

---

# Diagramas de Interacción

1

III. El Paradigma OO: Diagramas de Interacción

## Interacción

- Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones. Los diagramas de interacción muestran cómo se comunican los objetos en una interacción
- Existen dos tipos de diagramas de interacción: el Diagrama de Colaboración y el Diagrama de Secuencia

2

## Mensajes

### Sintaxis para mensajes:

predecesor / guarda secuencia: retorno := msg(args)

- **Predecesor** es una lista separada por coma de los números de secuencia de mensajes que deben ocurrir antes del mensaje especificado.
- La **guarda** representa una condición para el envío del mensaje
- **Secuencia** representa el nivel de anidamiento procedural. Por ejemplo el mensaje 3.1.4 es posterior al mensaje 3.1.3 dentro de la activación 3.1. También se pueden añadir nombres para especificar mensajes concurrente, por ejemplo, el mensaje 3.1a y el mensaje 3.1b son concurrentes dentro de la activación 3.1.
- Ejemplos:
  - 2: mostrar(x,y) mensaje simple
  - 1.3.1: p: = encontrar(espec) llamada anidada con valor de retorno
  - [x<0] 4: invertir(x, color) mensaje condicional
  - A3, B4/ C3.1\*: actualizar sincronización con otros hilos de ejecución, iteración

3

## Diagramas de interacción

- El Diagrama de Secuencia es más adecuado para observar la perspectiva cronológica de las interacciones
- El Diagrama de Colaboración ofrece una mejor visión espacial mostrando los enlaces de comunicación entre objetos
- El D. de Colaboración puede obtenerse automáticamente a partir del correspondiente D. de Secuencia (o viceversa)

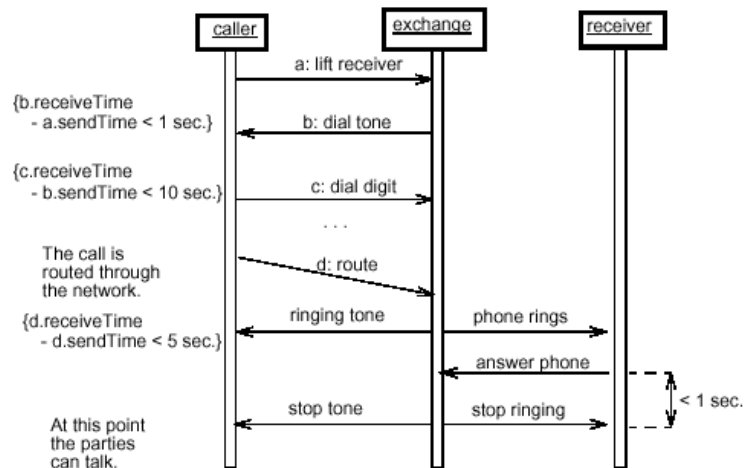
4

## Diagrama de Secuencia

- Muestra la secuencia de mensajes entre objetos durante un escenario concreto
- Cada objeto viene dado por una barra vertical
- El tiempo transcurre de arriba abajo
- Cuando existe demora entre el envío y la atención se puede indicar usando una línea oblicua

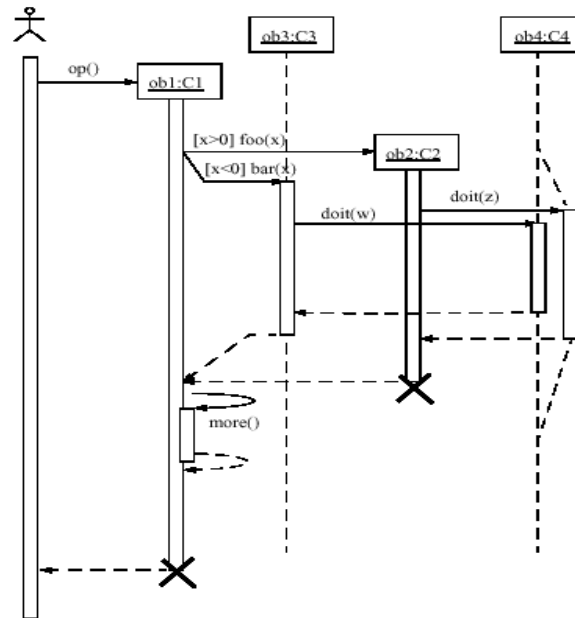
5

## ... Diagrama de Secuencia



6

Diagrama de Secuencia  
mostrando foco de control,  
condiciones, recursión  
creación y destrucción  
de objetos



7

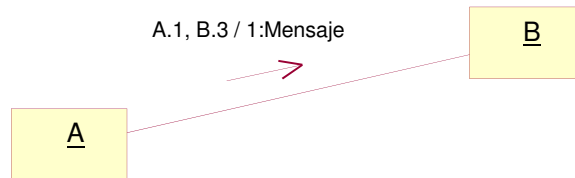
## Diagrama de Colaboración

- Son útiles en la fase exploratoria para identificar objetos
- La distribución de los objetos en el diagrama permite observar adecuadamente la interacción de un objeto con respecto de los demás
- La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces

8

## Mensajes

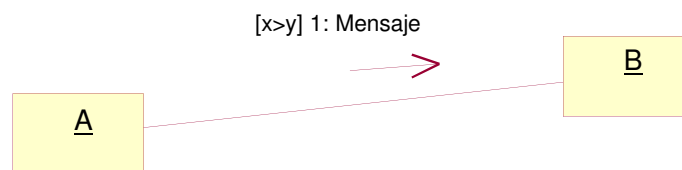
- Un mensaje desencadena una acción en el objeto destinatario
- Un mensaje se envía si han sido enviados los mensajes de una lista (sincronización):



9

## ... Mensajes

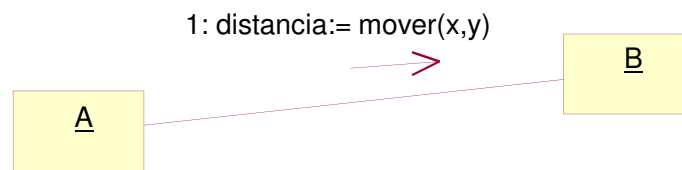
- Un mensaje se envía de manera condicionada:



10

## ... Mensajes

- Un mensaje que devuelve un resultado:



11

## Diagrama de Clases

12

## Clasificación

- El mundo real puede ser visto desde abstracciones diferentes (subjetividad)
- Mecanismos de abstracción:
  - Clasificación / Instanciación
  - Composición / Descomposición
  - Agrupación / Individualización
  - Especialización / Generalización
- La clasificación es uno de los mecanismos de abstracción más utilizados

13

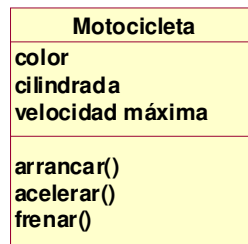
## Clases

- La clase define el ámbito de definición de un conjunto de objetos
- Cada objeto pertenece a una clase
- Los objetos se crean por instanciación de las clases

14

## Clases: Notación Gráfica

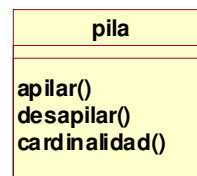
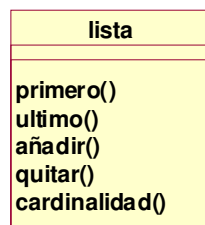
- Cada clase se representa en un rectángulo con tres compartimientos:
  - nombre de la clase
  - atributos de la clase
  - operaciones de la clase



15

## Clases: Notación Gráfica

- Otros ejemplos:



16



## Clases: Encapsulación

- La encapsulación presenta dos ventajas básicas:
  - Se protegen los datos de accesos indebidos
  - El acoplamiento entre las clases se disminuye
  - Favorece la modularidad y el mantenimiento
- Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos

17







## ... Clases: Encapsulación

- Los niveles de encapsulación están heredados de los niveles de C++:
  - **(-) Privado** : es el más fuerte. Esta parte es totalmente invisible (excepto para clases *friends* en terminología C++)
  - **(#)** Los atributos/operaciones **protegidos** están visibles para las clases *friends* y para las clases derivadas de la original
  - **(+)** Los atributos/operaciones **públicos** son visibles a otras clases (cuando se trata de atributos se está transgrediendo el principio de encapsulación)

18

## ... Clases: Encapsulación

- Ejemplo:

| Reglas de visibilidad   |                              |
|---|------------------------------|
|  | Atributo público : Integer   |
|  | Atributo protegido : Integer |
|  | Atributo privado : Integer   |
| <hr/>   |                              |
|  | Operación pública"()         |
|  | Operación protegida"()       |
|  | Operación privada"()         |

19

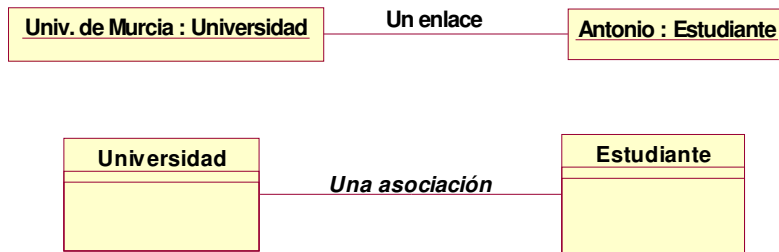
## Relaciones entre Clases

- Los enlaces entre de objetos pueden representarse entre las respectivas clases
- Formas de relación entre clases:
  - Asociación y Agregación (vista como un caso particular de asociación)
  - Generalización/Especialización
- Las relaciones de Agregación y Generalización forman jerarquías de clases

20

## Asociación

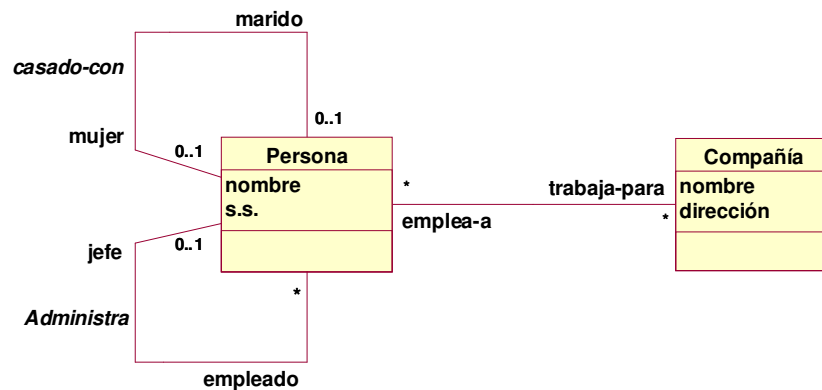
- La asociación expresa una conexión bidireccional entre objetos
- Una asociación es una abstracción de la relación existente en los enlaces entre los objetos



21

## ... Asociación

- Ejemplo:



22

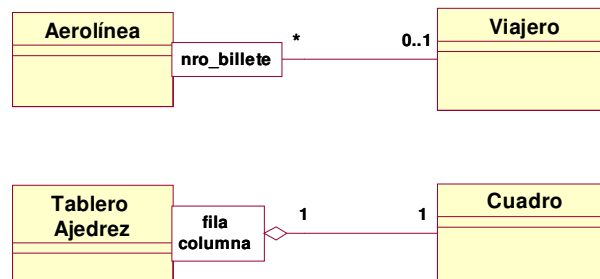
## ... Asociación

- Especificación de multiplicidad (mínima...máxima)
 

|      |                                     |
|------|-------------------------------------|
| 1    | Uno y sólo uno                      |
| 0..1 | Cero o uno                          |
| M..N | Desde M hasta N (enteros naturales) |
| *    | Cero o muchos                       |
| 0..* | Cero o muchos                       |
| 1..* | Uno o muchos (al menos uno)         |
- La multiplicidad mínima  $\geq 1$  establece una restricción de existencia

23

## Asociación Cualificada



Reduce la multiplicidad del rol opuesto al considerar el valor del cualificador

24

## Agregación

- La agregación representa una relación *parte\_de* entre objetos
- En UML se proporciona una escasa caracterización de la agregación
- Puede ser caracterizada con precisión determinando las relaciones de comportamiento y estructura que existen entre el objeto agregado y cada uno de sus objetos componentes

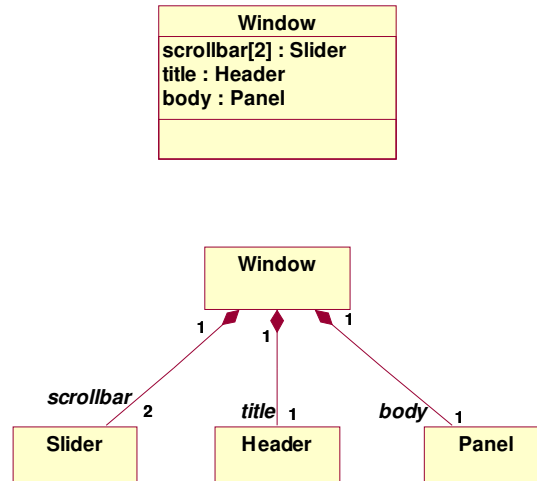
25

## ... Agregación: Caracterización

- ▣ En UML sólo se distingue entre agregación y composición (*aggregate composition*), siendo esta última disjunta y estricta (Cardinalidad de componente-compuesto 1:1)

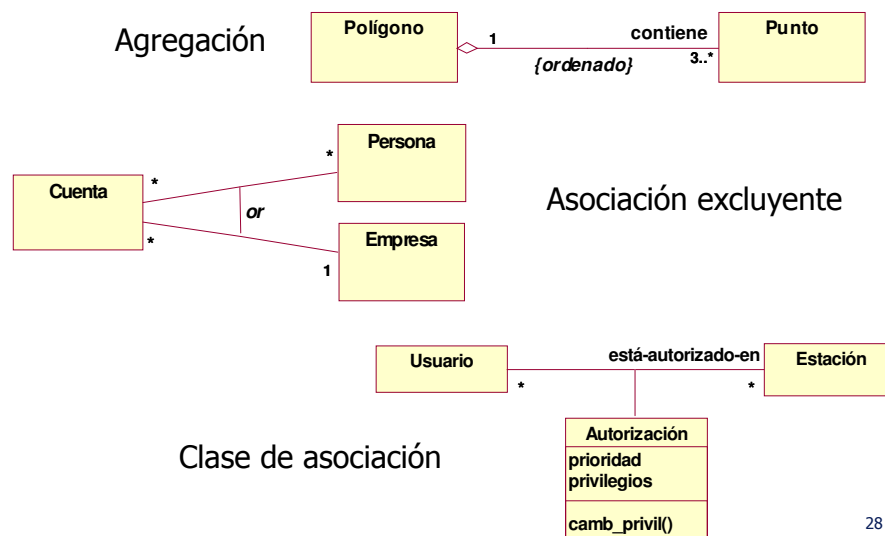
26

## Ejemplos



27

## ... Ejemplos



28

## Clases y Objetos

- Diagrama de Clases y Diagramas de Objetos pertenecen a dos vistas complementarias del modelo
- Un Diagrama de Clases muestra la abstracción de una parte del dominio
- Un Diagrama de Objetos representa una situación concreta del dominio

29

## Generalización

- Permite gestionar la complejidad mediante un ordenamiento taxonómico de clases
- Se obtiene usando los mecanismos de abstracción de Generalización y/o Especialización
- La Generalización consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general

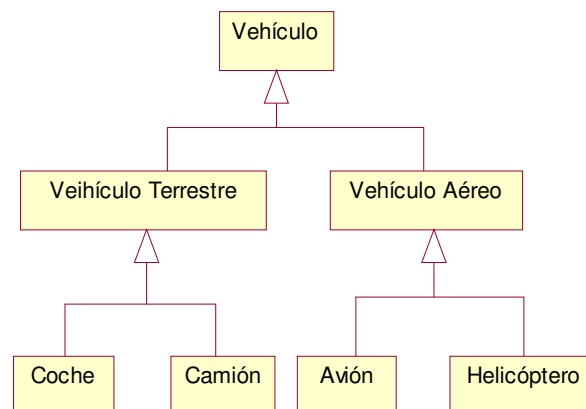
30

## ... Generalización

- Nombres usados: clase padre - clase hija.  
Otros nombres: superclase - subclase, clase base - clase derivada
- Las subclases **heredan** propiedades de sus clases padre, es decir, atributos y operaciones (y asociaciones) de la clase padre están disponibles en sus clases hijas

31

## ... Generalización

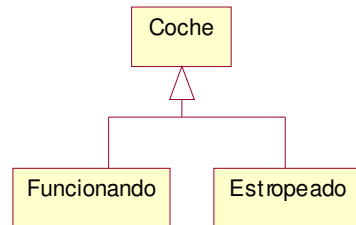


32



## ... Generalización

- La especialización es una técnica muy eficaz para la extensión y reutilización



- Restricciones predefinidas en UML:
  - disjunta - no disjunta
  - total (completa) - parcial (incompleta)

33

## ... Generalización

- La noción de clase está próxima a la de conjunto
- Dada una clase, podemos ver el conjunto relativo a las instancias que posee o bien relativo a las propiedades de la clase
- Generalización y especialización expresan relaciones de inclusión entre conjuntos

34

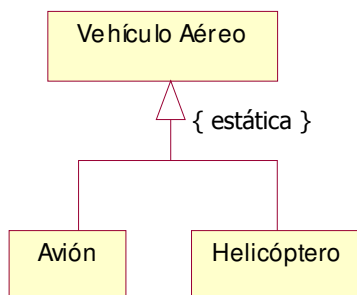
## ... Generalización

- Particionamiento del espacio de objetos => **Clasificación Estática**
- Particionamiento del espacio de estados de los objetos => **Clasificación Dinámica**
- En ambos casos se recomienda considerar generalizaciones/especializaciones disjuntas

35

## ... Generalización

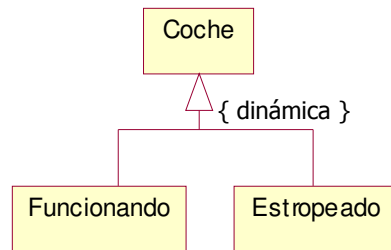
- Un ejemplo de Clasificación Estática:



36

## ... Generalización

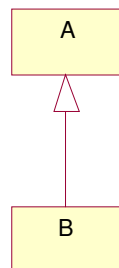
- Un ejemplo de Clasificación Dinámica:



37

## ... Generalización

- **Extensión:** Posibles instancias de una clase
- **Intensión:** Propiedades definidas en una clase



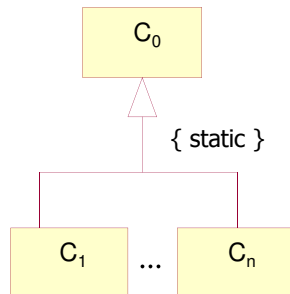
$$\text{int}(A) \subseteq \text{int}(B)$$

$$\text{ext}(B) \subseteq \text{ext}(A)$$

38

## ... Generalización

### ■ Clasificación Estática



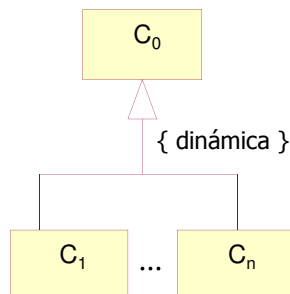
$\text{ext}(C_0) = \cup \text{ext}(C_i) \Rightarrow \text{completa}$

$\text{ext}(C_i) \cap \text{ext}(C_j) = \emptyset \Rightarrow \text{disjunta}$

39

## ... Generalización

### ■ Clasificación Dinámica



$\text{ext}(C_0) = \cup \text{ext}(C_i) \Rightarrow \text{completa}$

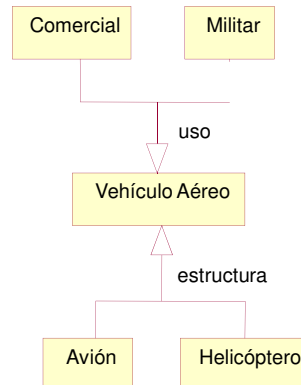
$\text{ext}_t(C_i) \cap \text{ext}_t(C_j) = \emptyset \Rightarrow \text{disjunta en } t$

$\text{ext}_{t_1}(C_i) \cap \text{ext}_{t_2}(C_j) \neq \emptyset \Rightarrow \text{posiblemente no disjunta en diferentes instantes}$

40

## ... Generalización

- Ejemplo: varias especializaciones a partir de la misma clase padre, usando **discriminadores**:



41

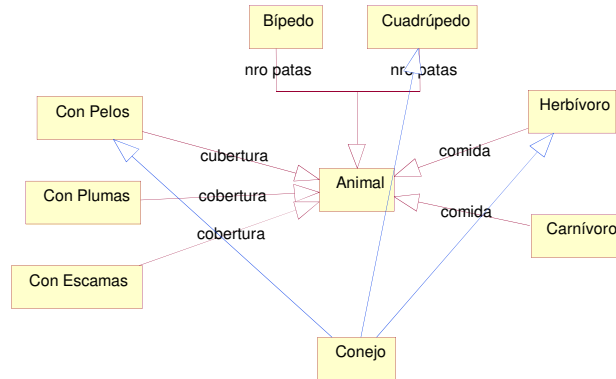
## Clasificación Múltiple (herencia múltiple)

- Se presenta cuando una subclase tiene más de una superclase
- La herencia múltiple debe manejarse con precaución. Algunos problemas son el conflicto de nombre y el conflicto de precedencia
- Se recomienda un uso restringido y disciplinado de la herencia. Java no ofrece soporte para herencia múltiple

42

## ... Herencia Múltiple

- Uso disciplinado de la herencia múltiple: clasificaciones disjuntas con clases padre en hojas de jerarquías alternativas



43

## Principio de Sustitución

- El Principio de Sustitución de Liskow (1987) afirma que:

*"Debe ser posible utilizar cualquier objeto instancia de una subclase en el lugar de cualquier objeto instancia de su superclase sin que la semántica del programa escrito en los términos de la superclase se vea afectado."*

44

## ... Principio de Sustitución

- Dado que los programadores pueden introducir código en las subclases redefiniendo las operaciones, es posible introducir involuntariamente incoherencias que violen el principio de sustitución
- El polimorfismo que veremos a continuación no debería implementarse sin este principio

45

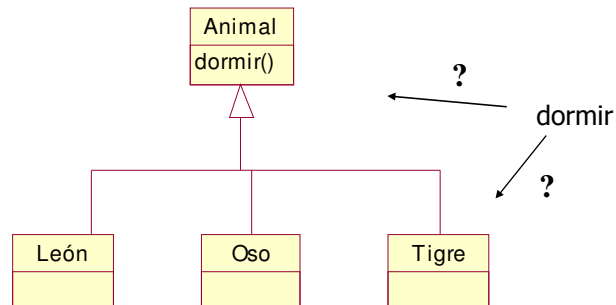
## Polimorfismo

- El término polimorfismo se refiere a que una característica de una clase puede tomar varias formas
- El polimorfismo representa en nuestro caso la posibilidad de desencadenar operaciones distintas en respuesta a un mismo mensaje
- Cada subclase hereda las operaciones pero tiene la posibilidad de modificar localmente el comportamiento de estas operaciones

46

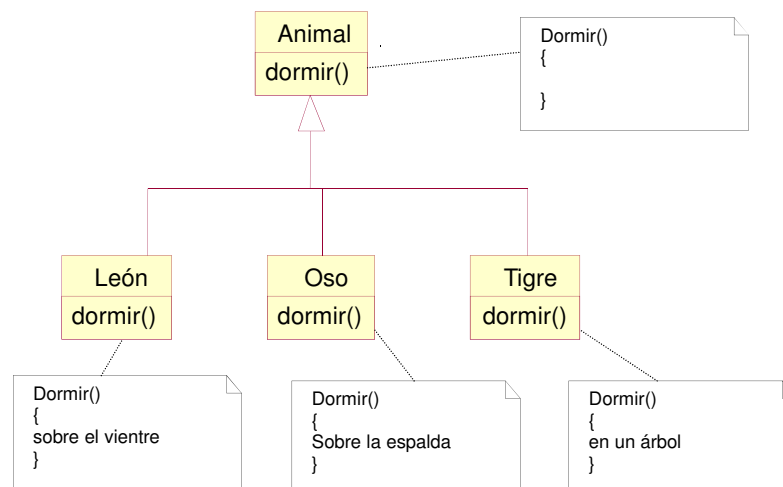
## ... Polimorfismo

- Ejemplo: todo animal duerme, pero cada clase lo hace de forma distinta



47

## ... Polimorfismo



48



## ... Polimorfismo

- La búsqueda automática del código que en cada momento se va a ejecutar es fruto del **enlace dinámico**
- El cumplimiento del Principio de Sustitución permite obtener un comportamiento y diseño coherente

49

## Clases

### Identificación de clases

El objetivo es identificar los conceptos significativos del dominio.

Dos posibles estrategias:

- A partir de una lista de categorías.
- A partir de identificación de frases nominales.

50

## Clases

### Lista de categorías conceptos

| Categoría del concepto                    | Ejemplo                     |
|---|-----------------------------|
| Objetos físicos tangibles                 | Camión                      |
| Especificaciones o descripciones de cosas | Descripción del producto    |
| Lugares                                   | Tienda, Almacén, Delegación |
| Transacciones                             | Venta, Pago, Reserva        |
| Línea o elemento de una transacción       | Línea de una Venta          |
| Papeles de las personas                   | Vendedor, Camionero         |

51

## Clases

### Lista de categorías conceptos

| Categoría del concepto           | Ejemplo                                       |
|----------------------------------|---|
| Contenedores de cosas            | Tienda, Almacén                               |
| Cosas dentro del contenedor      | Producto                                      |
| Otros sistemas software externos | Sistema de autorización de tarjeta de crédito |
| Conceptos abstractos             | Hambre  |
| Organizaciones                   | Dpto de Ventas                                |
| Eventos                          | Pago, Anulación                               |

## Clases

### Lista de categorías conceptos

| Categoría del concepto               | Ejemplo                              |
|--------------------------------------|--------------------------------------|
| Procesos                             | Venta de un producto                 |
| Reglas y políticas                   | Política de reembolso por anulación  |
| Catálogos                            | Catálogo de productos                |
| Documentos, libros                   | Manual de Personal, Ticket de compra |
| Instrumentos y servicios financieros | Existencias, Línea de crédito        |
|                                      |                                      |

53

## Clases

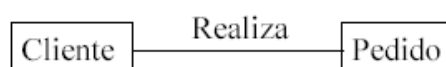
### Identificación de frases nominales

Este método consiste en identificar en las descripciones textuales del dominio nombre o frases nominales y considerarlas como conceptos.

En esta estructura los verbos representan asociaciones entre conceptos.

Ejemplo.-

El *cliente* realiza los *pedidos*



54

---

## Clases

### Errores y problemas en la identificación de clases

- Incorporación de documentos como clases.
  - ❖ Incorporarlos sólo si cumplen un papel especial respecto a las reglas del negocio (ejemplo.- un recibo de compra puede ser necesario para realizar una devolución).
- Distinción entre atributo y clase.
  - ❖ Si el concepto identificado no se describe mediante un simple número o texto descriptivo, posiblemente sea una clase.

55

## Asociación

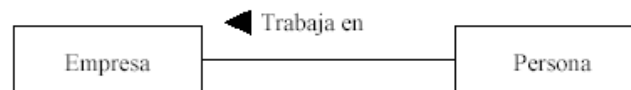
- Especifica que los objetos de una clase están conectados con objetos de otra clase.
- Puede verse como:
  - ❖ *“unión o conexión de ideas”*
  - ❖ *“establece las relaciones entre los objetos necesarios para llevar a cabo un conjunto de requerimientos”*
- Pueden incluirse cuatro adornos a la asociación.
  - ❖ **Nombre.**- Se utiliza para describir la naturaleza de la asociación.
  - ❖ **Rol.**- Es el papel específico que juega una clase en dicha relación.
  - ❖ **Multiplicidad.**-Indica cuántos objetos pueden conectarse a través de una instancia de la asociación.
  - ❖ **Agregación.**-Permite modelar relaciones especiales de tipo “todo/parte”.

56

## Asociación

**Nombre.-** describe la relación existente entre las clases .

Para aclarar su significado suele ser interesante indicar la dirección de lectura.



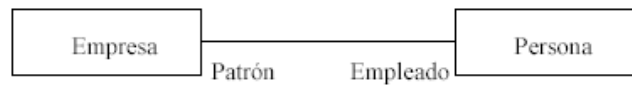
Puede no ser necesario su inclusión si se indican los nombre de los roles.

57

## Asociación

**Rol.-** es la cara que la clase de un extremo de la relación presenta a la clase del otro extremo.

Una clase puede jugar el mismo o diferentes roles en otras asociaciones.



58

## Asociación

**Multiplicidad.-** indica el número de objetos que puede participar en una instancia de la relación.

En una relación se indican tantas multiplicidades como clases participen en la asociación.

En la multiplicidad se indican los límites inferior y superior de los objetos participantes.

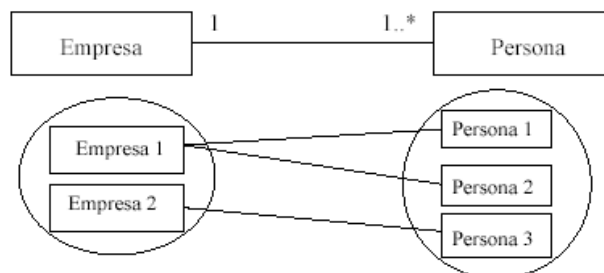
- ❖ 1 -> exactamente 1
- ❖ 0,1 -> cero o uno
- ❖ 0..4 -> entre cero y cuatro
- ❖ 3,7 -> tres o siete
- ❖ 0..\* -> mayor o igual de cero (por defecto)
- ❖ 1..\* -> mayor o igual a uno
- ❖ 0..3, 7, 9..\* -> cualquier número menos 4, 5, 6 y 8

59

## Asociación

### Multiplicidad.-

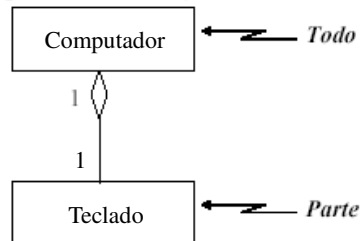
Cuando se indica una multiplicidad en un extremo de la asociación, se está especificando que, para cada objeto de la clase en el extremo opuesto, debe haber tantos objetos en este extremo



60

## Asociación

**Agregación.-** describe asociaciones en las que existe una jerarquía de composición en la que una clase representa el todo y otras las partes que lo constituyen.



La existencia de una agregación no liga la existencia del todo y sus partes.

61

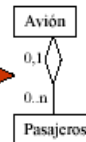
## Asociación

**Agregaciones típicas.-**

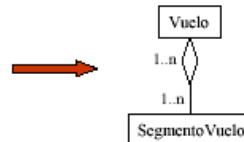
❖ Partes que componen un objeto de nivel superior



❖ Elementos contenidos en otro nivel superior



❖ Miembros de una colección o conjunto.

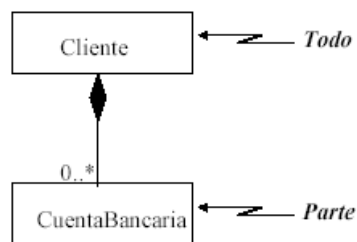


62

## Asociación

**Composición.**- es un tipo especial de agregación en la que la existencia de las partes está ligada a la del todo.

El objeto parte puede pertenecer a un todo único, es más se espera que las parte vivan y mueran con el todo.



63

## Asociación

### Lista de categorías asociaciones

| Categoría de asociación                            | Ejemplo                             |
|--|-------------------------------------|
| A es una parte física de B                         | Ala-Avión;                          |
| A es una parte lógica de B                         | TramoVuelo-RutaVuelo                |
| A está físicamente contenido en B                  | Producto-Estante;<br>Pasajero-Avión |
| A está contenido lógicamente en B                  | Producto-Catálogo                   |
| A es una descripción de B                          | DescripciónProducto-Producto        |
| A es un elemento de una línea en una transacción B | LíneaPedido-Pedido                  |

64



## Asociación

### Lista de categorías asociaciones

| Categoría de asociación                                  | Ejemplo  |
|--|--|
| A se conoce/introduce/<br>registra/presenta/captura en B | Reserva-ListaPasajeros;<br>Venta-Caja            |
| A es miembro de B  | Piloto-Avión;<br>Vendedor-Tienda                 |
| A es una subunidad<br>organizacional de B                | Departamento-Tienda;<br>Mantenimiento-LíneaAérea |
| A usa o dirige a B                                       | Piloto-Avión                                     |
| A se comunica con B                                      | Cliente-Vendedor;<br>AgenteReserva-Pasajero      |
| A se relaciona con una<br>transacción B                  | Pago Pedido;<br>Pasajero-Billete                 |

65

## Asociación

### Lista de categorías asociaciones

| Categoría de asociación                                    | Ejemplo                            |
|--|------------------------------------|
| A es una transacción relacionada<br>con otra transacción B | Pago Venta;<br>Reserva-Cancelación |
| A está contiguo a B  | Ciudad-Ciudad                      |
| A es propiedad de B  | Avión-LíneaAérea;                  |

66

---

## ... Generalización/Especialización

### ¿Cuándo deberíamos *definir un subtipo*? (Especialización)

- Debemos asegurarnos que esta partición es útil en el dominio del problema.
- Motivos para realizar la partición:
  - ❖ El subtipo tiene otros atributos.
  - ❖ El subtipo tiene otras asociaciones.
  - ❖ Se puede especificar un comportamiento específico para el subtipo o se reacciona ante él de manera diferente a como se haría ante el supertipo.

67

## ... Generalización/Especialización

### ¿Cuándo deberíamos *definir un supertipo*? (Generalización)

- Motivos para realizar la generalización:
  - ❖ Los conceptos asociados a subtipos potenciales representan variaciones de un concepto semejante.
  - ❖ Los conceptos comparten entre sí varios atributos o comportamientos semejantes.
  - ❖ Existen relaciones compartidas por los conceptos candidatos a subtipos que pueden generalizarse y asociarse al supertipo.

68