

# Orientación a Objetos

## AGENDA

- Breve introducción histórica a la OO
- Que es la Orientación a Objetos?
- Que es un Objeto?
- Que es una Clase?
- Conceptos Adicionales / Conceptos Avanzados
- Abstracción
- Encapsulamiento
- Herencia
- Polimorfismo
- UML
- Notación de Clases y Objetos
- Compartimientos de las Clases
- Relaciones

### Una forma nueva de pensar

- Programación Orientada a Objetos (POO) ≠ **Unas cuantas** características nuevas añadidas a un lenguaje de programación.
- Programación Orientada a Objetos (POO) = **Una nueva forma de pensar** acerca del proceso de descomposición de problemas y de desarrollo de soluciones de programación.

*La Programación Orientada a Objetos surge en la historia como un intento para dominar la complejidad que, de forma innata, posee el software.*

- Tradicionalmente, la forma de enfrentarse a esta complejidad ha sido empleando lo que llamamos Programación Estructurada, que consiste en descomponer el problema objeto de resolución en subproblemas y mas subproblemas hasta llegar a acciones muy simples y fáciles de codificar.

- Se trata de descomponer el problema en acciones, en verbos.
  - Ejemplo: en un programa que resuelve ecuaciones de segundo grado, descomponímos el problema en las siguientes acciones: primero, pedir el valor de los coeficientes a, b y c; después, calcular el valor del discriminante; y por ultimo, en función del signo del discriminante, calcular ninguna, una o dos raíces.

Un poco de historia (cont.)

- Este nuevo método de descomposición (POO) es la descomposición en objetos;
- Vamos a fijarnos no en lo que hay que hacer en el problema, sino en cuál es el escenario real del mismo, y vamos a intentar simular ese escenario en nuestro programa.

## ***¿Qué es la Orientación a Objetos?***

- La Orientación a Objetos es un paradigma de programación
- La Orientación a Objetos es “simplemente”, **una forma de ver las cosas, o bien...**
- Es una forma de entender un problema identificando las entidades principales que se encuentran en él...
- La programación orientada a objetos (POO) es por tanto Una forma de desarrollar un sistema, pensando en las entidades principales del problema que dicho sistema pretende resolver...
- El propósito en la **programación orientada a objetos** consiste en... proporcionar una solución informática identificando los **conceptos relevantes presentes en el problema**.
- Identificar los conceptos relevantes o las entidades involucradas en un problema significa... reconocer las características de estos y las acciones que realizan o bien que producen algún efecto sobre ellos.

***¿Qué es un objeto?***

Un Objeto es... cualquier cosa ... ... de la que se pueda emitir un concepto

... en el mundo está lleno de objetos reales, los cuales se pueden representar como tales en una solución computarizada.

***Veamos...***

Este es un objeto...



***...es un automóvil, blanco, con 4 llantas y 5 puertas***

En la imagen, cada uno de los elementos que vemos es considerado un **OBJETO**



Sin embargo de todos ellos podemos emitir un concepto conocido y para ello usamos la palabra GLOBO

A ese **concepto conocido que representa una agrupacion de objetos** ... lo llamamos **Clase**

La Clase es como un molde de galletas...



...determina la forma y las características que las galletas (el objeto) va a tener, sin ser el objeto real.

El molde (la clase) no determina qué sabor tiene cada una de las galletas, .....tampoco por cuánta cantidad de ingredientes estará compuesta cada una.

**Los objetos son todos los que podamos crear mentalmente a partir de dicha clase o concepto.**

### Ejercicio I

Tomemos el concepto de la clase PERSONA

Este grupo de personas tiene un conjunto de **características y comportamientos comunes:**

**Características:**

- Nombre
- Edad
- Color de piel
- Profesión
- Estado civil

**Comportamientos:**

- Hablar
- Caminar
- Mirar
- Nacer
- Morir

### Ejercicio I

Un objeto de la clase persona seria:

**Características:**

Nombre: Sandra  
Edad: 22  
Color de piel: Morena  
Profesión: Deportista  
Estado civil: Soltera

**Comportamientos:**

Hablar  
Caminar  
Mirar  
Nacer  
Morir

### En resumen

**Una clase está compuesta por características (atributos y propiedades) y por comportamientos (acciones y métodos)**

### **Otros conceptos**

A los valores que tienen los atributos de un objeto se los conoce como el estado del objeto y a los métodos que ofrece se los conoce como la interfaz.

Al código usado para construir las clases se lo conoce como la implementación de la clase.

Los **objetos** se **comunican** con otros a través de **mensajes**.

Un **mensaje** es una comunicación dirigida a un objeto, que le ordene que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó

Los objetos tienen distintos tipos de **relaciones**:

*Asociación  
Agregación / Composición*

### ***Conceptos avanzados***

Los principios que dirigen la orientacion a objetos son...

... ***la modularidad y la reusabilidad***

En términos simples **modularidad** significa trabajar por partes

Y la **reusabilidad** significa que no se invente la rueda!

Lo que ya está hecho es para usarse, y algunas de las cosas que no están hechas, deben construirse pensando en que alguien necesitará usarlo alguna vez.

### **Más conceptos...**

Con respecto a la comunicación, tenemos mas principios:

- Alta cohesión, y
- Bajo acoplamiento

**Alta cohesión**

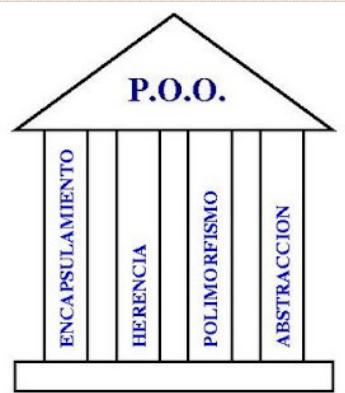
Cuando decimos que un componente tiene alta cohesión hablamos de que todos los componentes dentro de él están estrechamente relacionados

**Bajo acoplamiento**

Cuando decimos que un componente tiene bajo acoplamiento hablamos del nivel de independencia que tiene un componente con respecto a otros

## Características de la OO

- Abstracción
- Encapsulamiento
- Herencia
- Polimorfismo

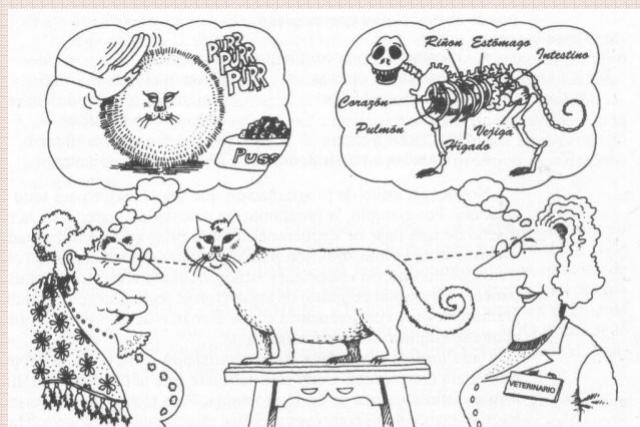


### **Abstracción**

Expresa las características esenciales de un objeto, las cuales distinguen al objeto de los demás, definiendo precisas fronteras conceptuales, relativas al observador.

- Surge del reconocimiento de similitudes entre ciertos objetos, situaciones o procesos en el mundo real.
- Decide concentrarse en estas similitudes e ignorar las diferencias.
- Enfatiza detalles con significado para el usuario, suprimiendo aquellos detalles que, por el momento, son irrelevantes o distraen de lo esencial.

### **Abstracción**



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

## Encapsulamiento

*Un usuario de un objeto no tiene que preocuparse por la estructura de los datos ocultos en el interior del objeto, sólo por lo que el objeto puede hacer, es decir, con los servicios que ofrece a otros objetos.*

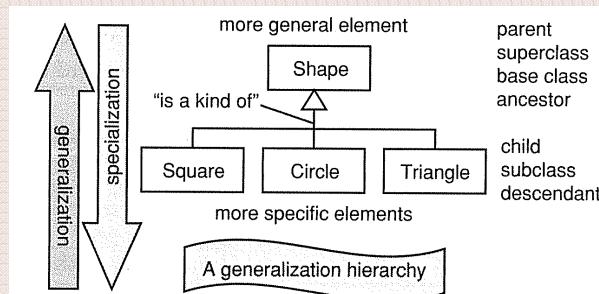
## Encapsulamiento

El encapsulamiento es el proceso de ocultar todos los detalles de un objeto que no contribuyen a sus características esenciales.

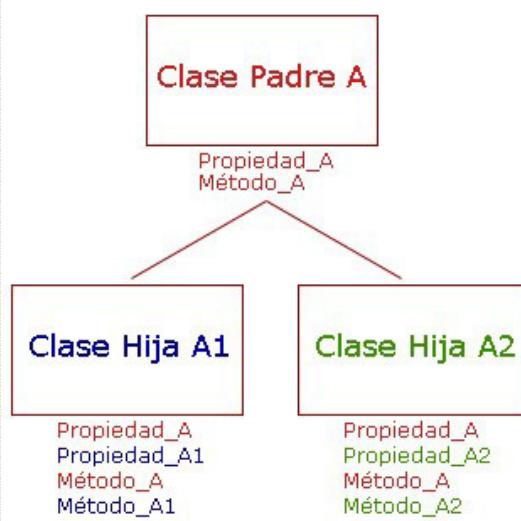
Un gato ronronea, maúlla, detecta perros **COMO?**  
→ **NO IMPORTA**

## Herencia

La **generalización** es una relación entre algo más general y una algo más específico, donde el elemento más específico es *enteramente consistente con* el elemento más general, pero *contiene más información*.



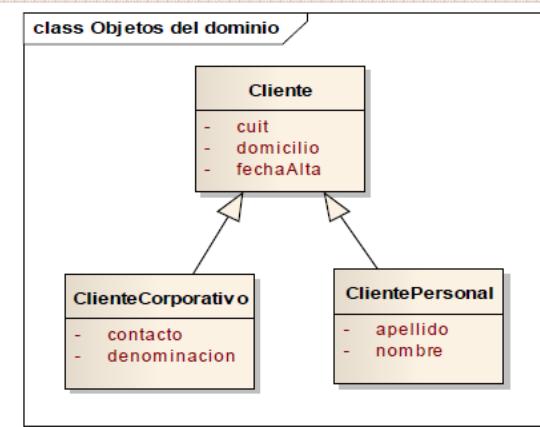
## Herencia de Clases

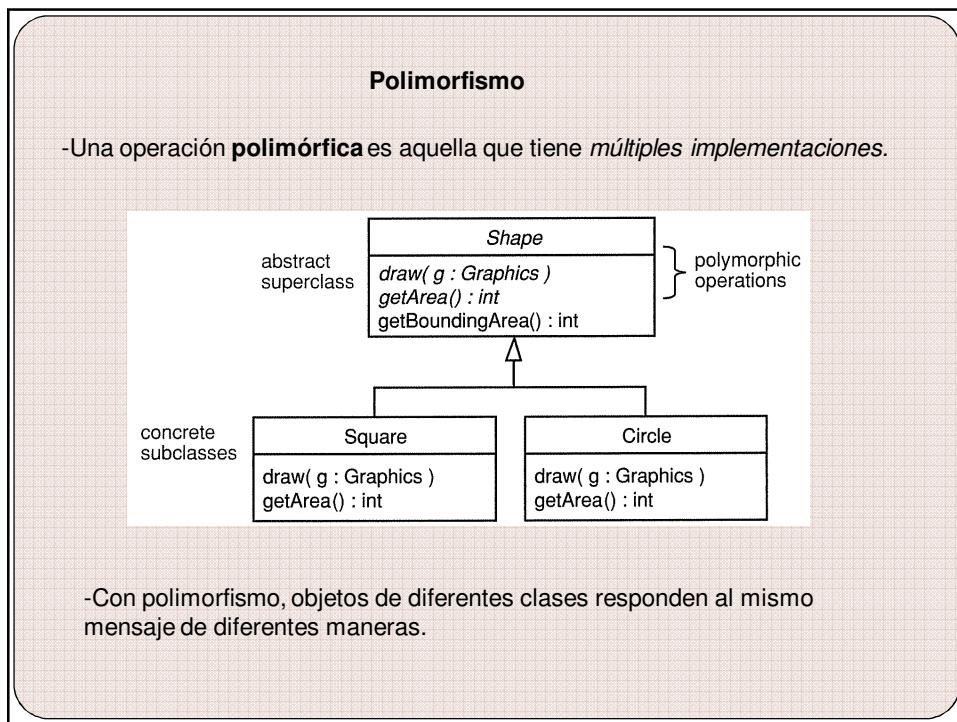


### Herencia – Un Ejemplo

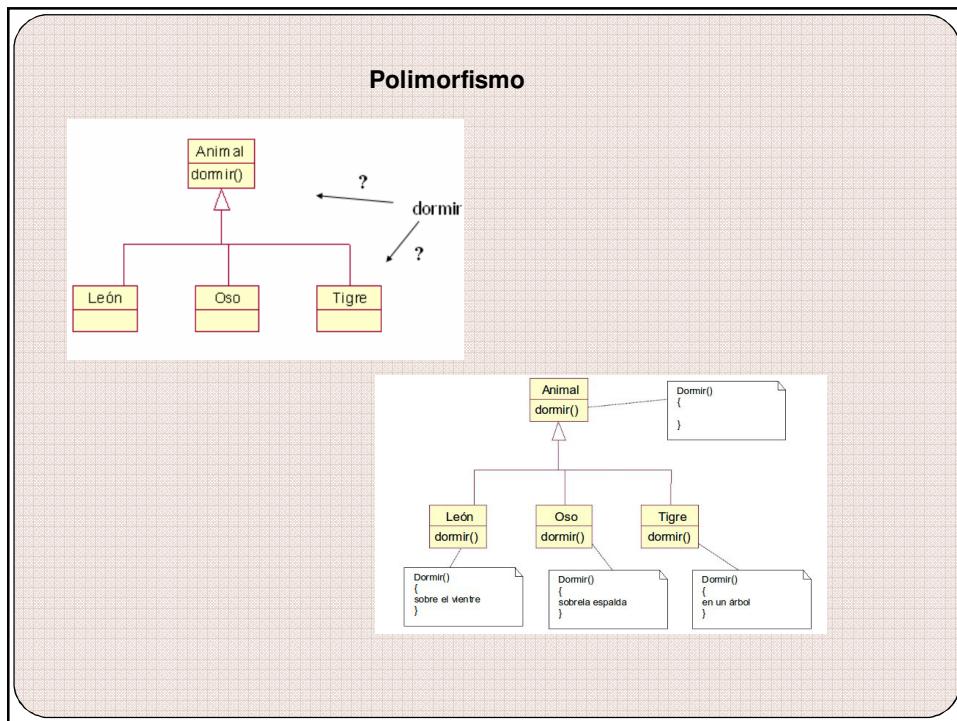
- Una organización tiene clientes y estos clientes pueden ser personas o corporaciones.
- Todos son clientes, pero cada uno tiene particularidades.
- Por ser clientes tienen características comunes y compartidas, como por ejemplo un domicilio, una cuit y una fecha en la cual comenzaron a ser clientes.
- Las personas tienen un nombre.
- Las organizaciones tienen una denominación e información de un contacto.
- Un cliente corporativo también es un cliente, es decir, que tiene las características propias (atributos y métodos) mas las que hereda de cliente.
- De manera similar, un cliente personal también es un cliente.

### Herencia – Un Ejemplo





-Con polimorfismo, objetos de diferentes clases responden al mismo mensaje de diferentes maneras.



### Unified Modeling Language-UML

-El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual para sistemas.

-UML se diseñó para incorporar las mejores prácticas en las técnicas de modelado y la ingeniería de software.

-UML no nos proporciona ningún tipo de metodología. Tampoco está unido a ninguna metodología específica o ciclo de vida y se puede utilizar con todas las metodologías existentes.

### Unified Modeling Language-UML

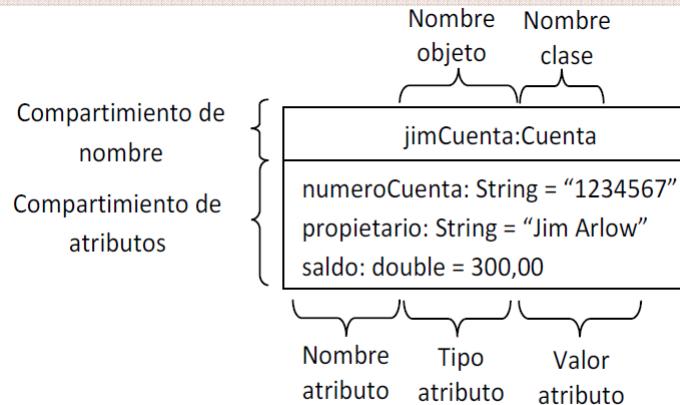
-La premisa básica de UML es que podemos modelar software y otros sistemas como colecciones de objetos que interactúan.

-Existen dos aspectos en un modelo UML:

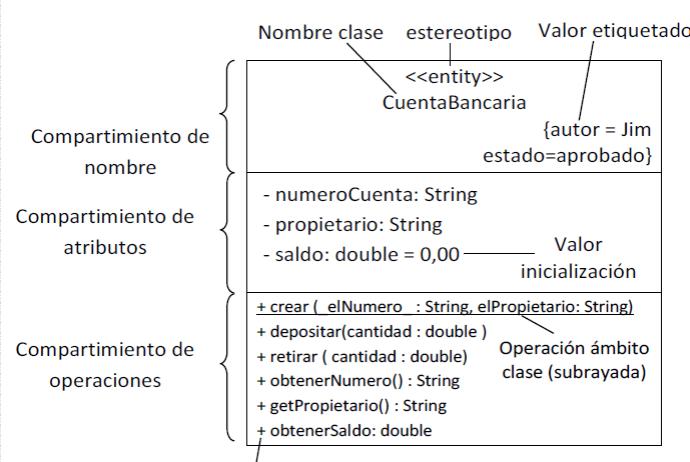
**Estructura estática:** describe qué tipo de objetos son importantes para modelar el sistema y cómo se relacionan.

**Comportamiento dinámico:** describe los ciclos de vida de estos objetos y cómo interactúan entre sí para entregar la funcionalidad del sistema requerida.

## Notación de objeto (UML)



## Notación de clase (UML)



### Compartimientos de la clase

**Nombre de la Clase:** empieza con una letra mayúscula y luego en una combinación mayuscula-minuscula, con cada palabra empezando en mayúscula. Se deben evitar abreviaciones.

**Atributos:** la única parte obligatoria de la sintaxis de atributo es el nombre.

**Visibilidad:** este adorno se aplica a atributos y operaciones.

### Visibilidad

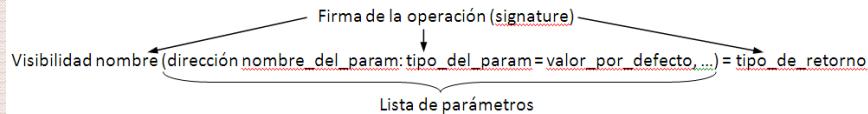
Adorno	Nombre	Semántica
+	Pública	Cualquier elemento que puede acceder a la clase puede acceder a cualquiera de sus características con visibilidad pública.
-	Privada	Solamente las operaciones dentro de la clase pueden acceder a características con visibilidad privada.
#	Protegida	Solo las operaciones dentro de la clase o dentro de hijos de la clase pueden acceder a características con visibilidad protegida.
~	De Paquete	Cualquier elemento que esté en el mismo paquete que la clase, o en un subpaquete anidado, puede acceder a cualquiera de sus características con visibilidad de paquete.

Tipo	
Tipo primitivo	Semántica
Integer (Entero)	Un número entero.
Natural ilimitado	Un número entero $\geq 0$ . Infinito.
Bolean (Booleano)	Puede tomar el valor verdadero o falso.
String (Cadena)	Una secuencia de caracteres. Los literales de cadena van entre comillas. Ej: "jim".

Compartimiento de atributos					
<b>-Multiplicidad:</b>					
<table border="1"> <tr> <td>DetallePersona</td> <td></td> </tr> <tr> <td>           -nombre: String [2..*]            -direccion: String [3]            -direccionEmail: String [0..1]         </td><td>           Nombre se compone de dos o más String            Dirección se compone de tres String            direcciónEmail se compone de una String o null         </td></tr> </table>	DetallePersona		-nombre: String [2..*] -direccion: String [3] -direccionEmail: String [0..1]	Nombre se compone de dos o más String Dirección se compone de tres String direcciónEmail se compone de una String o null	
DetallePersona					
-nombre: String [2..*] -direccion: String [3] -direccionEmail: String [0..1]	Nombre se compone de dos o más String Dirección se compone de tres String direcciónEmail se compone de una String o null				
<b>Valor inicial:</b>	le permite especificar el valor que tomara un atributo cuando se instancia un objeto desde la clase.				

## Construcción y destrucción de objetos

### -Multiplicidad:



**-Dirección de los parámetros:** in, inout, out

**-Valor por defecto de los parámetros:**

Canvas
drawCircle (origin : Point = Point (0,0), radius : Integer)
drawSquare (origin : Point = Point (0,0), size : Dimension)

## Construcción y destrucción de objetos

- Los constructores son operaciones especiales que crean nuevas instancias de las clases.
- Una clase puede tener muchos constructores, todos con el mismo nombre pero con distinta lista de parámetros.
- El constructor sin parámetros se conoce como constructor por defecto.
- Los destructores son operaciones especiales que "limpian" cuando los objetos son destruidos.

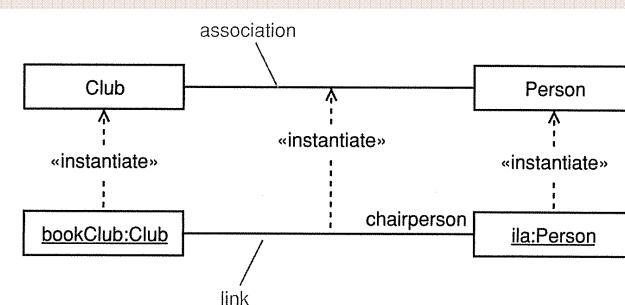
## Relaciones

Los **objetos** tienen distintos tipos de **relaciones**, dentro de las que se destacan:

- Asociación**
- Agregación**
- Composición**

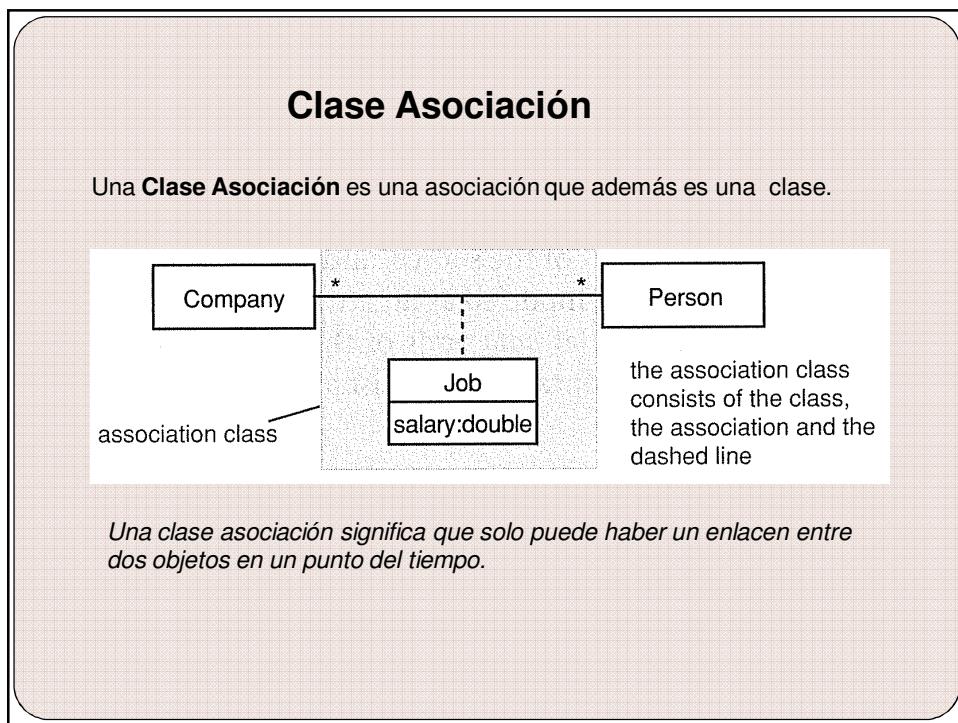
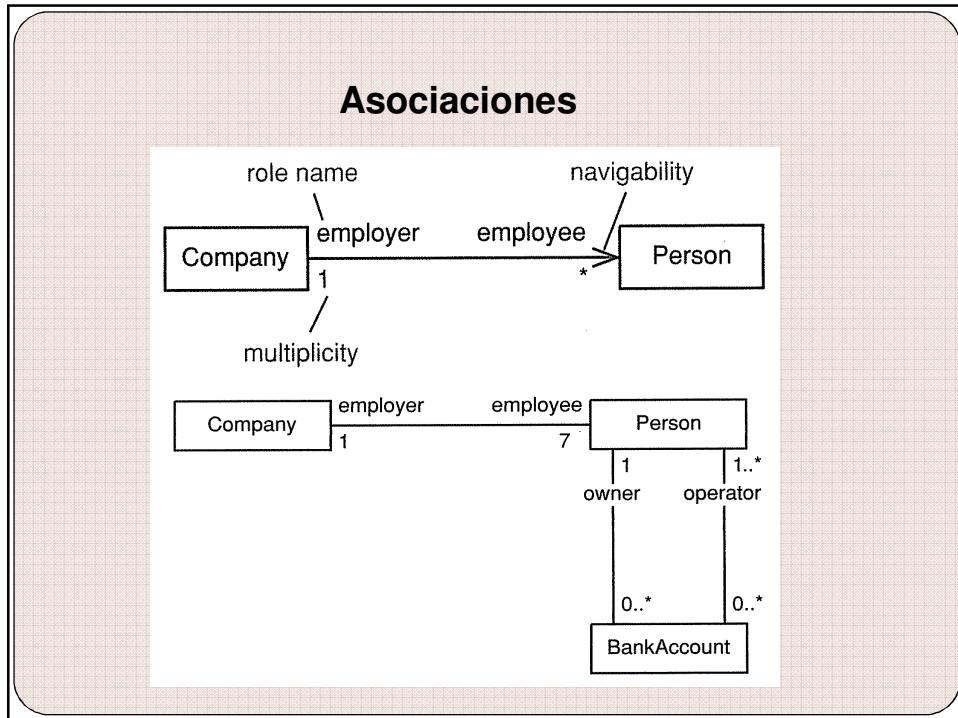
## ¿Qué es una Asociación?

- Las **Asociaciones** son relaciones entre clases



### - Sintaxis:

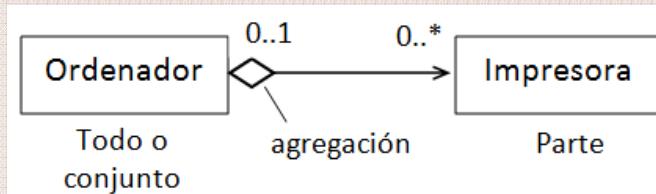
- Nombre de la asociación
- Nombre de roles
- Multiplicidad
- Navegabilidad



## Agregación

*Este es el tipo de relación más normal entre objetos; un ejemplo podría ser un ordenador y sus periféricos.*

Es un tipo de relación todo-parte en la que el conjunto se compone de muchas partes. Un objeto (el todo) utiliza los servicios de otro objeto (la parte).



## Agregación

*Semántica de la Agregación:*

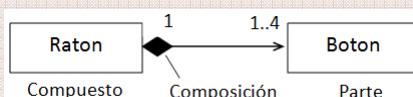
- El conjunto puede existir algunas veces independientemente de las partes, algunas veces no.
- Las partes pueden existir independientemente del conjunto.
- El conjunto está en cierto sentido incompleto si falta alguna de las partes.
- Es posible tener propiedad compartida de las partes por varios conjuntos.
- La agregación es transitiva.
- La agregación es asimétrica: un objeto nunca puede ser directa o indirectamente parte de si mismo.

## Composición

*Este es un tipo muy fuerte de relación entre objetos; es como un árbol y sus hojas.*

- La composición es una forma mas fuerte de agregación y tiene semántica similar, pero mas restringida.
- La diferencia clave es que en la composición las partes no tienen vida independiente fuera del todo.
- En composición, cada parte pertenece al menos a un y solo a un todo, mientras que en una agregación una parte se puede compartir entre los todos.

## Composición



### Semántica de la composición:

- Las partes pueden pertenecer solamente a un conjunto cada vez; no existe posibilidad de propiedad compartida de una parte.
- El conjunto tiene responsabilidad única para la disposición de todas su partes: responsabilidad para su creación y destrucción.
- El conjunto puede liberar partes, siempre y cuando la responsabilidad para ellas la asuma otro objeto.
- Si se destruye el conjunto, debe destruir todas sus partes o pasar la responsabilidad a algún otro objeto.

## Resumen

*Las clases y los objetos son los “bloques de construcción” de los sistemas OO.*

- Cada objeto es una instancia de una clase. Una clase define las características comunes compartida por todos los objetos de esa clase.
- Encapsulamiento: los datos dentro de un objeto están ocultos y solamente se pueden manipular al invocar una de las funciones del objeto.

## Resumen

- Todo objeto tiene las siguientes características:
- Identidad, su existencia única: utilice referencias de objeto para referirse de forma única a objetos específicos.
- Estado: un conjunto significativo de valores y relaciones de atributo para el objeto en un punto en el tiempo.
- Comportamiento: los servicios que el objeto ofrece a otros objetos. Se modelan como un conjunto de operaciones.
- Los objetos colaboran para generar el comportamiento del sistema. Esta interacción se logra con objetos enviando mensajes.

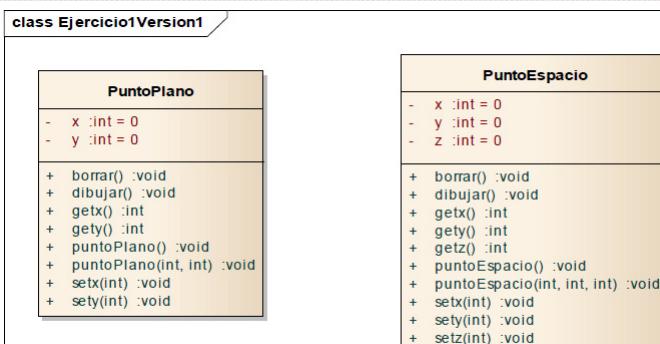
## EJERCICIOS

### Ejercicio 1:

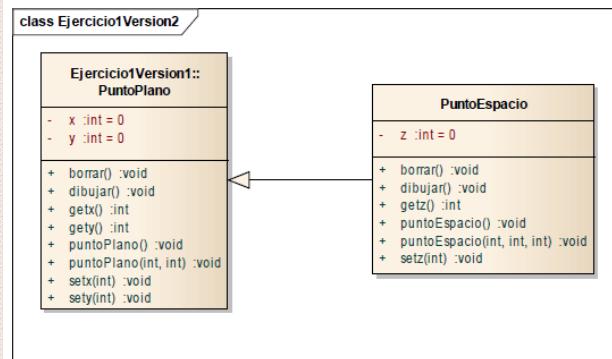
El punto es una “figura geométrica” adimensional: no tiene longitud, área, volumen, ni otra dimensión. El mismo describe posición en el espacio de acuerdo a un sistema de coordenadas preestablecido.

Cómo se podría diseñar el punto teniendo como referencia el sistema de coordenadas cartesianas, sabiendo que el mismo se representa con dos números (x, y) en el plano? Y, si se lo quiere representar en el espacio de tres dimensiones(x, y, z)?

### Solución 1



## Solución 2



### Ejercicio 2:

Un rectángulo puede ser definido por 4 puntos y puede entender los siguientes mensajes:

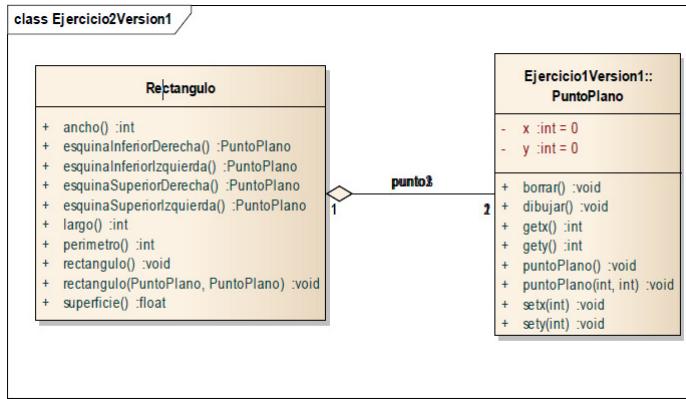
- esquinaSuperiorIzquierda
- esquinaSuperiorDerecha
- esquinaInferiorIzquierda
- esquinaInferiorDerecha

donde estos cuatro primeros mensajes devuelven el punto de la esquina correspondiente.

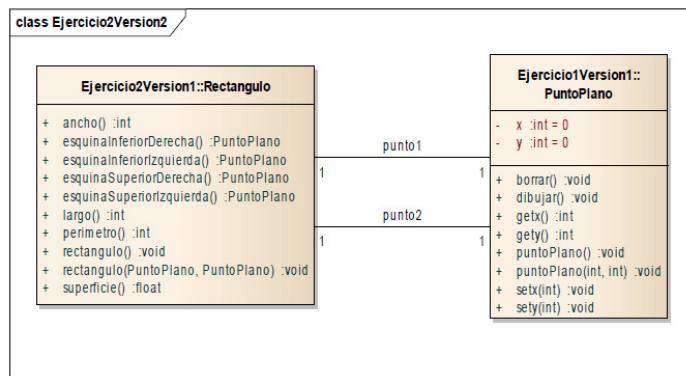
- largo
- alto
- superficie
- Perímetro

donde estos cuatro mensajes devuelven un valor que representa cada una de las características solicitadas.

## Solución 1



## Solución 2



### “Modelado Conceptual de Objetos”

- El modelo Conceptual
- Identificación de Clases
- Agregado de Relaciones
- Agregado de Atributos

### Modelado Conceptual

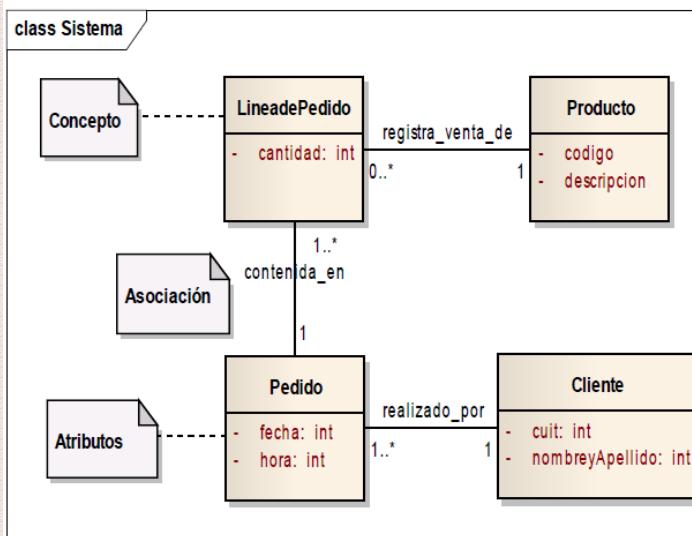
- Un **Modelo Conceptual** es una representación de conceptos en un dominio del problema



- Se utiliza un **Diagrama de Clases**
- Una cualidad esencial que debe ofrecer un modelo conceptual es que **representa cosas del mundo real**, no componentes de software.

**El modelo conceptual nos muestra:**

- Conceptos Clases
- Asociaciones entre conceptos → Relaciones
- Atributos de conceptos Atributos



### ¿Como se construye un modelo conceptual?

- 1.Liste los conceptos idóneos usando la Lista de categorías de conceptos y la identificación de la frase nominal relacionadas con los requerimientos en cuestión.
- 2.Dibujelos en un modelo conceptual.
- 3.Incorpore las asociaciones necesarias para registrar las relaciones para las cuales debe reservar un espacio de memoria.
- 4.Agregue los atributos necesarios para cumplir con las necesidades de información.

### ¿Como se construye un modelo conceptual?

#### Sugerencia ...

Prepare un modelo conceptual inspirándose en la metodología de trabajo de un cartógrafo:

- Utilice los nombre existentes en el territorio
- Excluya las características irrelevantes
- No agregue cosas que no existan

## Identificación de conceptos

La identificación de conceptos se lleva a cabo de dos formas (deben utilizarse ambas):

- **A partir de una lista de categorías:** se repasan las categorías y se identifican los conceptos correspondientes en el dominio.
- **A partir de las frases nominales:** se identifican las frases nominales o sustantivas en las descripciones textuales del dominio y se consideran conceptos o atributos.

## Lista de Categorías del Concepto

Categoría del Concepto	Ejemplos
objetos físicos o tangibles	Avión
especificaciones, diseño o descripciones de cosas	DescripcionDeVuelo
lugares	Aeropuerto
transacciones	Reservación
línea o renglón de elemento de transacción	-
rol de las personas	Piloto
contenedores de otras cosas	Avión
cosas dentro de un contenedor	Pasajero
Otros sistemas de cómputo o electromecánicos externos al sistema	ControlDeTraficoAereo

### Lista de Categorías del Concepto

Categoría del Concepto	Ejemplos
conceptos de nombre abstracto	Acrofobia
organizaciones	LíneaAérea
eventos	Vuelo, Accidente, Aterrizaje
procesos (a menudo no están representados como conceptos, pero pueden estarlo)	ReservaciónAsiento
reglas y políticas	PolíticasDeReservación
cátalagos	CatálogoDeMantenimientos
registros de finanzas, de trabajo, de contratos de asuntos legales	BitácoraDeMantenimiento
instrumentos y servicios financieros	Existencia
manuales, libros	ManualDeReparaciones

### Agregado de relaciones

La identificación de asociaciones se lleva a cabo de dos formas (deben utilizarse ambas):

- **A partir de asociaciones en que el conocimiento de la asociación ha de ser preservado algún tiempo.**
- **A partir de asociaciones provenientes de la lista de asociaciones comunes:** se revisan las categorías y se identifican asociaciones existentes entre los conceptos del dominio.

## Agregado de relaciones

### Sugerencias ...

- Concentrarse en las asociaciones en que el conocimiento de la relación ha de preservarse durante algún tiempo (asociaciones que "es necesario conocer").
- Es mas importante identificar los conceptos que las asociaciones.
- Muchas asociaciones tienden a confundir el modelo conceptual en vez de aclararlo. A veces se requiere mucho tiempo para descubrirlas, y los beneficios son escasos.
- No incluir las asociaciones redundantes ni las derivables.

## Lista de Categorías de Asociaciones

Categorías	Ejemplos
A es una parte física de B	Ala—Avión
A es una parte lógica de B	TramodeVuelo—RutadeVuelo
A está físicamente contenido en B	Pasajero—Avión
A está contenido lógicamente en B	Vuelo—ProgramadeVuelo
A es una descripción de B	Descripcionde Vuelo—Vuelo
A es un elemento de línea en una transacción o reporte B	TrabajodeMantenimiento—Mantenimiento
A se conoce/introduce/registra/presenta/captura en B	Reservación—ListadePasajeros
A es miembro de B	Piloto—Avión

### Lista de Categorías de Asociaciones

Categorías	Ejemplos
A es una subunidad organizacional de B	Mantenimiento—Línea Aérea
A usa o dirige a B	Piloto—Avión
A se comunica con B	Agente de Reservaciones—Pasajero
A se relaciona con una transacción B	Pasajero—Boleto
A es una transacción relacionada con otra transacción B	Reservación—Cancelación
A está contiguo a B	Ciudad—Ciudad
A es propiedad de B	Avión—Línea aérea

### Asociaciones de alta prioridad

Categorías	Ejemplos
A es una parte física de B	Ala—Avión
A es una parte lógica de B	Tramo de Vuelo—Ruta de Vuelo
A está físicamente contenido en B	Pasajero—Avión
<b>Categorías de asociaciones de Alta Prioridad que siempre conviene incluir</b>	
A se conoce/introduce/registra/presenta/captura en B	Reservación—Lista de Pasajeros

## Agregado de atributos

La identificación de los atributos se debe realizar teniendo como premisa la **representación de cosas reales** y teniendo en cuenta además que:

- Es preferible que los atributos sean atributos simples, por ejemplo, Booleano, Fecha, Numero, Cadena (Texto), Hora.
- Los conceptos deben relacionarse con una asociación, no a través de atributos.
- Ningún atributo debe incluirse como clave foránea.

## Agregado de atributos

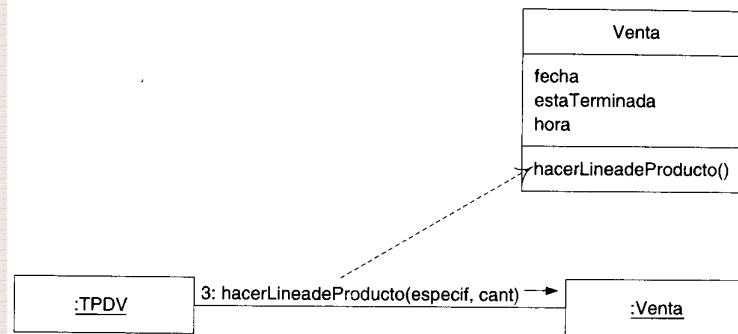
### Sugerencias ...

Represente como un nuevo concepto lo que inicialmente puede considerarse un tipo simple (un numero o una cadena, por ejemplo), si:

- Se compone de secciones independientes → numero telefónico, nombre de persona
- Normalmente se asocian a él operaciones como el análisis o la validación → numero del seguro social
- Posee otros atributos → el precio promocional podría tener fecha de inicio y de terminación
- Es una cantidad con una unidad → el importe del pago tiene una unidad monetaria

## Agregado de los métodos

Agregar un método en la clase A por cada mensaje enviado en los diagramas de interacción a algún objeto de la clase A.



## Agregado de los métodos - Excepciones

Nombre de los métodos: **crear**

- El mensaje crear es la forma independiente que tiene UML para representar la instanciación de objetos de una clase.
- Como cada lenguaje lo implementa de maneras muy diferentes, se aconseja no incluirlos en los diagramas de diseño. Algo similar ocurre con los métodos de inicialización.

## Agregado de los métodos - Excepciones

Nombre de los métodos: **métodos de acceso**

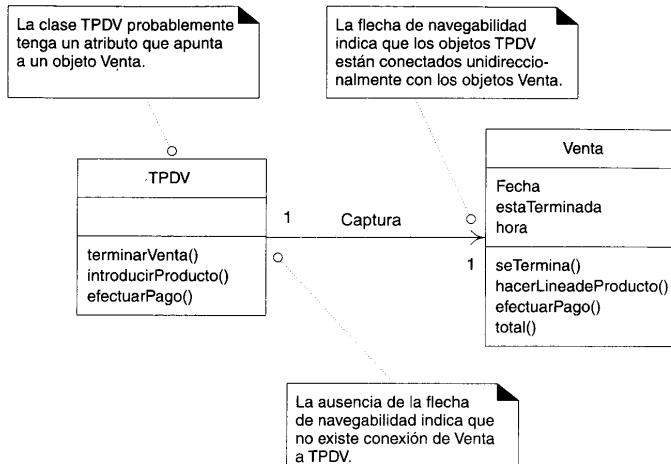
- No incorporar en el diagrama los métodos asociados a obtener (get) y modificar (set) los valores de los atributos, también llamados getters y setters, ya que si hay N atributos se deberían agregar  $2*N$  métodos.

## Agregado de asociaciones y navegabilidad

La navegabilidad es una propiedad del rol e indica la posibilidad de navegar unidireccionalmente en una asociación.

- Navegabilidad implica Visibilidad
- Ejemplos de necesidad de navegabilidad de A a B
  - A envía un mensaje a B
  - A crea una instancia de B
  - A necesita mantener una conexión con B

## Agregado de asociaciones y navegabilidad



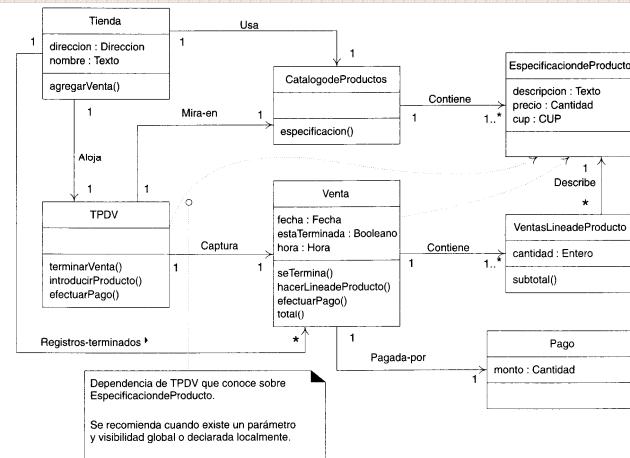
## Agregado de relaciones de dependencia

- El lenguaje UML incluye una **relación general de dependencia**, la cual indica que un elemento de cualquier tipo conoce la existencia de otro. Se denota con una línea punteada y con flecha

- Sirve para representar una relación de conocimiento o visibilidad cuando no existe una asociación. Los casos son:

- Pasaje como parámetro
- Globales

## Agregado de relaciones de dependencia



## Notación para agregar detalle a atributos y métodos

-Visibilidad (atributos y métodos):

- - **Privada:** solo accesible por el objeto
- + **Pública:** accesible por cualquier objeto
- # **Protegida:** accesible por los objetos que comparten una jerarquía de herencia

-Valores iniciales (atributos)

## Notación para agregar detalle a atributos y métodos

### **Tipo de Atributos:**

- **De Clase:** mantienen un único valor para todas las instancias de la clase
- **De Instancia:** cada instancia u objeto tiene su valor propio (son los que vimos hasta ahora)

### **Tipo de Métodos:**

- **De Clase:** los mensajes se envían a las clases
- **De Instancia:** los mensajes se envían a cada instancia u objeto

## Notación para agregar detalle a atributos y métodos

### Nombre de la clase

```

atributo
atributo : tipo
atributo : tipo = valor inicial
atributoDeClase
/atributoDerivado
...
metodo1()
metodo2(lista de parametros)
: tipo de retorno
metodoAbstracto()
+metodoPublico()
-metodoPrivado()
#metodoProtegido()
metododeClase()
...

```

### java.awt.Font

<pre> plain : Integer = 0 bold : Integer = 1 name : String style : Integer = 0 ... </pre>
<pre> +getFont(name : String) : Font +getName() : String ... </pre>

### java.awt.Toolkit

<pre> #createButton(target : Button) : ButtonPeer ... +getColorModel() : ColorModel ... </pre>
--