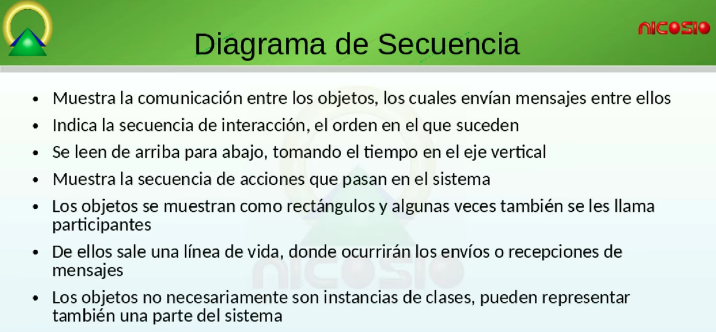
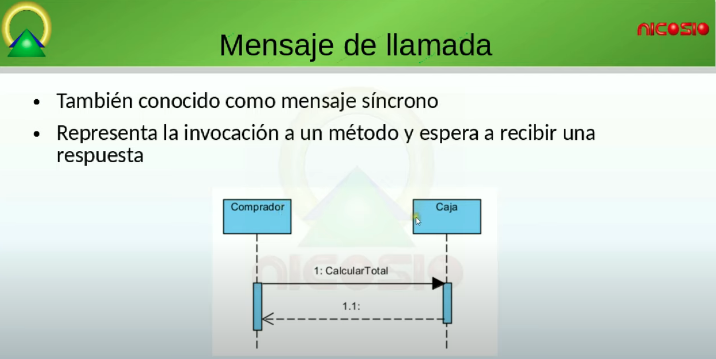
**Diagrama de Secuencia**



**Ejemplo s**



Tenemos dos (2) ***objetos:*** **comprador y caja**. A partir de dichos objetos, se puede visualizar que surjen, para cada uno, una línea vertical (de cada objeto siempre sale una línea vertical); esas *líneas verticales* son las llamadas: ***líneas de vida.*** No es casual que las líneas sean verticales; pues, se debe recordar, que el *tiempo* será leído de arriba hacia abajo.

**Mensaje de llamada**

El mensaje de llamada se representa por medio de una flecha puntero de cabeza o punta rellena en la que se solicita algo; en este caso, lo que se está pidiendo (invocando) con ella es un método de otro objeto para que, éste último, le pase una respuesta a raiz de dicho llamado; en este caso, es el objeto *comprador* quien está invocando un método del objeto *caja.* Al mensaje enviado, donde se solicita dicho método, se le asigna un nombre; es decir, al metodo solicitado se le pasa siempre un nombre; ese nombre que recibe se le conoce como: *etiqueta.* En este caso, la etiqueta del método solicitado es: *CalcularTotal.*

Entonces, hasta ahora, *comprador* está invocando de *caja* el método *CalcularTotal.* Ahora, si se ha dado cuenta, *CalcularTotal* viene acompañado por un número, este número se coloca de manera intencional para, justamente, conocer el orden y la *secuencia de interacción* en la cual se están llevando a cabo los mensajes; entonces, como la etiqueta *CalcularTotal* es nuestro primer mensaje se le asigna un número *1.*

Un parentesis: Las etiquetas pueden venir acompañadas de parametros en caso que, la solicitud de nuestro método, requiera de una respuesta más precisa, puntual o más concreta. Por ejemplo, si el método *CalcularTotal* tuviera parámetros en su solicitud se vería así: *CalcularTotal(parámetros)* -entre parentesis se colocan los parámetros de interés.

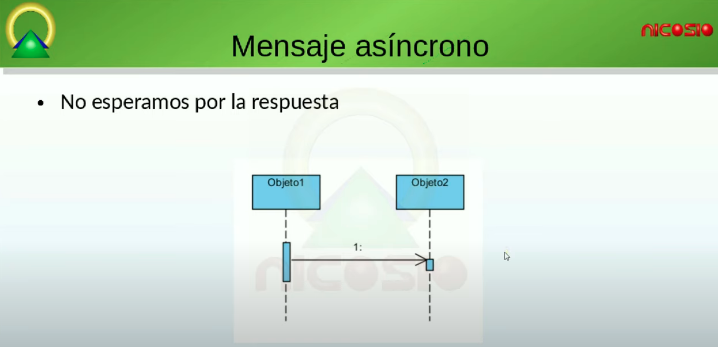
Retomando, en el mensaje de llamada hay un objeto A que invoca un método del que se espera una respuesta asistida por un objeto B.

Luego de un tiempo, objeto B responde a objeto A por medio de un mensaje de respuesta a su solicitud 1.

La respuesta a la solicitud 1. se representa númericamente como *1.1* debido a que su existencia depende o se debió inicialmente de la solicitud del mensaje de llamada; entonces, el primer *1* es para dejar claro que la respuesta se da en relación a una solicitud número 1 (justamente); mas, como es la primera respuesta a esa solicitud 1, entonces, se pone: *1.1* (y no 1.2 o 1.3, etc).

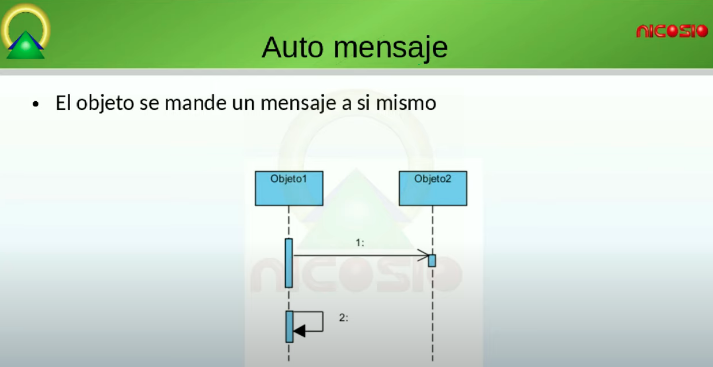
Ahora, la respuesta a ese mensaje de llamada se representa, gráficamente, con una una flecha invertida; pero, sin su cabeza o punta rellenada (tal como se ve en el ejemplo).

**Ejemplo 2: Mensaje asíncrono**



Acá no se espera respuestas. Si bien Objeto A manda un mensaje, invoca un método del objeto B, no se espera una mensaje de respuesta por parte de Objeto B en cuestión.

**Ejempl o 3: Auto Mensaje**



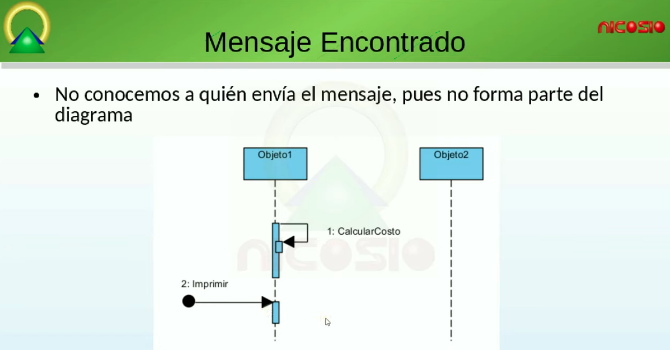
Acá el Objeto A se manda un mensaje así mismo. Atrás su representación gráfica. El auto-mensaje también se puede rotular, pasarle una etiqueta, justo después de la enumeración número 2., donde se explicaría de qué trata el mensaje particular; por ejemplo, “2: descripción, por medio de etiqueta, de este auto-mensaje”.

**Ejempl o 4: Mensaje recursivo**



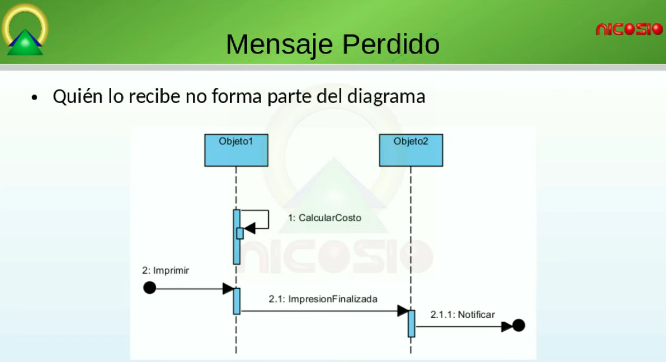
Es prácticamente un auto-mensaje con recursión, visualmente la diferencia se nota por la existencia de un *mini-rectangulo* superpuesto sobre otro mayor en la misma línea de vida del Objeto A. Este *mini-rectangulo* nos indica recursión. Lo otro particular que me causa intriga es que, acá, sí se evidencia una *auto-solicitud* de un método, el cual es: *CalcularCosto.*

**Ejempl o 5: Mensaje encontrado**



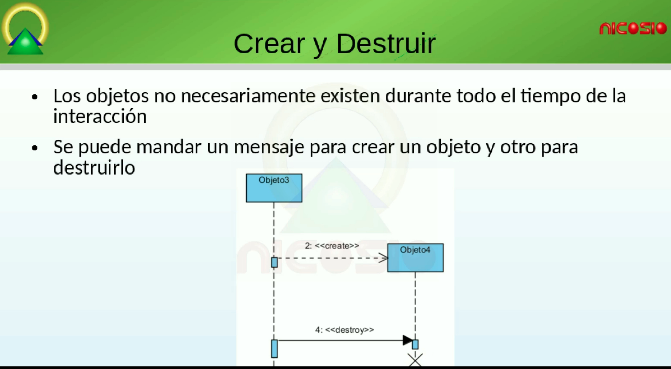
Un Mensaje Encontrado consiste en que, uno de nuestros Objetos que hacen parte de nuestro diagrama, recibe un mensaje donde se *invoca* un método; pero, el Objeto que envía dicho mensaje realmente no hace parte del diagrame, sí del sistema en su conjunto, pero no del diagrama (está fuera de nuestro análisis dentro del diagrama secuencia); entonces, sepa siempre que, a ese Objeto *unknown* se le representa con un circulo negro relleno y que, luego, se complementa con una flecha puntera *que apunta* hacia el Objeto de interés que sí hace parte de nuestro diagrama y al que se le solicita la respuesta de un método; en este caso, el método es: *Imprimir* (tal como se ve en en nuestro ejemplo).

**Ejempl o 6: Mensaje pérdido**



Un Mensaje Perdido es todo lo contrario a un mensaje Encontrado; es decir, seremos nosotros los que vamos a tener un Objeto dentro de nuestro diagrama que enviará un mensaje, pero, quién lo estará recibiendo, será un Objeto que no hace parte del diagrama en sí mismo (sí hace parte del sistema, mas no del diagrama) y, recuerde, todo Objeto que no haga parte del diagrama se representa con un circulo relleno de color negro. En nuestro caso el Mensaje Pérdido será el método que lleva por etiqueta el nombre de: *Notificar;* se espera entonces que, el método invocado *Notificar*, sea respondido por un Objeto que está por fuera del diagrama.

**Crear & Destruir**



Quien tiene la facultad de crear y destruir Objetos son los mismos Objetos ya existentes; por ejemplo, si un Objeto A crea un nuevo Objeto será el mismo Objeto A quién pueda destruirlo eventualmente (no otro Objeto). El mecanismo es más o menos así: un Objeto A ya existente puede mandar un mensaje-sentencia de creación, que es ***<<create>>,***con el cual se está *instanciando* un nuevo Objeto B. Pero, adicionalmente, podría ser el mismo Objeto A quién decidiría, si así lo quiere, la destrucción del Objeto B que creó; esto por medio de un mensaje-sentencia de destrucción, el cual es ***<<destroy>>.***

Ahora, gráficamente el mensaje-sentencia *creación* emitido se ve así:



Un Objeto A ya existente, y en algún momento de la trayectoria de su línea de vida, decide *crear* un nuevo objeto B por medio de un nuevo *mini-rectangulo.* Entonces, el mensaje-sentencia no está solo, se acompaña de una flecha que apunta al nuevo Objeto B creado. Es preciso decir que el trazado de la flecha no es completamente limpio (tiene intermitencia) y su cabeza o punta no es rellena.

Luego, el objeto B creado a partir de un Objeto A, adquiere su propia vida y tiene sus propias interacciones (puede emitir como recibir mensajes, etc); después, eventualmente con el tiempo, Objeto A ya no necesitará más dentro del diagrama a Objeto B y puede proceder a destruirlo.

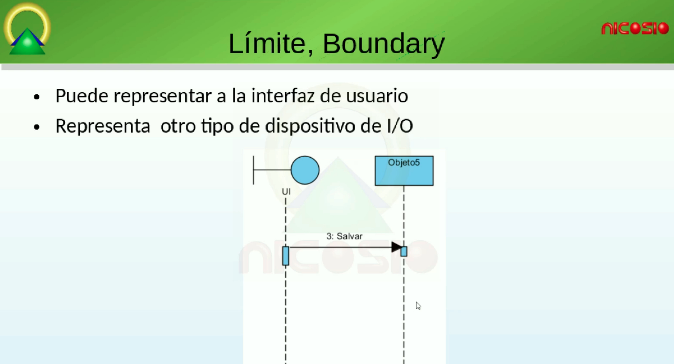
Ahora, gráficamente el mensaje-sentencia *destrucción* emitido se ve así:



De nuevo, el Objeto A que creó a Objeto B, en algún momento de la trayectoria de su línea de vida, decide ahora *destruir* al Objeto B mismo; esto mediante un *mini-rectangulo* también. Entonces, el mensaje-sentencia no está solo, se acompaña de una flecha que apunta al Objeto B ya existente. Ahora, el trazado de la flecha sí es completamente limpio y su cabeza o punta está rellena. Al final, se coloca una X al cierre de la línea de vida del Objeto B ya removido.

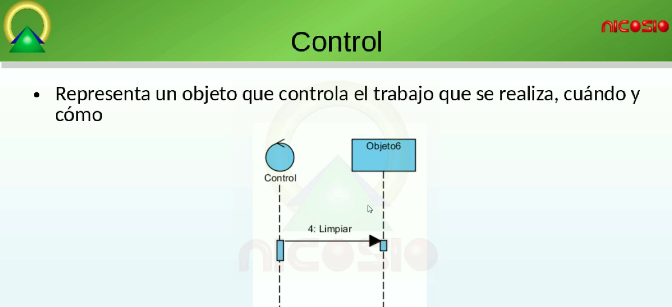
**Límite, Boundary**

Usualmente el *límite* representa a la UI; pues, el usuario como tal, desde luego, también puede emitir mensajes, solicita respuestas (en este caso, *salvar*) a los Objetos para que éstos puedan ejecutar alguna acción en particular.



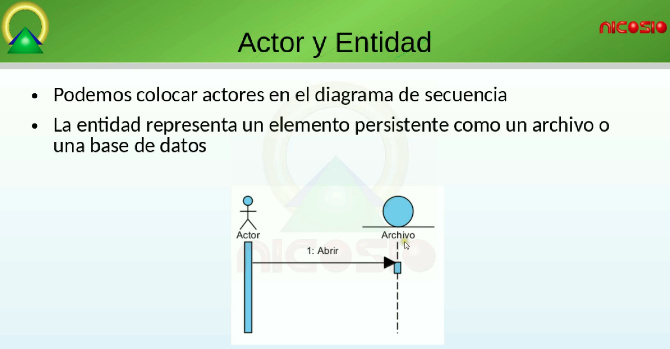
El límite , en este caso, está representando a una interfaz de usuario (User Interface, UI); es decir, a un usuario como tal . Entonces, el usuario (ui) en algún momento de su línea de vida, puede solicitar respuestas a un Objeto mediante un tipo de mensaje que le esté enviando.

**Control**



**Actor y Entidad**

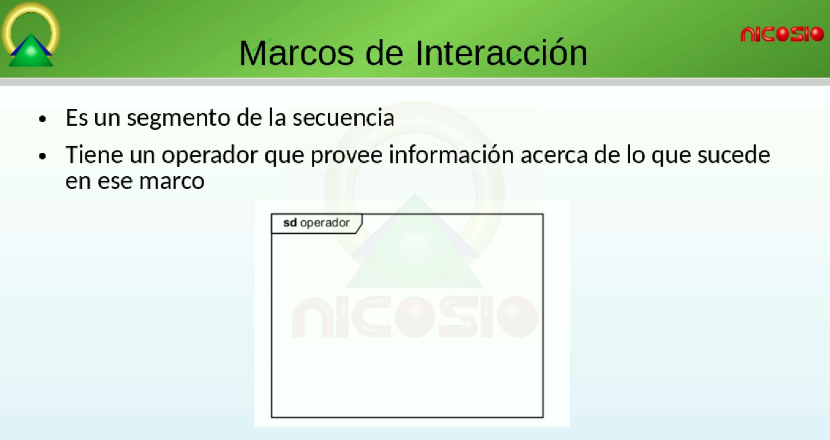
Además de Objetos, en nuestro diagrama de secuencia podemos colocar Actores; en ese sentido, los *actores* también podrían emitir mensajes a los diferentes Objetos de su diagrama y poseen, al igual, una línea de vida. *La representación del Actor es igual a la representación que ve, visualmente, en UML.*



En los diagramas de secuencia también podremos notar la existencia de Objetos de tipo Entidad, éstas sirven para almacenar el estado de un Objeto en cuestión y, en definición, se conocen por ser *elementos persistentes.* Su representación visual es:  y, en este caso, la entidad que tratamos es un Archivo. 

Siempre se pone por debajo, de la figura de *Entidad,* su indicación (qué entidad como tal se está tratando, en este caso, un *archivo*). En contexto, hay un actor que le está mandando un mensaje de *abrir* a nuestro Objeto de tipo Entidad llamado Archivo.

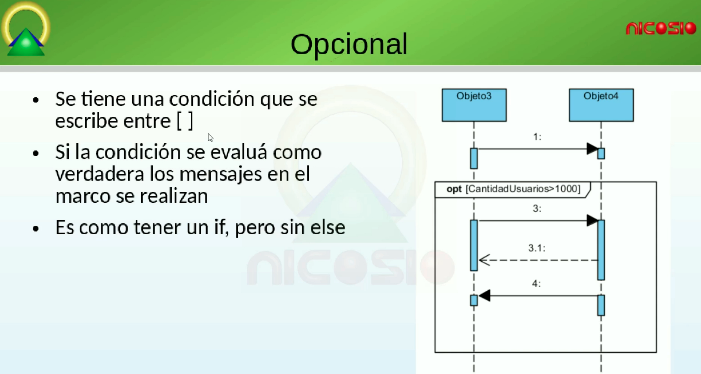
**Marcos de Interacción**



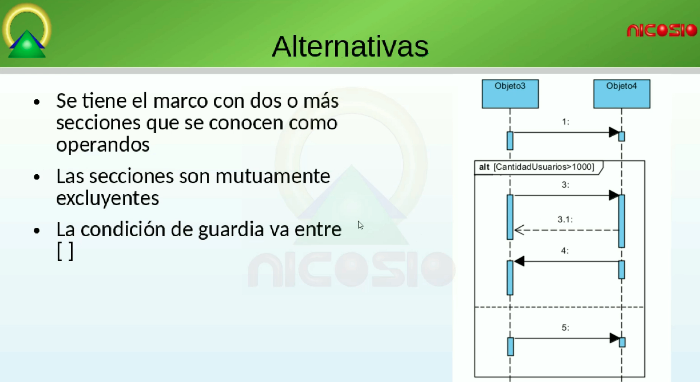
Los marcos de interacción nos van a servir para que nosotros podamos *segmentar* la secuencia, “*partir* el diagrama” antes de avanzar con su secuencia natural; más precisamente, con los marcos de interacción se definen puntos o espacios transitorios en el tiempo donde se conforman estructuras básicas de programación y de decisión antes de continuar con la natural fluidez del diagrama de secuencia en cuestión. Es común encontrarse como *estructura básica de programación,* dentro de un marco de interacción, estructuras de tipo condicional y/o cíclica; los cuales, trabajan de la mano con un *operador.*

**Marcos de Interacción de estructuras de tipo *condicional.***

**Opcional (opt).**

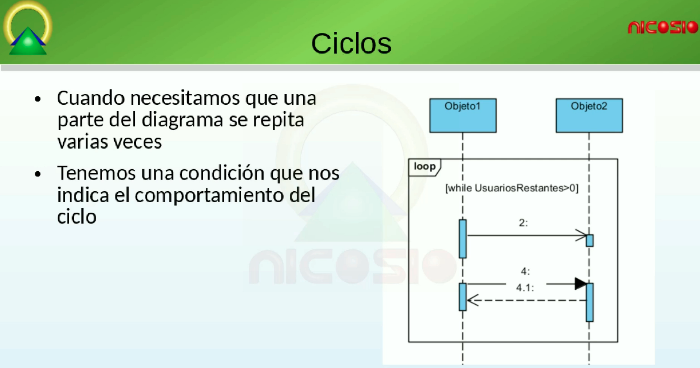


**Alternativas (alt).**



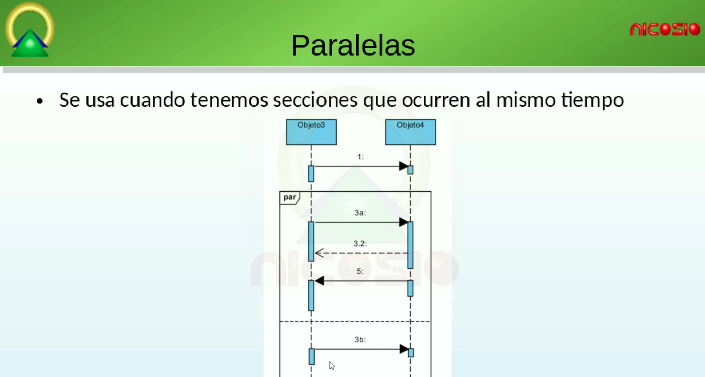
**Es como tener un if; pero, ahora sí, con un Else. Si no se cumple una sección, se cumple otra.**

**Marcos de Interacción de estructuras de tipo *cíclica.***



En caso que no se desee trabajar con una estructura cíclica de tipo *while,* podemos establecer un intervalo *(range #a to #b)* que haga el simil de un bucle *for* (en caso que se desee trabajar, justamente, con una estructura cíclica for).

**Marcos Paralelos**



Cuando queremos que dos o más de dos secciones, relativamente independientes entre sí, se ejecuten al mismo tiempo y no de forma secuencial; es decir, que primero no tenga que ejecutarse *A* para que luego se ejecute *B;* se establecen, entonces, marcos *paralelos.*

**Otros marcos**

Neg - Para una interacción inválida

Ref - Hace una referencia a una interacción que está definida en otro diagrama

Region - Es una región crítica, es decir que solamente un hilo puede ejecutarla a la vez

Sd - Indica que es un diagrama de secuencia y envuelve a todo el diagrama.

***Ante todo ...***

***un Diagrama de Secuencias muestra las interacciones, que hay entre objetos y actores, en el orden que son llevadas a cabo. En otras palabras, muestra una secuencia de eventos.***