



**Universidad Nacional Autónoma de México**  
**Facultad de Ingeniería**  
**Programación orientada a objetos**

**Tema 4:**  
**Lenguaje de modelado unificado**

## 4 Lenguaje de modelado unificado

**Objetivo:** Clasificar las diferentes vistas en el diseño orientado a objetos para aplicarlo en la solución de problemas.

# **4 Lenguaje de modelado unificado**

**4.1 Diseño estático.**

**4.2 Diseño dinámico.**

## Bibliografía



***Aprendiendo UML En 24 Horas. Joseph Schmuller, ISBN:  
968444463X, Prentice-Hall, noviembre 2001***

# **4 LENGUAJE DE MODELADO UNIFICADO**



**La ingeniería de software se define como el uso y establecimiento de principios de ingeniería sólidos, a fin de obtener un software que sea económicamente fiable y funcione eficientemente en máquinas reales.**

**La ingeniería de software provee métodos que indican cómo generar software. Estos métodos abarcan una amplia gama de tareas:**

- **Planeación y estimación del proyecto.**
- **Análisis de requerimientos del sistema y software**
- **Diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico**
- **Codificación**
- **Pruebas y mantenimiento (validación y verificación).**

**Las soluciones de ingeniería de software a menudo presentan un lenguaje orientado especial o una notación gráfica e introducen un conjunto de criterios para la calidad de software.**

## **Ciclo de vida del software**

**La ISO (International Organization for Standardization) en su norma 12207 define al ciclo de vida de un software como:**

**Un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso.**

**El ciclo de vida del software está constituido por las fases de análisis, diseño, implementación e instalación.**

**La calidad de software que se genera se mide en dos grandes rubros:**

- **Factores de calidad externos:** Correcto, sólido o robusto, confiable, extensible, reutilizable, compatible, eficiente, portable, fácil de usar.
- **Factores de calidad internos:** Modular y legible.

## ¿Qué es el análisis?

**Es el proceso para averiguar qué es lo que requiere un cliente de un sistema de software (análisis de requisitos) y de definir estos requisitos en forma clara y concisa (especificación de requisitos).**

**El productor de software se enfrenta al principio de un proyecto con un documento escrito por el cliente, en el cual expresa, en términos de la aplicación, qué se requiere del sistema. Este documento se conoce como declaración de requisitos.**

**La especificación del sistema debe contener una declaración de las funciones del sistema así como las restricciones con las cuales tendrá que trabajar el productor de software.**

**También debe contener información adicional como los detalles del hardware que se usará y la capacitación que tendrá que proporcionar el productor, así como una especificación de las herramientas de software especiales que se usarán en el proyecto.**

## **¿Qué es el diseño?**

**Se refiere al manejo de distintas técnicas y principios con el propósito de definir a detalle un producto para poder realizarlo físicamente.**

**El diseño representa el más alto nivel de abstracción y se puede seguir hasta requisitos más específicos funcionales, de datos o de comportamiento, es decir, permite generar una representación técnica del software a desarrollar.**

**El diseño debe ser una guía entendible tanto para los desarrolladores del código, como para los que se encargan de realizar pruebas, así como para los encargados del mantenimiento del software.**

**Por lo tanto, debe proporcionar una idea completa de la funcionalidad y comportamiento del software desde el punto de vista de la implementación.**



How the customer explained it



How the Project Leader understood it



How the Analyst designed it



How the Programmer wrote it



How the Business Consultant described it



How the project was documented



What operations installed



How the customer was billed



How it was supported

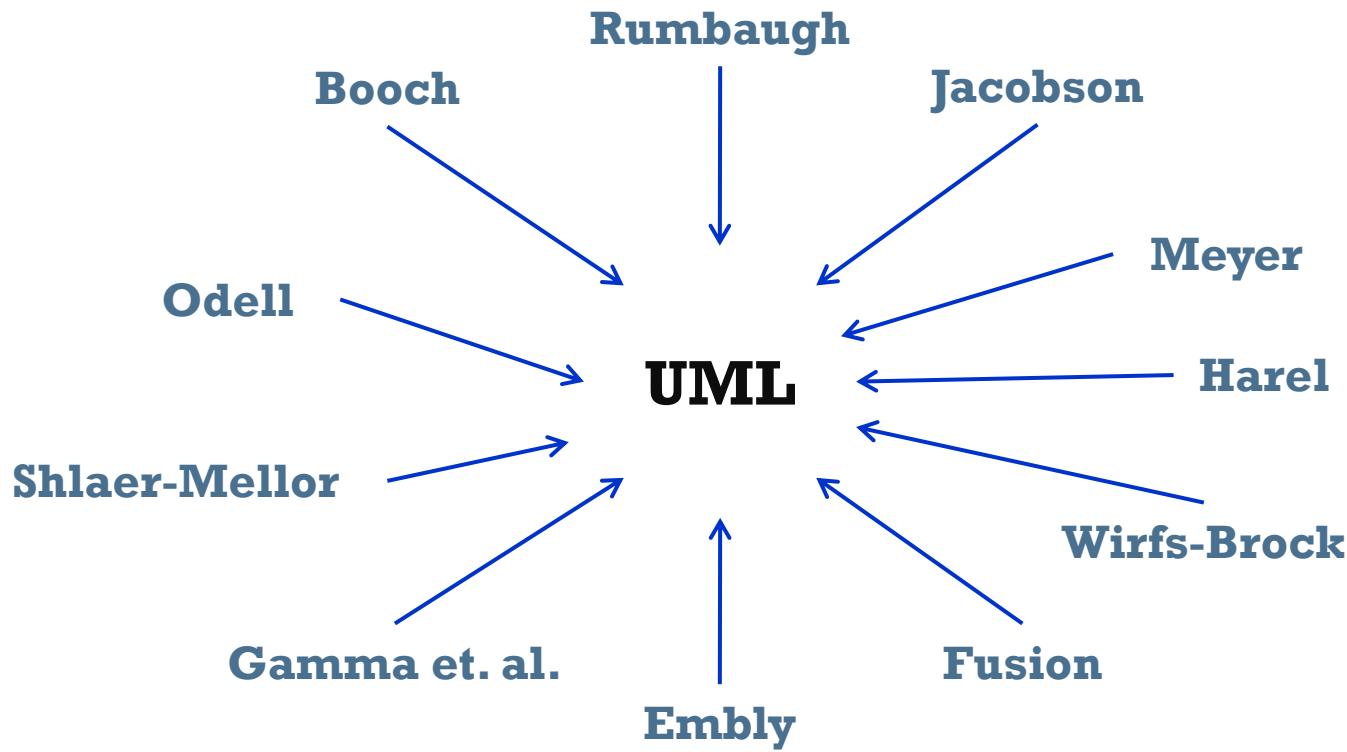


What the customer really needed

## Notación UML



**El Lenguaje de Modelado Unificado (UML - Unified Modeling Language) es un lenguaje gráfico que permite visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software.**



**Los diagramas UML permiten modelar aspectos conceptuales como procesos de negocios o funcionalidades del sistema, cumpliendo con los siguientes objetivos:**

- **Visualizar:** expresa de forma gráfica la solución y/o flujo del proceso o sistema.
- **Especificiar:** muestra las características del sistema.
- **Construir:** generar soluciones de software.
- **Documentar:** especificar la solución implementada.

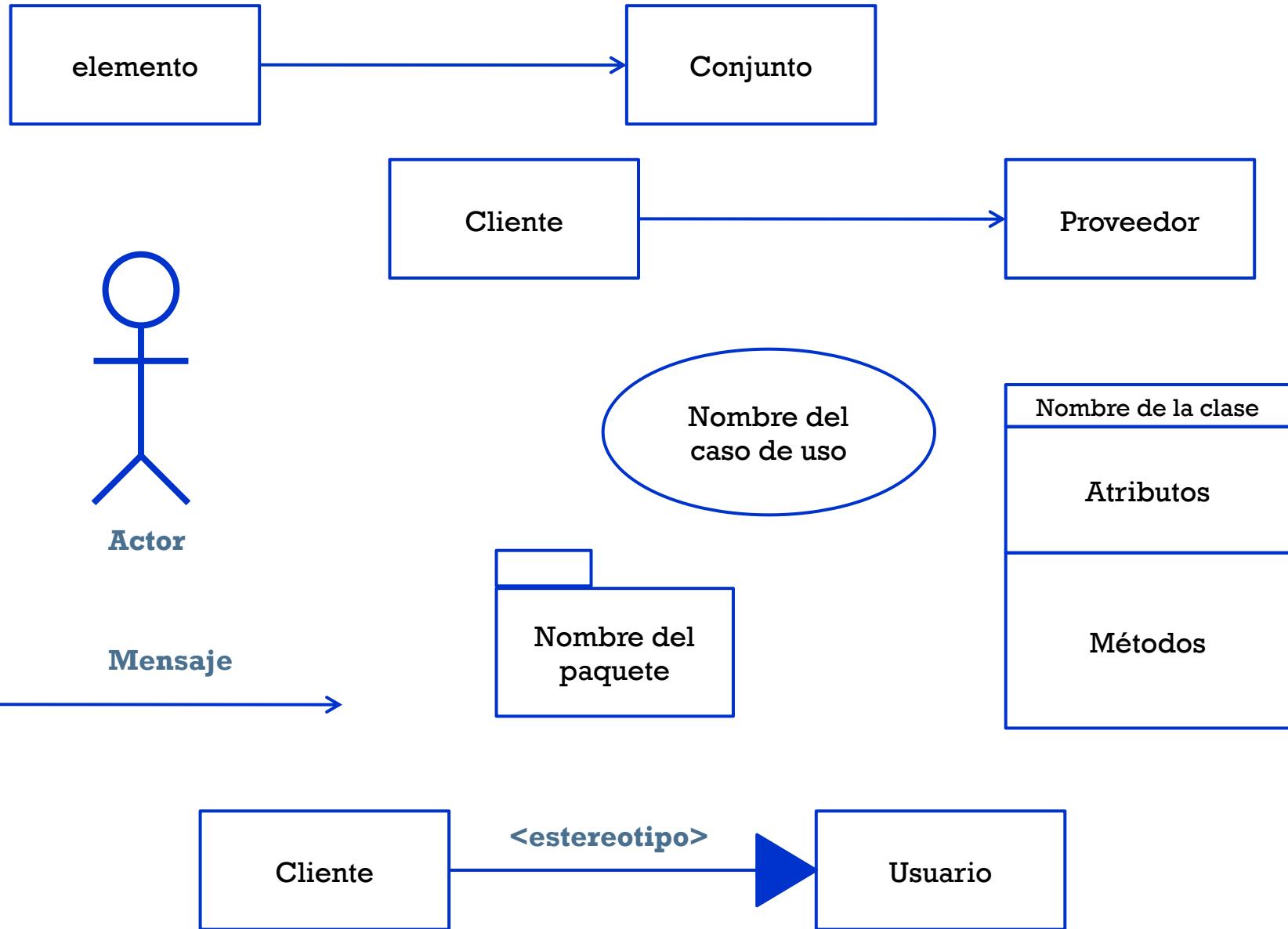
**El lenguaje UML está compuesto por tres bloques generales:**

- **Elementos:** son representaciones de entes reales (usuarios) o abstractos (objetos, acciones, clases, etc.).
- **Relaciones:** es la unión e interacción entre los diferentes elementos.
- **Diagramas:** muestra a todos los elementos con sus relaciones.

**Los elementos describen los componentes que van a interactuar dentro del diagrama, como pueden ser los actores, las clases, los paquetes, etc.**

**Las relaciones permiten unir y/o comunicar los diferentes elementos entre sí.**

**Los elementos y las uniones crean un conjunto de esquemas que pueden dibujar un flujo, una dependencia, una interacción, una comunicación, etc, entre clases. Estos esquemas son llamados diagramas y es el objetivo final de la parte del diseño.**



# DISEÑO ESTÁTICO



**Los diagramas UML estáticos o estructurales aportan una visión fija del sistema.**

**Reflejan el esqueleto básico de un sistema de software.  
Permiten representar los componentes mayores del dominio  
del problema.**

**Los diagramas que permiten modelar estas características son:**

- **Diagrama de casos de uso**
- **Diagrama de clases**
- **Diagrama de objetos**
- **Diagrama de componentes**
- **Diagrama de despliegue**

## Diagramas de casos de uso

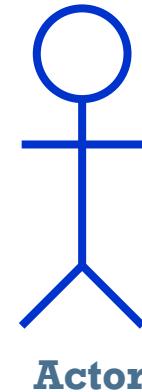
**Un diagrama de caso de uso define la manera en la que el usuario del sistema interactúa con éste.**

**Los diagramas de casos de uso permiten modelar el comportamiento de un sistema, un subsistema o, incluso, una clase.**

**Un diagrama de casos de uso está compuesto por 3 elementos básicos:**

- **Actor(es)**
- **Caso(s) de uso**
- **Relación(es) (uso, herencia y comunicación)**

**El elemento actor permite modelar el rol del usuario en el sistema. Permite conocer las interacciones con los casos de uso.**



**Actor**

Nombre  
del caso  
de uso

**El elemento caso de uso se refiere a una acción, operación o tarea específica que se realiza tras la orden de un agente externo, es decir, la ejecución de la acción puede venir de un actor o de otro caso de uso.**

**Existen diferentes tipos de relaciones para los diagramas de casos de uso:**

- **Asociación.** Indica la invocación desde un actor o caso de uso a otra operación (caso de uso). 
- **Dependencia o instanciación.** Denota la relación entre clases, es decir, cuando una clase instancia otra. 
- **Generalización.** Permite hacer uso o heredar, dependiendo del estereotipo que se especifique. Esta relación está orientada a casos de uso, únicamente. 

## Ejemplo 1

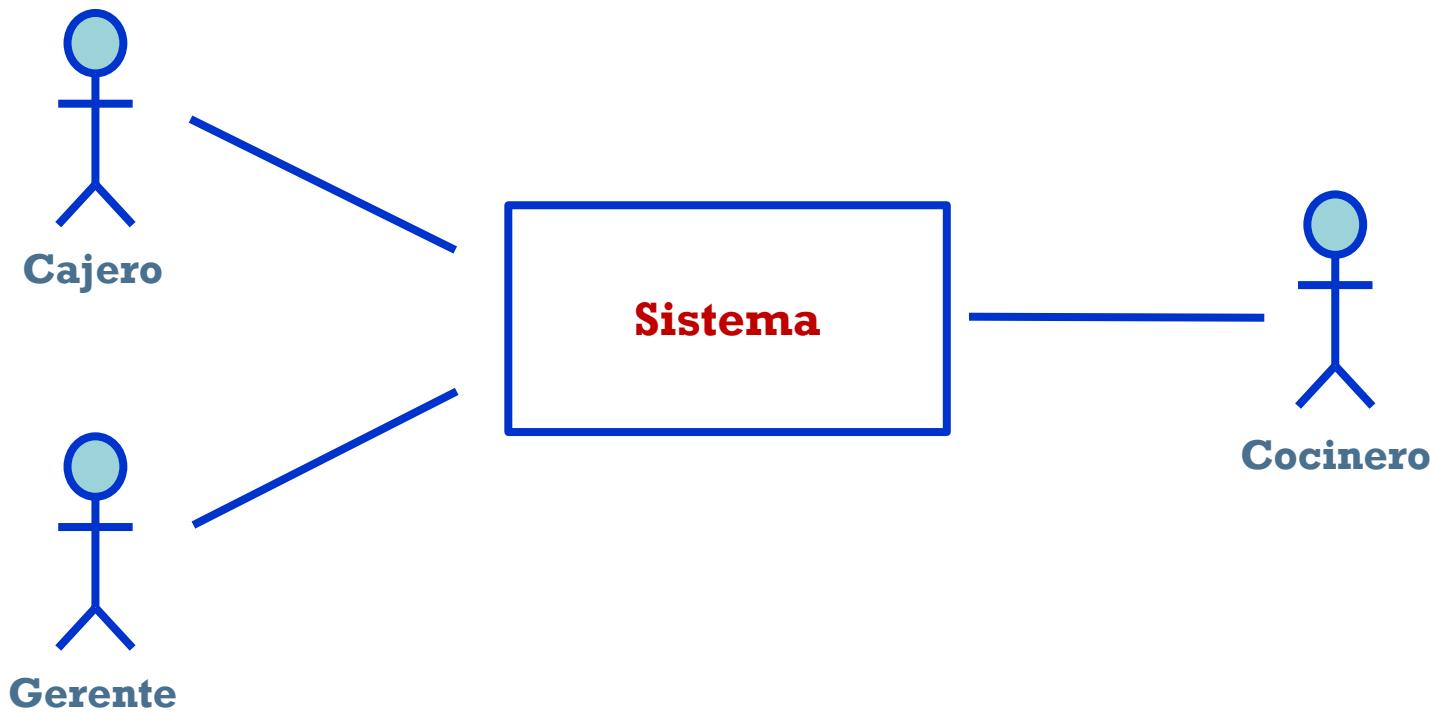
**La comida rápida “El rey de las hamburguesas” tiene un sistema para atender a sus clientes, mediante un proceso repetitivo.**

**Los clientes llegan y pueden ordenar un paquete o cada elemento por separado, el cajero se encarga de tomar la orden. El sistema captura la orden y calcula el monto a pagar por el usuario. En caso de que el cajero se equivoque o el usuario cambie de opinión, el gerente puede cancelar la orden.**

**Una vez confirmado el pedido y realizado el pago, el sistema envía la solicitud a la cocina para la preparación de los alimentos.**

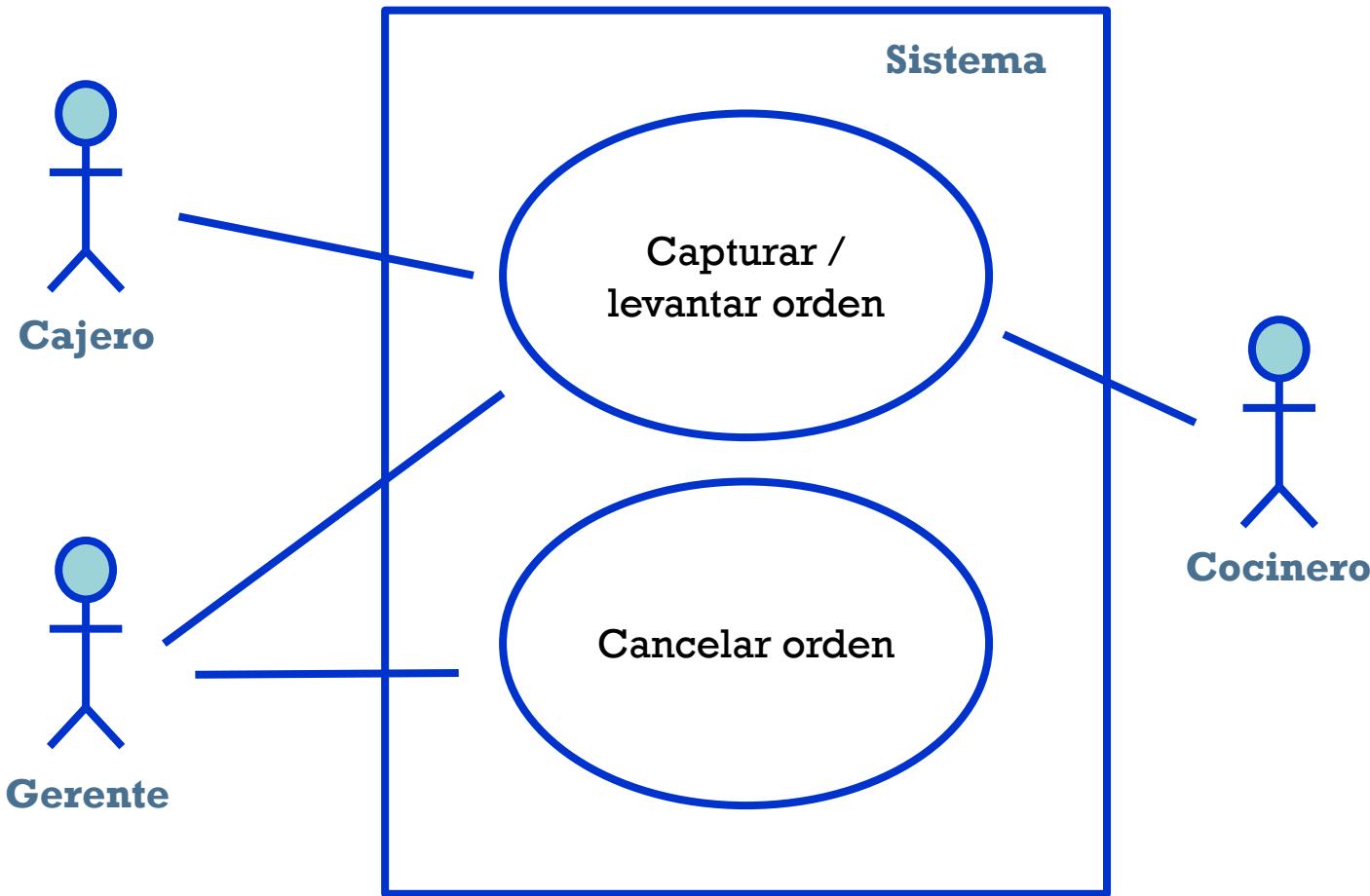
## Ejemplo 1

Para realizar los diagramas de casos de uso del sistema anterior primero se identifican a los actores que interactúan con el sistema



## Ejemplo 1

También es posible generar los diagramas de las acciones que puede realizar cada actor.



## Ejemplo 1

### Escenario Capturar / levantar orden

**Autor:** Jorge A. Solano

**Fecha:**

**Descripción:** Permite capturar una nueva comanda.

**Actor(es):** Cajero, Gerente

#### **Precondiciones:**

**Flujo normal:** El comensal selecciona los elementos que van a componer su orden y se lo hace saber a la persona que lo atiende (cajero o gerente). El encargado ingresa los elementos que conforman la orden en el sistema. Al final de recibir todos los elementos que conforman la orden, el encargado confirma la orden y, de estar correcta, proporciona el monto total del encargo.

Una vez recibido el pago por la orden, se envía una confirmación para que el sistema haga llegar la misma al cocinero.

**Excepciones:** Si el número de elementos que ordena el comensal es mayor al número de elementos que se tiene en bodega, la orden no puede ser surtida.

#### **Post-condiciones:**

## Ejemplo 1

### Escenario Cancelar orden

**Autor:** Jorge A. Solano

**Fecha:**

**Descripción:** Permite cancelar una comanda.

**Actor(es):** Gerente

**Precondiciones:** Se debe tener una comanda capturada en el sistema.

**Flujo normal:** El comensal selecciona un alimento y es agregado a su orden. Posteriormente, el comensal decide cancelar ese alimento de su orden.

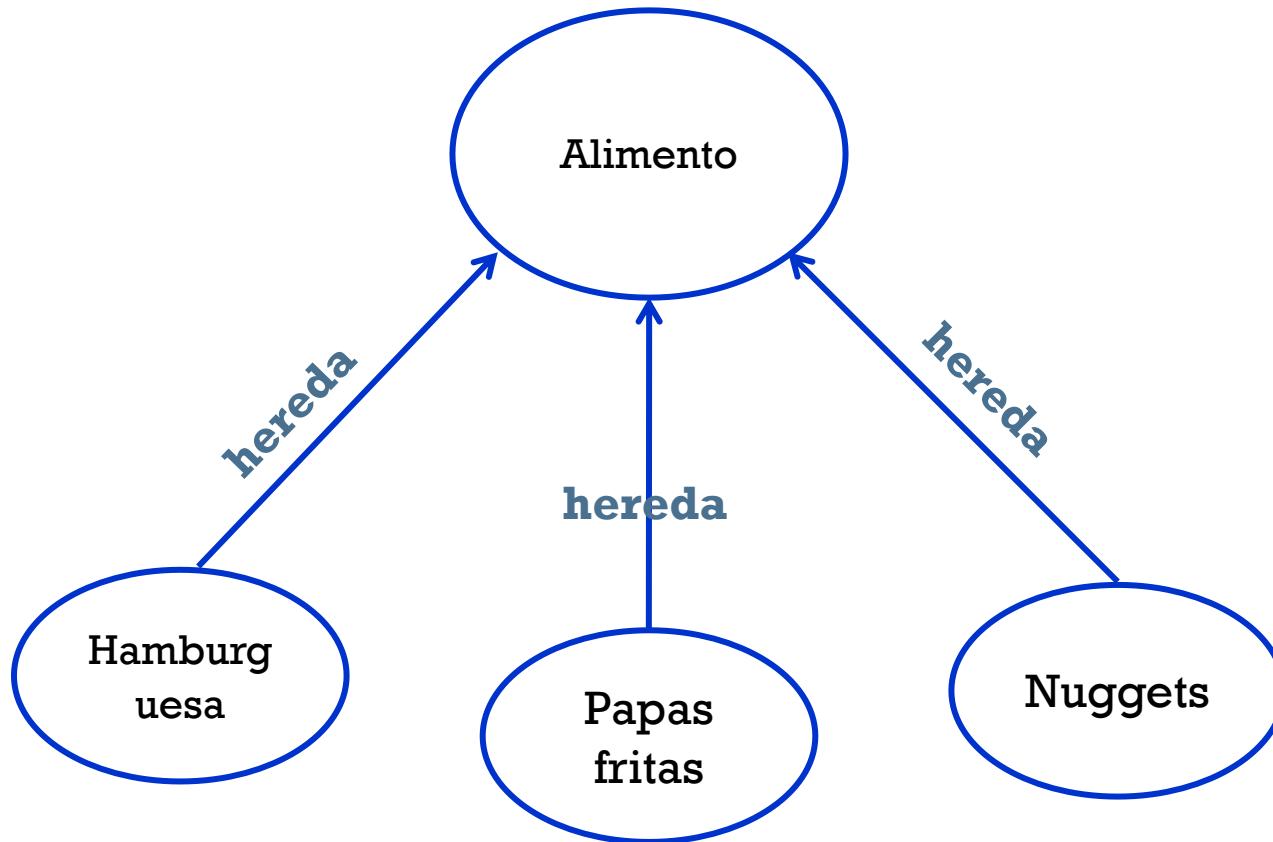
El gerente debe ingresar su número de trabajador para que el sistema valide que la cancelación está autorizada por el mismo.

**Excepciones:** Si la nota ya fue impresa ya no es posible cancelar la orden, debido a que la orden ya fue enviada al cocinero.

**Post-condiciones:**

## Ejemplo 1

Una orden puede constar de diferentes alimentos, los cuales pueden tener características comunes:



## Diagrama de clases

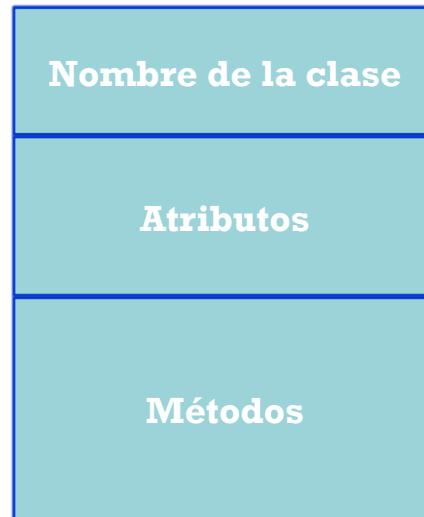
**Un diagrama de clases permite modelar las características de las clases que componen al sistema. Dentro de cada clase se pueden visualizar los atributos y métodos que contiene la clase. El conjunto de clases permite observar las relaciones que existen entre ellas dentro del sistema.**

**Un diagrama de clases está compuesto por 3 elementos básicos:**

- **Clase(s)**
- **Relación(es)**
- **Cardinalidad**

## Clase

La clase es la unidad básica que encapsula toda la información de un objeto y, por ende, permite modelar todos los objetos que se produzcan a partir de esa clase. Su representación gráfica en lenguaje UML es:



**Existen diferentes permisos de acceso tanto a los atributos como a los métodos:**

- **public (+):** el atributo o método es visible para todas las clases, dentro o fuera de el paquete de la clase.
- **private (-):** el atributo o método es visible sólo dentro de la clase donde está declarado.
- **protected (#):** el atributo o método es visible para todas las clases y subclases que pertenecen al mismo paquete donde está declarada la clase.
- **friendly:** indica que el atributo será accesible desde cualquier otra clase dentro de su paquete.

## **Cardinalidad**

**La cardinalidad se refiere al nivel de dependencia que puede existir entre dos clases. Existen 3 tipos diferentes de cardinalidad:**

- **1 ... \*:** La cual representa una unión de uno a muchos, lo que implica que por lo menos hay una relación.
- **0 ... \*:** La cual representa una unión de cero a muchos, lo que implica que puede o no existir relación.
- **n:** La cual representa una unión de un número fijo de elementos.

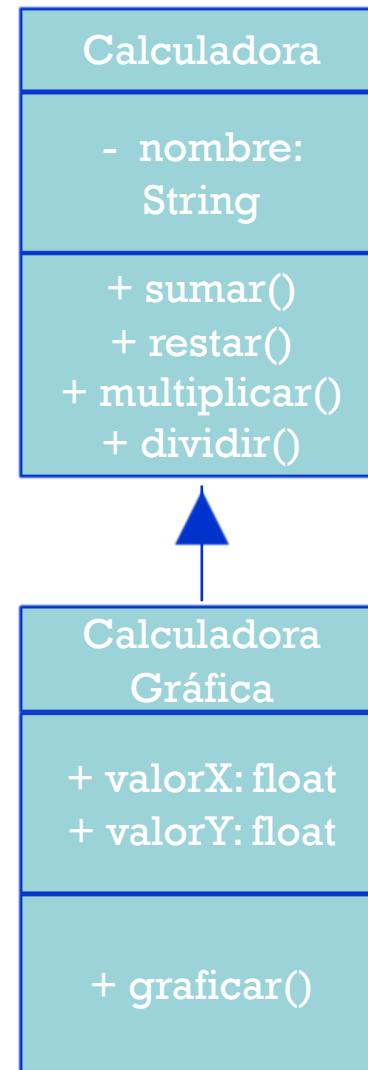
## **Relaciones**

**Dentro de los diagramas de clase existen 4 relaciones básicas entre las clases:**

- **Herencia**
- **Agregación**
- **Asociación**
- **Dependencia o instanciaión.**

## Herencia

**Indica la relación que existe entre la clase base y la súper clase, es decir, los atributos y métodos que hereda la clase base de la súper clase.**

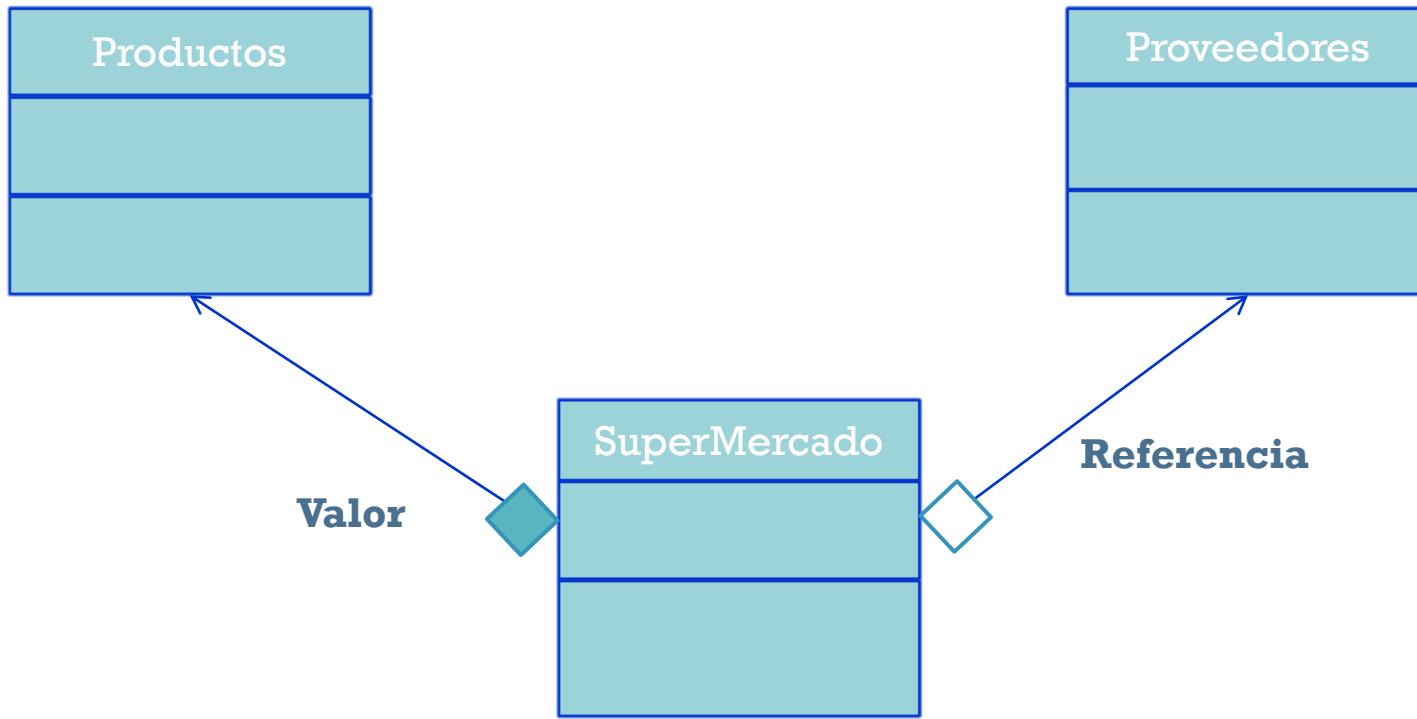


## Agregación

**Permite modelar objetos acoplados, es decir, aquellos tipos de datos que son instancias de otras clases. Se tienen dos tipos de relaciones:**

- **Por valor: o composición, donde el objeto base construye al objeto incluido, siendo ambos parte del todo.**
- **Por referencia: o agregación, donde el objeto base utiliza al objeto incluido en su funcionamiento o desempeño.**

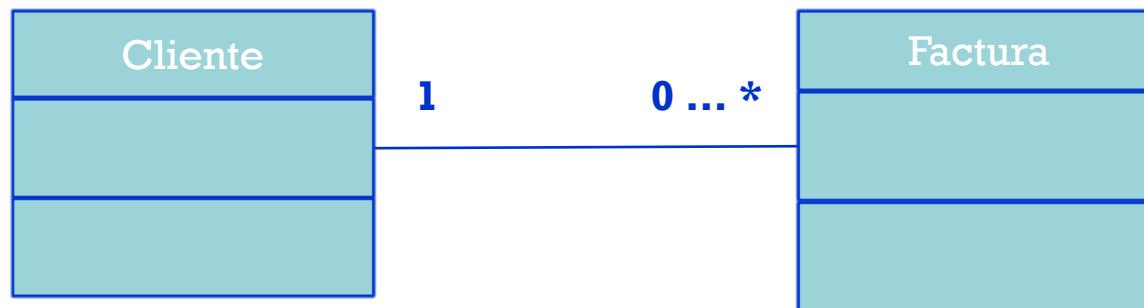
**Un súper mercado posee productos y proveedores.**



**Si el súper mercado quiebra, los productos se eliminan con él, sin embargo, los proveedores siguen trabajando con otros clientes.**

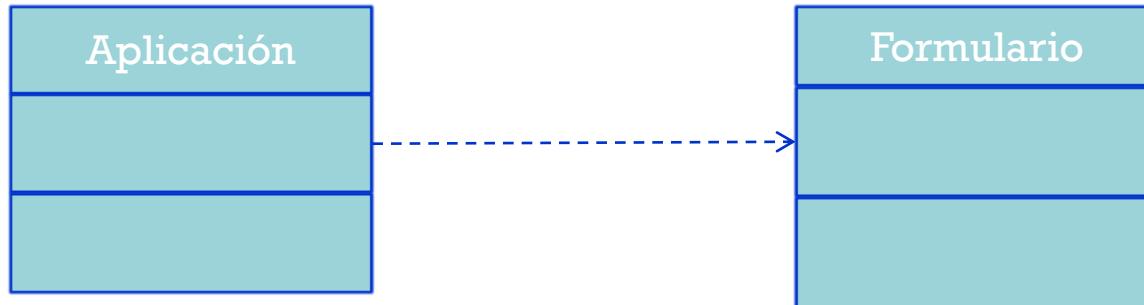
## Asociación

Permite unir objetos que colaboran entre sí. Las uniones permiten establecer la cardinalidad entre las clases.



## Dependencia o instancia

Representa el acoplamiento entre clases, es decir, al crear un objeto se debe crear también el objeto referenciado.

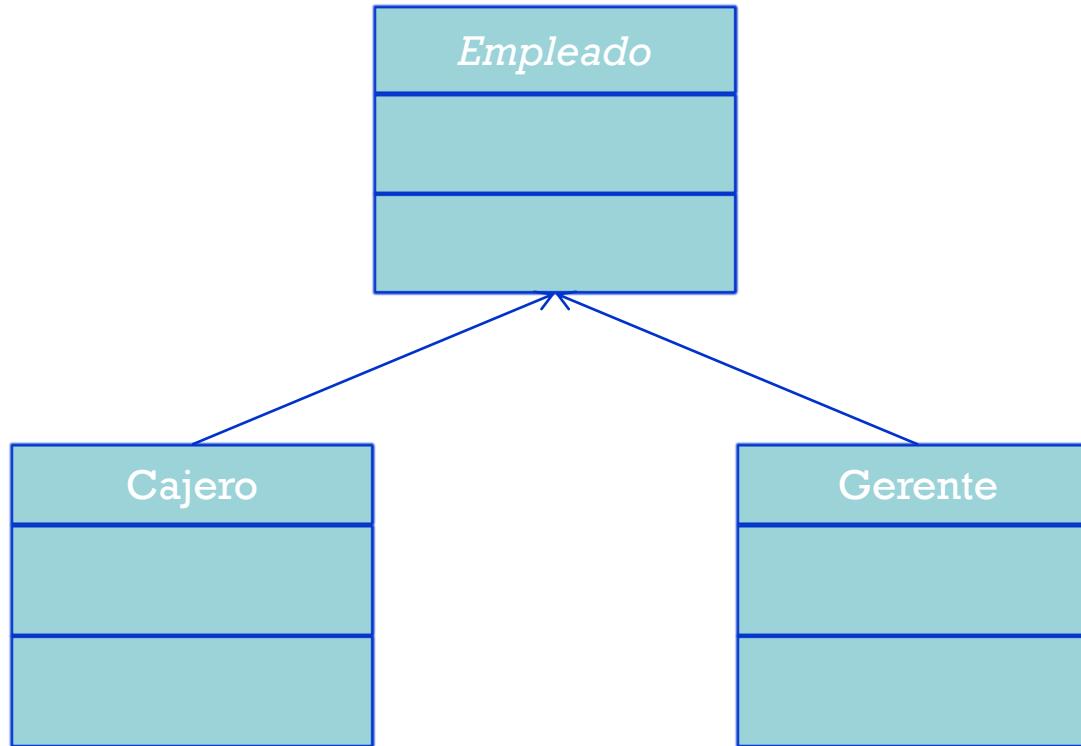


El objeto ventana crea un objeto tipo formulario, pero el formulario no se almacena en el objeto ventana.

## Abstracciones

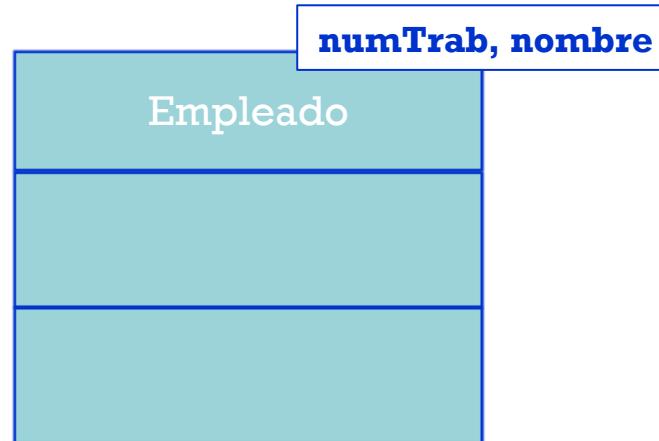
**Permiten definir atributos y métodos comunes a cierto tipo de entes para, posteriormente, implementarlos y poder definirlos según cada ente quiera.**

**El nombre de las abstracciones se escribe en cursiva.**



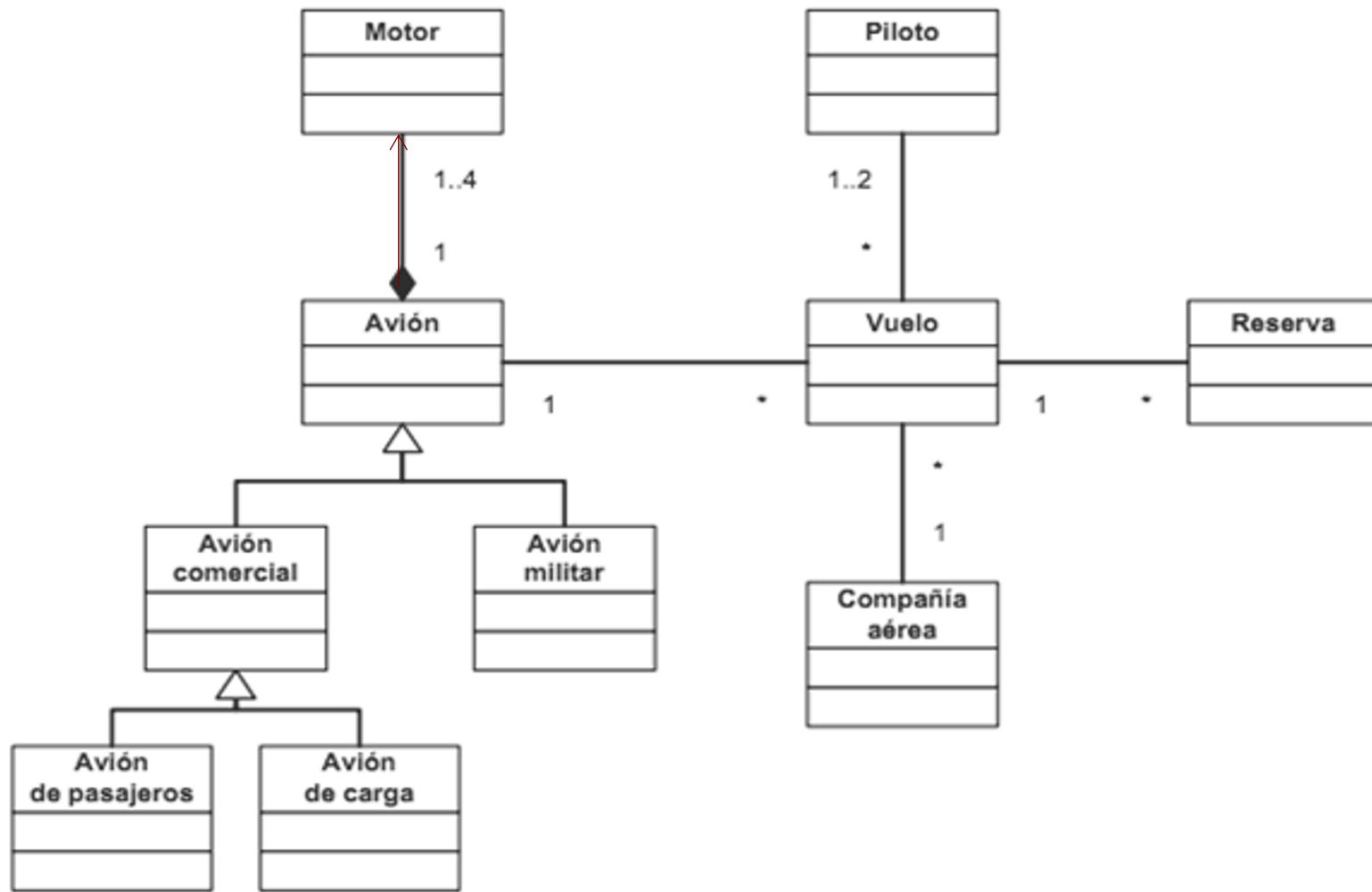
## Clases con parámetros

Son aquellas clases que necesitan parámetros de entrada al momento de ser instanciadas. Los parámetros se denotan con un recuadro arriba del nombre de la clase.



## Ejemplo 2

### Diagrama de clases de un vuelo



## Diagrama de objetos

**Los diagramas de objetos modelan las instancias generadas a través de las clases y se utilizan para describir al sistema en un instante de tiempo (o acción) en particular.**

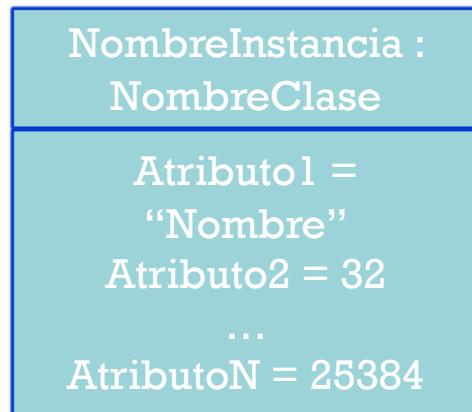
**Permiten mostrar los objetos y las relaciones entre ellos en un momento dado, por lo tanto, representa la parte estática de la interacción entre objetos (una situación específica en un momento determinado).**

**Los elementos utilizados para generar este tipo de diagramas son:**

- **Objeto(s)**
- **Asociación(es)**

## Objeto

**Los objetos se representan con dos rectángulos. En el primer rectángulo se indica el nombre del objeto y el nombre de la clase separados por dos puntos. En el segundo rectángulo se indican el nombre del (los) atributo(s) y el (los) valor(es) asignado(s).**

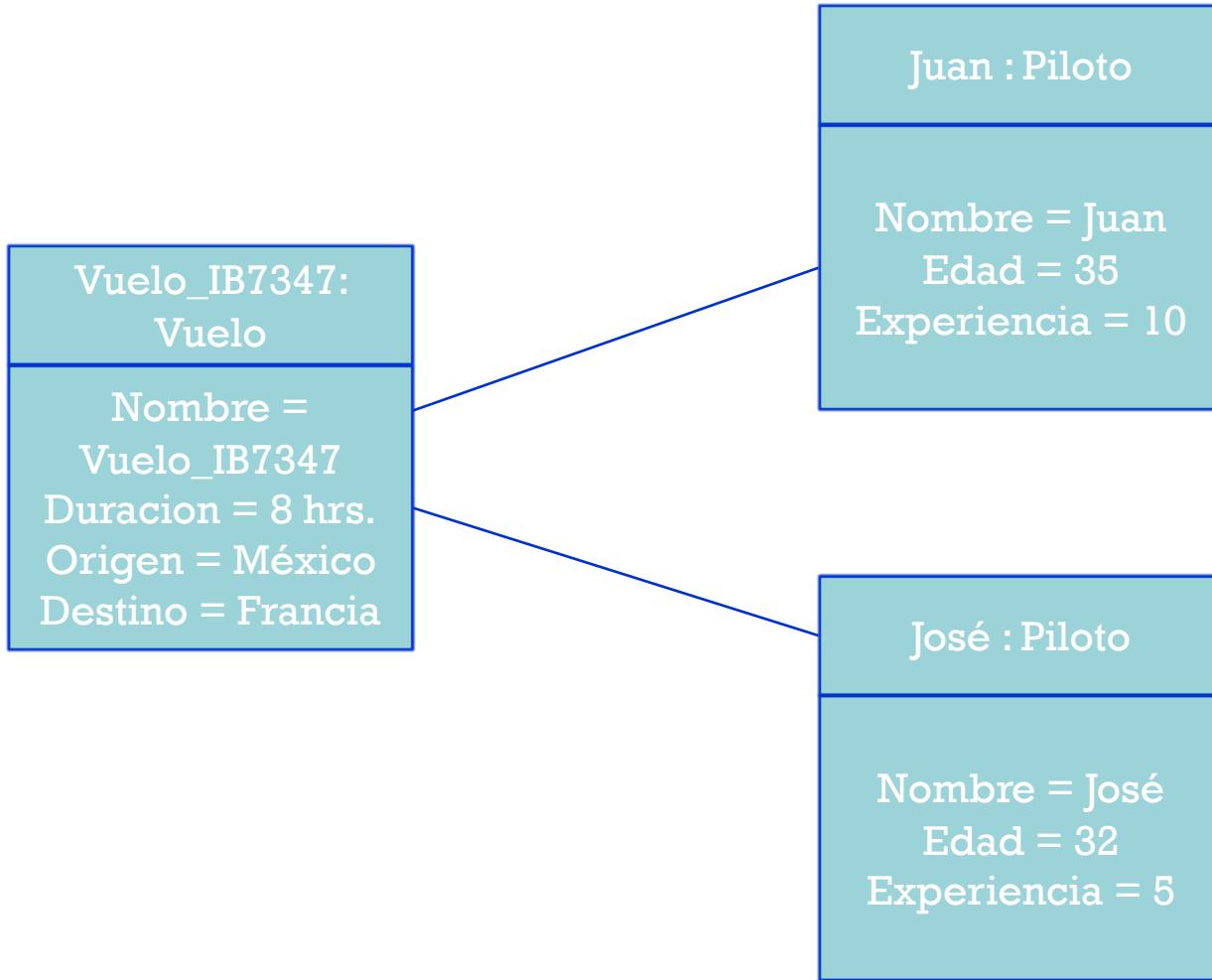


## Asociación

**Los objetos se unen utilizando una línea continua entre las instancias relacionadas.**

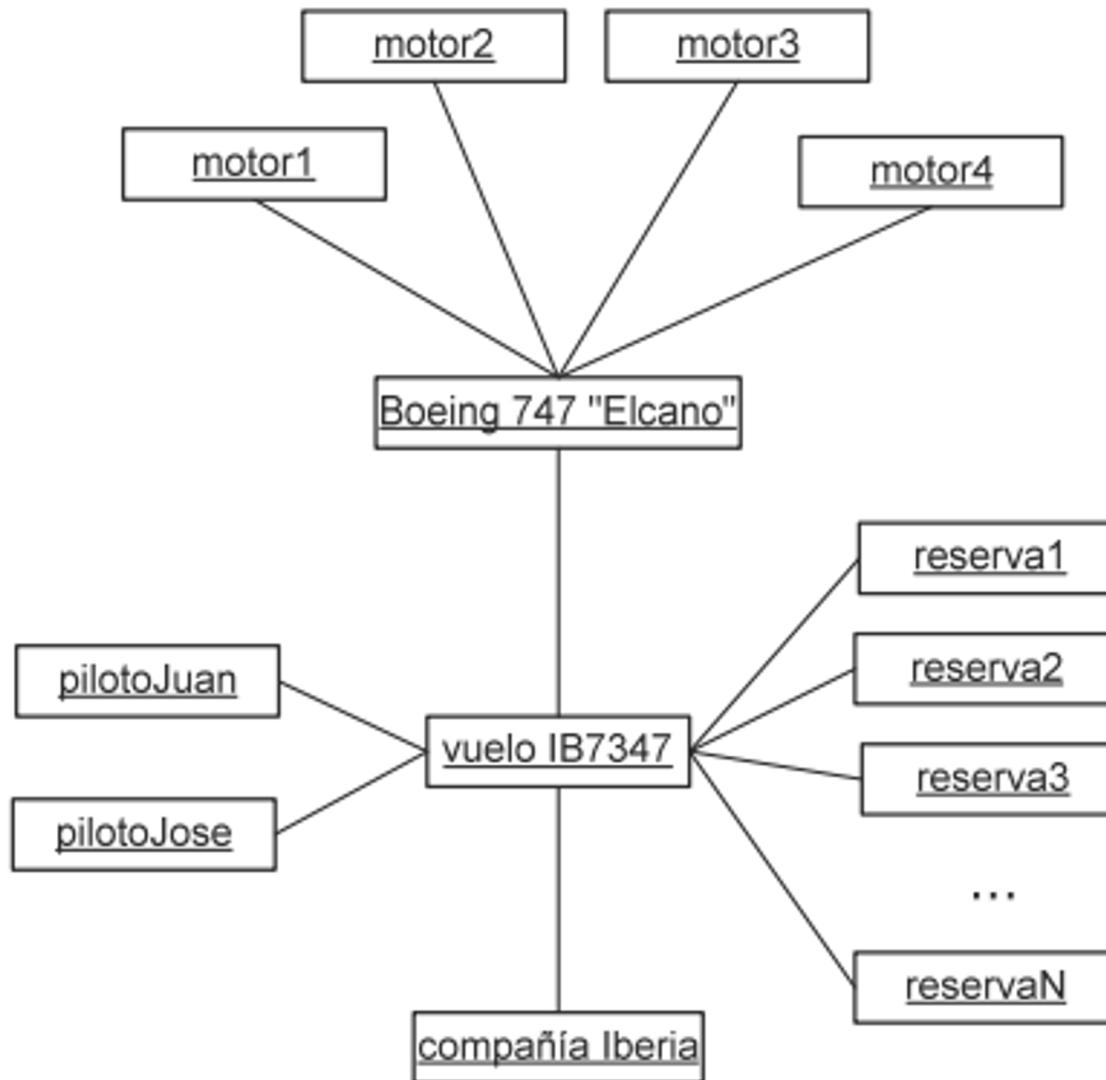


## Objetos de tipo Piloto



### Ejemplo 3

## Objetos de un vuelo



## Diagrama de componentes

**Un diagrama de componentes permite describir los elementos de software que constituyen al sistema, así como las relaciones que deben existir entre éstos.**

**Los elementos básicos de un diagrama de componentes son:**

- **Componente(s)**
- **Enlace(s)**

## Componentes

**Un componente es una parte lógica (un archivo ejecutable, una biblioteca, un archivo binario, etc.) de un sistema. Los componentes permiten modelar los elementos de software que van a conformar al sistema. Su representación gráfica en lenguaje UML es:**



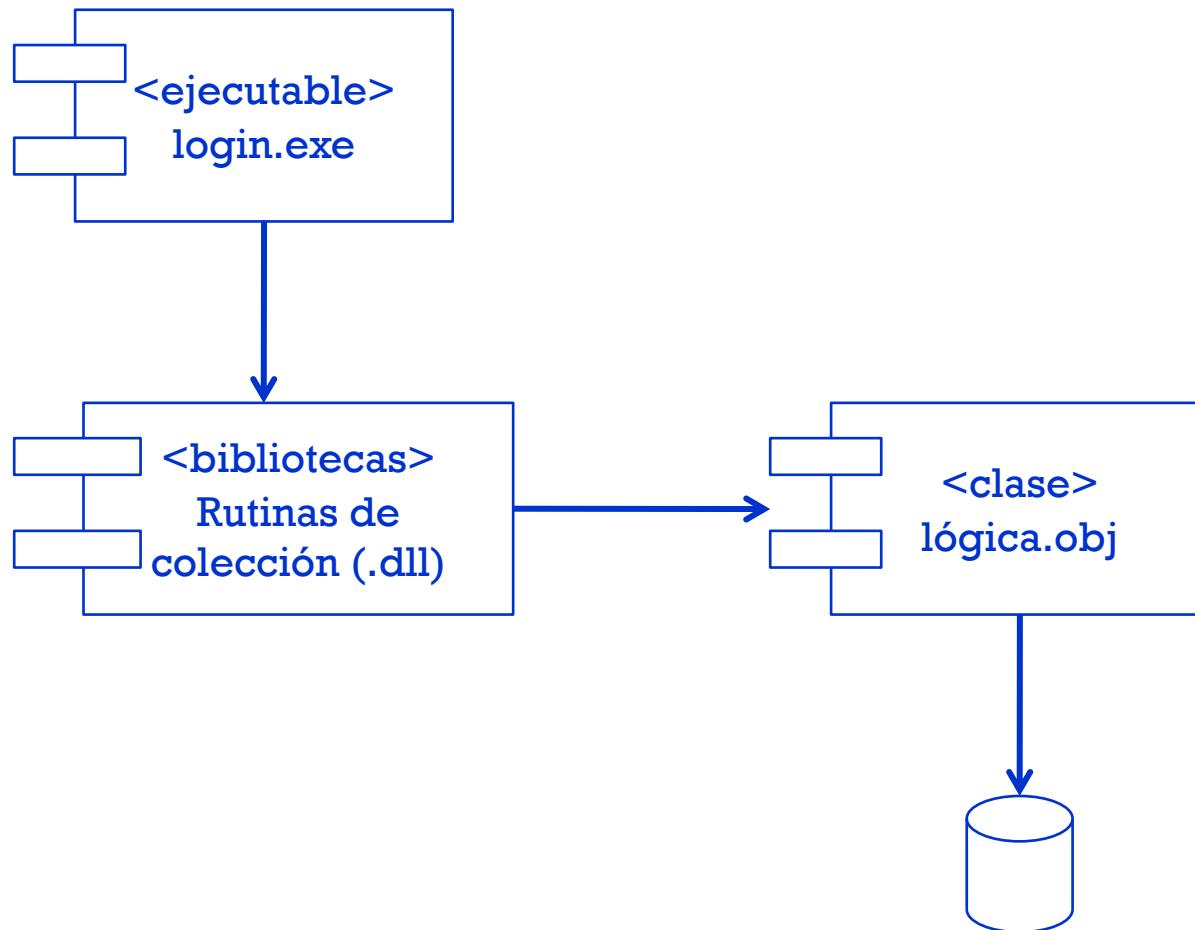
## Enlace

**Los enlaces permiten unir componentes del diagrama con el fin de saber qué otros componentes de software son necesarios para su funcionamiento. Se representan a través de una flecha:**



## Ejemplo 4

### Mensajero



## Diagrama de despliegue / distribución

**Un diagrama de despliegue o distribución permite mostrar la disposición física de los distintos elementos (nodos) que se requieren para que el sistema se pueda poner en marcha, es decir, permite visualizar el hardware necesario sobre el cual se ejecutará el sistema.**

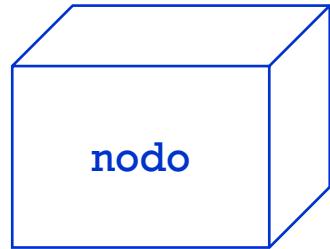
**Los componentes básicos de un diagrama de despliegue son:**

- **Nodo(s)**
- **Relación(es)**

## Nodo

**Un nodo es un elemento físico que se requiere para ejecutar la aplicación. Representa un recurso computacional.**

**Los nodos se utilizan para modelar la topología del hardware sobre la que se ejecuta el sistema.**



## **Enlace**

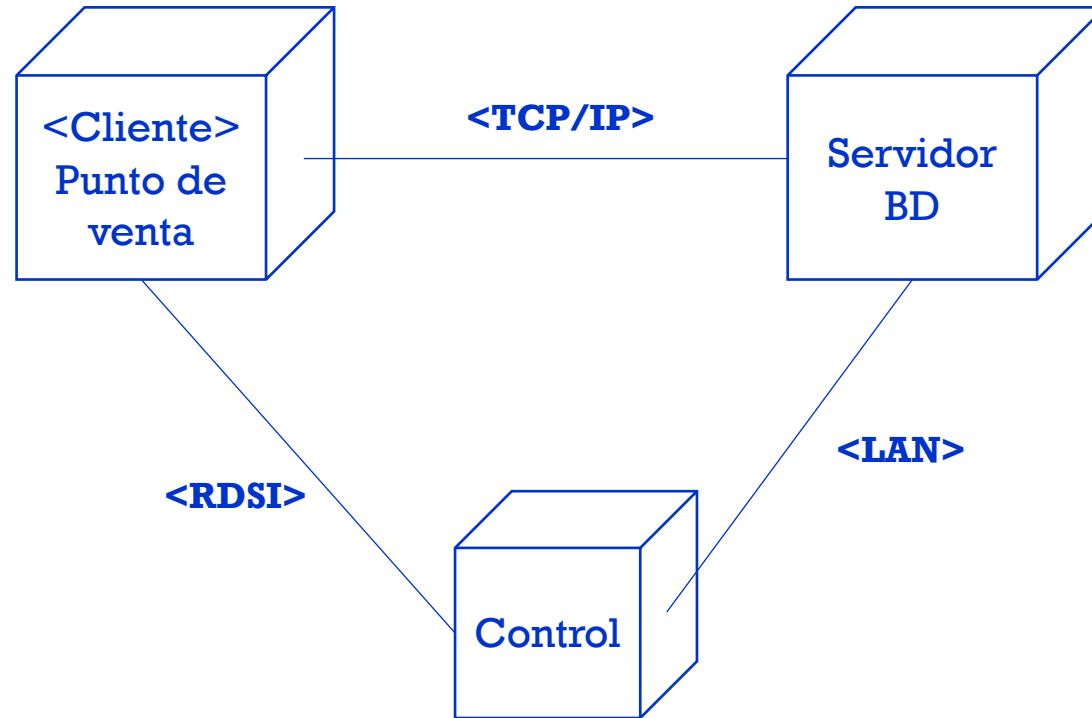
**Un enlace representa la unión que existe entre los nodos. Los enlaces pueden indicar el tipo de conexión o alguna especificación técnica que se requiere para la comunicación física entre los nodos.**

### **<TIPO COMUNICACIÓN>**

---

## Ejemplo 5

### Realizar una compra con tarjeta bancaria



# DISEÑO DINÁMICO



**Los diagramas dinámicos o de comportamiento permiten visualizar la comunicación entre elementos del sistema para un proceso específico. Los diagramas que permiten modelar estas características son:**

- **Diagrama de estados**
- **Diagrama de actividades**
- **Diagrama de interacción**

## Diagrama de estados

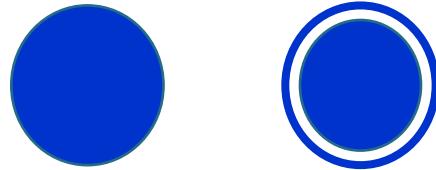
**Los diagramas de estados describen las transiciones por las que puede pasar un objeto durante su tiempo de vida en la aplicación, así como la manera en la que cambia de estado el objeto.**

## **Los elementos de un diagrama de estados son**

- Nodos de entrada y salida.**
- Estado(s) del objeto.**
- Transición(es) entre estados.**

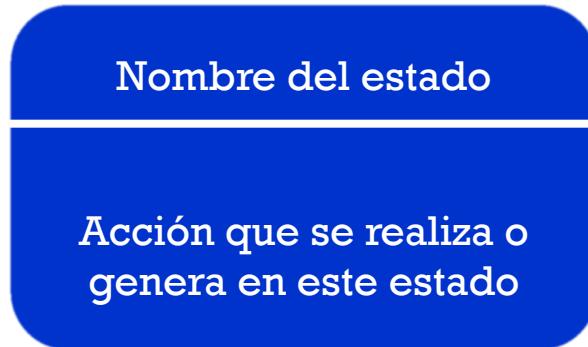
## Nodos

Los nodos de entrada y salida representan el inicio y final del diagrama. El símbolo de inicio está representado por una circunferencia rellena y el símbolo de final está representado por dos circunferencias, una exterior (sin llenar) y una interna (rellena).



## Estado

**Un estado se representa como un rectángulo redondeado que posee dos separaciones, en la primera se escribe el nombre del estado y en la segunda se escriben las acciones de entrada, salida o internas que se tienen en dicho estado.**



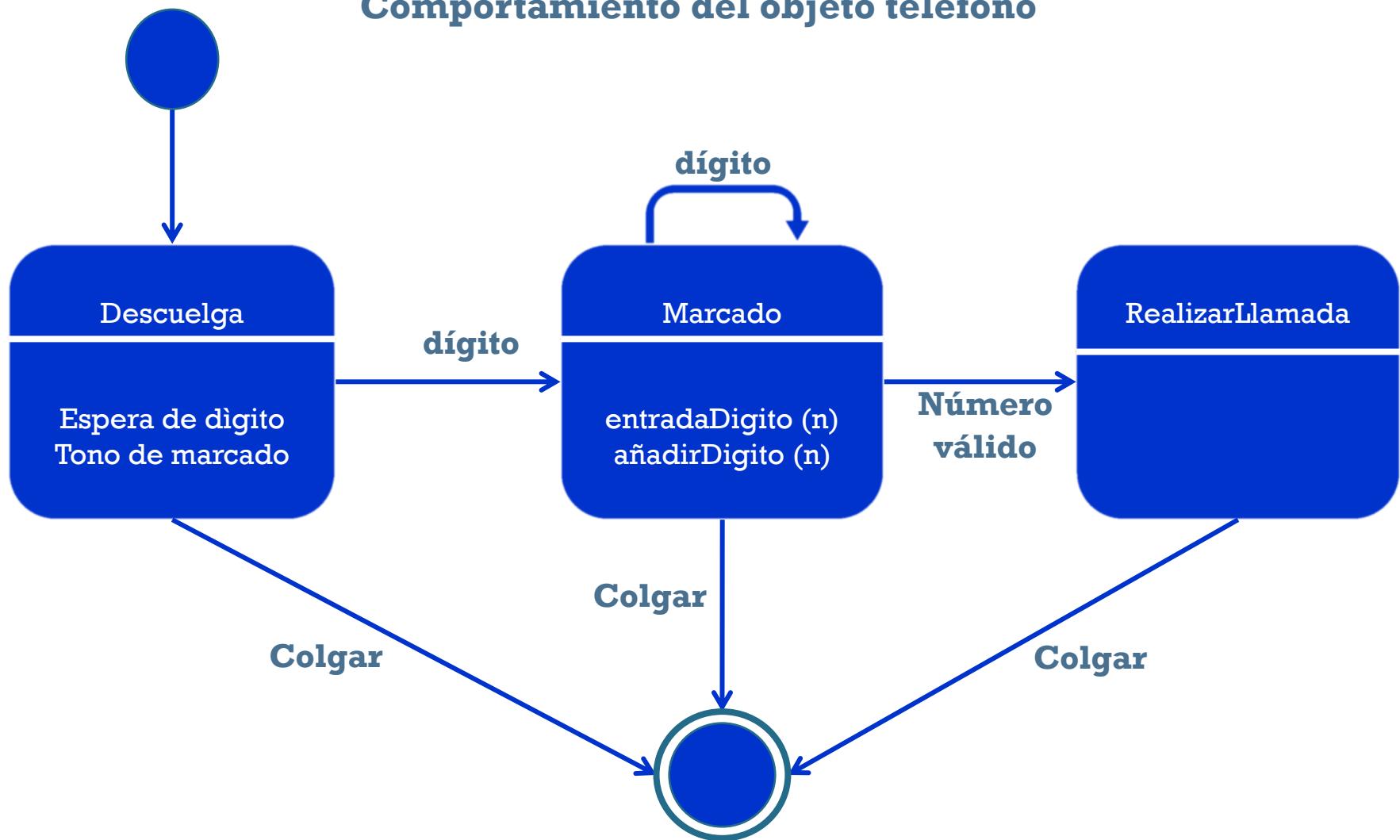
## Transición

Una transición entre estados representa el flujo que puede seguir el objeto dependiendo de la entrada recibida. Se representa mediante una flecha desde el estado origen hacia el estado destino.



## Ejemplo 6

### Comportamiento del objeto teléfono



## Diagrama de actividades

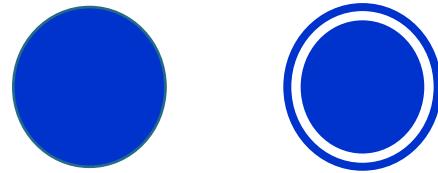
**El diagrama de actividades muestra el flujo de acciones (operaciones que se ejecutan) y los objetos involucrados. Permite visualizar el orden en el que se van realizando tareas dentro de un sistema, así como los objetos involucrados en la comunicación.**

**Los componentes básicos de un diagrama de actividades son:**

- **Nodos de entrada y salida.**
- **Actividad(es).**
- **Transiciones.**

## Nodos de entrada y salida

**Los nodos de entrada y salida representan el estado inicial y el estado final del diagrama. El símbolo de inicio está representado por una circunferencia rellena y el símbolo de final está representado por dos circunferencias, una exterior (sin llenar) y una interna (rellena).**



## Actividad

**Una actividad representa los diferentes comportamientos por los que puede transitar un objeto de manera interna, describiendo con más detalle las acciones que desencadena una actividad.**

Actividad

## Transición

**Las transiciones representan los flujos de acción que puede seguir una actividad. Se representan mediante una línea que va del estado origen al estado destino. Existen 3 tipos de transiciones:**

- **Secuencial o sin disparadores**
- **Bifurcación**
- **Unión**

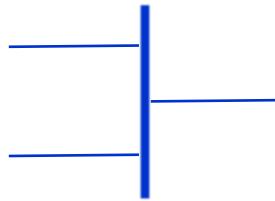
## Transición secuencial

Muestra el flujo de una actividad sin divisiones.



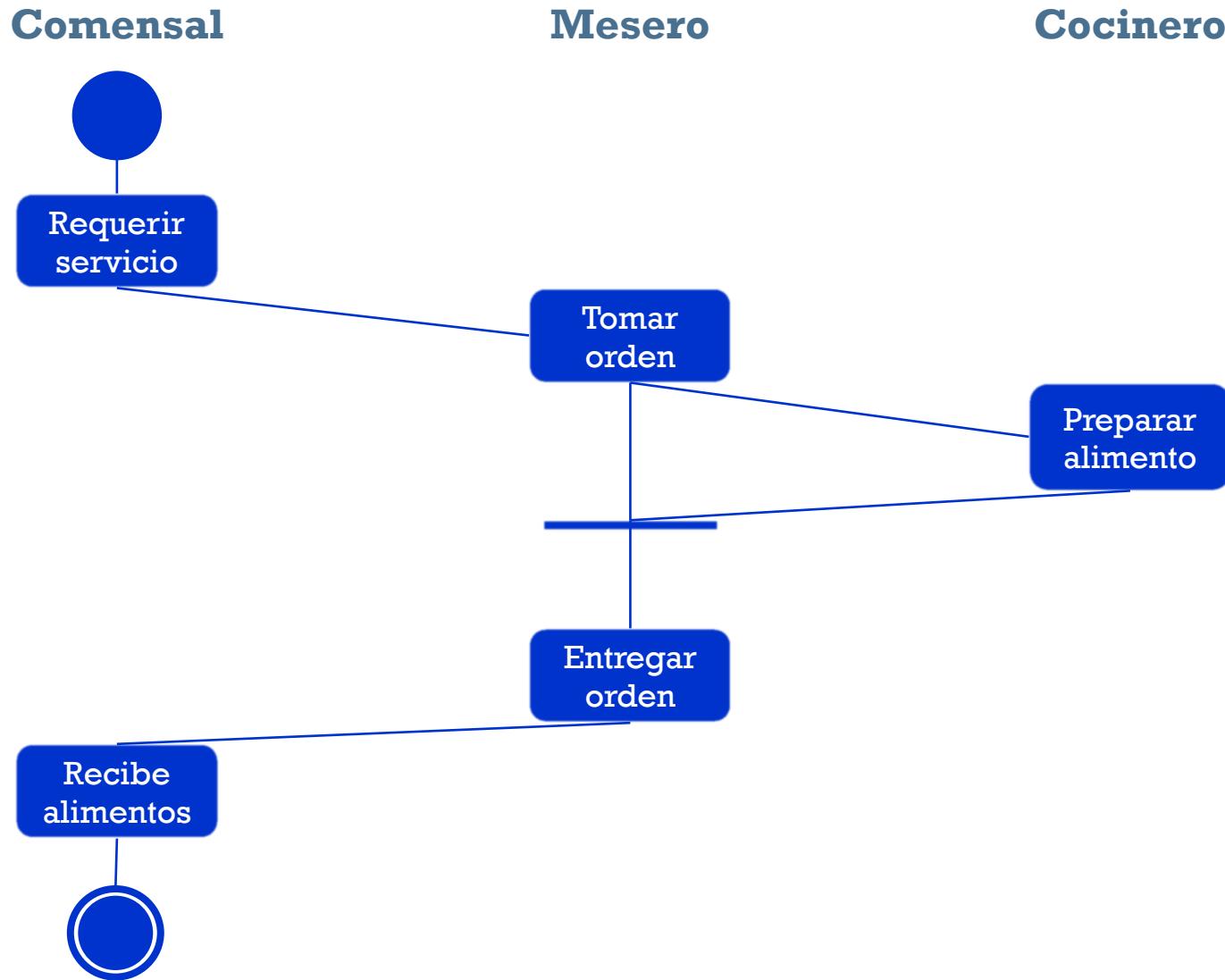
## Transición de unión

**Permite unir un camino bifurcado para regresar a un camino secuencial.**



## Ejemplo 7

### Generar comanda en un restaurante



## Diagrama de interacción

**Estos diagramas representan la comunicación que se lleva a cabo entre un cliente (actor) o un objeto (clase) cuando se ejecuta una acción en el sistema.**

**Los elementos básicos de los diagramas de interacción son:**

- **Objeto o actor.**
- **Enlace(s).**
- **Mensaje(s).**

**Los diagramas de interacción, a su vez, se dividen en dos tipos:**

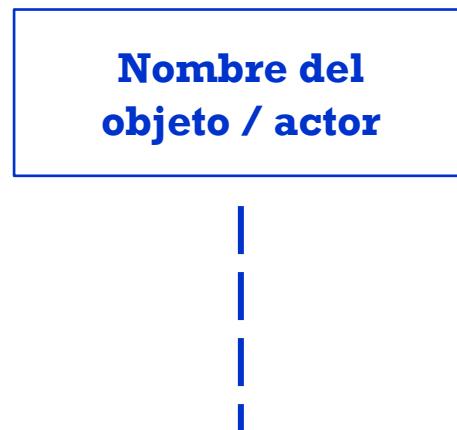
- **Diagrama de secuencia.**
- **Diagrama de colaboración.**

## Diagrama de secuencia

**Los diagramas de secuencia muestran una interacción ordenada de eventos, visualizando los objetos participantes en cada interacción, así como los mensajes que intercambian entre ellos.**

## Objeto o actor

**Los objetos o actores se representan mediante un rectángulo (donde se indica el nombre del objeto o del actor) y con una línea punteada, la cual representa el tiempo de vida del objeto o actor.**



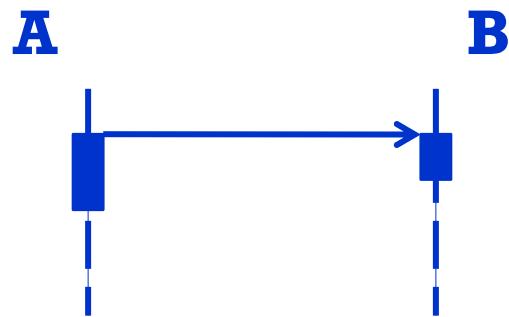
## Enlace

Un enlace representa la llamada de un método a otro (o al mismo) y se representa a través de una flecha. Los métodos se representan con rectángulos dispuestos dentro de la línea de tiempo del actor/objeto.

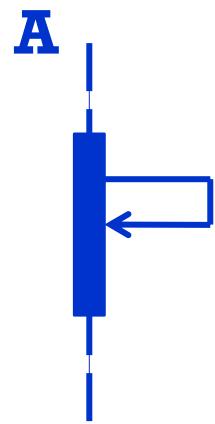


## Mensaje

**Los mensajes entre objetos se representan por una flecha que va de un objeto y otro. La flecha parte de un rectángulo desde el objeto/actor de origen hacia otro rectángulo en el objeto/actor de destino.**

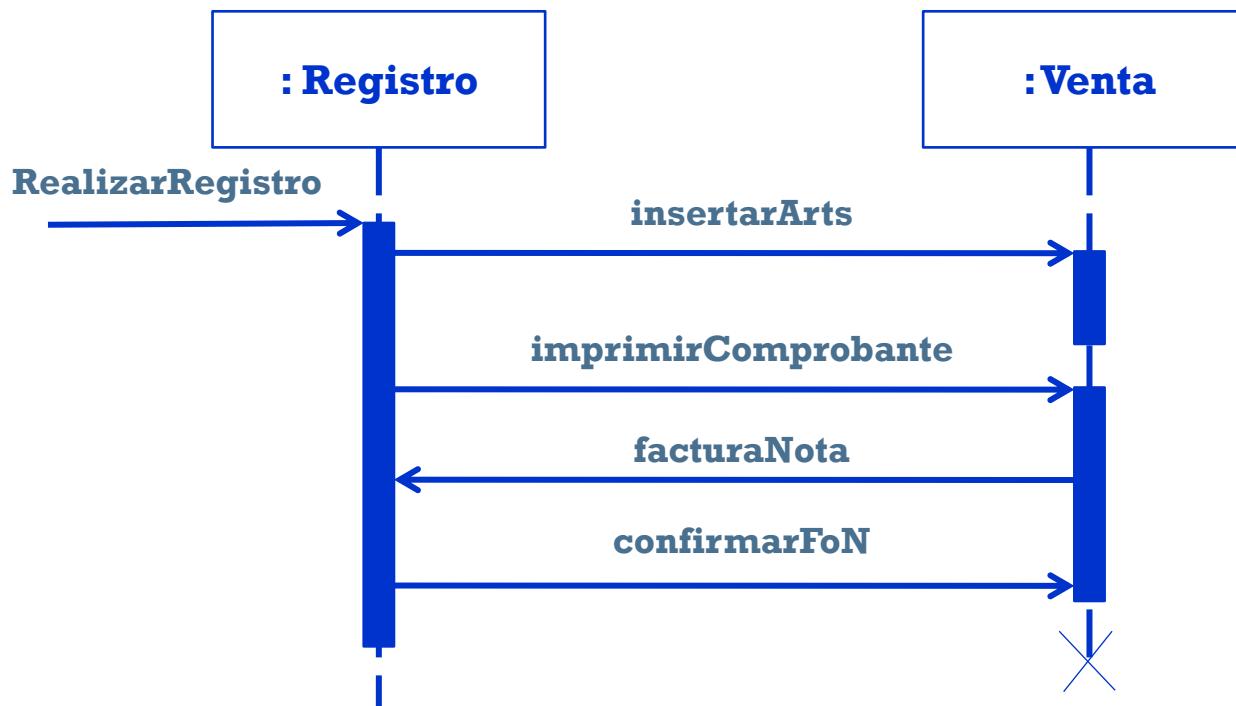


**Es posible visualizar las llamadas a métodos propios del objeto, es decir, mensajes al mismo objeto. En este caso la flecha parte de un rectángulo desde el objeto/actor de origen hacia el mismo rectángulo (recursividad) o hacia otro rectángulo del mismo objeto/actor.**



## Ejemplo 8

### Crear nota o factura de una venta



## **Diagrama de colaboración**

**Los diagramas de colaboración muestran la interacción entre varios objetos y el orden que existen entre ellos.**

**La secuencia de los mensajes y los flujos de ejecución concurrentes se determinan mediante números secuenciales.**

## Objeto

**Representa un elemento dentro del sistema que puede ser invocado directamente por el usuario o de manera interna. Permite identificar los objetos involucrados en la interacción. Se representa con un rectángulo.**

**objeto : Clase**

## Enlace

**Los objetos se enlazan mediante una línea indicando el sentido de la invocación, es decir, quién invoca a quién. Normalmente el flujo de acción se indica mediante una fecha.**

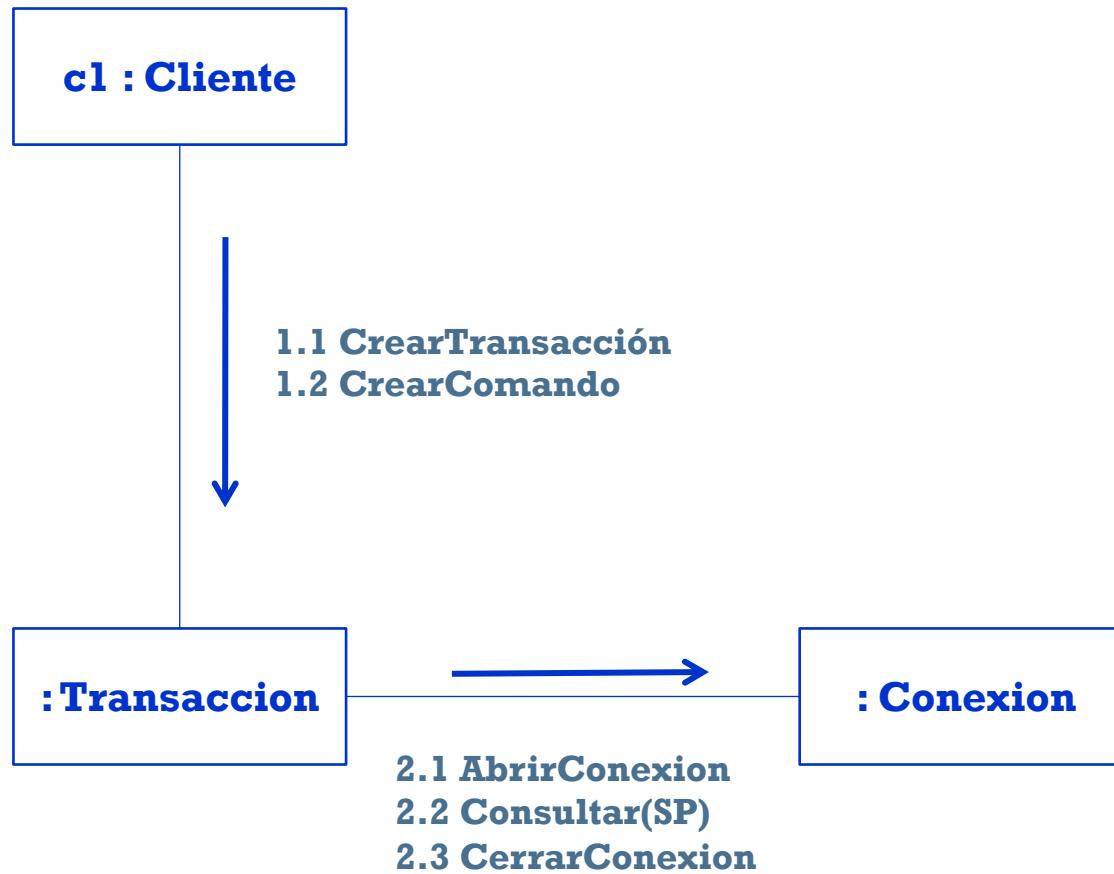


## Mensaje

**Los mensajes permiten visualizar el método al que invoca un objeto, así como el orden de invocación.**

1.1 método1  
1.2 método2

## Consultar datos de una Base de Datos



**Para realizar diagramas UML existen diversas herramientas, entre los programas más conocidos se pueden encontrar:**

- **Dia**
- **Umbrello**
- **BoUML**
- **ArgoUML**
- **IBM Rational Rose**
- **Poseidon**
- **SmartDraw**
- **Enterprise Architect**

## Diagramas UML

**Para un mensajero (messenger, skype, google talk, etc.), realizar los siguientes diagramas:**

- **Los diagramas de casos de uso, de clases y de despliegue de la aplicación.**
- **El diagrama de estados de una conversación.**
- **Realizar el diagrama de interacción (secuencia) al establecer una conversación.**
- **Realizar un diagrama de actividades que describa la acción de iniciar sesión.**

**Los diagramas se envían en un archivo en formato PDF.**

## 4 Lenguaje de modelado unificado

**Objetivo:** Clasificar las diferentes vistas en el diseño orientado a objetos para aplicarlo en la solución de problemas.

4.1 Diseño estático.

4.2 Diseño dinámico.