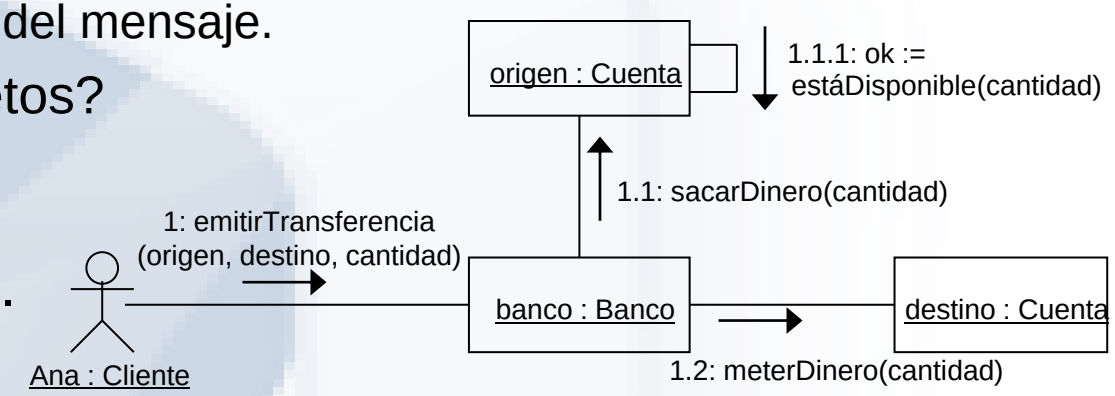
# Modelado Dinámico Avanzado

Universidad
Carlos III de Madrid

# *Enlaces de Comunicación: Estructurales y Contextuales*

- El envío de mensajes tiene lugar en un **contexto** determinado, normalmente la **ejecución de una operación**. El contexto delimita los destinos válidos.
- ¿A quién puedo enviar un mensaje? **Ley de Demeter:**
  - A un objeto conectado mediante un **enlace** navegable (instancia de asociación).
  - A un objeto recibido como **parámetro** en esta activación.
  - A un objeto creado localmente en esta ejecución, o **variable local**.
  - A mí mismo, el **emisor** del mensaje.
- ¿Cómo nombrar los objetos?
  - Nombre arbitrario.
  - Valor identificador.
  - Nombre de parámetro.
  - Nombre de variable.

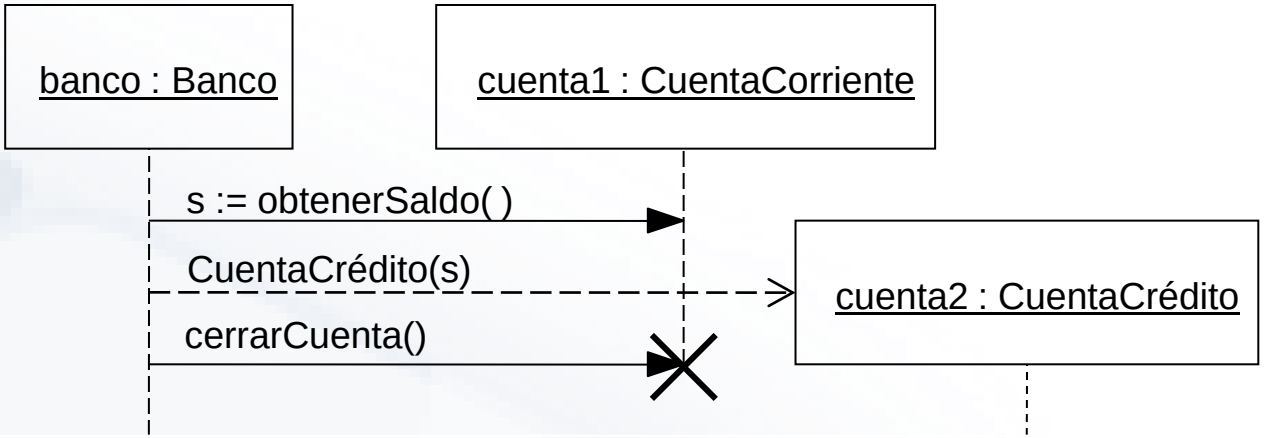


# *Diagramas de Interacción vs. Diagramas de Clases*

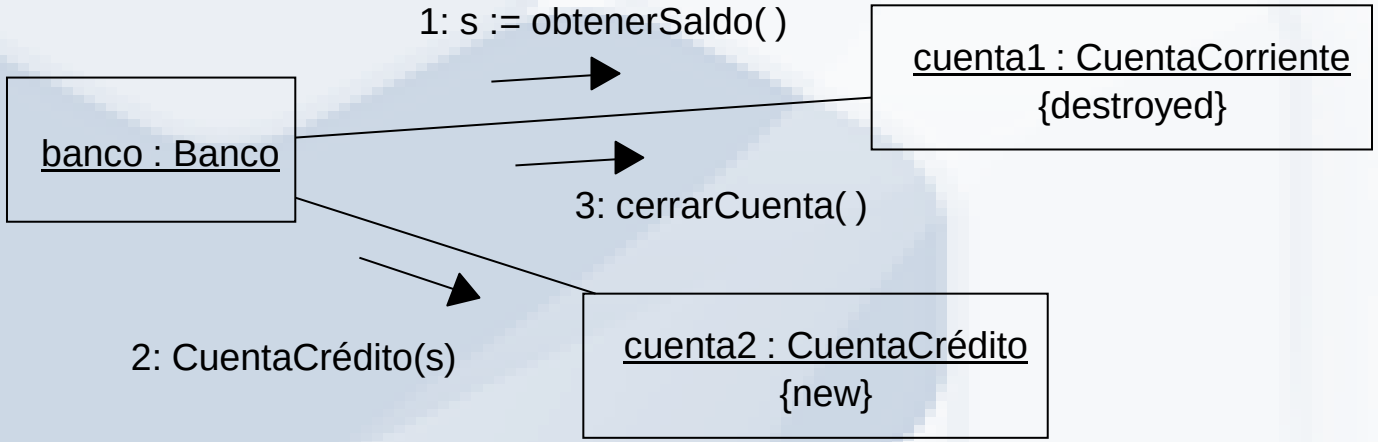
- Un diagramas de clases **especifica** la estructura del sistema, que sirve de base para su comportamiento: asociaciones, operaciones.
- Un diagrama de interacción **ilustra** un comportamiento posible del sistema.
- Las diversas interacciones ayudan a detectar las clases, asociaciones y operaciones requeridas: **reglas de coherencia**.

| **Diagramas de interacción (modelo dinámico)** | **Diagramas de clases (modelo estático)** |
|---|---|
| Línea de vida | Instancia de clase |
| Conector | Instancia de asociación (excepto enlaces contextuales) |
| Mensaje | Operación o señal visible en la (super)clase receptora |

# Creación y Destrucción de Objetos




Cancelar una cuenta y transferir el saldo a una nueva de otro tipo.




{transient}
=
{new} + {destroyed}

# *Fragmentos Combinados*

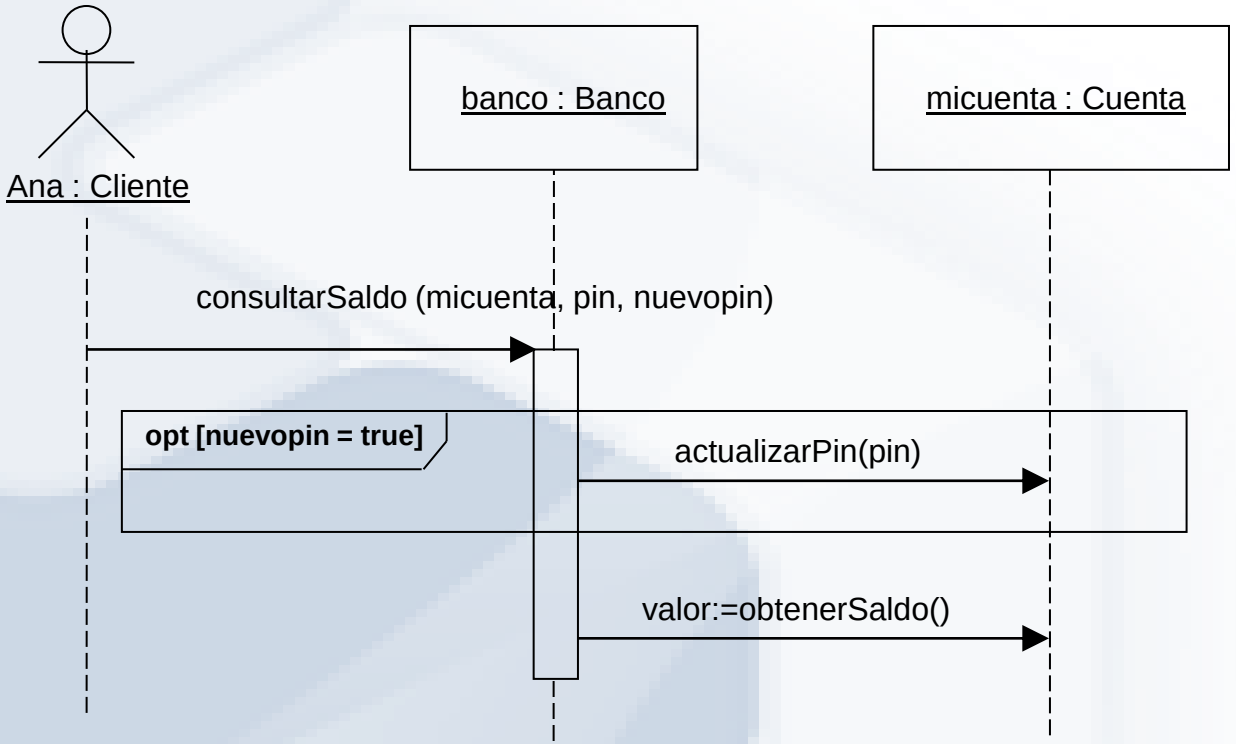
- Un fragmento de interacción es un trozo de interacción que conceptualmente constituye en sí una interacción.
- Un fragmento combinado es un fragmento de interacción que se define con un operador y sus correspondientes operandos.
- Los operadores que se van a tratar en este curso son: **opt**, **alt** y **loop**.
- Un operando es un fragmento de interacción que **opcionalmente** puede llevar una **condición** (guarda) con la siguiente sintaxis: '[' (<condición> | 'else') ']'.
- Un operando contiene a su vez un conjunto ordenado de fragmentos de interacción.
- Los distintos operandos del fragmento combinado se separan por líneas discontinuas horizontales.

# *Ejecución Opcional (I)*

- La ejecución opcional en UML 2 se realiza con el fragmento combinado cuyo operador es **opt**.
- Uso de **variables** locales, valores devueltos por mensajes anteriores, etc. en la condición (guarda).
- El operador **opt** permite que se dé el operando si se cumple la condición.
- Un único operando
- Posibilidad de añadir **comentarios** explicativos al margen.

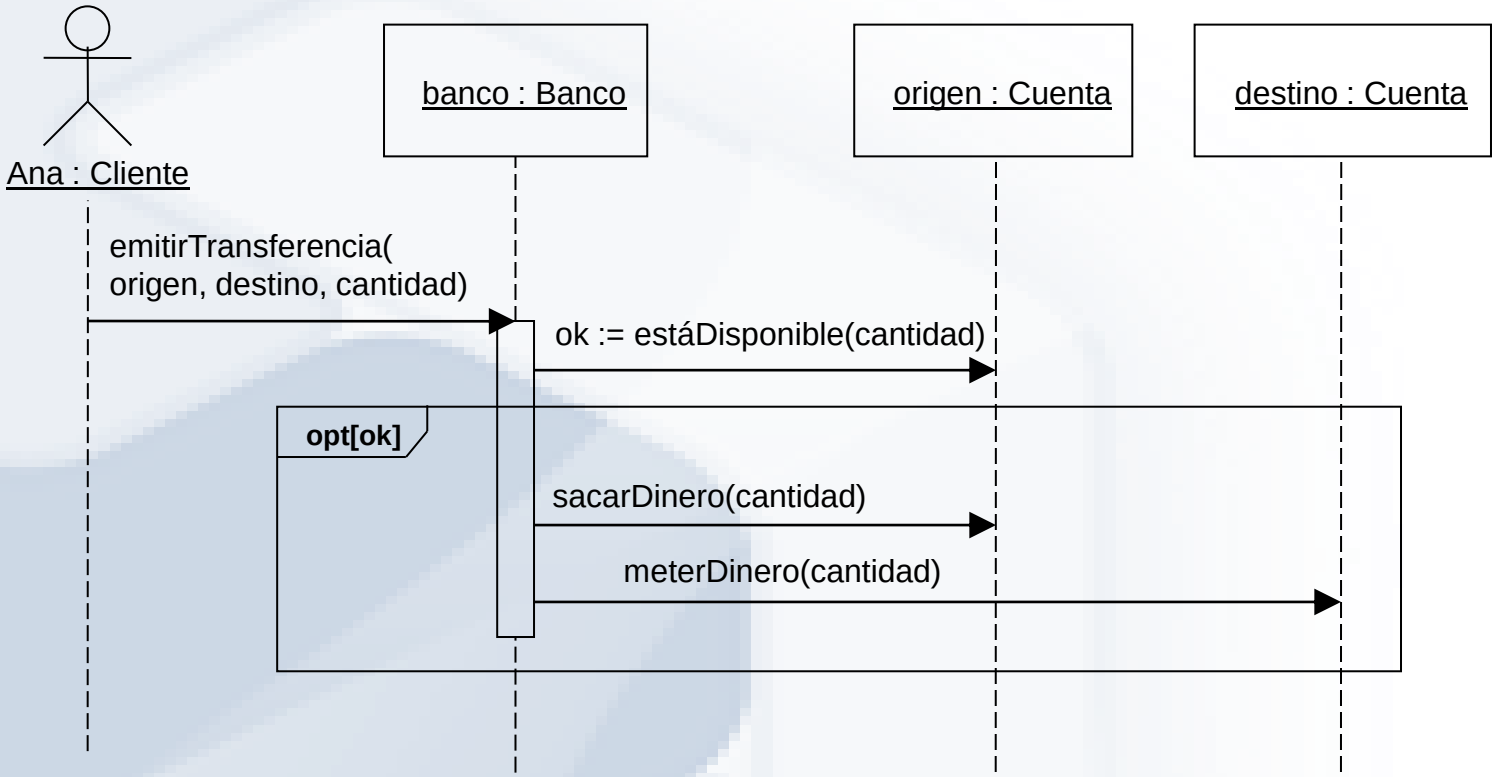
# Ejecución Opcional (II)

- Sintaxis: **opt '[' <condición> ']'**



# Ejecución Opcional (y III)

- Si la cantidad deseada está disponible, sacarla de la cuenta origen y meterla en la cuenta destino, si no, bloquear la cuenta origen.

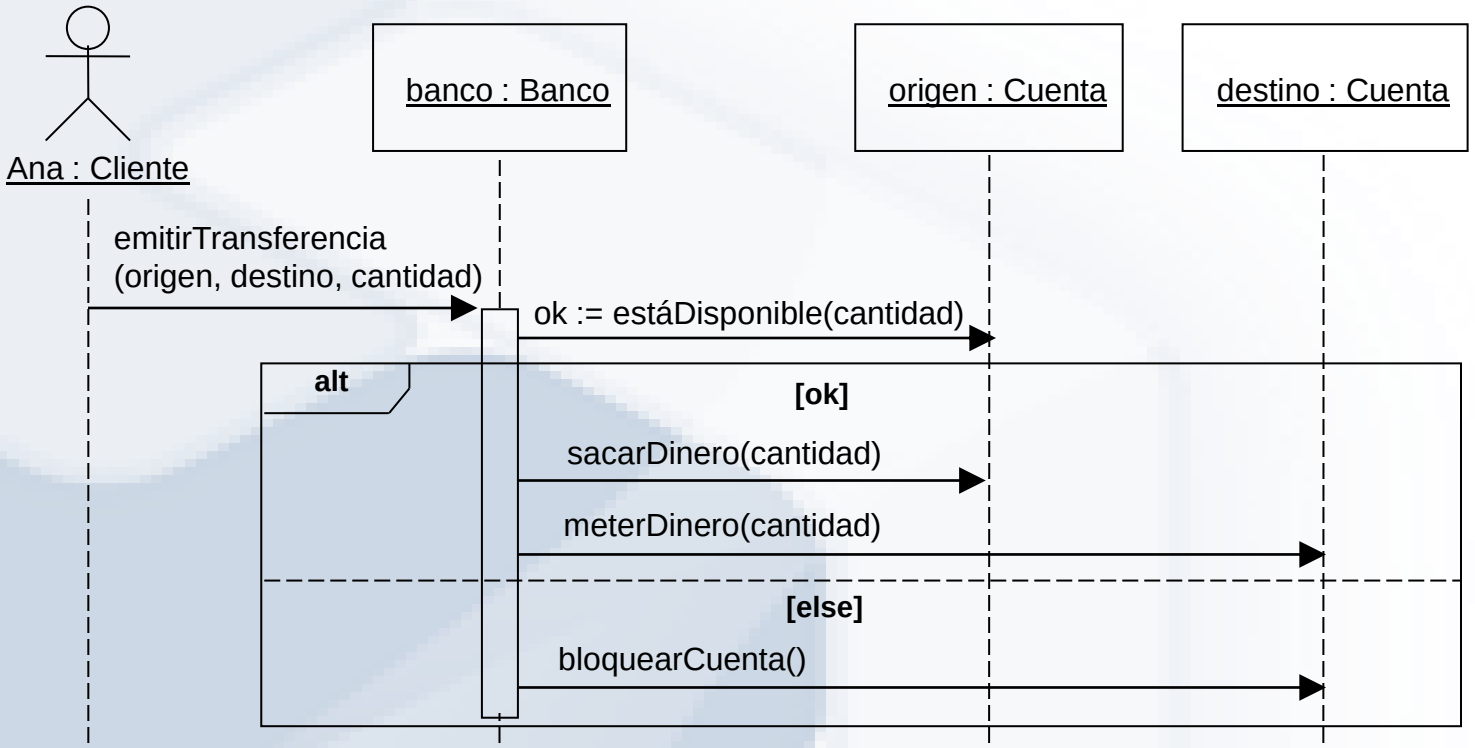


# *Ramificación Condicional (I)*

- La ramificación condicional en UML 2 se realiza con el fragmento combinado cuyo operador es **alt**.
- Uso de **variables** locales, valores devueltos por mensajes anteriores, etc. en las condiciones (guardas).
- El operador **alt** permite indicar varios operandos, pero sólo se puede cumplir una condición como máximo (es decir, sólo se puede dar un operando).

# Ramificación Condicional (y II)

- Si la cantidad deseada está disponible, sacarla de la cuenta origen y meterla en la cuenta destino, si no, bloquear la cuenta origen.

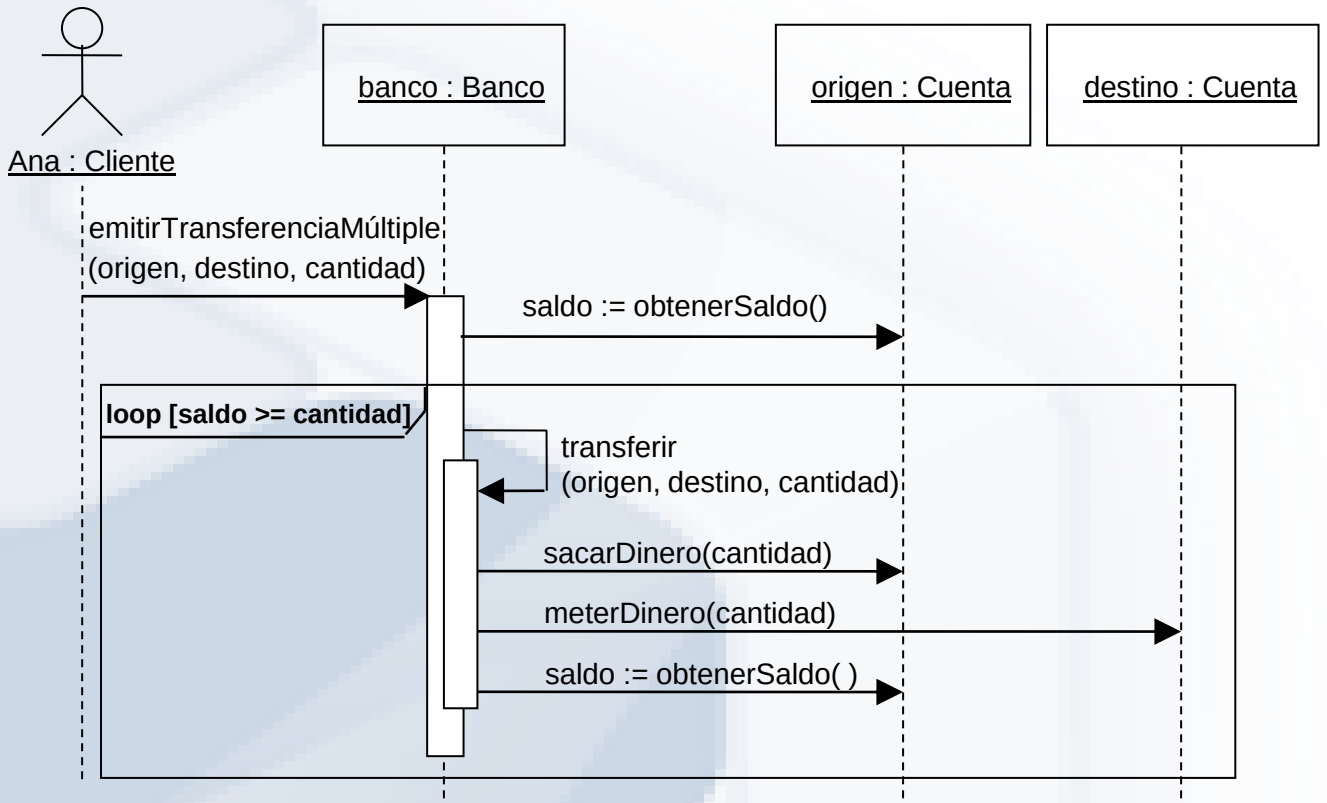


# *Iteración o Repetición de Mensajes (I)*

- La iteración en UML 2 se realiza con el fragmento combinado cuyo operador es **loop**.
- El operando se da **repetidas veces** según especifique el límite del operador **loop**.
- Si no se especifica límite se considera un ejecución infinita.
- El límite superior puede tener el valor '*' que indica valor infinito.
- Si se especifica límite inferior pero no superior, entonces se considera que el límite superior es igual al límite inferior.
- Sintaxis: **loop [ '(' <límite_inferior> [',' <límite_superior> ] ')' ] [ '[' <condición> ']' ]**
- La condición del operador permite simular varios tipos de cláusula: contador, mientras, hasta que, etc.:
  - while (true) => **loop** o **loop (0,*)**
  - for (i= n to m) => **loop (n,m)**
  - while (<condición>) => **loop [<condición>]**
  - do ... while (<condición>) => **loop (1,*) [<condición>]**

# Iteración o Repetición de Mensajes (y II)

- Mientras haya saldo, realizar transferencias por la cantidad especificada



# Polimorfismo de Mensajes

- El envío múltiple de mensajes es particularmente expresivo cuando la operación invocada se ejecuta en forma **polimórfica** en los receptores, lo que permite tratar de modo uniforme un conjunto de objetos que satisfacen una misma **interfaz** (entienden todos el mismo mensaje, cada uno a su manera).
- El uso adecuado de polimorfismo facilita el mantenimiento de la aplicación cuando hay que **añadir nuevas clases**, y hace innecesarias las instrucciones de **ramificación múltiple** ya que la ramificación es implícita: cada objeto interpreta el mensaje a su manera, según cuál sea su clase.