

Ingeniería de Software II

UML – Leguaje Unificado de Modelado

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología

Lic. Paolo Orundés Cardinali
orundescardinali.paolo@uader.edu.ar

Diagrama de objetos

Los **diagramas de objetos** en UML son una representación de instancias específicas de clases en un momento determinado. Se usan para visualizar cómo se relacionan los objetos en un sistema en un estado concreto.

Características principales

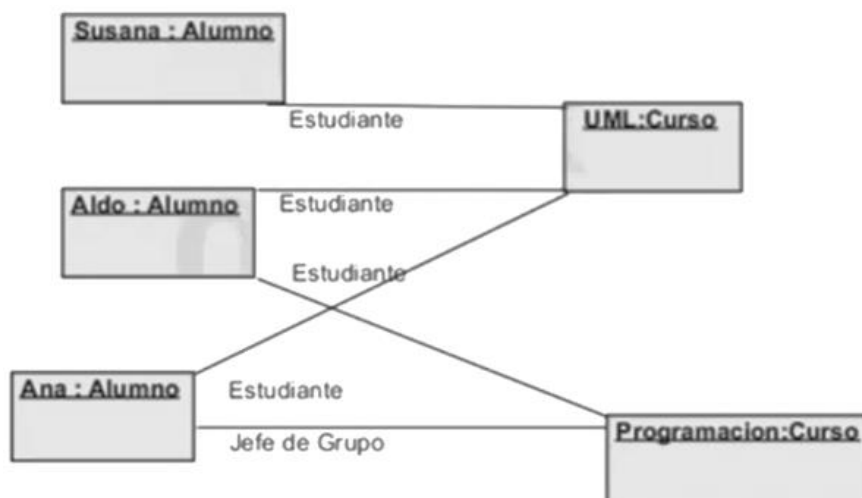
- Muestran instancias de clases en lugar de las clases mismas.
- Representan valores concretos de atributos en un instante de tiempo.
- Permiten visualizar relaciones entre objetos, como asociaciones y agregaciones.
- Son útiles para ilustrar ejemplos específicos de interacción entre objetos.
- Representan la lógica del negocio
- Se les conoce también como diagramas de instancia



- Se usan para mostrar el estado del sistema en un momento concreto
- El diagrama se enfoca en la parte lógica y no tanto a lo estructural.
- **El nombre del objeto y la clase van subrayados**

Conectando objetos

- Los objetos se conectan por medio de una línea que es nombrada
- Cuando hay una relación entre objetos o clases, ésta debe de mostrarse en ambos diagramas



- Si en el diagrama de clases aparece una asociación, en el diagrama de objetos puede aparecer un **enlace**

Ejemplos

Diagrama de clases

Nos da información acerca de la estructura

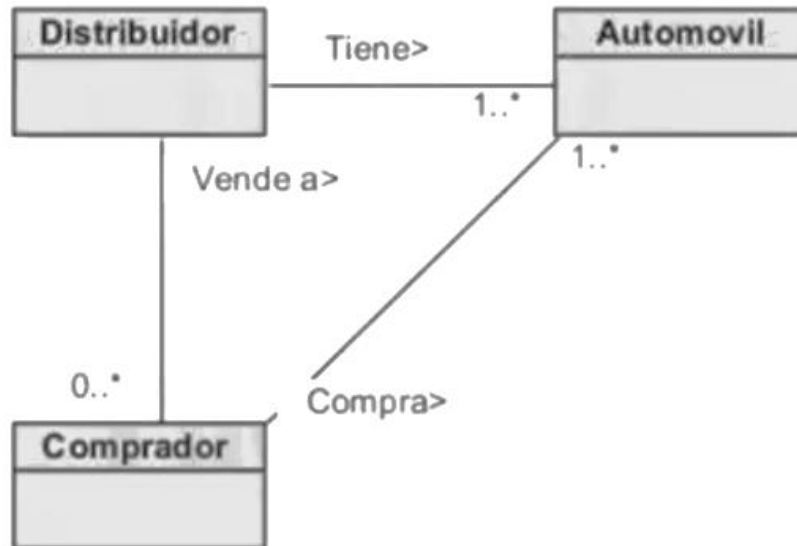


Diagrama de Objetos

Nos da más información acerca de la lógica como tal

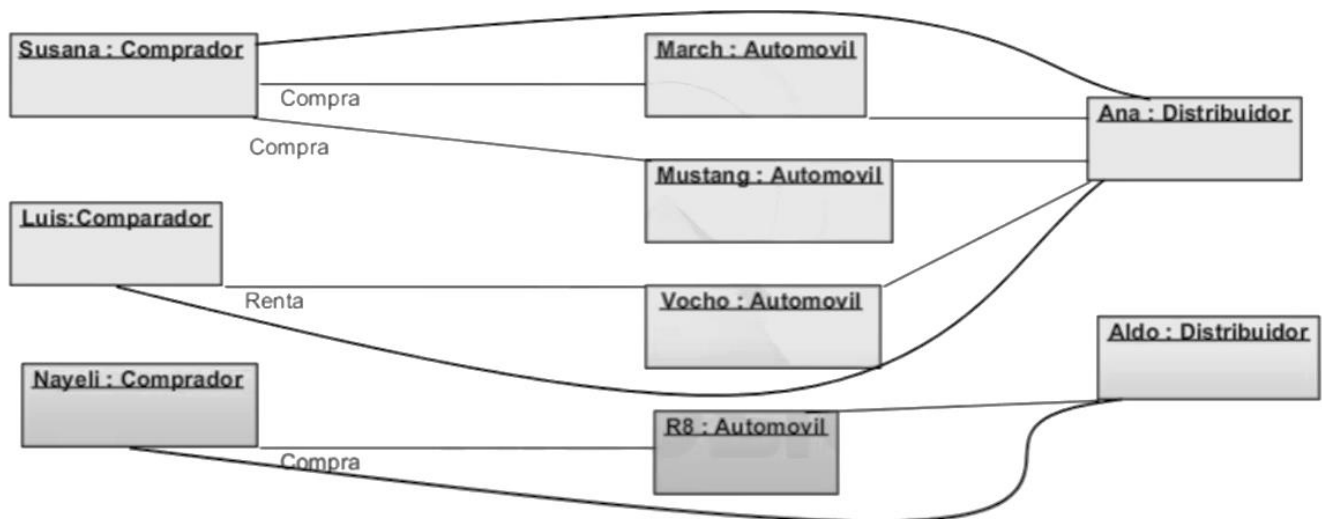


Diagrama de estados

El **diagrama de estados** en UML representa los diferentes estados por los que pasa un objeto a lo largo de su ciclo de vida, así como los eventos o condiciones que causan los cambios de un estado a otro. Es útil para modelar el comportamiento dinámico de un sistema.

Elementos principales

1. **Estados:** Representan las diferentes situaciones en las que puede estar un objeto. Se dibujan como rectángulos redondeados.
2. **Transiciones:** Indican el paso de un estado a otro, representadas por flechas.
3. **Eventos o transición:** Condiciones o sucesos que activan una transición.
4. **Estados inicial y final:**
 - **Estado inicial:** Representado por un círculo negro sólido.
 - **Estado final:** Representado por un círculo negro con un contorno.
5. **Actividades:** representan las acciones o procesos que ocurren mientras un objeto se encuentra en ese estado

Usos del diagrama de estados

- Modelar el ciclo de vida de un objeto.
- Representar sistemas que cambian de estado, como protocolos de comunicación o flujos de trabajo.
- Comprender el comportamiento de clases con estados complejos.

Entendiendo el concepto, estado y transición

La máquina de estados tiene un número finito de estados, es decir **se conocen de antemano**. Nos vamos a concentrar en el de comportamiento.

- Nos sirve para mostrar el comportamiento del sistema, ya sea:
 - Caso de uso
 - Clase
 - Sistema completo
- El estado lleva en su interior actividades
- La transición lleva una etiqueta que se conoce como firma
 - `<gatillo>[<guarda>]/<efecto>`
 - Gatillo es el evento
 - Guarda es la condición
 - Efecto es la actividad que sucede (no después) durante la transición

Actividades

- El estado puede estar activo o inactivo
- Tenemos tres posibles actividades
 - Entry
 - Do
 - Exit



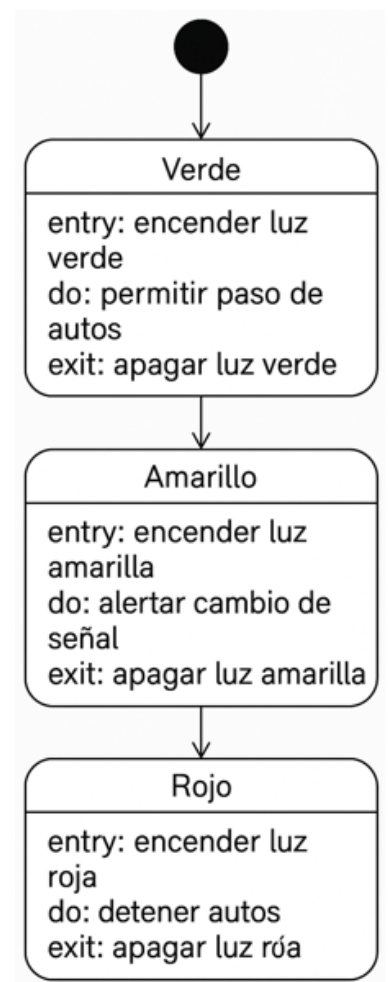
Ejemplos



Ejemplo: Máquina de estados de un semáforo

1. **Estados:**
 - Verde
 - Amarillo
 - Rojo
2. **Transiciones:**
 - Verde → Amarillo (cuando el tiempo del verde se agota)
 - Amarillo → Rojo (cuando el tiempo del amarillo se agota)
 - Rojo → Verde (cuando el tiempo del rojo se agota)

Este diagrama muestra cómo cambia el semáforo dependiendo de los eventos que ocurren.



Ejemplo: petición a un servidor

1. Estados:

- Entrada del usuario
- Eco del servidor

2. Transiciones:

- Entrada / envío al servidorVerde → Amarillo (cuando el tiempo del verde se agota)
- Llegada del eco / muestra por pantalla

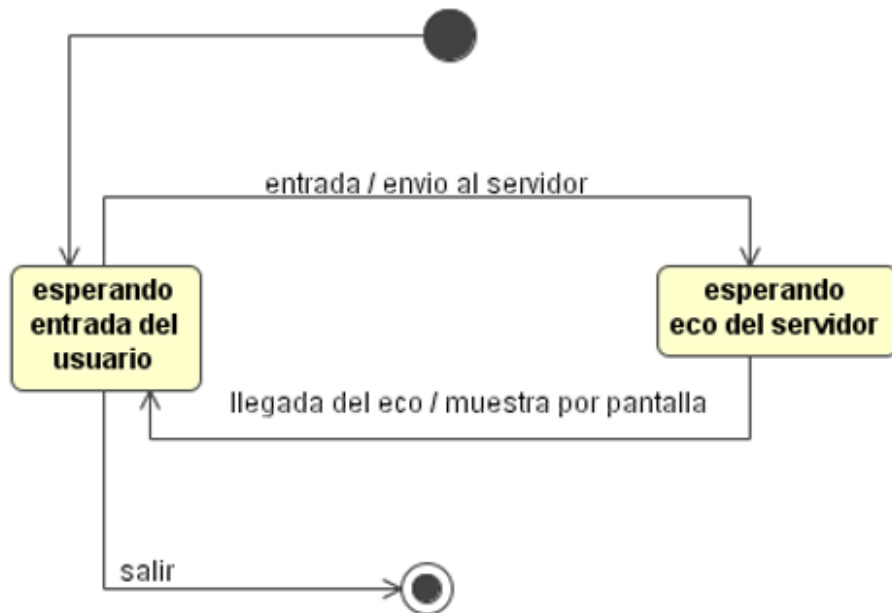


Diagrama de secuencia

Los **diagramas de secuencia** en UML representan la interacción entre objetos en un sistema en orden cronológico. Son ideales para modelar cómo fluyen los mensajes entre los diferentes componentes de un sistema en un escenario específico.

Usos del diagrama de secuencia

- Modelar el comportamiento dinámico de un sistema.
- Visualizar la comunicación entre objetos.
- Entender el flujo de ejecución de un caso de uso.
- Detectar posibles errores en la lógica antes de programar.

Características

- Los diagramas de secuencia muestran como los objetos se intercambian mensajes a lo largo del tiempo
- Indica la secuencia de interacción, el orden en que suceden
- Se leen de arriba hacia abajo, tomando el tiempo en el eje vertical
- Muestra la secuencia de acciones que pasan en el sistema
- Los objetos se muestran como rectángulos y algunas veces también se los llama participantes
- De ellos sale una línea de vida, donde ocurrirán los envíos o recepciones de mensajes

- Los objetos no necesariamente son instancias de clases, pueden representar también una parte del sistema
- Los mensajes pueden estar escritos en lenguaje natural o con una sintaxis más precisa y cercana a los lenguajes de programación dependiendo del nivel de abstracción

Elementos principales

1. Objetos/Participantes:

- Se representan con rectángulos con nombres subrayados.
- Pueden ser clases, actores o sistemas externos.

2. Líneas de vida (Lifelines):

- Son líneas verticales debajo de cada objeto.
- Representan la existencia del objeto durante la interacción.

3. Mensajes:

- Se representan con flechas entre líneas de vida.
- Pueden ser sincrónicos (línea sólida con punta llena) o asincrónicos (línea sólida con punta abierta).

4. Activación:

- Se muestra como un rectángulo delgado sobre la línea de vida.
- Indica cuándo un objeto está realizando una acción.

5. Regiones alternativas y de repetición:

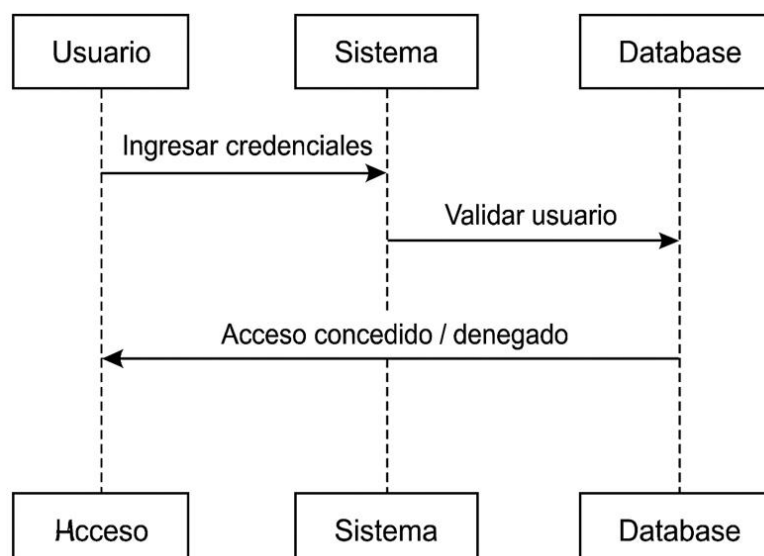
- Se encierran en cuadros con etiquetas como `alt` (alternativa) y `loop` (bucle).
- Representan decisiones y repeticiones dentro del proceso.

Ejemplo: Inicio de Sesión

Supongamos que queremos modelar el proceso de inicio de sesión en un sistema.

1. El **usuario** ingresa sus credenciales.
2. El **sistema** valida la información.
3. Si los datos son correctos, muestra el **panel de usuario**.
4. Si son incorrectos, devuelve un **mensaje de error**.

Representado en un diagrama de secuencia, se vería así:



Clasificación

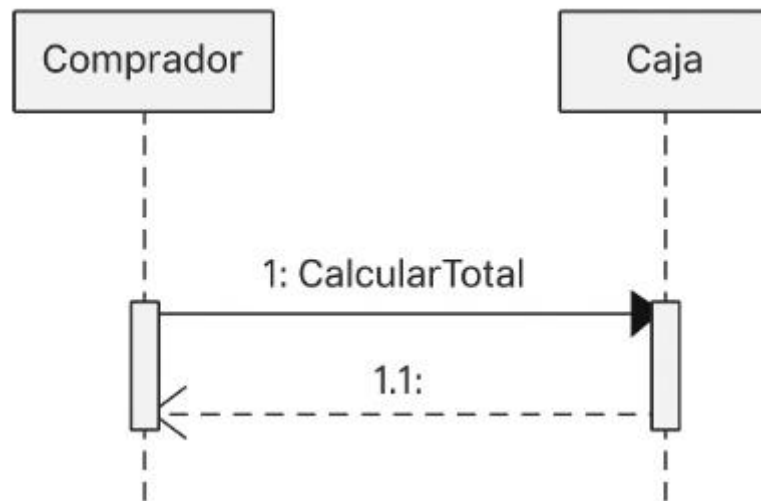
- Mensaje de llamada
- Mensaje asincrónico
- Auto mensaje
- Mensaje recursivo
- Mensaje encontrado
- Mensaje perdido
- Mensajes para crear y destruir objetos
- Límite
- Control
- Actor y Entidad
- Marcos de interacción
 - Opcional
 - Alternativo
 - Ciclos
 - Paralelas
 - Otros marcos

Mensaje de llamada

Los **mensajes de llamada** en UML son un tipo de mensaje en los diagramas de secuencia que representan la invocación de un método o función en un objeto. Estos mensajes indican que un objeto solicita a otro que ejecute una operación específica.

Características

- Se representan con una flecha sólida con punta llena (→) que va desde el objeto que envía el mensaje hasta el objeto que lo recibe.
- Pueden llevar parámetros, representando valores enviados al método.
- Pueden ser síncronos o asíncronos:
 - Síncronos: El remitente espera una respuesta antes de continuar (línea sólida con punta llena).
 - Asíncronos: El remitente no espera respuesta y sigue su ejecución (línea sólida con punta abierta).



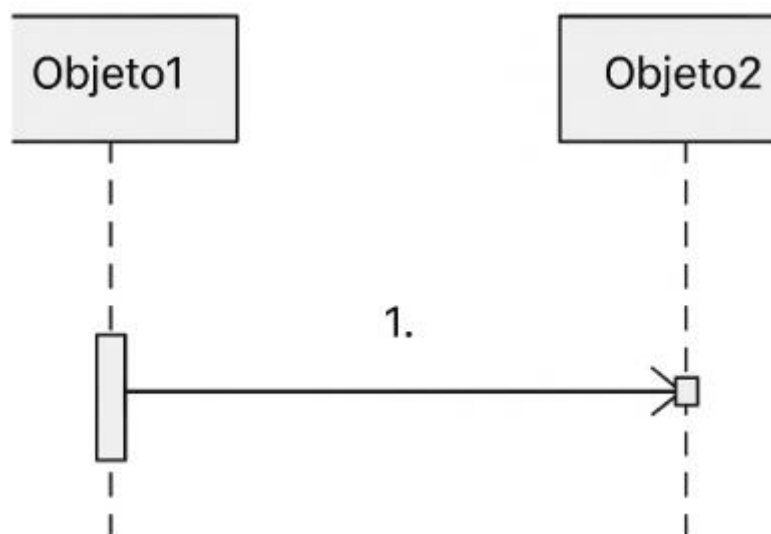
- Las líneas punteadas verticales se las conoce como Línea de Vida
- Comprador invoca un método de Caja, se representa con flecha y cabeza llena. Y el mensaje es 1: CalcularTotal, donde 1 representa el número de mensaje, es decir es el primero.
- 1.1: Es el mensaje de retorno y el 1: es el mensaje asociado. Se representa con línea punteada y flecha abierta

Mensaje asíncrono

Los **mensajes asíncronos** en UML representan una comunicación entre objetos donde el **emisor no espera una respuesta inmediata** del receptor antes de continuar su ejecución. Son útiles para modelar sistemas concurrentes, eventos y comunicaciones entre procesos independientes.

Características

- Se representan con una flecha de línea sólida con punta abierta (→ con punta en "V").
- El emisor sigue ejecutando otras acciones sin esperar que el receptor termine su tarea.
- El receptor puede procesar el mensaje más tarde, en paralelo con otras acciones.
- Común en sistemas distribuidos y aplicaciones concurrentes, como mensajes en colas, eventos en interfaces gráficas y sistemas de mensajería.



Diferencia entre Mensajes Sincrónicos y Asincrónicos

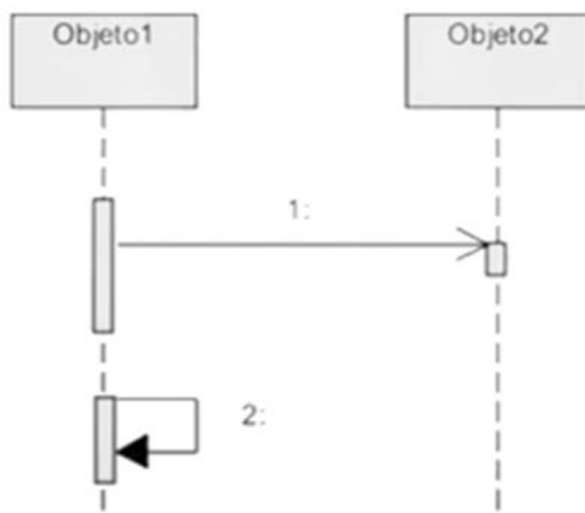
| Tipo de Mensaje | Representación | Espera Respuesta | Ejemplo |
|-----------------|-------------------------------------|------------------|--|
| Sincrónico | Flecha sólida con punta llena (→) | Sí | Llamada a un método donde se espera una respuesta antes de continuar |
| Asincrónico | Flecha sólida con punta abierta (⇢) | No | Un sistema envía un evento a otro sin esperar respuesta inmediata |

Automensaje

Un **automensaje** en UML es un mensaje que un objeto se envía a sí mismo dentro de un **diagrama de secuencia**. Representa una llamada a un método dentro del mismo objeto.

Características

1. El emisor y el receptor son el mismo objeto.
2. Se representa con una flecha que regresa al mismo objeto, formando un **bucle**.
3. Útil para representar llamadas a métodos internos o procesos recursivos.

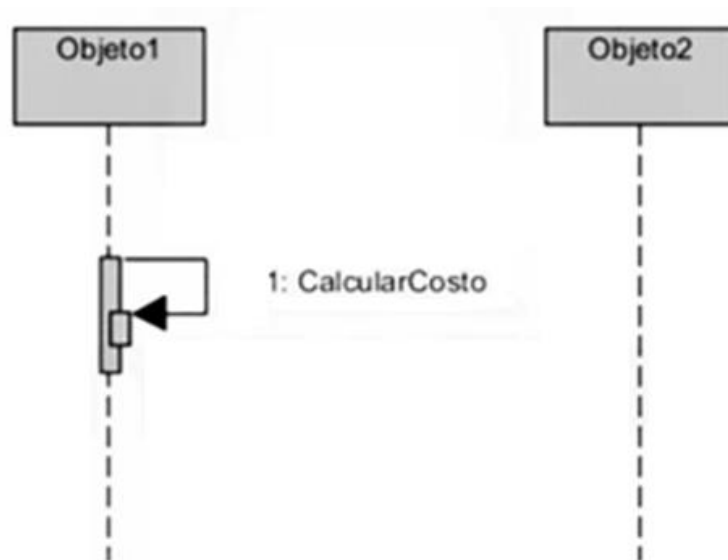


Mensaje recursivo

Un **mensaje recursivo** en UML es un tipo de **automensaje**, donde un objeto se envía un mensaje a sí mismo y **llama repetidamente al mismo método** dentro de su propia ejecución. Se usa para representar **recursión** en algoritmos o procesos iterativos.

Características

1. El emisor y receptor son el mismo objeto.
2. Se representa con una flecha que **vuelve al mismo objeto**, formando un **bucle**.
3. Se usa en **algoritmos recursivos**, como búsqueda en árboles, factorial, Fibonacci, etc.



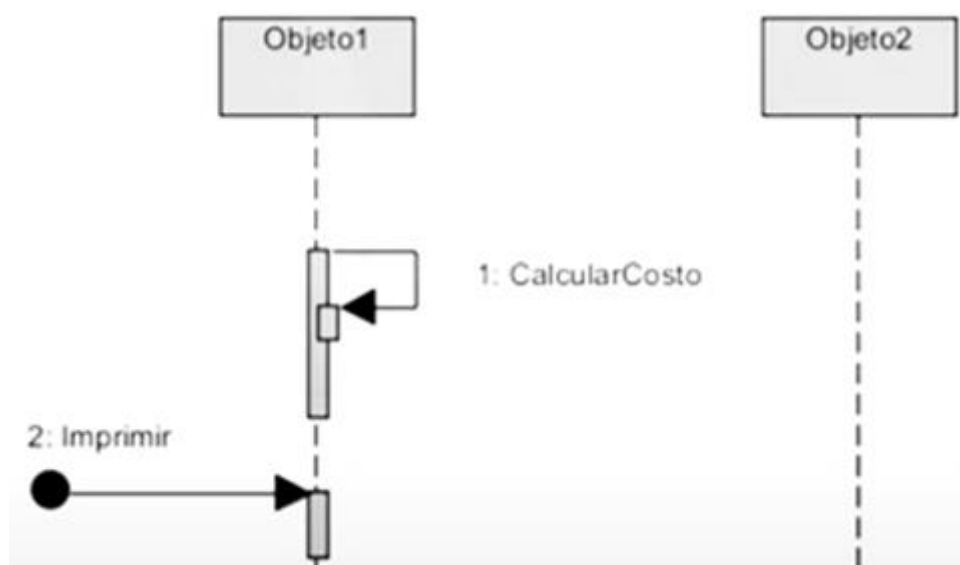
- La recursión lleva el cuadrado pequeño a diferencia del automensaje

Mensaje encontrado

Los **mensajes encontrados** en UML representan un mensaje que **llega a un objeto desde una fuente desconocida o no especificada** dentro del diagrama de secuencia. Se utilizan cuando no es relevante mostrar el origen del mensaje o cuando proviene de un sistema externo no modelado en el diagrama.

Características

1. **Se representan con una flecha que inicia fuera del diagrama**, apuntando a la línea de vida del objeto receptor.
2. **No se conoce o no se representa el origen del mensaje.**
3. **Se usan para indicar eventos externos** que afectan el sistema, como interrupciones o datos entrantes.



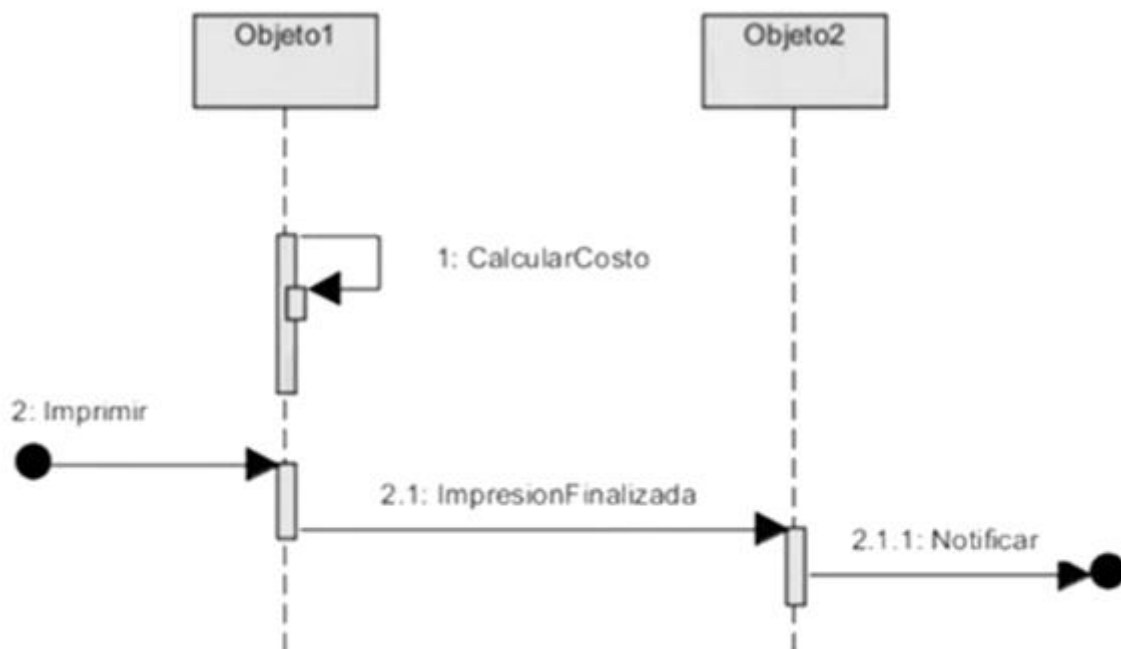
- Lleva una flecha de mensaje encontrado hacia el objeto y se representa con un círculo lleno
- No conocemos a quién envía mensaje, pues no forma parte del diagrama

Mensaje perdido

Los **mensajes perdidos** en UML representan un mensaje que **se envía desde un objeto, pero no se conoce o no se muestra su destino** en el diagrama de secuencia. Se utilizan cuando el destino del mensaje no es relevante, no está modelado o no se sabe si realmente lo recibe alguien.

Características

1. Se representan con una flecha que sale de la línea de vida de un objeto y apunta fuera del diagrama.
2. No se conoce o no se muestra el receptor del mensaje.
3. Se usan para modelar mensajes enviados a sistemas externos, redes, logs o eventos sin respuesta garantizada.



- Quién lo recibe no forma parte del diagrama

Diferencia entre Mensajes Encontrados y Perdidos

| Tipo de Mensaje | Representación | Descripción | Ejemplo |
|---------------------------|---|--|--|
| Mensaje Encontrado | Flecha que entra desde fuera del diagrama | Un mensaje llega desde una fuente desconocida | Un usuario recibe una notificación push sin mostrar el origen |
| Mensaje Perdido | Flecha que sale fuera del diagrama | Un mensaje se envía a un destino desconocido o irrelevante | Un sistema envía datos a una API sin saber si fueron recibidos |

Diagrama de Actividad

El **diagrama de actividad** en UML representa el **flujo de trabajo** o la **secuencia de actividades** en un proceso. Se usa para modelar **procesos de negocio**, **algoritmos**, o **flujos de eventos** dentro de un sistema

Características

- Muestra el flujo de control entre actividades en un proceso.
- Puede incluir decisiones, concurrencia y sincronización.
- Se usa para modelar procesos en software y negocios.
- Se parece a un diagrama de flujo, pero con elementos propios de UML.

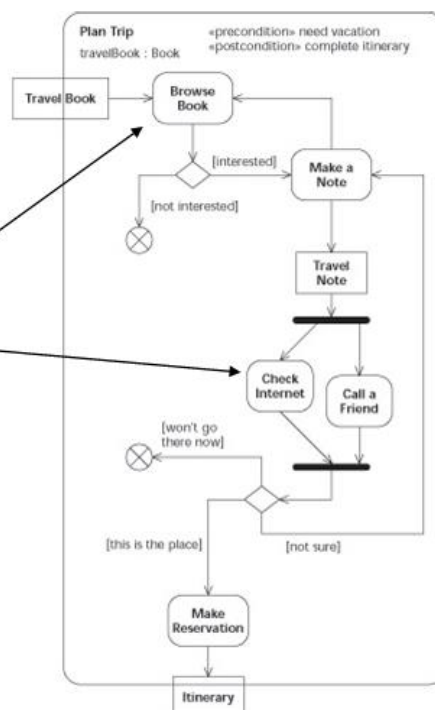
Elementos del Diagrama de Actividad

1. **Nodo de Inicio**
 - Representa el punto donde comienza la actividad.
2. **Actividades (Acciones)**
 - Representan tareas o procesos dentro del flujo.
 - Se dibujan como rectángulos redondeados.
3. **Flujo de Control**
 - Representado por flechas, indica el orden de ejecución.
4. **Decisión y Fusión**
 - Representadas con rombos, indican bifurcaciones (**if-else**).
 - Pueden fusionarse en un solo flujo después de la decisión.
5. **División y Unión (Fork & Join)**
 - Representadas con barras gruesas.
 - Modelan procesos que ocurren en paralelo (hilos o tareas concurrentes).
6. **Nodo de Fin**
 - Representa el final del flujo de actividad.

En resumen, podemos utilizar

- Nodos Básicos
- Flujos paralelos
- Nodo objeto
- Pins
- Transformación

acciones



actividad

El diagrama de actividad nos va a mostrar lo siguiente:

- Muestra lo que sucede durante una actividad u operación, proceso o caso de uso
- Recuerda a los diagramas de flujos
- La actividad es un proceso, cada paso del proceso es una relación

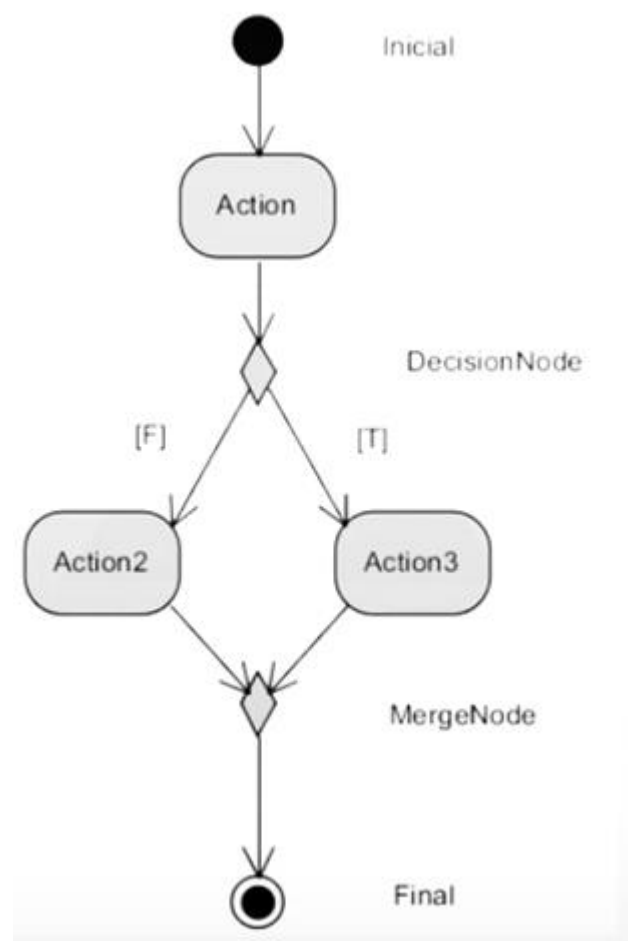
En síntesis, son los pasos que se tienen que seguir o llevar a cabo para realizar un proceso en particular, a estos pasos se los conocen como **acciones**

Nodos básicos

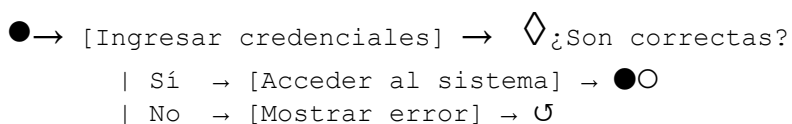
Los **nodos básicos** en un diagrama de actividad UML son los elementos fundamentales que representan el **flujo de ejecución** de un proceso. Se dividen en diferentes tipos según su función.

Tipos de nodos básicos

1. **Nodo de Inicio**
 - Representa el **punto de entrada** al proceso.
 - Solo puede haber **uno** en cada diagrama.
2. **Nodo de Acción o Actividad**
 - Representa una **tarea o proceso** dentro del flujo.
 - Se dibuja como un **rectángulo con bordes redondeados**.
 - Ejemplo: [Validar Usuario]
3. **Nodo de Decisión y Fusión**
 - Modela **bifurcaciones** (if-else) y **uniones** en el flujo.
 - Representado con un **rombo**.
 - Ejemplo: ¿Contraseña correcta?
 - Sí → Accede al sistema
 - No → Muestra error
4. **Nodo de División y Unión (Fork & Join)**
 - Representa tareas **paralelas o concurrentes**.
 - Se dibuja con una **línea gruesa horizontal o vertical**.
 - Fork: Divide un flujo en varias ramas.
 - Join: Reúne varias ramas en un solo flujo.
 - Ejemplo: Un sistema que **envía un email** mientras **registra un log**.
5. **Nodo de Finalización**
 - Indica el **fin del flujo de actividad**.
 - Puede haber **varios** en un diagrama.



Ejemplo: Autenticación de usuario



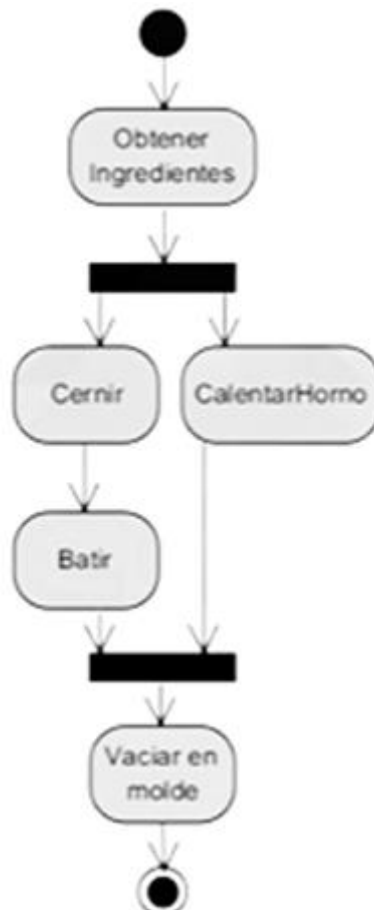
Flujos paralelos

Los **flujos paralelos** en un **diagrama de actividad UML** representan procesos que **se ejecutan simultáneamente**. Se utilizan cuando múltiples tareas pueden realizarse en paralelo en un sistema.

Dentro los flujos paralelos podemos encontrar los ya mencionados **Join y Fork**

Características

- Un flujo de entrada y se tiene dos o más de salida que suceden en paralelo
- Fork se usa para iniciar la concurrencia
- Join es usado para regresar a un solo flujo



A las representaciones de los nodos básicos, se agregan

- **Actividad en paralelo (Fork):** representado por el rectángulo en negro. Las actividades ocurren en paralelo
- **Nodo de Join:** representado por el rectángulo en negro. Las actividades vuelven a unirse en un solo flujo.

Ejemplo

Escenario: Un sistema de e-commerce procesa un pedido en paralelo:

- Registrar la orden
- Cobrar al cliente

- Preparar el paquete

Representación UML (simplificada):

```

● → [Pedido recibido] → — — (Fork)
    |→ [Registrar orden] →
    |→ [Cobrar cliente] →
    |→ [Preparar paquete] →
    — — (Join) → [Enviar pedido] → ●○
  
```

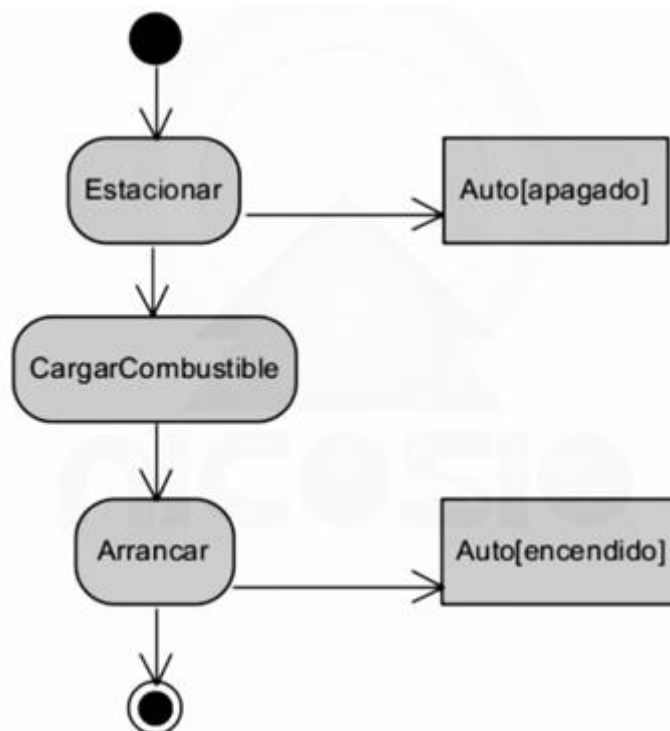
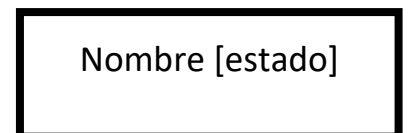
Aquí, las tres actividades (`Registrar orden`, `Cobrar`, `Preparar paquete`) **ocurren al mismo tiempo** y solo cuando todas terminan se continúa con `Enviar pedido`.

Nodo objeto

Un **nodo objeto** en un **diagrama de actividad UML** representa un **objeto que participa en el flujo del proceso**. Se usa para mostrar cómo los datos o entidades **se crean, modifican y transfieren** entre actividades.

Características

1. Representa un dato u objeto dentro del proceso.
2. Se representa con un rectángulo con la forma **Nombre del objeto [estado]**.
3. Puede estar conectado a actividades mediante flujos de objetos (flechas con datos).
4. Puede actuar como entrada o salida de una actividad.
5. Nos sirve para mostrar un objeto en particular que se relaciona con la actividad.
6. El objeto cambia de estado según lo que sucede en el proceso u operación.



Ejemplo

Escenario: Un sistema de compras en línea procesa pedidos.

- Un cliente crea un pedido.
- El sistema verifica el stock.
- Si hay stock, se confirma el pedido y se almacena.

Representación UML (simplificada):

```
● → [Ingresar pedido] → (Pedido) → [Verificar stock]
    | Sí → (Pedido confirmado) → [Guardar en base de datos] → ●○
    | No → [Mostrar error] → ●○
```

Aquí, los nodos `(Pedido)` y `(Pedido confirmado)` representan objetos que se **modifican y fluyen** entre las actividades.

Pins

Los **pins** en un diagrama de actividad UML son elementos que **representan los puntos de entrada o salida de datos en una actividad**. Se utilizan dentro de los **nodos de acción** para mostrar cómo se manipulan objetos o datos en el flujo del proceso.

Características

1. Indican cómo se usan los objetos en una actividad.
2. Se dibujan como pequeños cuadrados en los bordes de una actividad.
3. Pueden ser de entrada (reciben datos) o salida (producen datos).
4. Conectan flujos de objetos entre actividades.

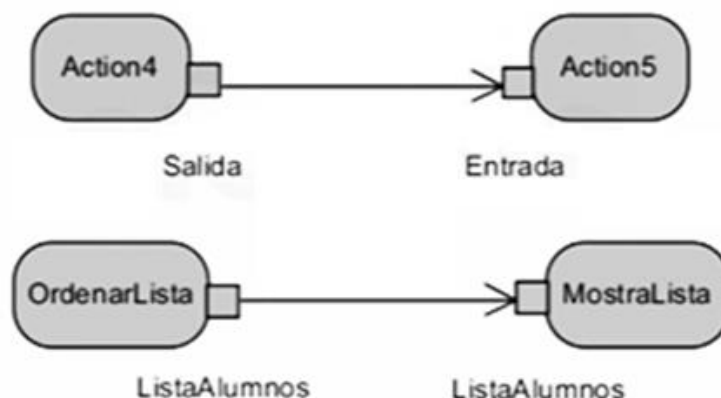
Tipos de Pins

1. Pin de Entrada

- Recibe un objeto o dato antes de que la actividad se ejecute.
- Ejemplo: `Pedido` como entrada para `Verificar Stock`.

2. Pin de Salida

- Devuelve un objeto o dato después de la ejecución de la actividad.
- Ejemplo: `Pedido confirmado` como salida de `Verificar Stock`.



- También podemos mostrar los objetos por medio de Pins.
- Pero en este caso indican que el objeto es una salida o una entrada.
- Por medio de ellos indicamos que un objeto sale de una acción y entra como parámetro a otra.

Ejemplo

Escenario: Un sistema procesa un pedido y lo almacena en la base de datos.

● → [Verificar Stock] ← (Pedido) → (Pedido Confirmado) → [Guardar Pedido] → ●○

Aquí:

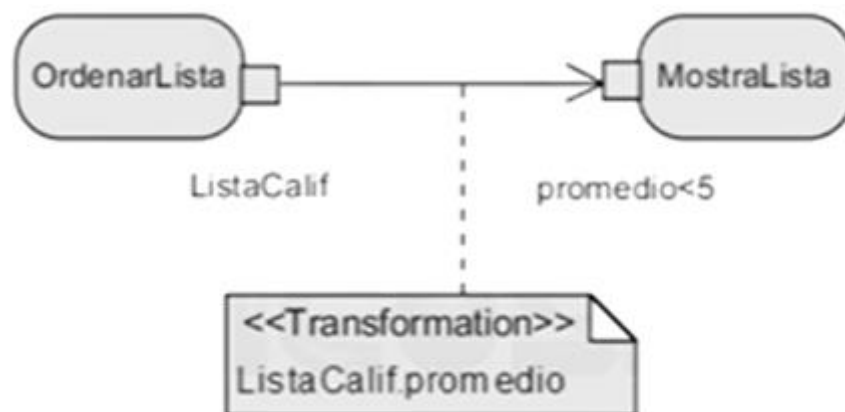
- (Pedido) entra como pin de entrada a [Verificar Stock].
- (Pedido Confirmado) es el pin de salida, que luego se usa como entrada para [Guardar Pedido].

Transformación

En UML, la transformación se refiere al cambio o conversión de un objeto en otro dentro de un diagrama de actividad. Se usa cuando un objeto pasa por una serie de modificaciones y termina con un estado o formato diferente.

Características

1. Ocurre cuando **un objeto cambia de forma, estado o representación**.
2. Se representa mediante **nodos de acción con pins de entrada y salida**.
3. Se pueden usar **flujos de objetos** para conectar el objeto antes y después del cambio.

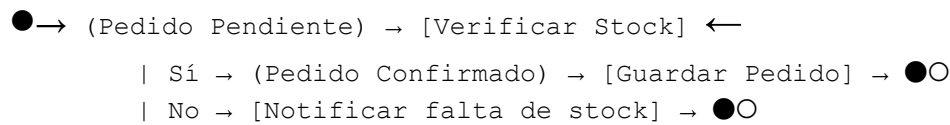


- Lo usamos para indicar que una acción solamente utilizará una parte del objeto enviado

En este caso únicamente vamos a utilizar el promedio de ListaCalif, y se mostraran para este caso aquellos con el condicional donde el promedio sea menor a 5.

Ejemplo: Un sistema de e-commerce convierte un **pedido pendiente** en un **pedido confirmado** después de verificar el stock.

Representación UML (simplificada):



Aquí, el objeto **"Pedido Pendiente"** se transforma en **"Pedido Confirmado"** tras la validación de stock.




Diagrama de comunicación

El **diagrama de comunicación** (antes llamado **diagrama de colaboración**) es un tipo de diagrama de interacción en UML que **muestra cómo los objetos colaboran entre sí** mediante el **intercambio de mensajes** para llevar a cabo un proceso o comportamiento.

Características Clave

- Representa **objetos (instancias)** conectados por **enlaces (relaciones)**.
- Los **mensajes** se indican con flechas **numeradas** según el orden de ejecución.
- Se enfoca en la **estructura** de la colaboración más que en el tiempo (a diferencia del diagrama de secuencia).
- Muy útil para visualizar cómo **los objetos se comunican entre sí** en tiempo de ejecución.

Elementos Principales

| Elemento | Descripción |
|---|---|
|  Objetos | Representados como rectángulos con nombre (por ejemplo: :Usuario). |
|  Enlaces | Líneas que conectan objetos para indicar una relación de comunicación. |
|  Mensajes | Flechas con números y etiquetas, que indican qué mensaje se envía y su orden. |

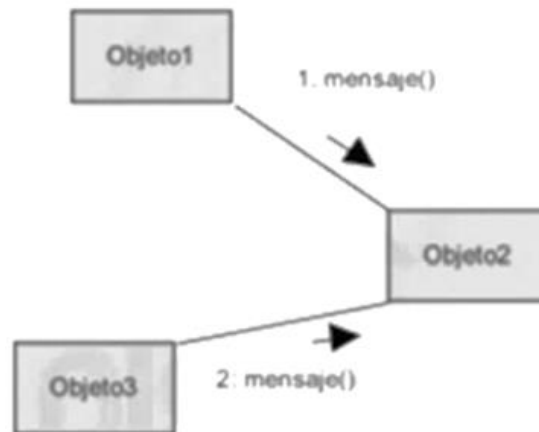
¿Cuándo se usa?

- ✓ Para mostrar **la estructura de la comunicación** entre objetos en una situación concreta.
- ✓ Cuando te interesa **ver cómo están relacionados los objetos**, no solo el orden temporal de los mensajes.
- ✓ Muy útil en el diseño orientado a objetos y para **complementar el diagrama de secuencia**.

A diferencia del diagrama de secuencia que hace enfoque en el tiempo, el diagrama de comunicación pone más énfasis en las ligas de los objetos y en el orden que suceden, pero no sabemos en qué momento o tiempo va a suceder.

Partes

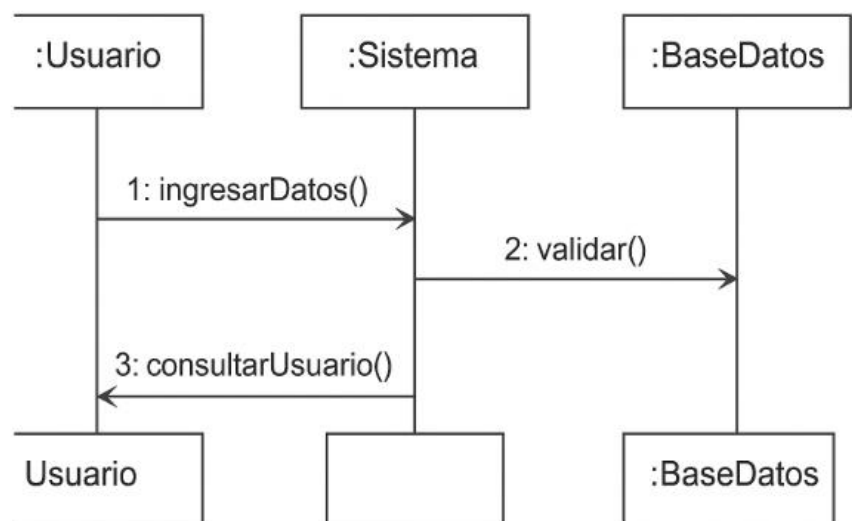
- Objeto
 - Interactúan entre ellos
 - Se muestra de la forma usual por medio de un rectángulo
- Liga
 - También se conocen como ligas de comunicación
 - Se muestran por medio de una línea
 - Si la línea conecta dos objetos eso significa que los objetos pueden comunicarse/interactuar entre ellos.
- Mensajes
 - Se colocan a lo largo de las ligas
 - Muestra la comunicación entre los objetos
 - Se usa una flecha
 - Los mensajes se nombran
 - Tienen un número que nos indica el orden de los mensajes



Ejemplo

Usuario :Usuario
Sistema :Sistema
BD :BaseDatos

1: ingresarDatos()
2: validar()
3: consultarUsuario()
4: devolverResultado()



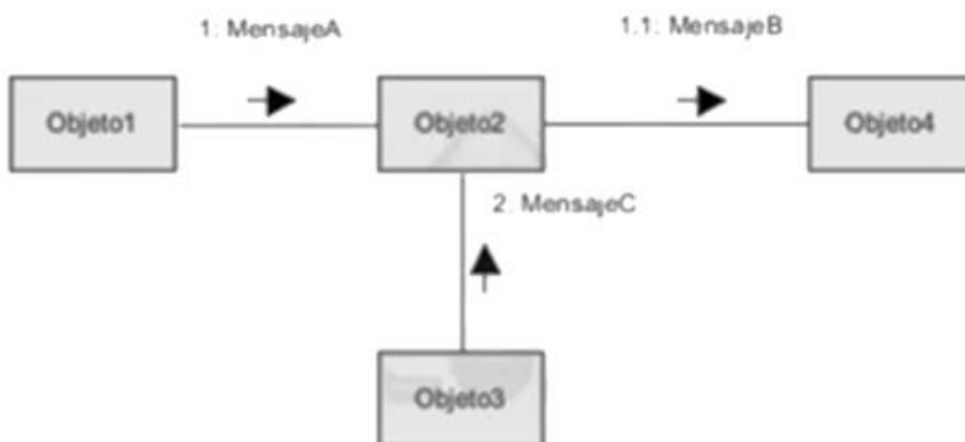
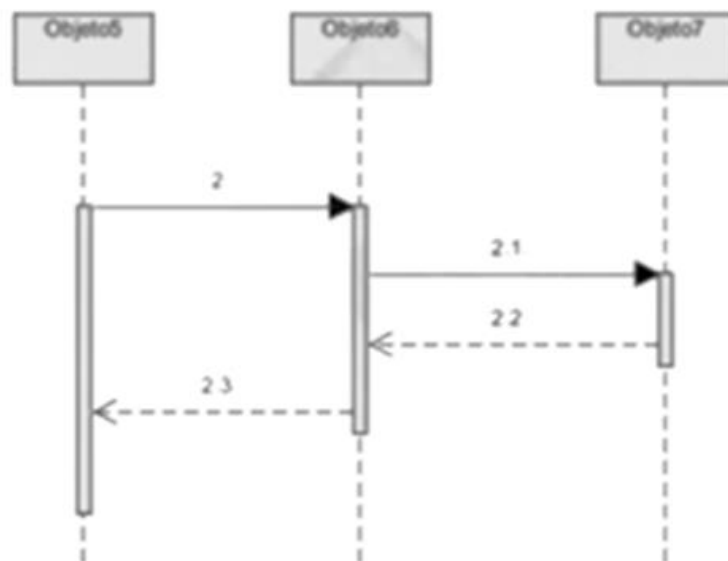
Mensajes anidados

Los **mensajes anidados** en un **diagrama de secuencia UML** representan llamadas **dentro de otras llamadas**, es decir, cuando una operación invoca a otra **como parte de su ejecución**.

Un **mensaje anidado** es una **subllamada** que se realiza **mientras se está ejecutando otra llamada**. Ayuda a representar cómo una acción **desencadena internamente otras acciones**.

Características

- Hay que tener cuidado con la numeración de los mensajes para mostrar el orden correctamente
- n.m



Ejemplo

Supongamos este flujo:

1. Usuario llama a Sistema para iniciar sesión.
2. Sistema, al recibir el mensaje, **llama a BaseDatos** para validar credenciales.

Esto se vería así:

Usuario → Sistema: iniciarSesion()

Sistema → BaseDatos: validarUsuario()

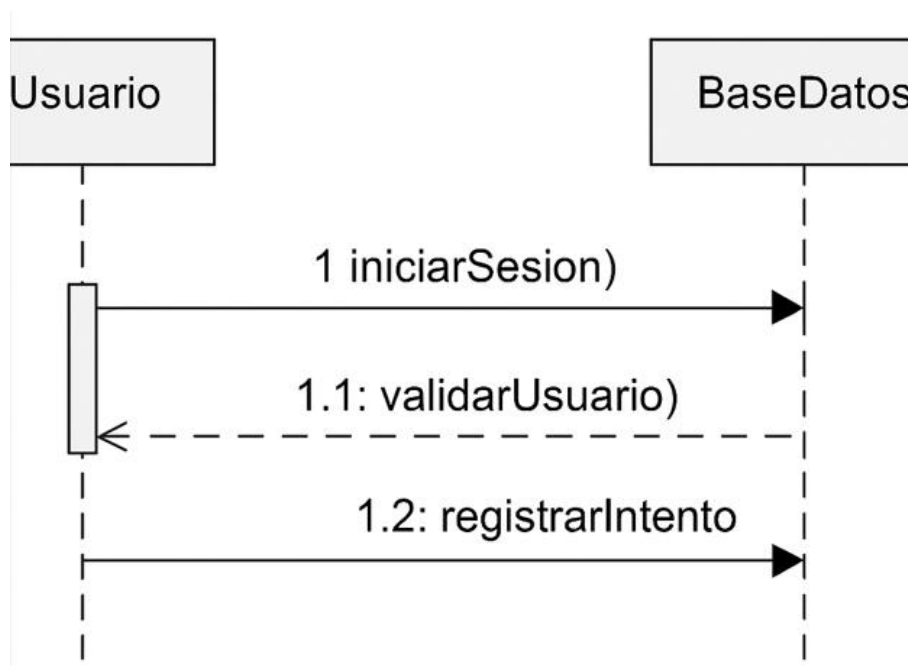
En este caso, validarUsuario() es un **mensaje anidado dentro de** iniciarSesion().

¿Cómo se muestran en el diagrama?

- En el **diagrama de secuencia**, los mensajes anidados aparecen **debajo** del mensaje principal, alineados en el tiempo.
- En un **diagrama de comunicación**, se muestran como **subnúmeros**:
 - 1: iniciarSesion()
 - 1.1: validarUsuario()
 - 1.2: registrarIntento()

¿Para qué sirven?

- Para **mostrar el detalle interno** de una operación.
- Para representar **procesos jerárquicos** o dependientes.
- Para seguir la **cadena de llamadas** dentro de un sistema.



Auto mensaje

Un **automensaje** es un tipo de mensaje en un **diagrama de secuencia UML** donde un **objeto se envía un mensaje a sí mismo**. Representa una llamada a un **método interno o recursivo** dentro del mismo objeto.

Un automensaje es cuando **una instancia de clase invoca una operación propia**, como si dijera: “yo me doy una orden a mí mismo”.

¿Cómo se representa?

- Como una **flecha curva** que **sale y vuelve al mismo objeto** (en el diagrama de secuencia).
- Se dibuja hacia abajo, como cualquier otro mensaje, para reflejar la secuencia temporal.



¿Cuándo se usa?

- ✓ Cuando una operación **activa otro método** de la **misma instancia**.
- ✓ Para representar **recursividad** o **lógica interna encapsulada**.
- ✓ En estructuras como **búsquedas**, **validaciones internas**, etc.

Ejemplo

Supongamos que una clase **Contador** tiene un método **incrementar()**, y dentro de ese método llama a **verificarLimite()**.

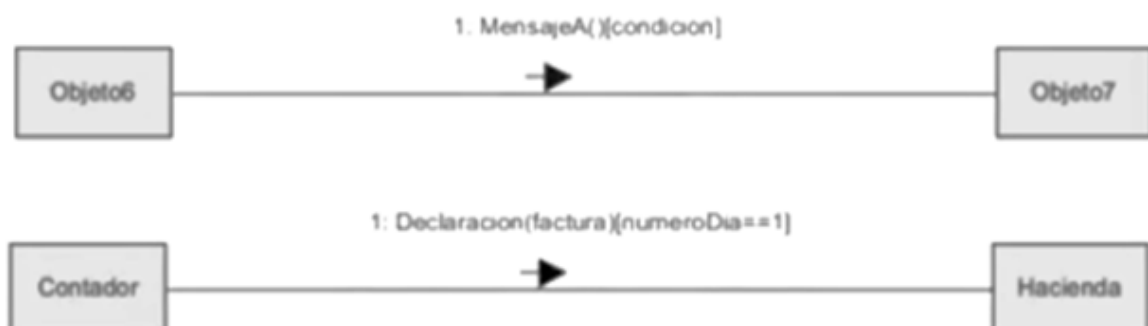
Mensajes condicionales

En un **diagrama de secuencia UML**, los **mensajes condicionales** son aquellos que **se envían solo si se cumple una determinada condición**. Representan **ramas de decisión** dentro del flujo de mensajes.

“Si pasa tal cosa, entonces mando este mensaje.”

Características

- El mensaje se lleve a cabo solo si la condición se evalúa como verdadera



¿Cuándo se usan?

- Para mostrar **decisiones lógicas** o **alternativas de flujo**.
- Para modelar respuestas como **sí/no**, **éxito/error**, **opciones múltiples**.

Ejemplo

Supongamos un proceso de inicio de sesión:

Usuario → Sistema: ingresarCredenciales()

Sistema → BD: verificarUsuario()

BD → Sistema: [usuario válido] true

→ [true] mostrarPantallaPrincipal()

→ [false] mostrarErrorLogin()

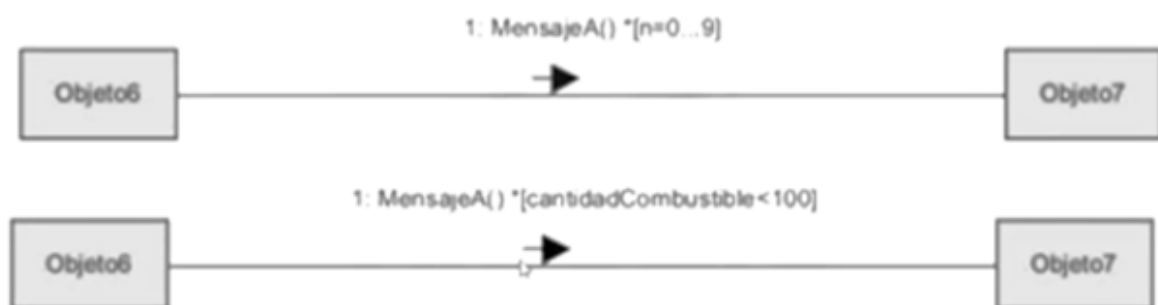
Ciclos para mensajes

En UML, específicamente en los **diagramas de secuencia**, un **ciclo para mensajes** representa una **repetición o bucle** de uno o varios mensajes, es decir, cuando una acción se **realiza varias veces** bajo una condición o durante un intervalo.

“Repite este mensaje mientras se cumpla cierta condición.”

Características

- Los usamos cuando vamos a invocar un mensaje múltiple veces
- Se usa * para indicar ciclo y luego la condición del ciclo



¿Cuándo usar ciclos en UML?

- Cuando querés mostrar que **una misma operación se repite** (por ejemplo: un for, while o foreach).
- Para modelar **procesamientos por lote, iteraciones sobre colecciones, o reintentos**.
- Para mantener el **flujo claro y ordenado** sin tener que repetir los mismos mensajes muchas veces.

Ejemplo

Un usuario consulta sus 5 últimas compras:

loop [para cada compra en historial]

→ mostrarCompra()

Mensajes paralelos

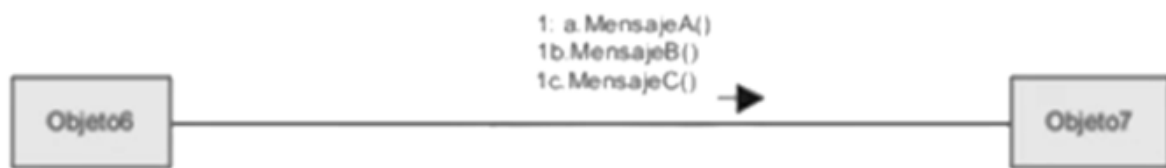
En UML, especialmente en **diagramas de secuencia**, los **mensajes paralelos** representan **acciones o mensajes que se ejecutan al mismo tiempo** (o al menos, que **no dependen entre sí para ejecutarse**).

“Son mensajes que se pueden enviar en paralelo a diferentes objetos, sin esperar que uno termine para comenzar el otro.”

Características

Aunque UML no tiene una notación súper explícita como para los bucles o condicionales, los mensajes paralelos se muestran como **líneas de mensaje que parten de un mismo punto de activación y siguen en paralelo hacia distintos objetos**.

Los objetos envían múltiples mensajes al mismo tiempo hacia el mismo objeto



¿Cuándo usar mensajes paralelos?

- Cuando dos o más acciones **pueden ocurrir al mismo tiempo**.
- Cuando las operaciones son **independientes entre sí**.
- Para representar **conurrencia o hilos en ejecución simultánea**.

Ejemplo práctico:

Un sistema de compras realiza en paralelo:

- Enviar correo de confirmación
- Registrar la venta en base de datos

Tabla de contenido

| | |
|--|----|
| Diagrama de objetos | 2 |
| Características principales..... | 2 |
| Conectando objetos | 2 |
| Diagrama de estados | 4 |
| Elementos principales | 4 |
| Usos del diagrama de estados | 4 |
| Entendiendo el concepto, estado y transición..... | 4 |
| Actividades | 5 |
| Diagrama de secuencia | 6 |
| Usos del diagrama de secuencia | 6 |
| Características | 6 |
| Elementos principales | 7 |
| Clasificación | 8 |
| Mensaje de llamada | 8 |
| Mensaje asíncrono | 9 |
| Diferencia entre Mensajes Sincrónicos y Asíncrónicos | 10 |
| Automensaje | 10 |
| Mensaje recursivo | 10 |
| Mensaje encontrado | 11 |
| Mensaje perdido | 12 |
| Diferencia entre Mensajes Encontrados y Perdidos..... | 12 |
| Diagrama de Actividad..... | 13 |
| Características | 13 |
| Elementos del Diagrama de Actividad | 13 |
| Nodos básicos..... | 14 |
| Tipos de nodos básicos | 14 |
| Flujos paralelos..... | 15 |
| Nodo objeto | 16 |
| Pins | 17 |
| Transformación | 18 |
| Diagrama de comunicación | 19 |
| Partes | 20 |
| Mensajes anidados | 21 |

| | |
|------------------------------|----|
| Auto mensaje | 22 |
| Mensajes condicionales | 23 |
| Ciclos para mensajes..... | 24 |
| Mensajes paralelos | 25 |