

CAPÍTULO 1

INTRODUCCIÓN AL PARADIGMA ORIENTADO A OBJETOS



Preguntas detonadoras



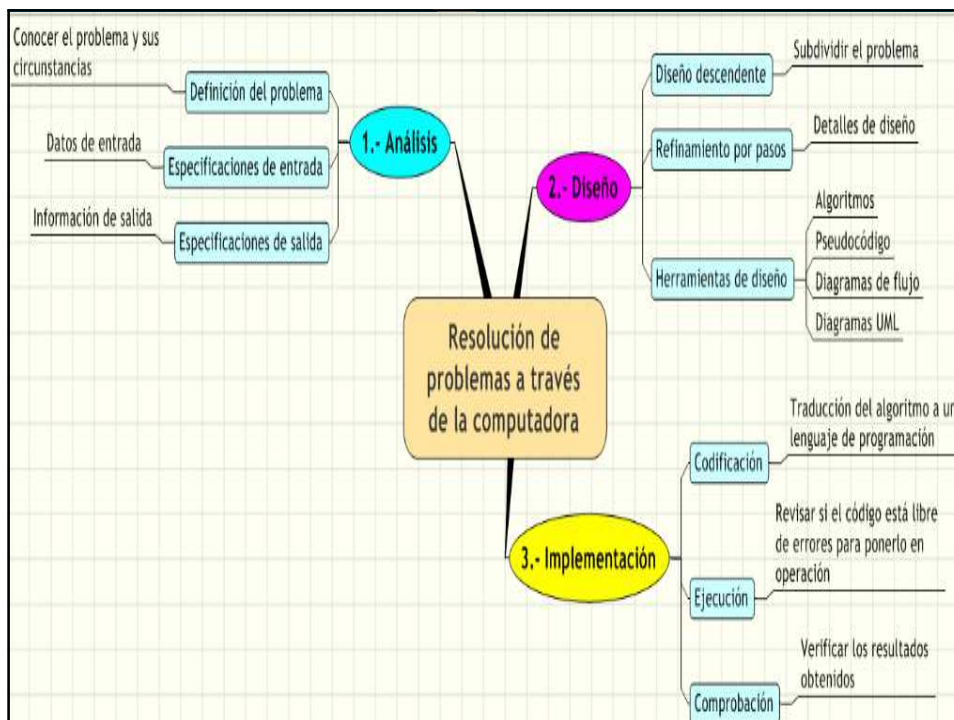
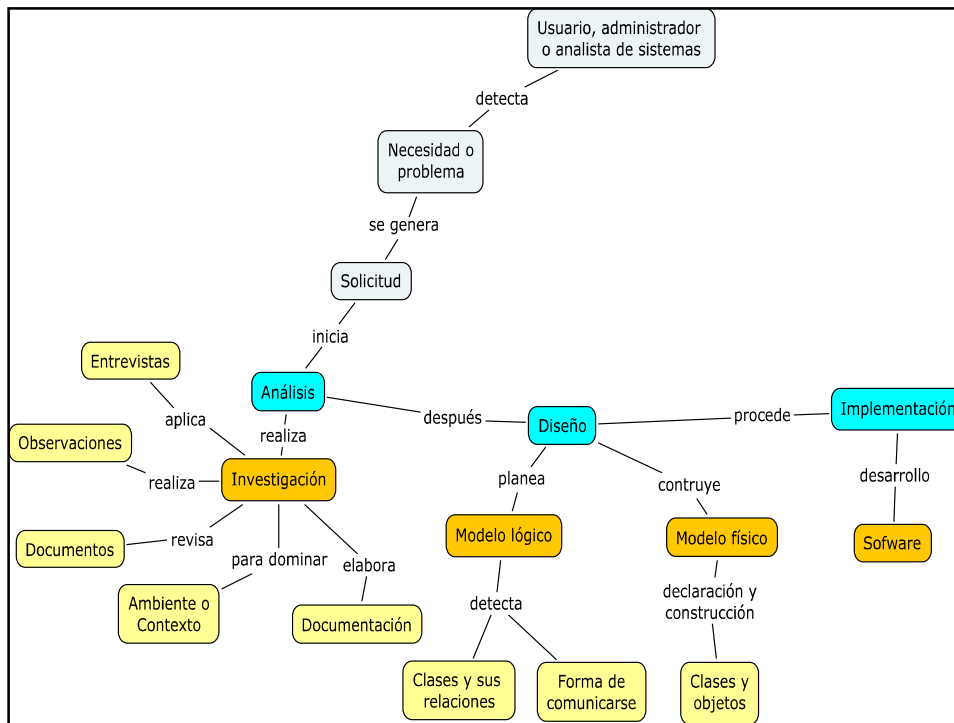
- ❑ ¿Qué es una clase?
- ❑ ¿Qué es un objeto?
- ❑ ¿Representa lo mismo una clase que un objeto?
- ❑ Diversos objetos creados a partir de la misma clase, ¿son iguales?
- ❑ ¿Qué significa el término instancia?
- ❑ ¿Cómo se logra que un objeto almacene datos y también realice acciones?
- ❑ ¿Qué es un atributo, propiedad y método?
- ❑ ¿Cuál es la diferencia entre atributo y propiedad?
- ❑ ¿Cómo se diseña el modelo de una aplicación orientada a objetos?

3

Resolución de problemas a través de la computadora

1. **Análisis:** ¿Qué ...?
 - ¿Qué problema debe resolverse?
 - ¿Qué datos se requieren?
 - ¿Qué resultados debe arrojar el Sistema?
2. **Diseño:** ¿Cómo ...?
 - ¿Cómo atacar el problema?
 - ¿Cómo plantear el modelo de solución?
 - ¿Cómo aplicar el modelo de solución?
3. **Implementación:** ¿Con qué ...?
 - ¿Con qué lenguaje se desarrolla el modelo?
 - ¿Con qué plataforma de desarrollo?
 - ¿Con qué recursos de hardware y software?

4



Programación Orientada a Objetos

- POO es un conjunto de técnicas que pueden utilizarse para desarrollar programas eficientemente.
- Los objetos son los elementos principales de construcción.
- La Orientación a Objetos (OO) es el estilo dominante de programación, descripción y modelado de hoy en día.

7

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 una instancia de alguna clase y cuyas clases son todas miembros de una jerarquía de clases unidas mediante relaciones”

Grady Booch

8

El modelo de Objetos


- Objetos en el mundo real
 - Atributos
 - Propiedades
 - Métodos
- Abstracción
- Clases y Objetos
- Encapsulamiento
- Mensajes
- Constructores
- Destructor
- Herencia
 - Simple
 - Múltiple



- Clases abstractas
- Clases parametrizadas
- Interfaces
- Sobrescritura
- Sobrecarga
- Polimorfismo

9

Objetos en el mundo real



Lavadora



Perro



Televisión



Persona



Factura

10



Podemos darnos cuenta que...

- Los objetos poseen **características** que los distinguen entre sí.
- Los objetos tienen **acciones** asociadas a ellos.

11

Ejemplo: PERRO



- **Características:**
 - Nombre: "FIDO"
 - Raza: "Chihuahua"
 - Color: "Café"
 -etc...
- **Acciones:**
 - Ladrar ["Guau Guau"]
 - Comer ["Chomp Chomp"]
 - Dormir ["Zzzzzzzzz"]
 - ...etc...

12

¿Cómo modelar un objeto real en un programa?

- Las “características” son **ATRIBUTOS** o datos.
- Las “acciones” son **MÉTODOS** u operaciones.



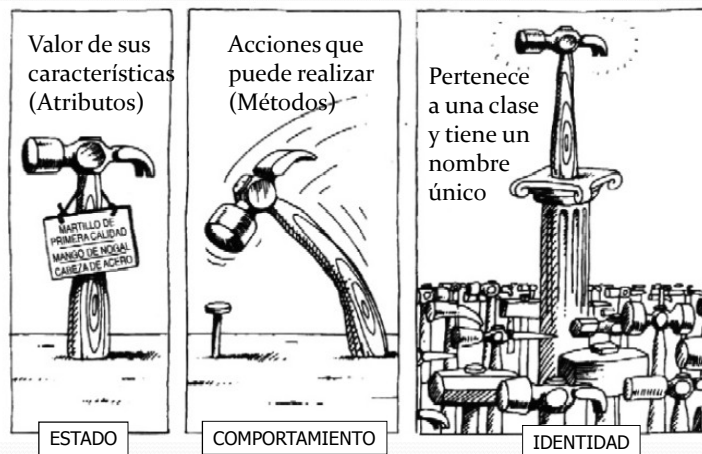
Objeto Perro “Real”

FIDO : Perro
Nombre: FIDO Raza: Chihuahua Color: Café
Ladrar() Comer() Dormir()

Abstracción de un objeto “Perro” en software

13

Todos los objetos tienen Estado, Comportamiento e Identidad



14

Abstracción

- Se refiere a “quitar” atributos, propiedades y métodos de un objeto y quedarse solo con aquellos que sean necesarios (relevantes para el problema a solucionar).



Objeto Perro “Real”:

Características o atributos:

(Nombre, Raza, Color, Edad, Tamaño, etc.)

Acciones o métodos:

(Ladear, Comer, Dormir, Jugar, Caminar, etc.)

FIDO : Perro

Nombre: FIDO
Raza: Chihuahua
Color: Café

Ladear()
Comer()
Dormir()

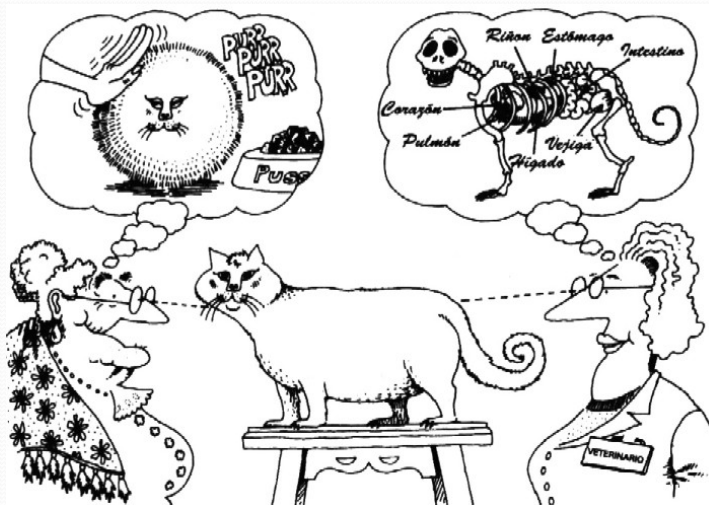
Nótese que en la “Abstracción” del perro quitamos varias características y acciones.

Abstracción de un “Perro”

15

Abstracción

La abstracción se centra en las características esenciales de algún objeto, en relación a la perspectiva del observador.



16

Abstracción

Las clases y objetos deben estar al nivel de abstracción adecuado: ni demasiado alto ni demasiado bajo.



17



Encapsulamiento

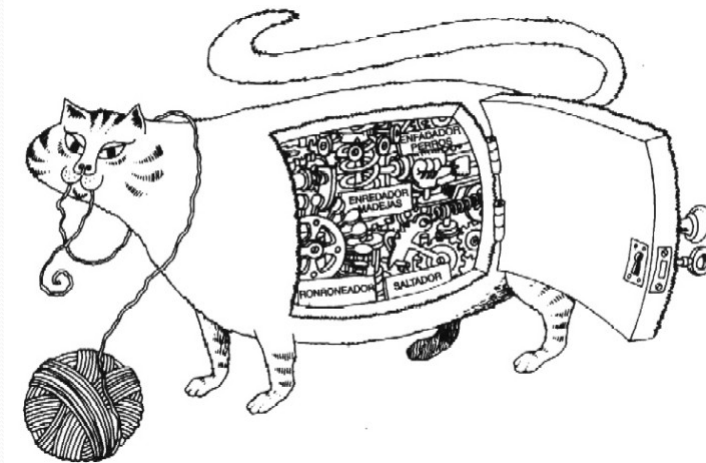


- Permite incluir en una sola entidad información y operaciones que controlan dicha información.
- Permite:
 - Componentes públicos [Accesibles, Visibles].
 - Componentes privados [No accesibles, Ocultos].
 - Restricción de accesos indebidos.

18

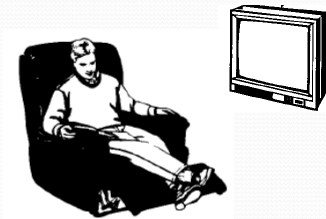
Encapsulamiento

El encapsulamiento oculta detalles de la implementación de un objeto.



19

Ejemplo: Encapsulamiento



La Televisión oculta algunos componentes y operaciones de la persona que la ve.

- Los objetos encapsulan lo que hacen. Ocultan la funcionalidad interna de sus operaciones, de otros objetos y del mundo exterior.

20

Ejemplo Encapsulamiento

Componentes privados - Ocultos
(NO Accesibles desde el exterior)
Circuitos, cables



Aunque TODOS los componentes de un objeto se comuniquen entre sí internamente, algunos componentes son visibles al exterior y otros permanecen ocultos por motivos de seguridad e integridad del objeto.

Componentes accesibles desde el exterior
(Interfaz público)
Botones para cambiar el canal, subir/bajar el volumen

21

Mensajes entre Objetos

Los objetos realizan acciones cuando reciben mensajes



Oprime el botón de encendido



Envía una orden de encendido



Mensaje recibido: Encender
Acción realizada: Se muestra imagen

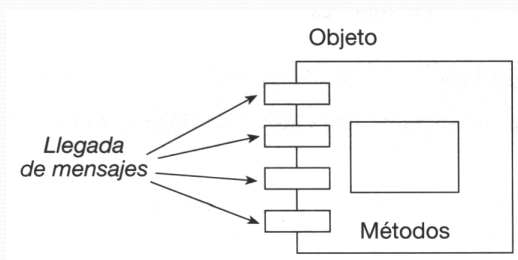
Mensaje recibido: Enciende la TV
Acción realizada: Envía orden de encendido a la TV

22

Mensajes: Comunicación entre objetos



- **Mensaje.-** Orden que se envía al objeto para indicarle realizar una acción.
- **Mensaje.-** Llamada a un método (o función) del objeto.



Al conjunto de mensajes a los cuales puede responder un objeto se llama “**Protocolo del Objeto**”

23

Clase



- Es una descripción de las características y acciones para un tipo de objetos.
- Una clase NO es un objeto. Es solo una plantilla, plano o definición para crear objetos.

24

Clase



- Contiene todas las características comunes de ese conjunto de objetos
- Clase = Modelo = Plantilla = Esquema = Descripción de la anatomía de los objetos.
- A partir de una clase se pueden crear muchos objetos independientes con las mismas características.

25

Objeto



- Unidad que combina datos y funciones.
 - Datos = Atributos = Características
 - Funciones = Métodos = Procedimientos = Acciones
- Un objeto es creado a partir de una clase.
- Los datos y funciones están Encapsulados.
- Posee un nombre único (identificador).
- Un objeto es del tipo de una clase
- “Un objeto es la instancia de una clase”
- Un objeto es un ejemplar específico creado con la estructura de una clase.

26

Instancia

- Es la creación o manifestación concreta de un objeto a partir de su clase



27

Clases y Objetos

- “FIDO” es UN “PERRO”
 - “FIDO” es del TIPO “PERRO”
 - “FIDO” es un OBJETO
 - “PERRO” es la CLASE de “FIDO”
-
- “CHESTER” es OTRO “PERRO”
 - “CHESTER” también es del TIPO “PERRO”
 - “CHESTER” es otro OBJETO
 - “PERRO” también es la clase de “CHESTER”



28

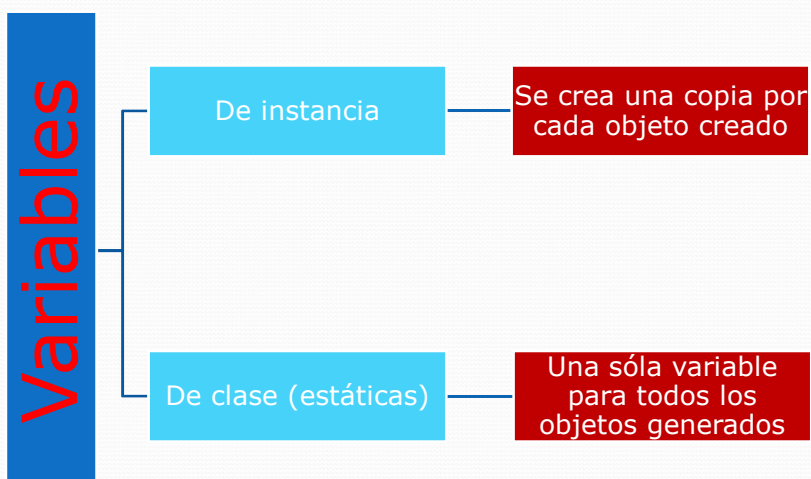
Atributos

- Representan los datos de los objetos
- Son controlados a través de la declaración de variables
- Es importante identificar el tipo de dato
- Se debe seleccionar sólo aquellos atributos necesarios para el modelo planteado (abstracción)



29

Atributos



30

Ejemplo: Atributos de un estudiante

□ Atributos:

- claveMatrícula: "A-233"
- nombre: "Bruno López Takeyas"
- grado: 3
- grupo: 'A'
- promedio: 87.4

```
string claveMatricula;  
string nombre;  
int grado;  
char grupo;  
float promedio;
```

31

Métodos

- Son las acciones que realizan los objetos y definen su comportamiento

□ Atributos:

- claveMatrícula: "A-233"
- nombre: "Bruno López Takeyas"
- grado: 3
- grupo: 'A'
- promedio: 87.4

□ Acciones:

- Leer()
- Investigar()

```
void Leer()  
void Investigar()
```

32

Propiedades

- Son mecanismos que permiten acceder a los atributos de un objeto.
- Algunos autores asumen que las propiedades son sinónimos de los datos
- En un sentido estricto, las propiedades actúan como un canal de comunicación para acceder a un atributo, ya sea para consultar o modificar su valor.
- Descriptores de acceso: **get** y **set**.

33

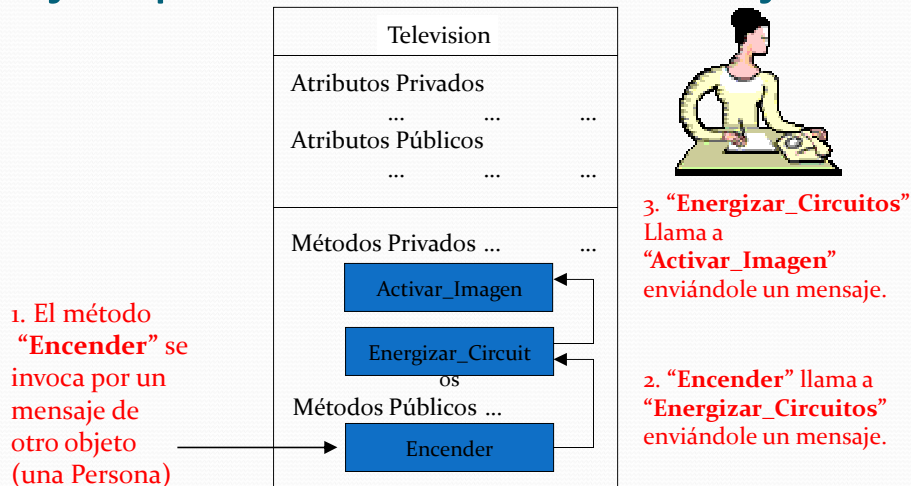


Anatomía de un mensaje

- Identidad del receptor
- Método que ha de ejecutar
- Información especial (argumentos o parámetros)
- Ejemplos:
 - **miTelevision.Encender()**
 - **miTelevision.Apagar()**
 - **miTelevision.CambiarCanal(45)**
 - **miPerro.Comer("Croquetas")**
 - **miEmpleado.Contratar ("Juan", 3500)**
 - **miFactura.Imprimir()**

34

Ejemplo de envío de mensajes



35

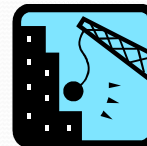
Constructores y Destructores

- Los objetos ocupan espacio en memoria; existen en el tiempo y deben crearse [instanciarse] y destruirse:

■ **Constructor.**- Operación que crea un objeto y/o inicializa su estado.



■ **Destructor.**- Operación que libera el estado de un objeto y/o destruye el propio objeto.



36

Ejemplo de constructor y destructor

Cada vez que se **enciende** la Television...

- Se deben energizar los circuitos
- Se debe activar el cinescopio
- ...Para posteriormente mostrar la imagen.



Cada vez que se **apaga** la Television...

- Se deben des-energizar los circuitos
- Se debe des-activar el cinescopio
- ...Para posteriormente apagar la imagen

37

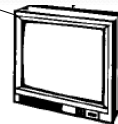
Herencia

Aparato_Electrodomestico

(Atributos:
Interruptor,
CableElectrico
Métodos:
Encender, Apagar)



Herendan características de
Aparato_Electrodomestico
e incorporan las suyas propias.



Lavadora

• Atributos:
(Interruptor, CableElectrico,
PerillaDeCiclosDeLavado,
CapacidadDeCarga)
• Métodos:
(Encender, Apagar,
LlenarConAqua, TirarAqua)

Televisión

• Atributos:
(Interruptor, CableElectrico,
BotonDeCanales, BotonDeVolumen)
• Métodos:
(Encender, Apagar,
CambiarVolumen, CambiarCanal)

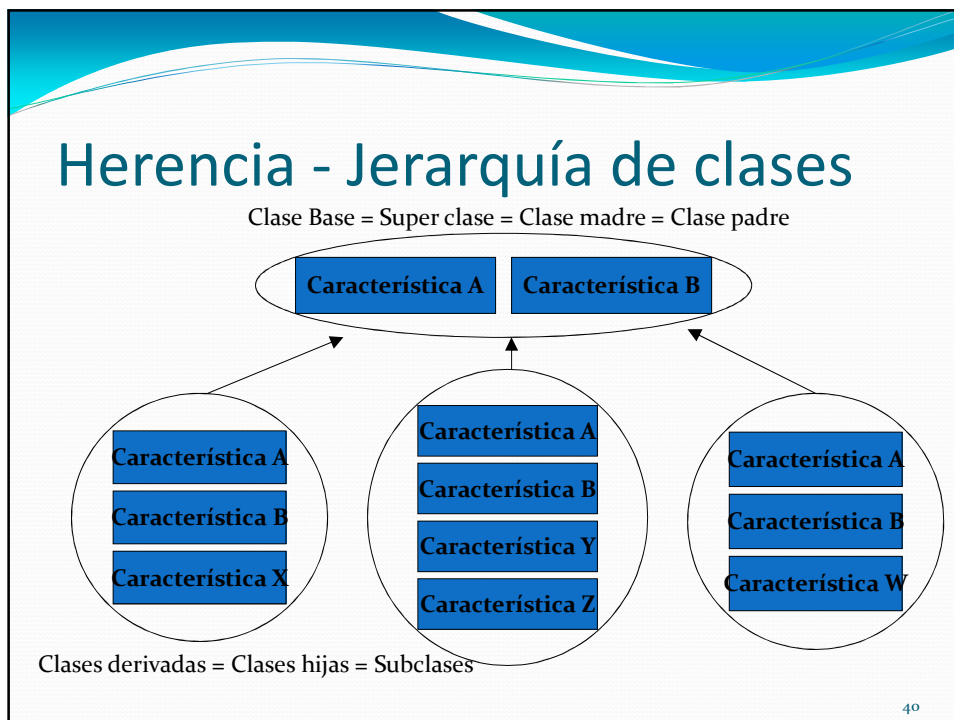
38



Herencia

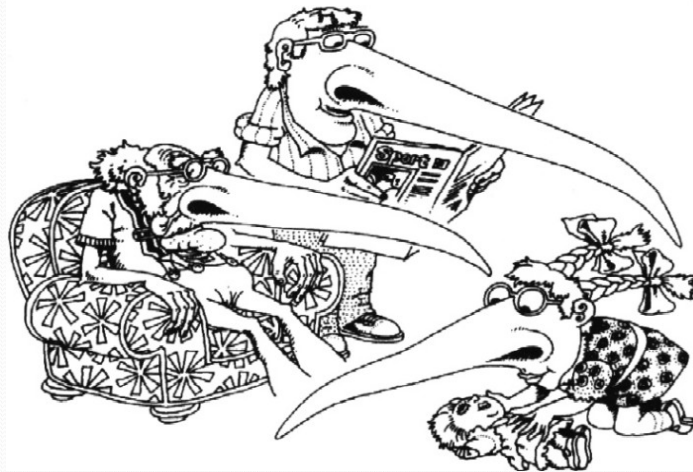
- Capacidad para utilizar características previstas en antepasados o ascendientes.
- Permite construir nuevas clases a partir de otras ya existentes, permitiendo que éstas les “transmitan” sus propiedades.
- Objetivo: Reutilización de código.

39



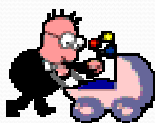
Herencia

Una subclase hereda el comportamiento y la estructura de su Super Clase

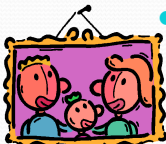


41

Tipos de Herencia



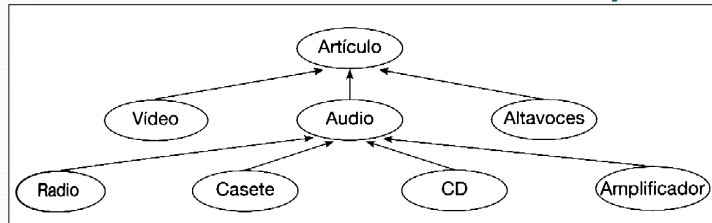
- **Herencia Simple.-** Una clase puede tener sólo un ascendiente. [Una subclase puede heredar de una única clase].



- **Herencia múltiple (en malla).-** Una clase puede tener más de un ascendiente inmediato. [Heredar de más de una clase].

42

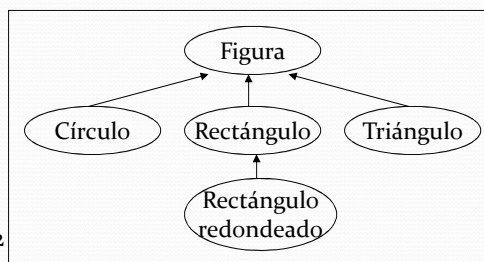
Herencia simple



Ejemplo 1

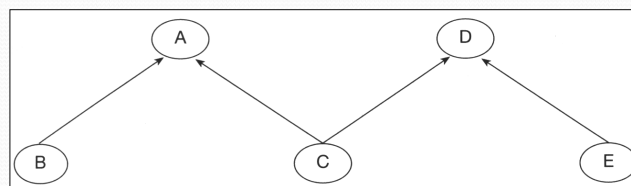


Ejemplo 2

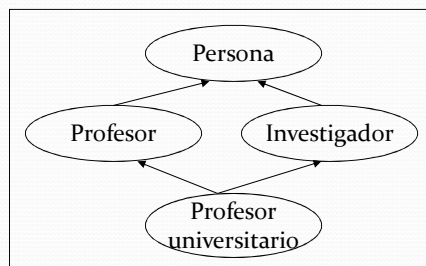


43

Herencia múltiple



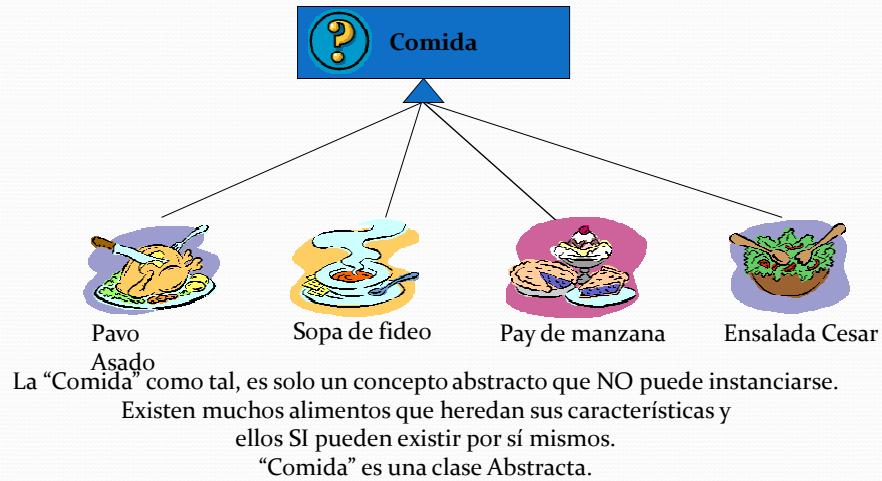
Ejemplo 1



Ejemplo 2

44

Clase abstracta



45

Clase abstracta



- Es una clase que sirve como clase base común, pero **NO** puede tener instancias.
- Una clase abstracta solo puede servir como clase base (solo se puede heredar de ella).
- Sus clases "hijas" SI pueden tener instancias.

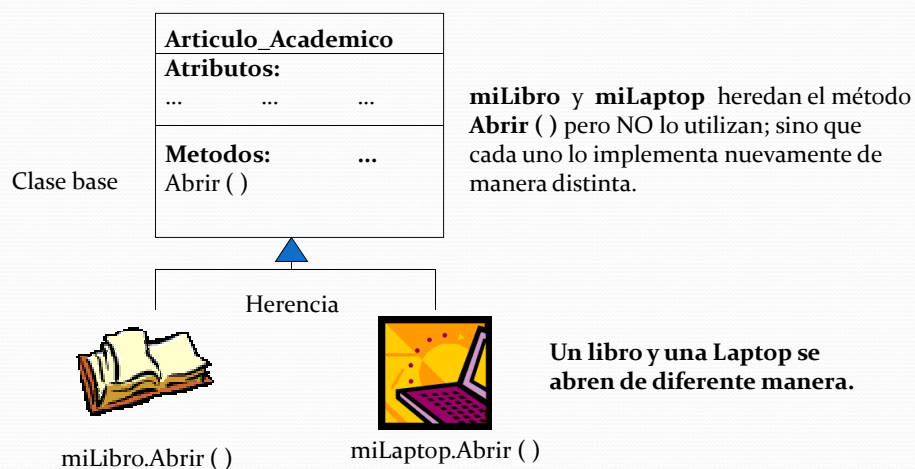
46

Anulación / Sustitución / sobrescritura [Overriding]

- Sucede cuando una clase “B” hereda características de una clase “A”, pero la clase “B” re-define las características heredadas de “A”.
- Propiedades y métodos pueden heredarse de una superclase. Si estas propiedades y métodos son re-definidos en la clase derivada, se dice que han sido “Sobrescritos”.

47

Anulación / Sustitución / sobrescritura [Overriding]



48

Sobrecarga [Overload]

- La sobrecarga representa diferentes maneras de realizar una misma acción.
- En los programas se usa el mismo nombre en diferentes métodos con diferentes firmas [número, orden y tipo de los parámetros].
- El código de programación asociado a cada sobrecarga puede variar.
- Ejemplos:
 - `miEmpleado.Contratar("Juan", "Ventas", 2500)`
 - `miEmpleado.Contratar("Juan")`
 - `miEmpleado.Contratar("Juan", 2500)`

49

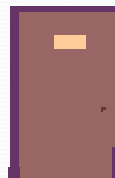
Ejemplo de Sobrecarga [Overload]



`miPuerta.Abrir (Adentro, Afuera)`




`miPuerta.Abrir (Afuera, Adentro)`



`miPuerta.Abrir ()`

50



Polimorfismo

Se refiere a:

1. Es el uso de un mismo nombre para representar o significar más de una acción.
 - La sobrecarga es un tipo de Polimorfismo.
2. Que un mismo mensaje pueda producir acciones totalmente diferentes cuando se recibe por objetos diferentes del mismo tipo.
 - Un usuario puede enviar un mensaje genérico y dejar los detalles de la implementación exacta para el objeto que recibe el mensaje en tiempo de ejecución.
 - Para este caso, se utiliza herencia y sobrescritura (Override).

51

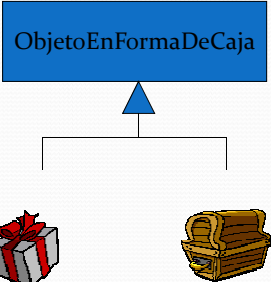
Polimorfismo



POLI = Múltiples MORFISMO = Formas



miRefrigerador.Abrir("Puerta de Abajo")
miRefrigerador.Abrir("Puerta de Arriba", "Mitad")



```
graph BT; ObjetoEnFormaDeCaja --> miRegalo; ObjetoEnFormaDeCaja --> miCofre;
```

miRegalo.Abrir() miCofre.Abrir()

52

Software

- **NClass** es software para el diseño de diagramas de clases.
- Puede descargarse de manera gratuita en:

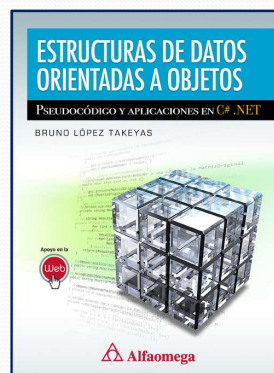
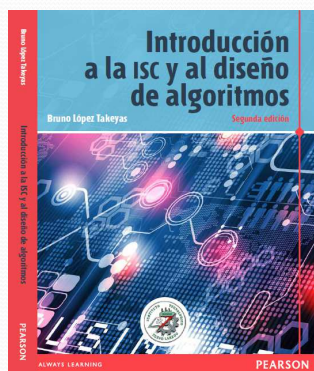


<http://nclass.sourceforge.net>

53

Otros títulos del autor

<http://www.itnuevolaredo.edu.mx/Takeyas/Libro>



takeyas@itnuevolaredo.edu.mx



Bruno López Takeyas