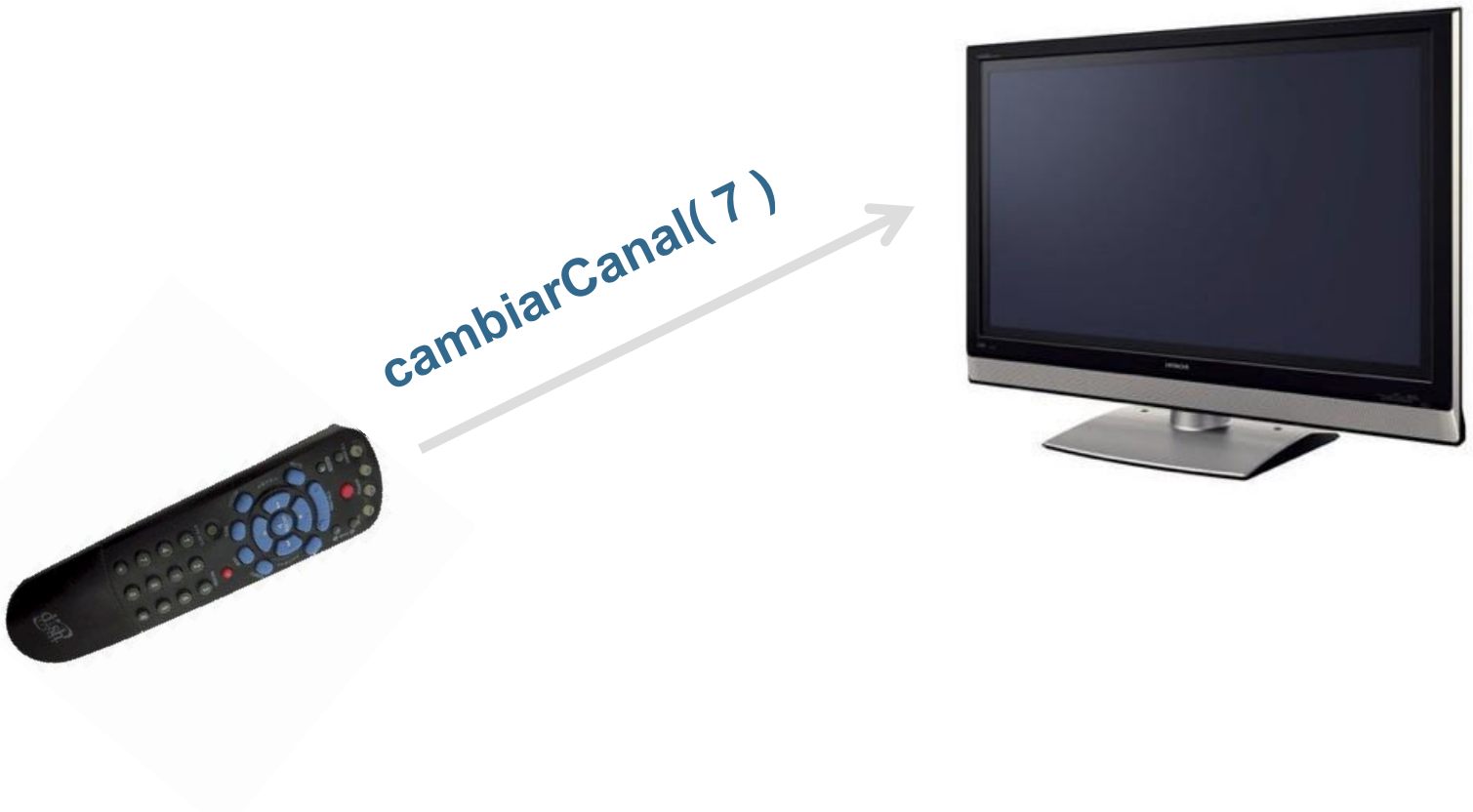
La programación orientada a objeto permite crear programas separando las partes por responsabilidades y haciendo que esas partes se comuniquen entre sí mediante mensajes.

Programación Orientada a Objetos

La programación orientada a objeto permite crear programas separando las partes por responsabilidades y haciendo que esas partes se comuniquen entre sí mediante mensajes.

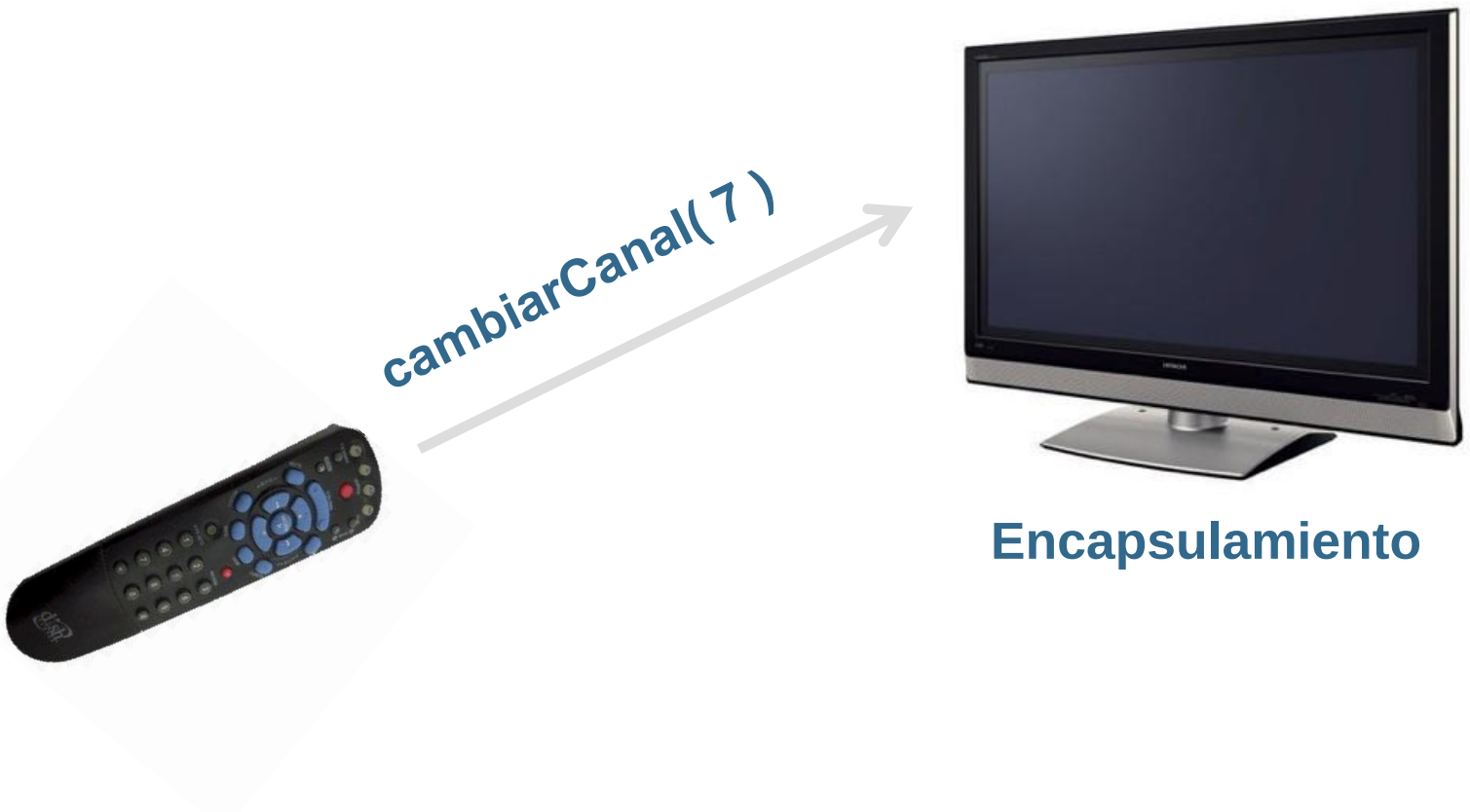
Programación Orientada a Objetos

La programación orientada a objeto permite crear programas separando las partes por responsabilidades y haciendo que esas partes se comuniquen entre sí mediante mensajes.



Programación Orientada a Objetos

La programación orientada a objeto permite crear programas separando las partes por responsabilidades y haciendo que esas partes se comuniquen entre sí mediante mensajes.



Encapsulamiento

Programación Orientada a Objetos

La programación orientada a objeto permite crear programas separando las partes por responsabilidades y haciendo que esas partes se comuniquen entre sí mediante mensajes.

cambiarCanal(7)

cambiarCanal(7)

Encapsulamiento

Encapsulamiento

La programación orientada a objeto permite crear programas separando las partes por responsabilidades y haciendo que esas partes se comuniquen entre sí mediante mensajes.

cambiarVarXUW(7)

Cambiar???

Encapsulamiento

Encapsulamiento

Métodos públicos

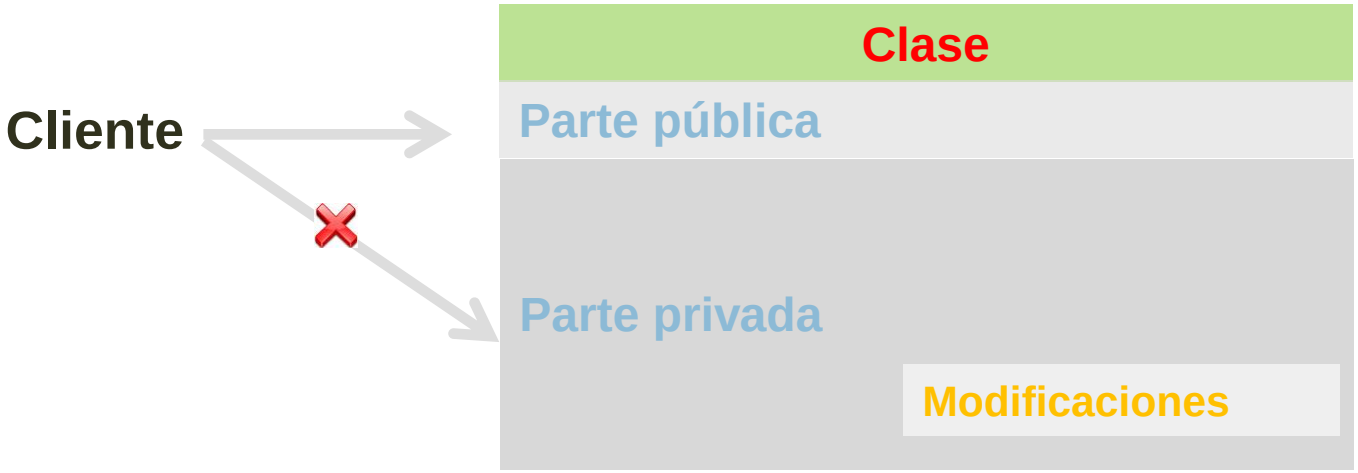
-CambiarCanal
-ModificarVolumen
-Encendido/apagado

Métodos privados

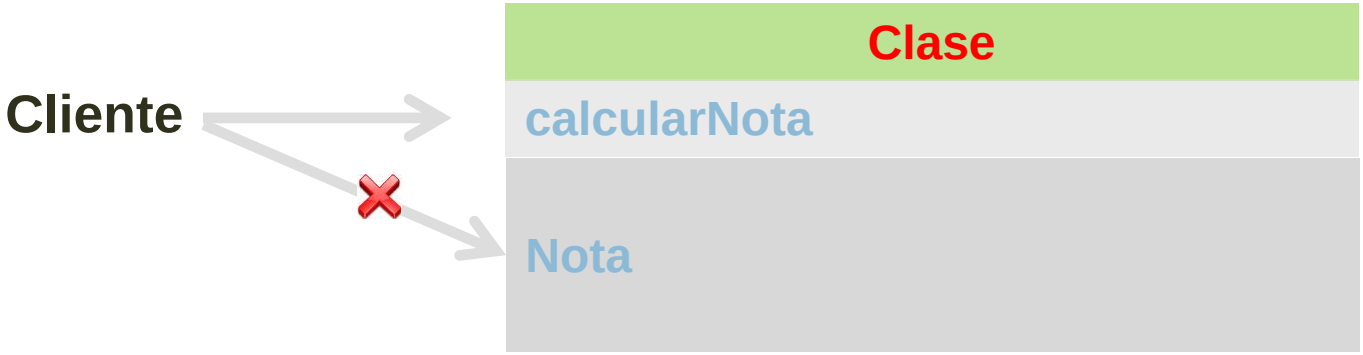
-SintonizarFrecuencia
-ModificarEstadoCI35
-CortarCorriente

Encapsulamiento - Ventajas

Capacidad de crear módulo



Protección de la información



Encapsulamiento en Java

Modificadores de acceso

-public

-private

Encapsulamiento en Java

Modificadores de acceso

```
package pol.formas;

public class Figura {
    public String color;
    public int tamaño;
    private String colorLinea;

    public void cambiarColor(String nuevoColor){
        this.color = nuevoColor;
    }

}
```

Encapsulamiento en Java

Modificadores de acceso

```
public class Figura {
    private String color;
    private int tamaño;
    private String colorLinea;
```

¿Cómo accedemos a los atributos de la clase?

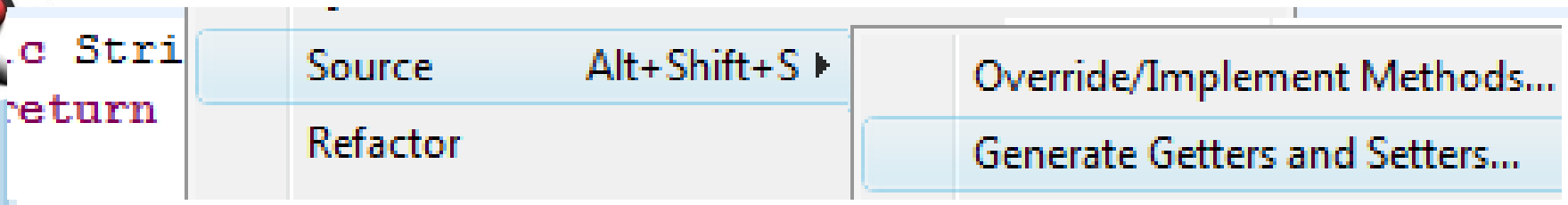
Encapsulamiento en Java

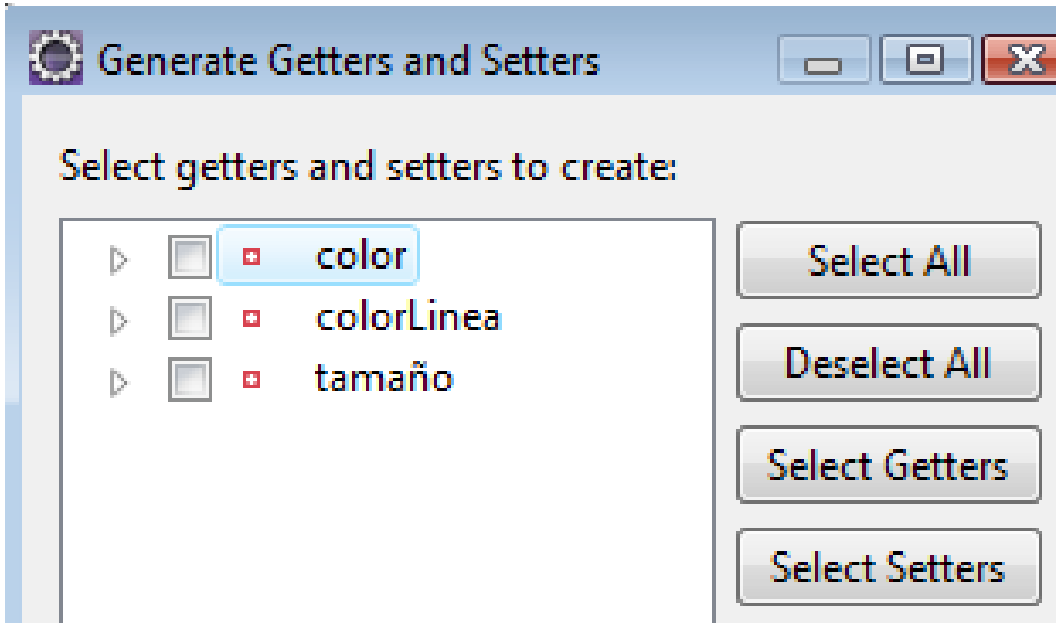
Getters y Setters

```
public class Figura {
    private String color;
    private int tamaño;
    private String colorLinea;

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
```

Getters y Setters en Eclipse

Click secundario sobre el código





Constructores

Los constructores son métodos que permiten inicializar un objeto al crearlo. El constructor es un método especial de la clase.

```
public class Figura {
    public String color;
    public int tamaño;
    private String colorLinea;

    public Figura(String color, int tamaño){
        this.color = color;
        this.tamaño = tamaño;
    }
```

El constructor debe coincidir con el nombre de la clase.

Constructor por defecto

Java crea automáticamente un constructor sin parámetros para cualquier método que se defina.

```
public class Figura {
    public String color;
    public int tamaño;
    private String colorLinea;

    public Figura(){

    }
```

Inicialización por defecto

Cuando no se especifican los valores para los atributos la inicialización realiza lo siguiente:

-Las variables de instancia de tipo numérico se inician en 0.

-Las variables de instancia de tipo char se inicializan en '\0'

-Las variables de instancia de tipo boolean se inicializan en false

-Las variables de instancia de cualquier tipo no primitivo se inicializan a null