



Análisis y Diseño de Sistemas 2

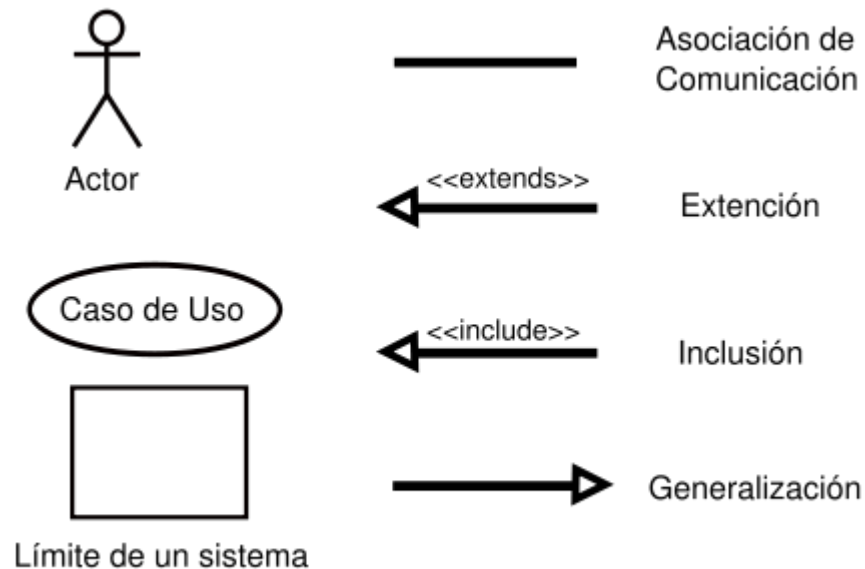
2015

Ing. Luis Alberto Arias Solórzano

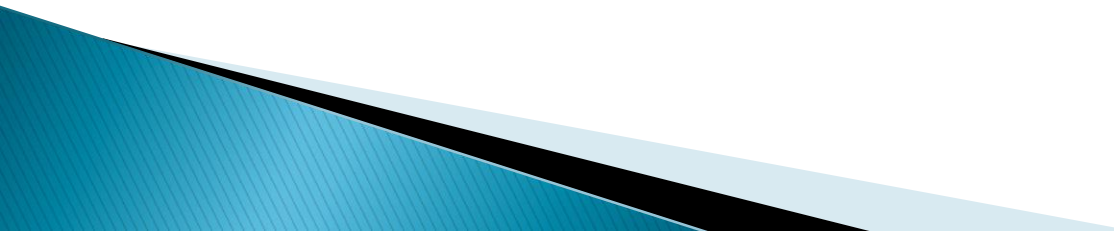
Unidad 3

Repaso – Casos de uso

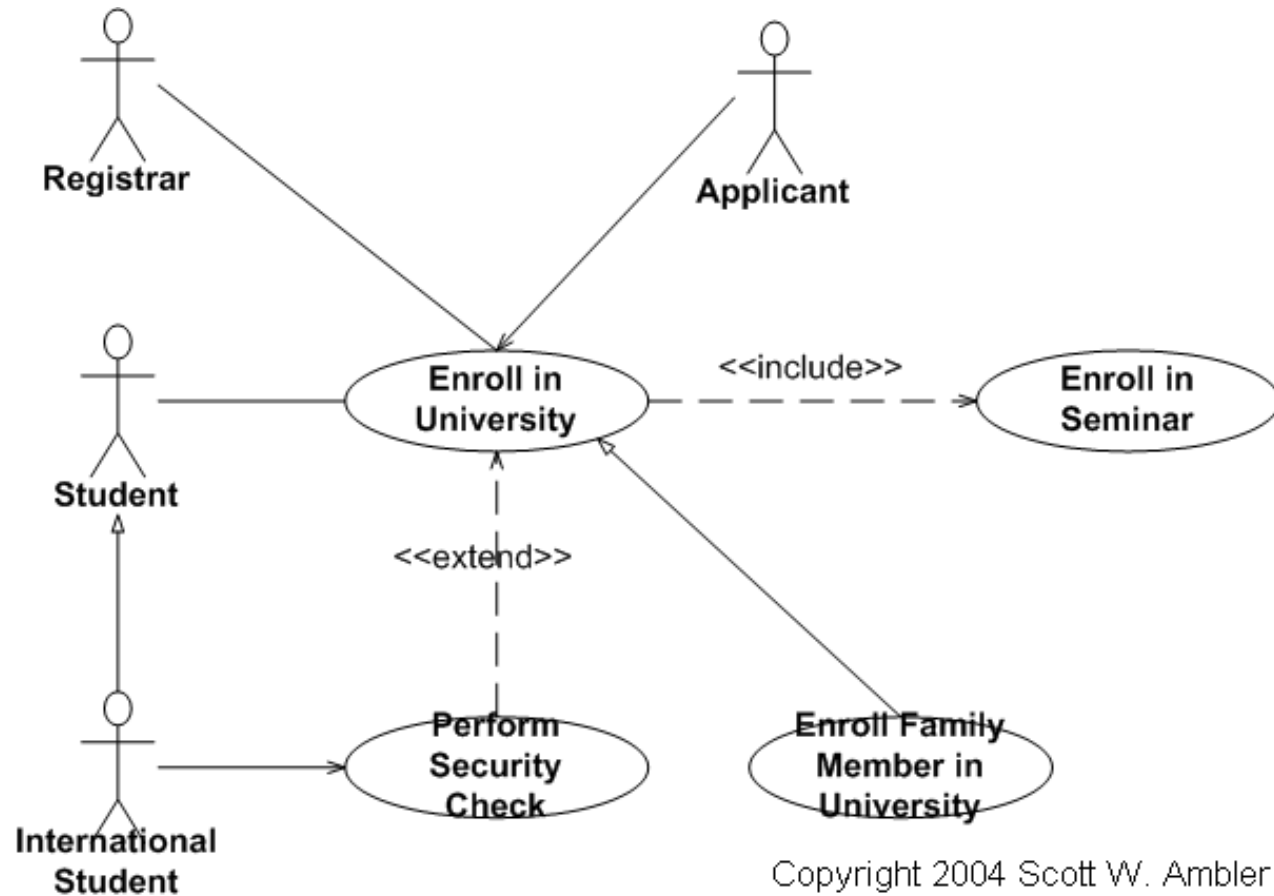
- Técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.



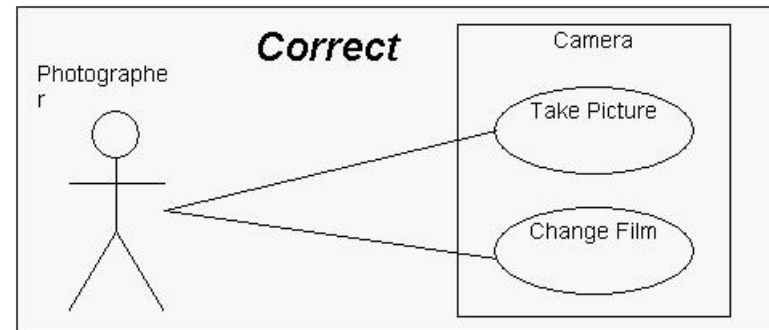
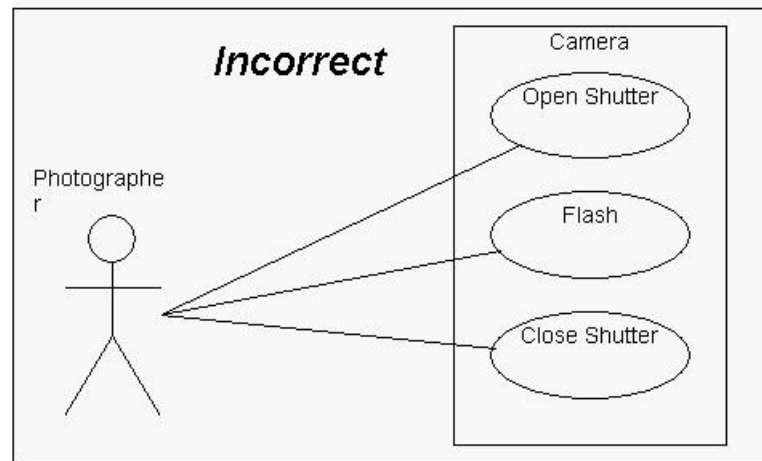
Repaso – Casos de uso

- ▶ **Caso de uso:** Describe una secuencia de acciones que provee algun valor medible para uno o varios actores
 - ▶ **Actor:** Persona, organizacion o sistema externo que juega un rol en una o mas interacciones con el sistema.
 - ▶ **Asociaciones:** Una asociacion existe siempre que un actor este involucrado con una interaccion descrita por un caso de uso.
- 

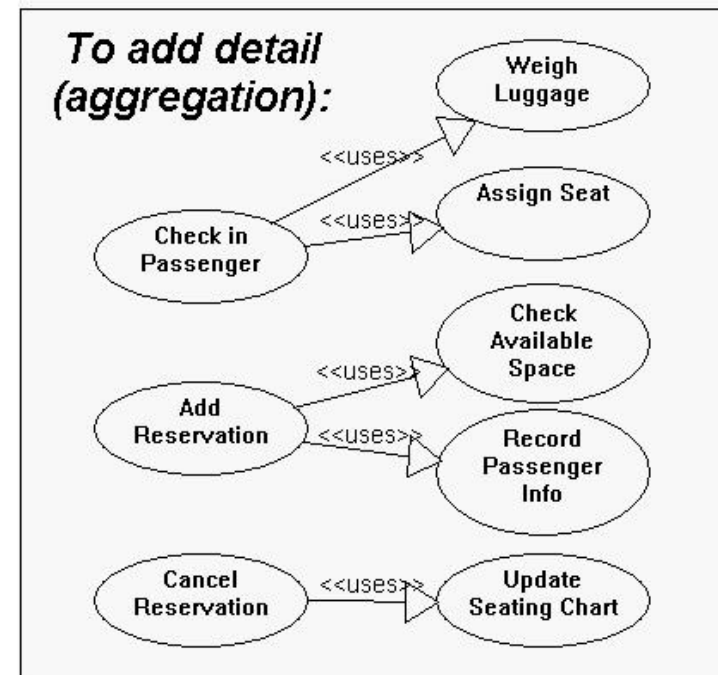
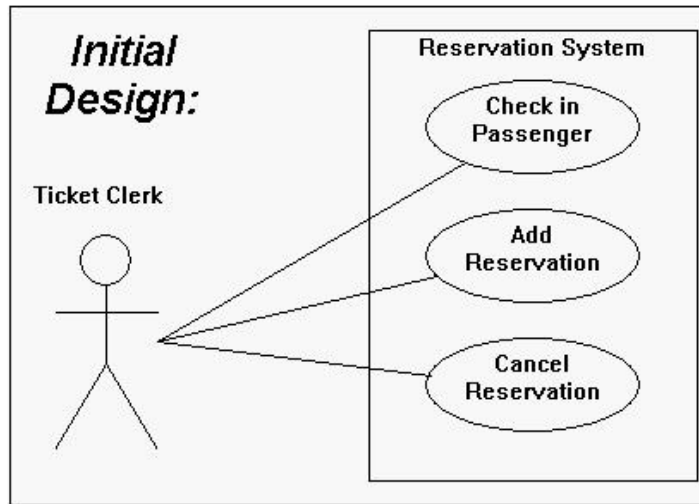
Repaso – Casos de uso



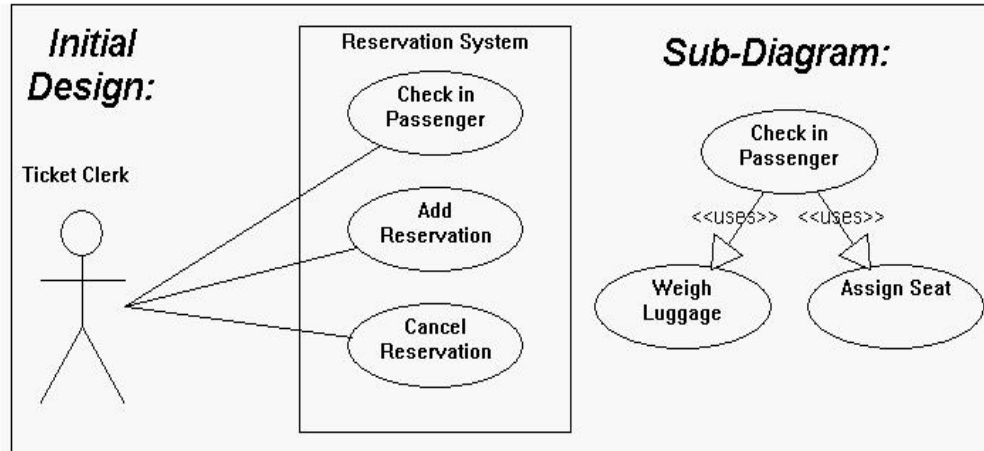
Repaso – Casos de uso



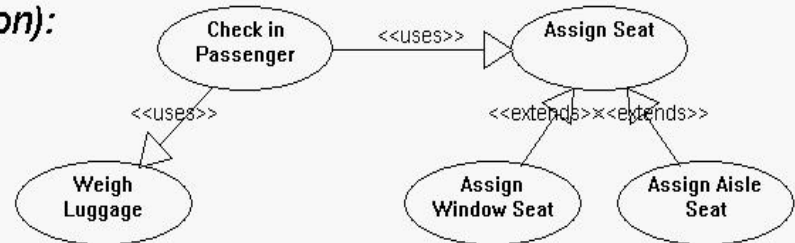
Repaso – Casos de uso



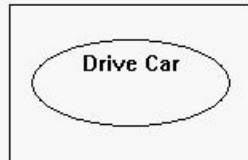
Repaso – Casos de uso



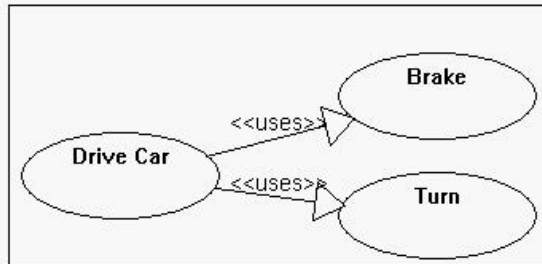
To add detail (extension):



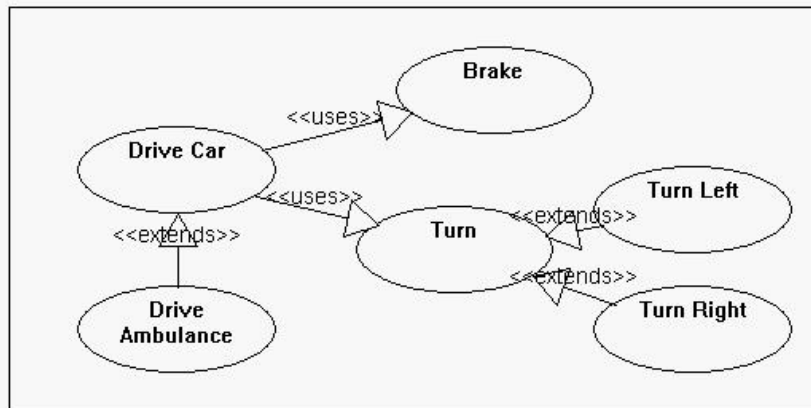
Repaso – Casos de uso



... Becomes ...



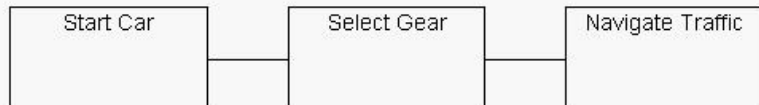
... Which Becomes ...



Evolution of a Traditional Flowchart Diagram



... Becomes ...



... Which Becomes ...



Repaso – Diagrama de clases

- ▶ Representa la **vista estática** de un sistema de software
- ▶ Los elementos que aparecen en éste son aquellos **conceptos que tienen significado** dentro de una aplicación
- ▶ Elementos principales del diagrama:
 - **Clasificadores:** Elementos que describen cosas
 - **Relaciones** entre clasificadores
- ▶ La identificación de cada **concepto del mundo real** se identifica con una **clase** de este diagrama.

Repaso – Diagrama de clases

■ Clases

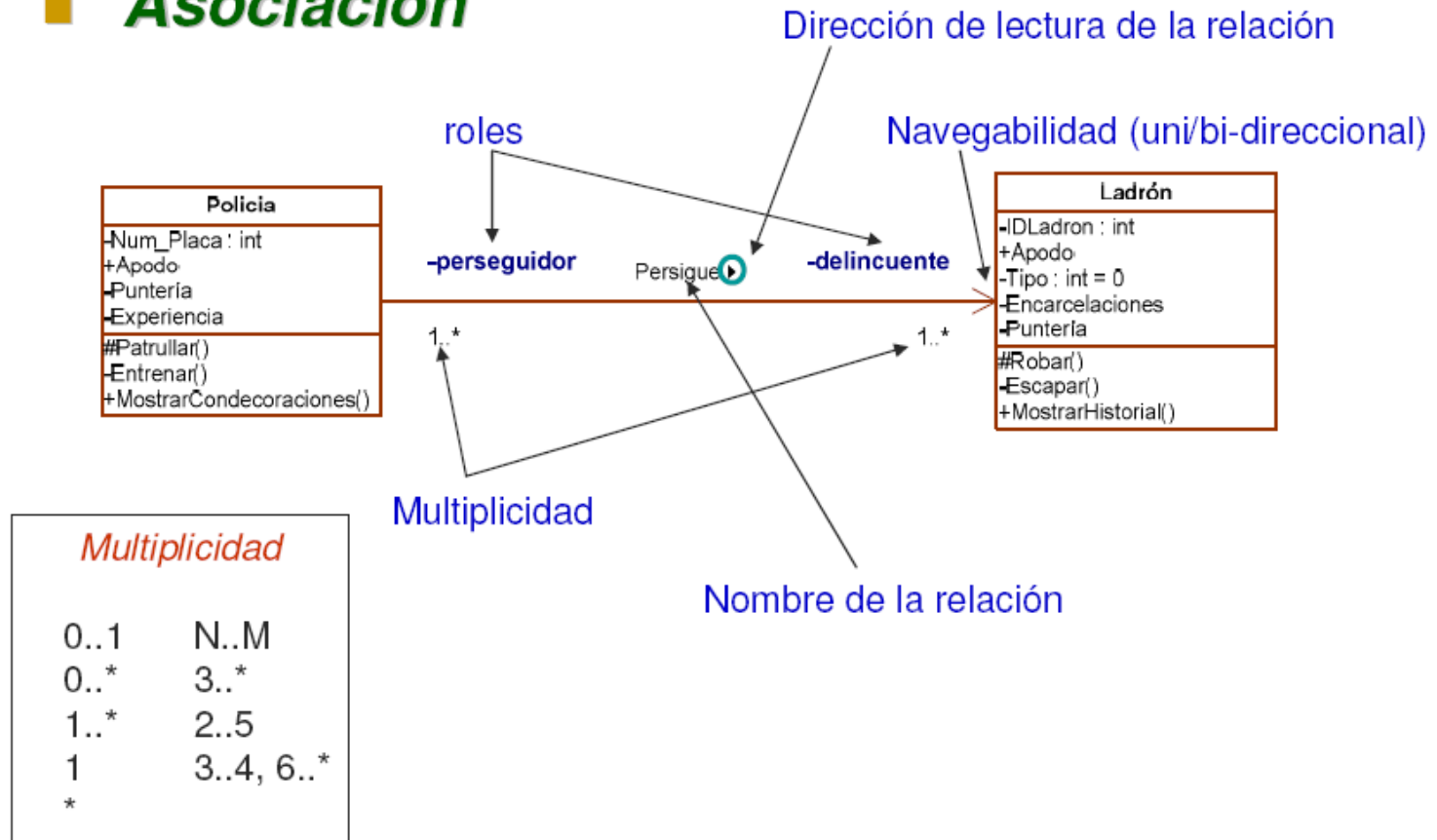


Visibilidad

+	público
#	protegido
-	privado

Repaso – Relación entre clases

■ Asociación



Repaso – Relación entre clases

- **Agregación:** relación entre un todo y sus partes
 - **Lógica:** la parte puede pertenecer a varios agregados

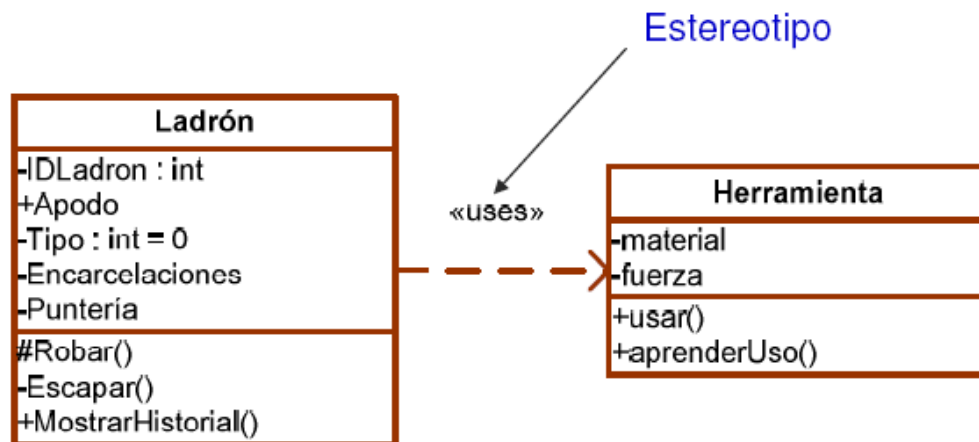


- **Física (o composición):** las partes sólo existen asociadas al compuesto (acceso a través de él)



Repaso – Relación entre clases

- **Dependencia:** relación entre cliente y un servidor (equivalente a una relación de uso)

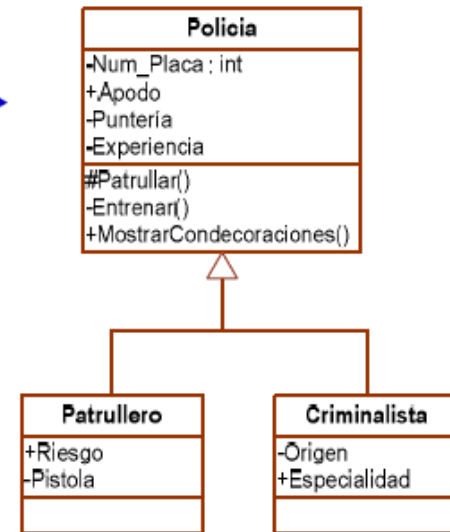
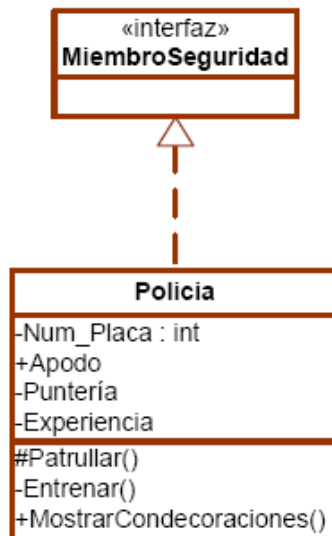
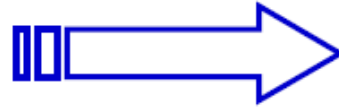


Mecanismos de extensión de UML

- Estereotipos <<excepción>>
- Valores etiquetados {versión=3.1}
- Restricción {edad>18}

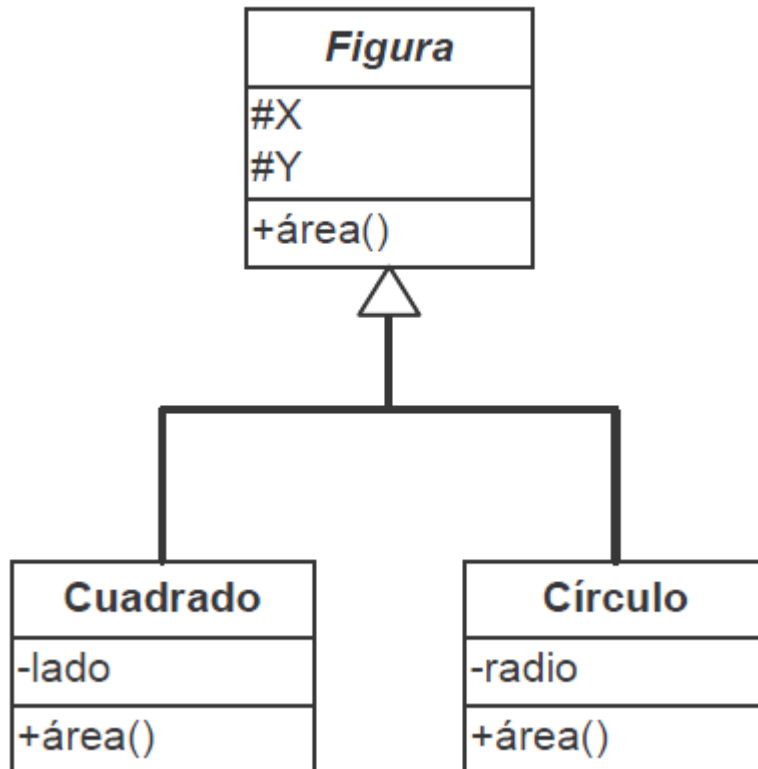
Repaso – Relación entre clases

**Herencia
(generalización)**



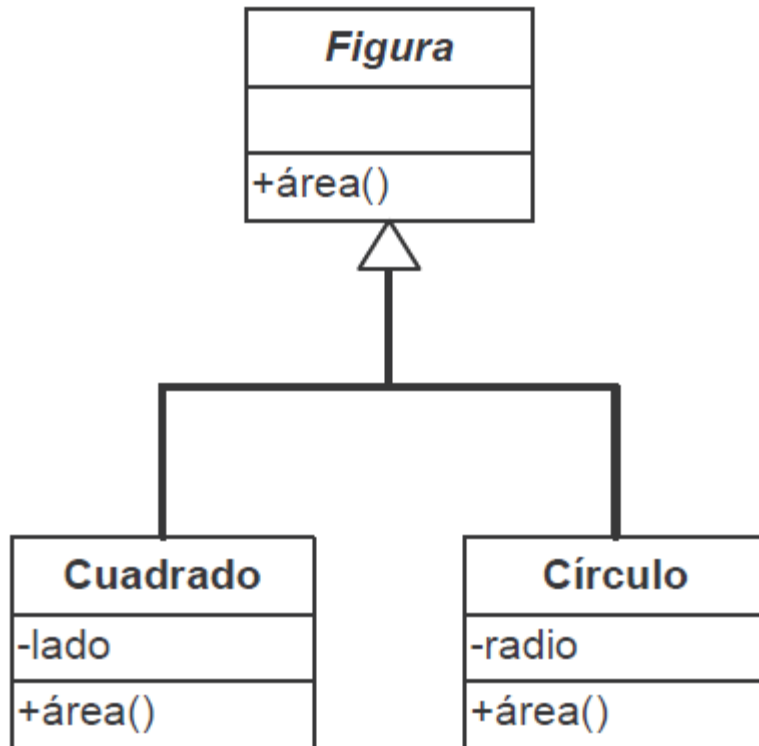
Realización

Repaso – Clase Abstracta



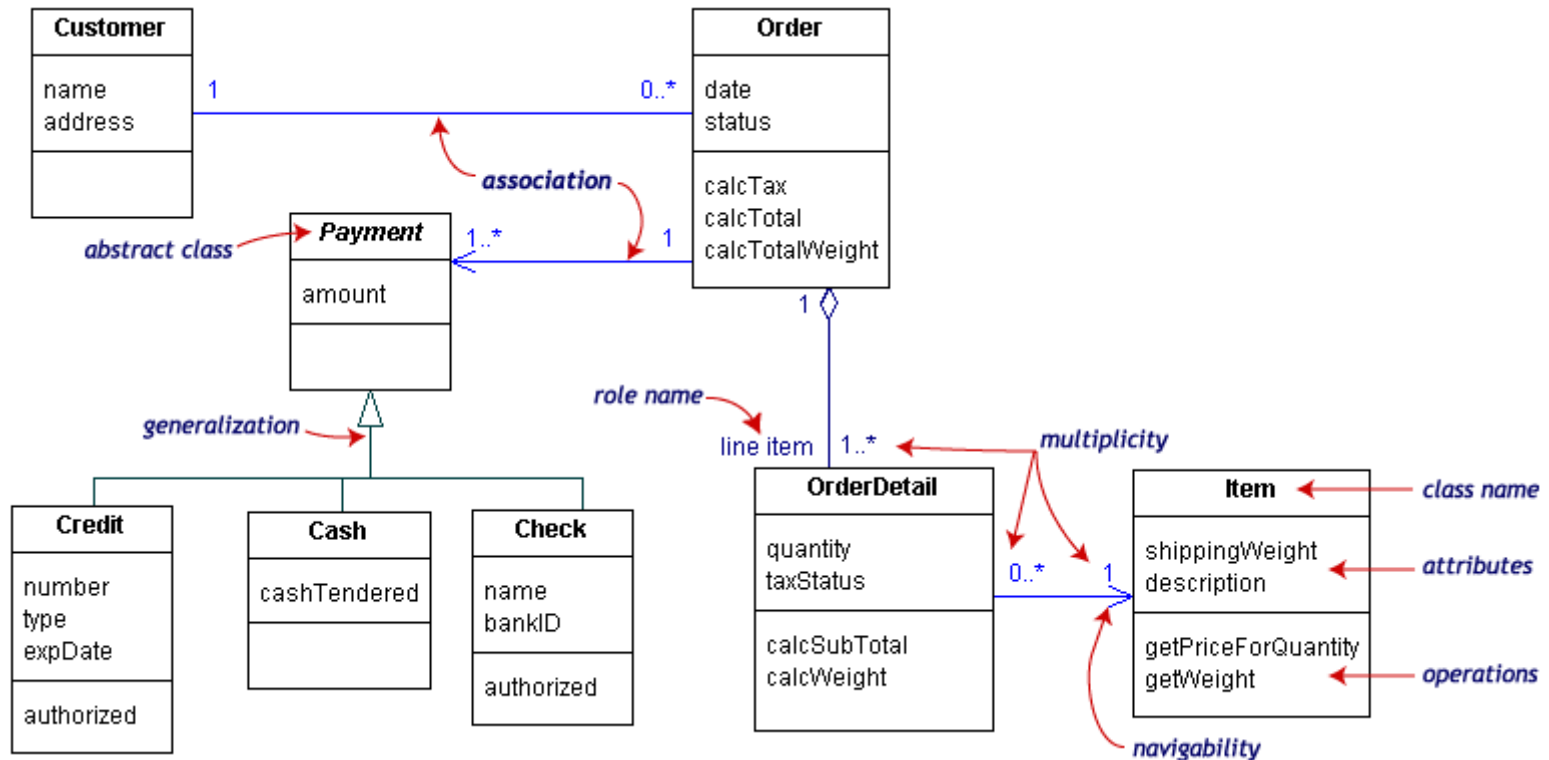
- Es una clase que no se puede instanciar
- Se usa unicamente para definir subclases
- Una clase es abstracta en cuanto uno de sus metodos no tiene implementacion
- Las utilizamos cuando deseamos definir una abstraccion que englobe objetos de distintos tipos y queremos hacer uso del polimorfismo

Repaso – Interfaces



- Una interfaz es una clase completamente abstracta (una clase sin implementación)
- Una interfaz no encapsula datos, solo define cuales son los métodos que han de implementar los objetos de aquellas clases que implementen la interfaz

Repaso – Diagrama de Clases



Análisis vs Diseño

Analysis

Order
Placement Date Delivery Date Order Number
Calculate Total Calculate Taxes

Design

Order
- deliveryDate: Date - orderNumber: int - placementDate: Date - taxes: Currency - total: Currency
calculateTaxes(Country, State): Currency # calculateTotal(): Currency getTaxEngine() {visibility=implementation}

Interacción de los Objetos – Responsabilidades

Durante el diseño de clases se asignan **responsabilidades**

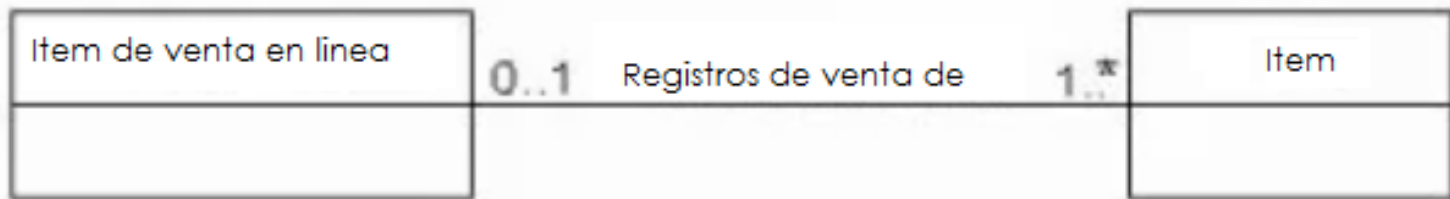
- Crear otros objetos

- Realizar cálculos

- Iniciar acciones en otros objetos

- Controlar y coordinar actividades en otros objetos

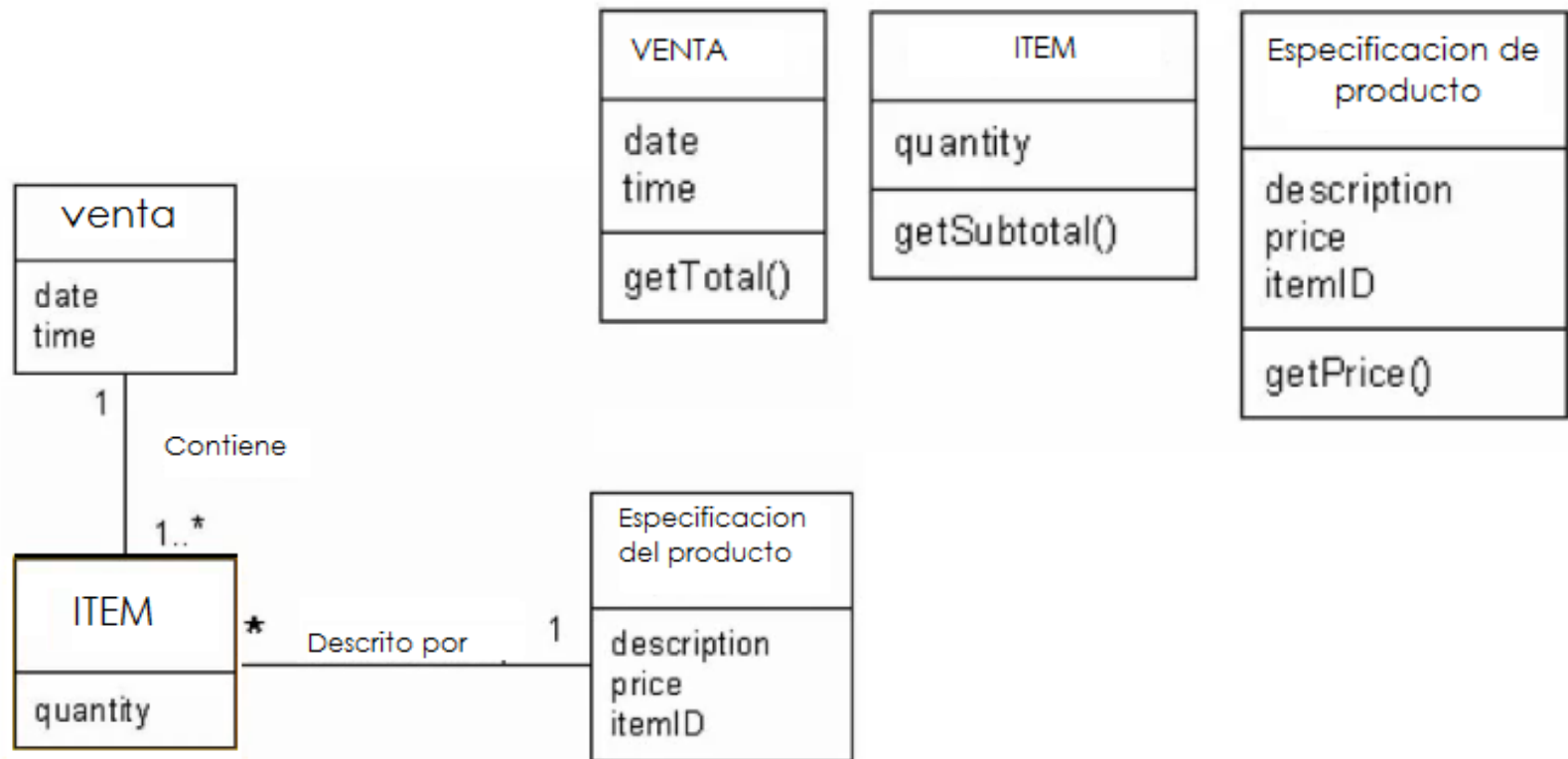
Ejemplo: Las VENTAS son responsables de saber la cantidad total de –Ítem De Venta En Línea–



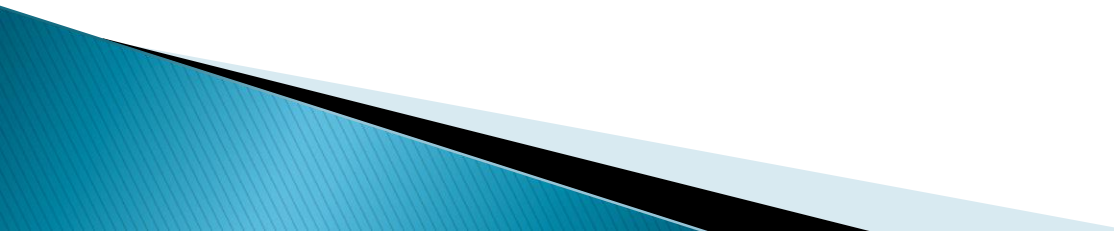
Interacción de los Objetos – Responsabilidades

- ▶ La traducción de las responsabilidades en clases y metodos esta influenciada por:
 - La granularidad de la responsabilidad
 - La responsabilidad NO es un metodo, los metodos cumplen las responsabilidades
 - Las responsabilidades son implementadas usando metodos que funcionan solos o colaboran con otros metodos y objetos
- ▶ Ej. La clase venta puede definir uno o mas metodos para hacer un seguimiento del total de ventas

Interacción de los Objetos – Responsabilidades



Interacción de los Objetos – Debilmente acoplado

- ▶ El acoplamiento es una medida de:
 - Que tan fuertemente esta un elemento conectado a...
 - Tiene el conocimiento de...
 - Depende de ...
 - ▶ Un elemento debilmente acoplado NO depende de muchos otros elementos
 - ▶ Un elemento fuertemente acoplado depende de otras clases, esto ocasiona:
 - Cambios dentro de clases relacionadas forzan cambios locales
 - Las clases son dificiles de entender y dificiles de reusar
- 

Interacción de los Objetos – Alta Cohesion

- ▶ Cohesion funcional es un metodo para medir *que tan fuertemente relacionadas y enfocadas* son las responsabilidades de un elemento
- ▶ Un elemento con baja cohesion posee muchas cosas no-relacionadas; hace mucho trabajo, problemas:
 - Dificil de comprender
 - Dificil de reusar
 - Dificil de mantener
 - Muy delicada; afectada constantemente por el cambio
 - Contiene responsabilidades que debieron **delegarse** a otros objetos

Interacción de los Objetos – Diseño modular

- ▶ Acoplamiento y cohesion
 - Clases altamente cohesivas
 - Modulos debilmente acoplados
 - Metodos con un solo y claro proposito
 - Agrupamiento de un conjunto relacionado de preocupaciones

Gracias