

## TEMA 3.4: DIAGRAMAS DE INTERACCIÓN

# Índice

- Introducción
- Diagramas de secuencia
- Diagramas de colaboración / comunicación UML 2

# Introducción

- Los **diagramas de interacción** proporcionan dos notaciones para un mismo objetivo:
  - Mostrar el modo en que un grupo de objetos interaccionan (se comunican) por medio de mensajes (comportamiento de varios objetos en un CU)
- Permiten **modelar la vista dinámica**
- Ayudan a implementar la lógica de los métodos
- Engloban dos tipos de diagramas:
  - Diagramas de secuencia
  - Diagramas de colaboración/comunicación

# Diagramas de secuencia

- Más adecuados para observar la **perspectiva cronológica** de las interacciones.
- Muestran la forma en que un objeto interacciona con otros a través del tiempo.
- Ideas principales:
  - Las interacciones entre los objetos se deben realizar en una secuencia establecida
  - Esta secuencia se debe tomar el tiempo necesario para terminar todo el proceso
- Incluye una nueva dimensión
  - Tiempo

# Diagramas de secuencia

- Elementos de un diagrama de secuencia:

– **Objetos**



– **Mensajes**



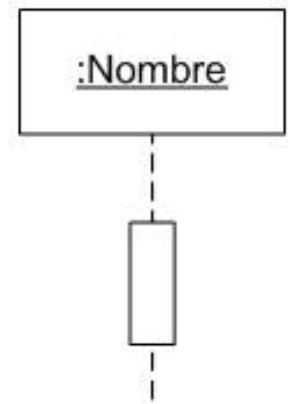
– **Tiempo**



# Diagramas de secuencia

- **Objetos**

- Se representan en la parte superior del diagrama como rectángulos y con el nombre subrayado
- Debajo de cada objeto hay una línea discontinua conocida como “línea de vida”
- Junto con la “línea de vida” se encuentra un pequeño rectángulo conocido como “**activación**”, que representa el período de tiempo en el cual el objeto está realizando una operación
- La longitud del rectángulo se interpreta como la duración de la “activación”

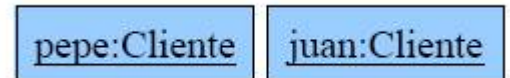


# Diagramas de secuencia

- Los **objetos** que participan en una interacción pueden ser:

- Elementos concretos (INSTANCIAS)

- Representan algo del mundo real



- Elementos prototípicos (ROLES)

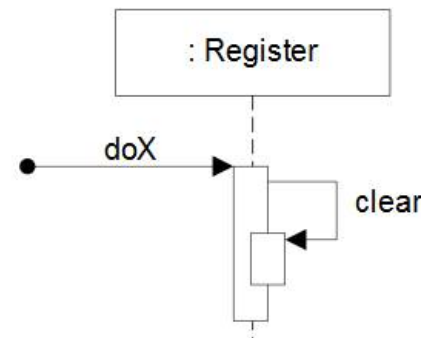
- Representan cualquier elemento de cierto tipo



# Diagramas de secuencia

- **Mensajes**

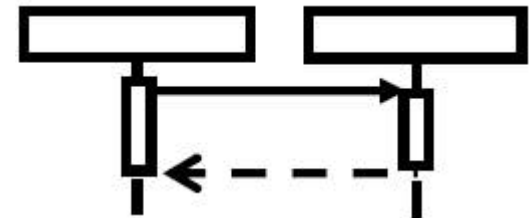
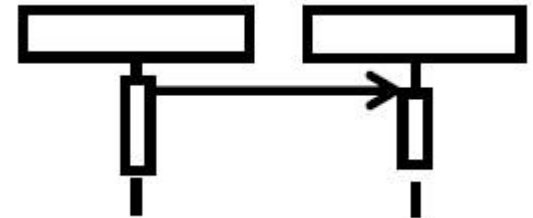
- Representan la forma de comunicación entre objetos
- Un mensaje va de un objeto a otro
- Pasa de la línea de vida de un objeto a la de otro
- Representados gráficamente por líneas continuas con una punta de flecha
- Un objeto puede enviarse un mensaje a sí mismo





# Diagramas de secuencia

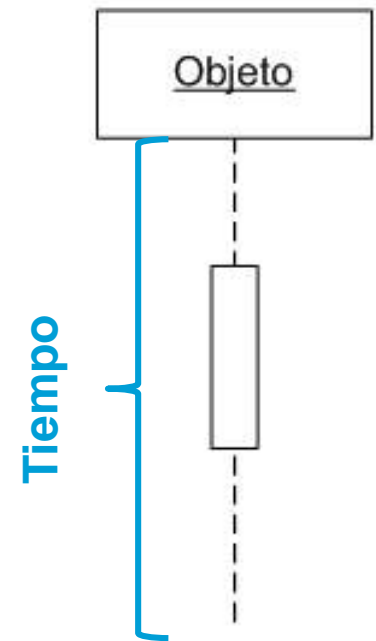
- Tipos de mensajes:
  - **Simple:** Se usa cuando no se conocen detalles del tipo de comunicación o cuando no resulta relevante en el diagrama
  - **Síncronos:** El objeto que envía el mensaje espera la respuesta a tal mensaje antes de continuar su trabajo
  - **Asíncronos:** El objeto que envía el mensaje no esperará una respuesta antes de continuar



# Diagramas de secuencia

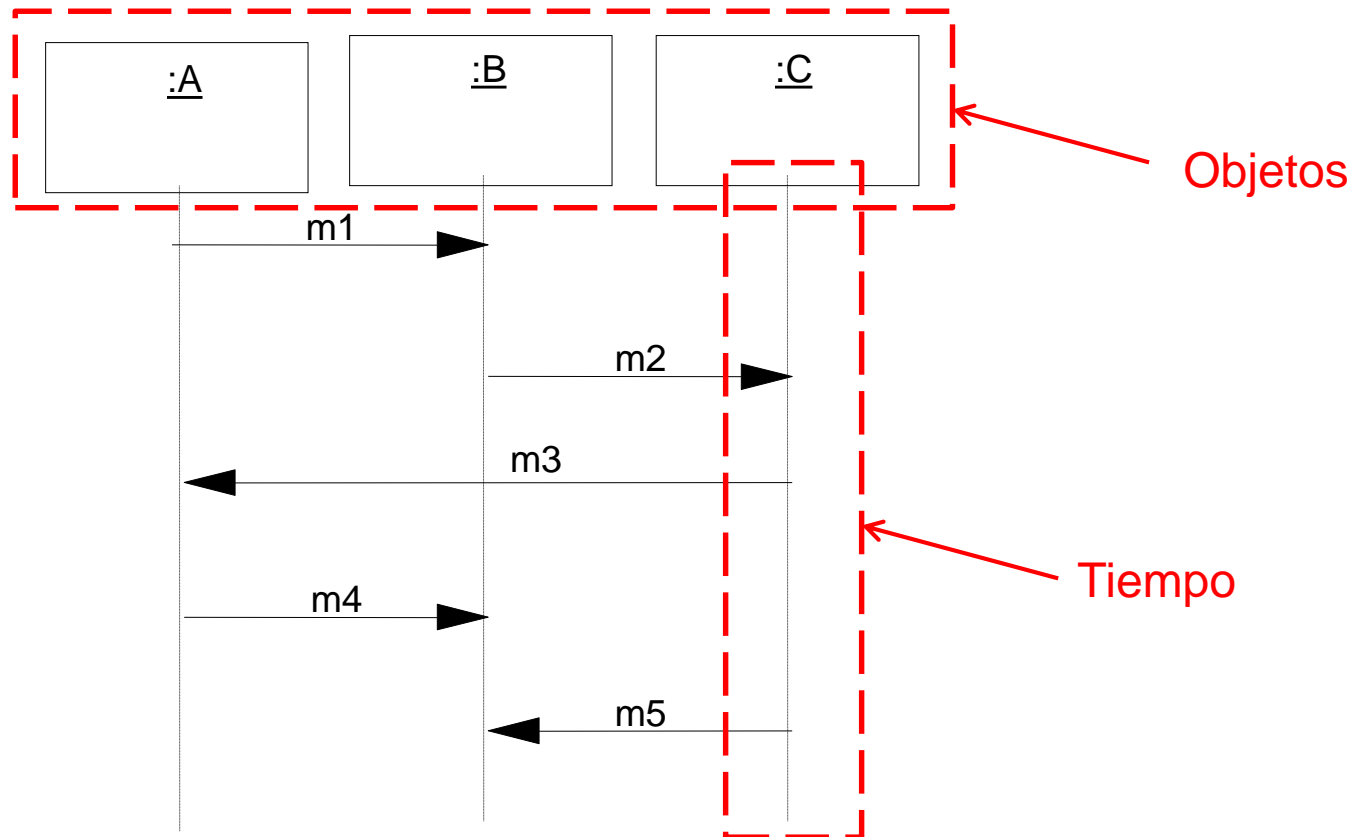
- **Tiempo**

- Representado por una progresión vertical
- El tiempo se inicia en la parte superior y avanza hacia la parte inferior
- Un mensaje que esté más cerca de la parte superior ocurrirá antes que uno que esté cerca de la parte inferior



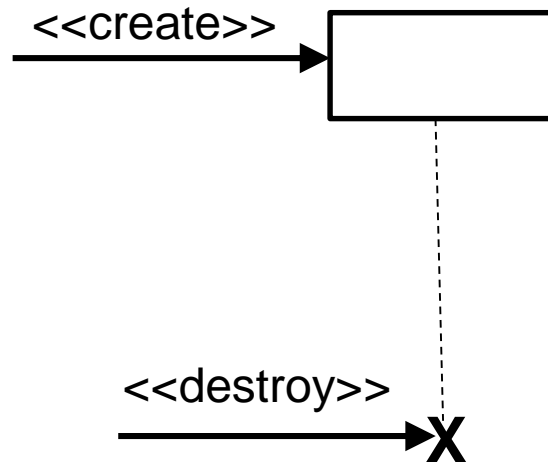
# Diagramas de secuencia

- El diagrama de secuencia tiene 2 dimensiones:
  - La dimensión horizontal es la disposición de objetos
  - La dimensión vertical muestra el paso del tiempo



# Diagramas de secuencia

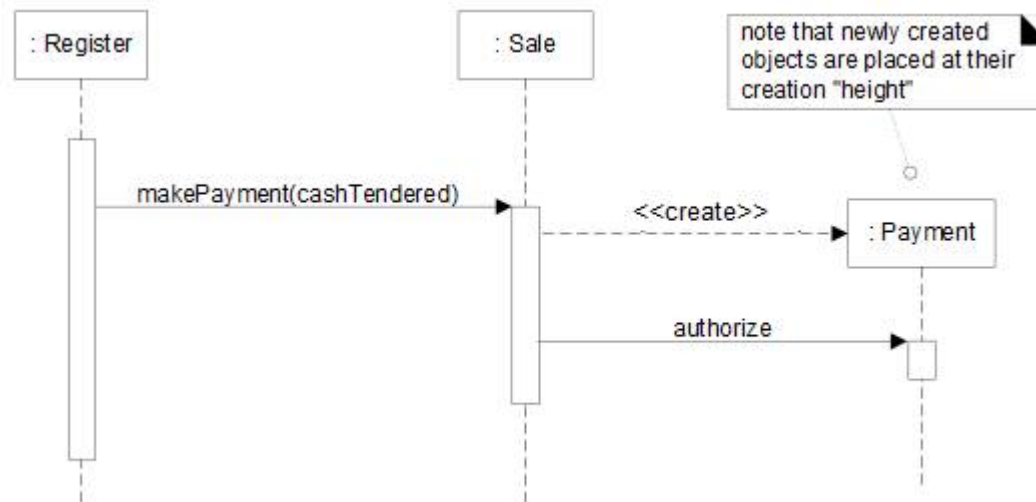
- Para la creación o destrucción de objetos se utilizan mensajes especiales
  - Se les añade el estereotipo <<create>> o <<destroy>>



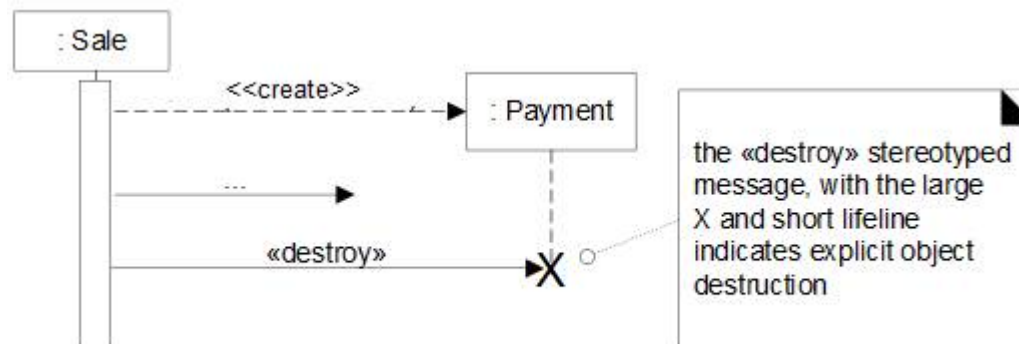
- Los objetos se crean cuando es necesario

# Diagramas de secuencia

- Creación

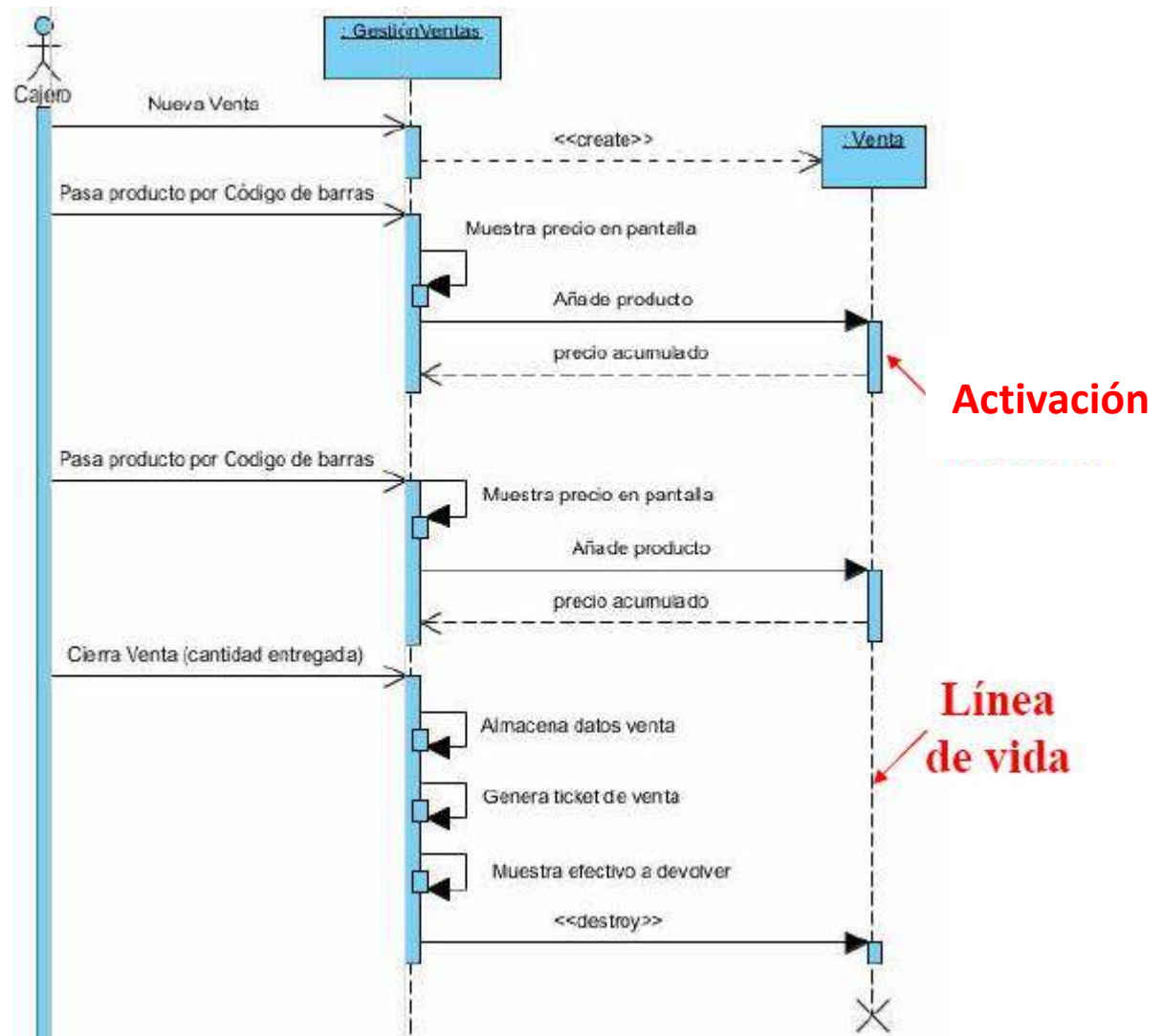


- Destrucción



# Diagramas de secuencia

- Diagrama de secuencia que modela la compra de dos productos en el supermercado con pago en efectivo



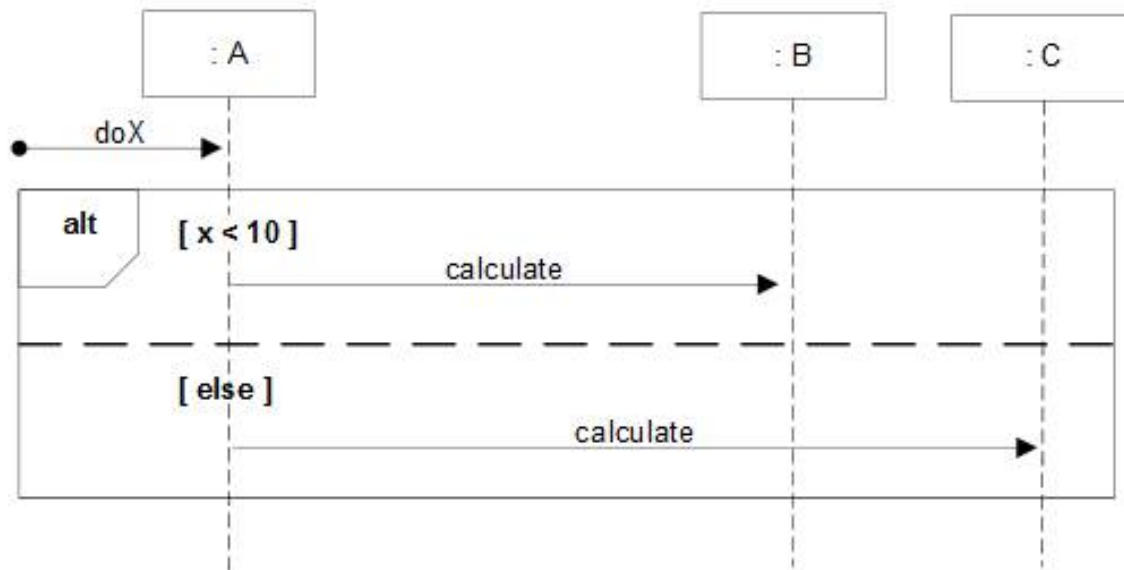
# Diagramas de secuencia

- Fragmentos (marcos de interacción)
  - Regiones rectangulares usadas para la especificación de bloques repetitivos, opcionales, alternativos, entre otros
  - Pueden rodear un diagrama completo o parte de él
  - Cada fragmento se etiqueta con una palabra específica
  - Principales tipos de fragmentos:

Operador	Significado
alt	Indica que el fragmento de diagrama es una <b>alternativa</b> (“Si– Sino”)
loop	Indica que el fragmento de diagrama se ejecuta <b>repetidas veces</b>
opt	Indica que el fragmento de diagrama es <b>opcional</b> (sólo “Si”)
par	Indica que el fragmento de diagrama <b>se ejecuta en paralelo</b>

# Diagramas de secuencia

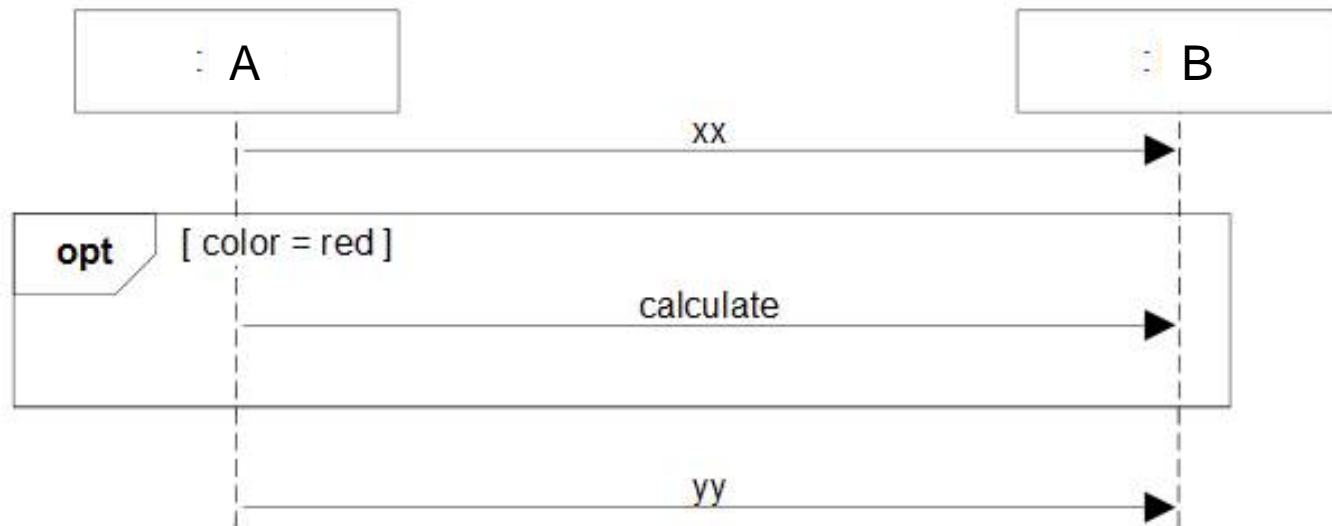
- Alternativas





# Diagramas de secuencia

- Opcionalidad

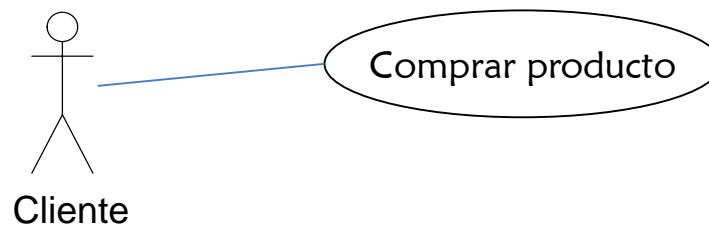


# Diagramas de secuencia

- Los diagramas de secuencia muestran las interacciones entre objetos en un escenario concreto o de un caso de uso en general
- Si se modela un escenario concreto podemos hablar de diagramas de secuencia de instancias
  - Sólo se centra en un escenario (una instancia) en un caso de uso

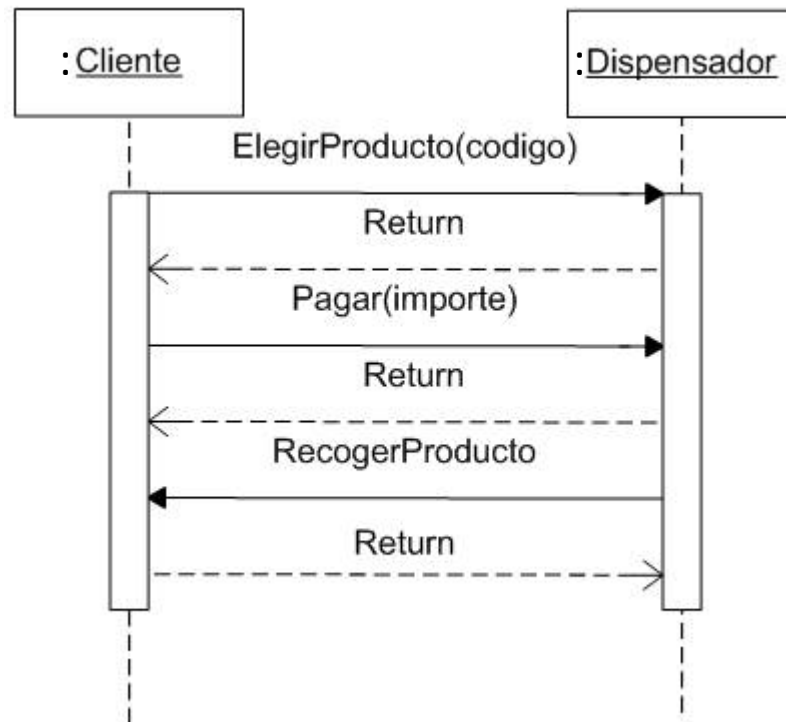
# Diagramas de secuencia

- Ejemplo:
  - Supongamos que queremos modelar las interacciones entre un cliente y una máquina expendedora de productos alimenticios



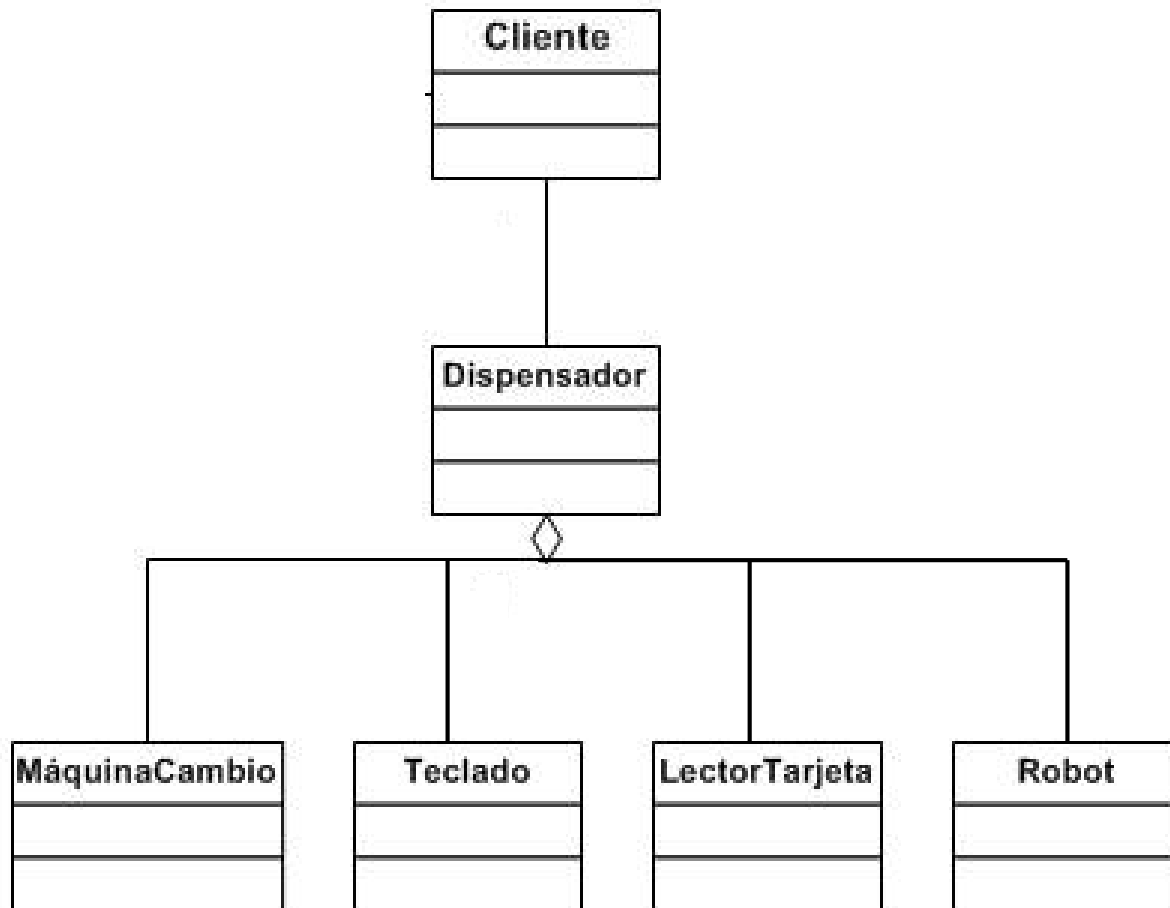
# Diagramas de secuencia

- A partir del diagrama de CU anterior se procedería a elaborar el diagrama de secuencia que modele las interacciones entre el cliente y el dispensador de productos

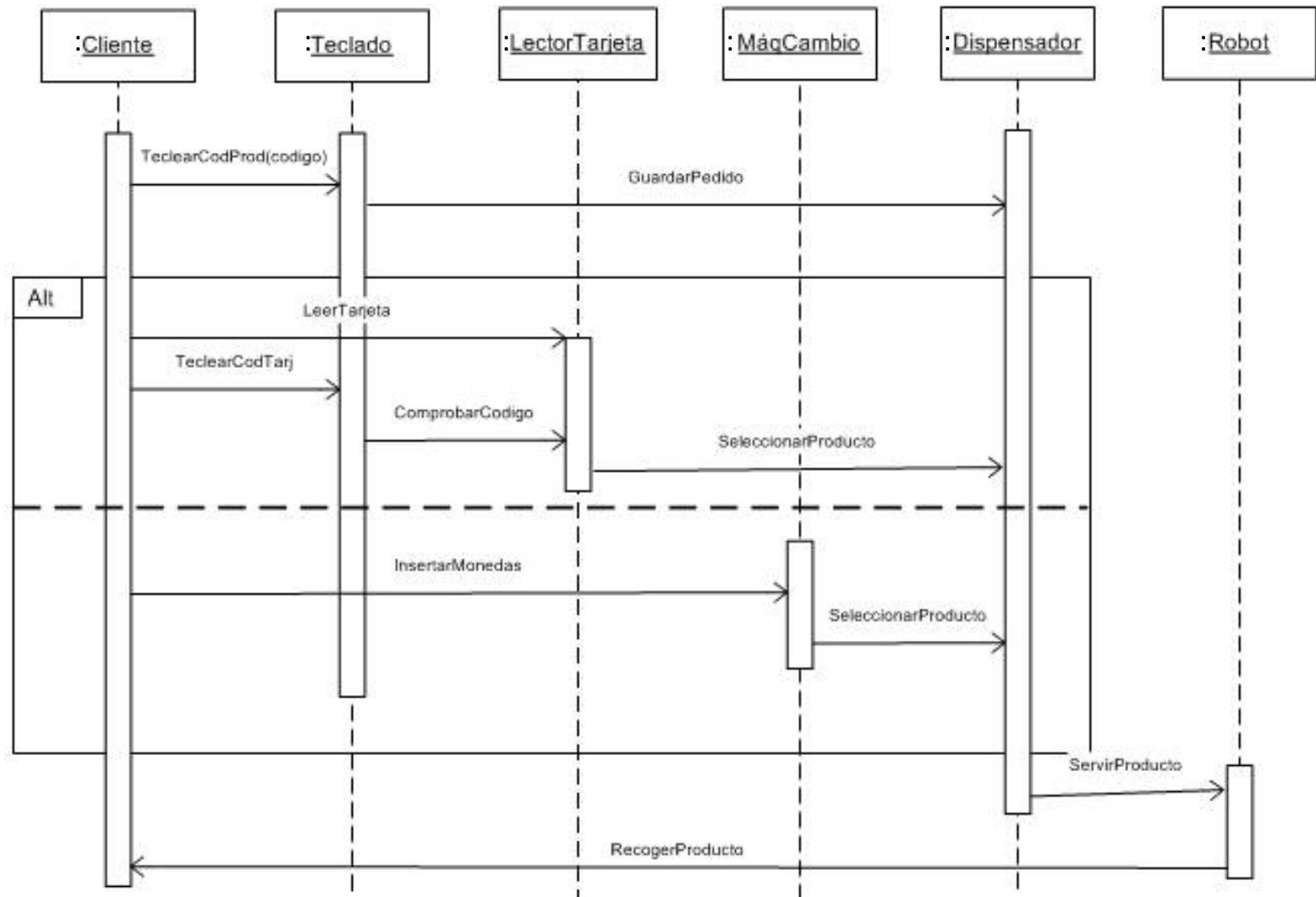


# Diagramas de secuencia

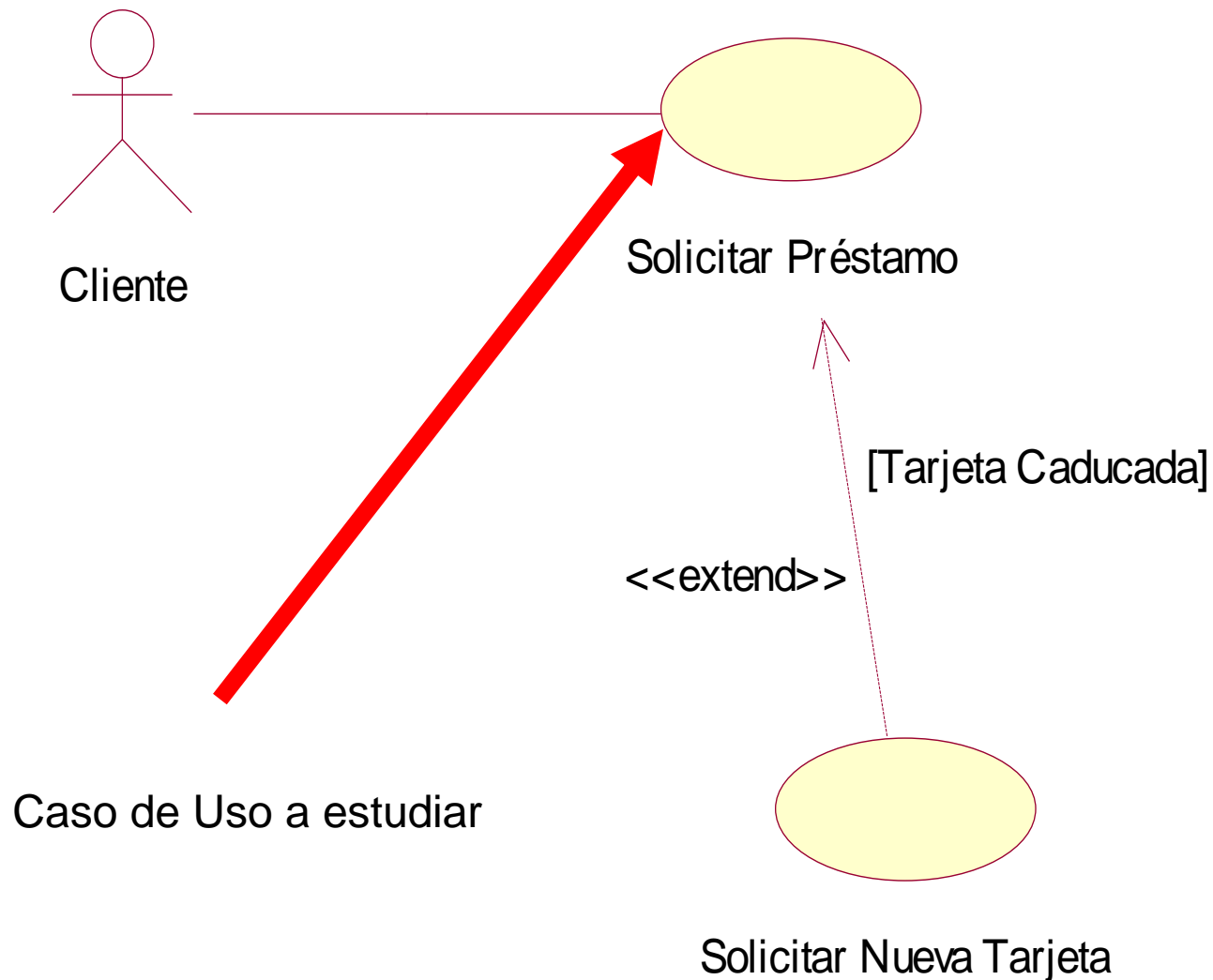
- Supongamos que queremos especificar las interacciones entre el cliente y la máquina expendedora según el siguiente diagrama de clases:



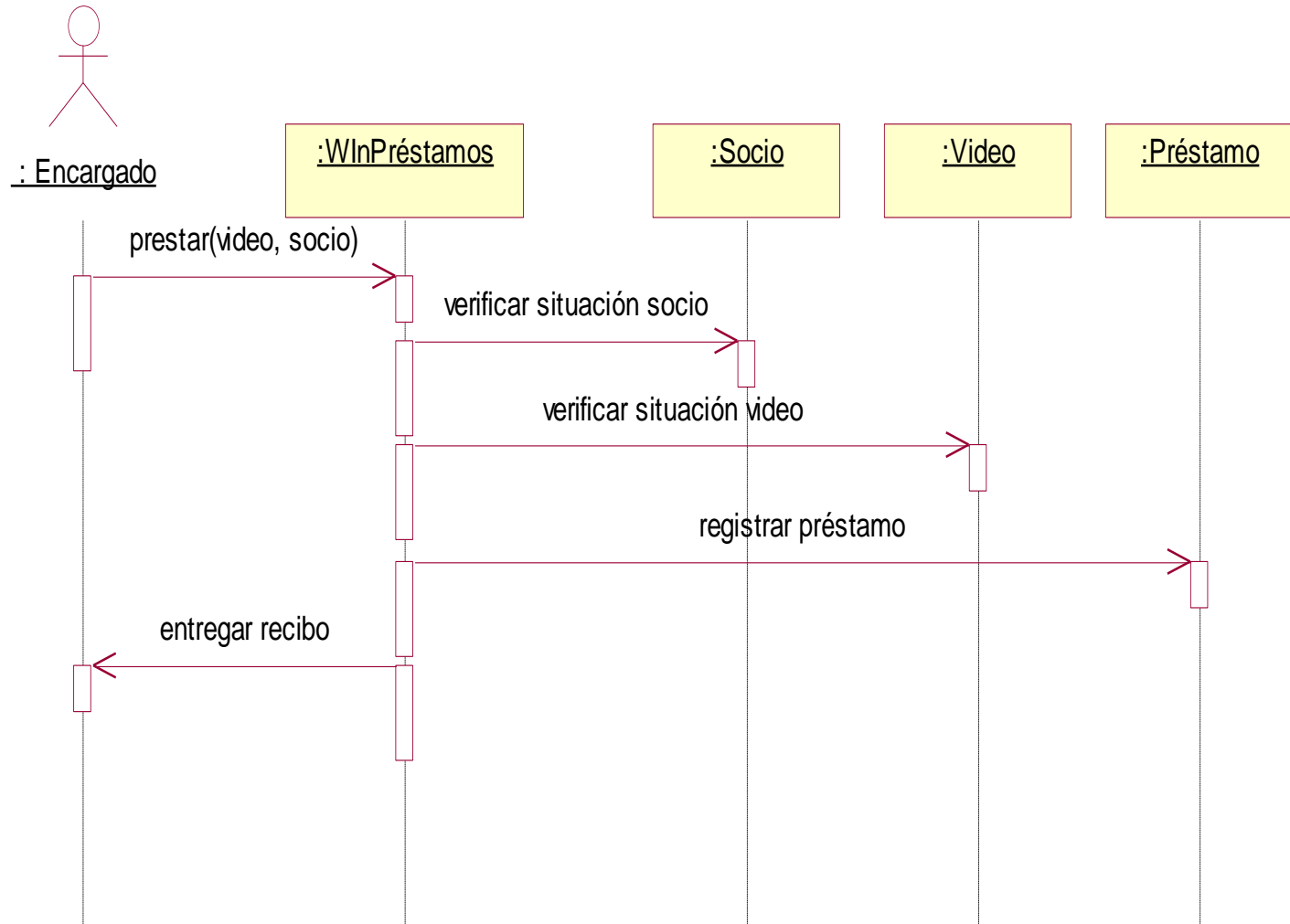
# Diagramas de secuencia



# Caso de uso y derivación en un diagrama de secuencia



# Diagrama de secuencia derivado





# Diagramas de secuencia

- Errores clásicos:
  - No hacer un diagrama de secuencia para cada caso de uso
  - No identificar todos los objetos necesarios
  - No proveer de texto a las flechas de mensajes
  - Dar más importancia a funciones get y set en lugar de enfocarse en los métodos importantes
  - No invocar a operaciones de clases

# Ejercicio

- Especificar el diagrama de secuencia de la operación “crearLaberinto”

```
public class JuegoLaberinto {  
    public Laberinto crearLaberinto () {  
        Laberinto lab = new Laberinto();  
        Habitacion h1 = new Habitacion();  
        Habitacion h2 = new Habitacion();  
        Puerta puerta = new Puerta(h1, h2);  
        lab.añadeHabitacion(h1);  
        lab.añadeHabitacion(h2);  
        h1.añadePuerta(puerta);  
        return lab;  
    }  
}
```

# Ejercicio

- **Especificar el diagrama de secuencia de la operación “crearLaberinto”**

```
public class JuegoLaberinto {  
    private Laberinto lab;  
    private boolean conVentana;  
    public JuegoLaberinto() {  
        lab = new Laberinto();  
        conVentana = true;  
    }  
    public void crearLaberinto () {  
        Habitacion h;  
        for (int i=0; i<10; i++) {  
            h = new Habitacion();  
            if (conVentana == true)  
                h.añadeVentana(new Ventana());  
            lab.añadeHabitacion(h);  
        }  
    }  
}
```

# Diagramas de colaboración

- Los diagramas de colaboración muestran la forma en que los **objetos colaboran** entre sí, al igual que ocurre en un diagrama de secuencia
- La diferencia entre ambos diagramas es:
  - Los diagramas de **secuencia** destacan la sucesión de las interacciones (**organizado respecto al tiempo**)
  - Los diagramas de **colaboración** destacan el contexto y organización general de los objetos que interactúan (**organizado respecto al espacio**)

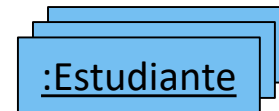
# Diagramas de colaboración

- Elementos de un diagrama de colaboración:

- **Objetos**



- **Grupo de objetos**



- **Mensajes**

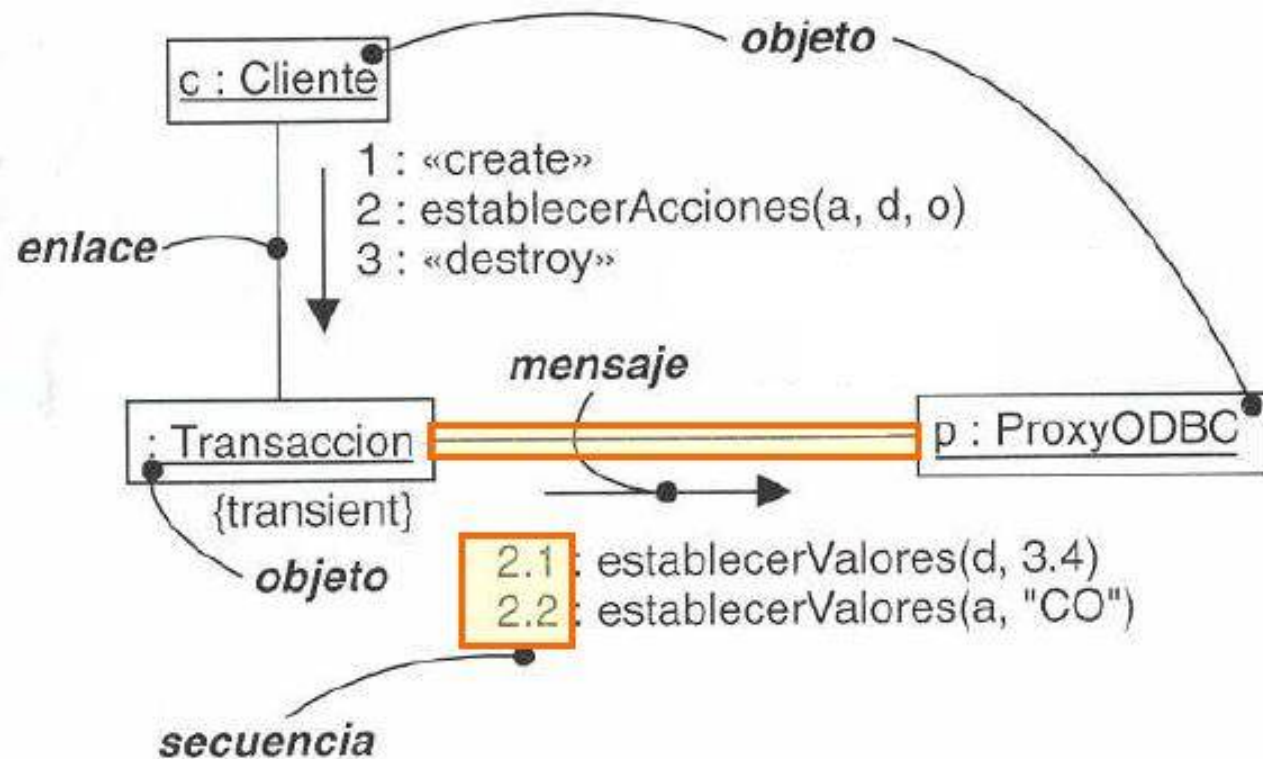
(se numeran para indicar la secuencia)



- **Asociaciones entre objetos**



# Diagramas de colaboración



# Diagramas de colaboración

- Un diagrama de colaboración es una extensión de un diagrama de objetos
- Además de las relaciones entre objetos, el diagrama de colaboración muestra los mensajes que se envían los objetos entre sí
- Para representar un mensaje se dibuja una flecha cerca de la línea de asociación entre dos objetos
  - La flecha apuntará al objeto receptor
- El mensaje indicará al objeto receptor que ejecute alguna de sus operaciones

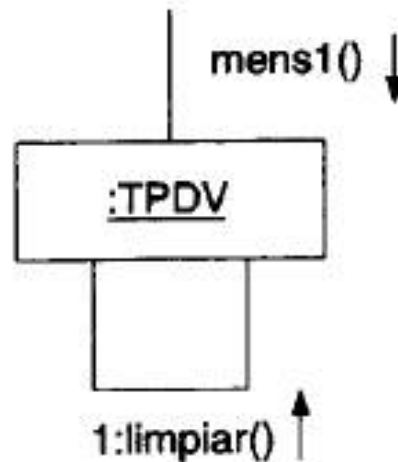
# Diagramas de colaboración

- Son útiles en la fase exploratoria para identificar objetos
  - La distribución de los objetos en el diagrama permite observar adecuadamente la interacción de un objeto con respecto de los demás.
  - La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces.



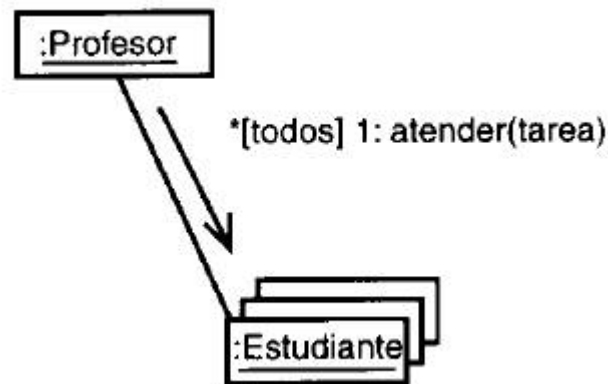
# Diagramas de colaboración

- Un objeto puede enviarse un mensaje a sí mismo
- Gráficamente se representa mediante un enlace a sí mismo



# Diagrama de colaboración

- En ocasiones un objeto envía un mensaje a diversos objetos de la misma clase
- Por ejemplo, un profesor pide a un grupo de estudiantes que entreguen una tarea
- En el diagrama de colaboración, diversos objetos se representan como una pila de rectángulos



# Diagramas de colaboración

- En algunos casos el orden del mensaje enviado es importante
- Por ejemplo, un cajero atenderá a cada cliente en orden de llegada
- Esta opción se representará como un mientras cuya condición implicará orden

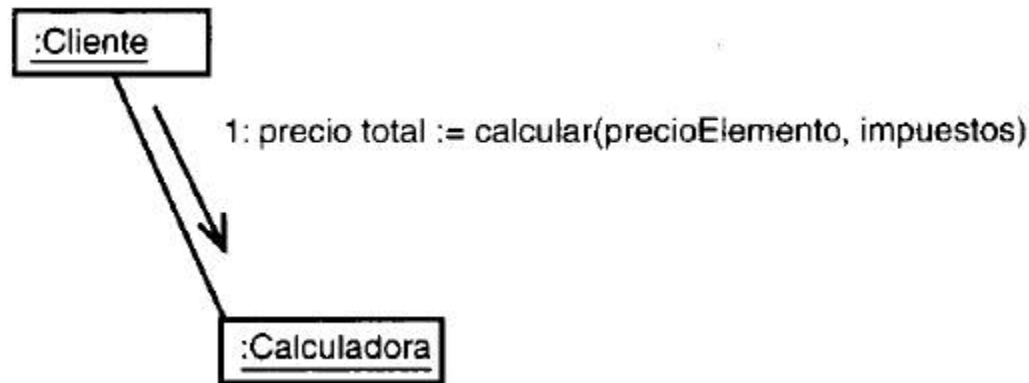


# Diagramas de colaboración

- Un mensaje podría ser una petición a un objeto para que realice un cálculo y devuelva un valor
- Para representar este caso se deberá escribir una expresión que tenga el nombre del valor devuelto a la izquierda, seguido de “:=”, a continuación el nombre de la operación y las cantidades con que opera para producir el resultado

# Diagramas de colaboración

- Un objeto Cliente podría solicitar a un objeto Calculadora que calcule el precio total de un producto con el impuesto asociado
- En este ejemplo, la expresión podría ser `precioTotal:=calcular(precioElemento, impuesto)`



# Diagramas de colaboración

- Sincronización
  - Un objeto sólo puede enviar un mensaje después de que otros mensajes hayan sido enviados
  - El objeto debe sincronizar todos los mensajes en el orden adecuado

# Diagramas de colaboración

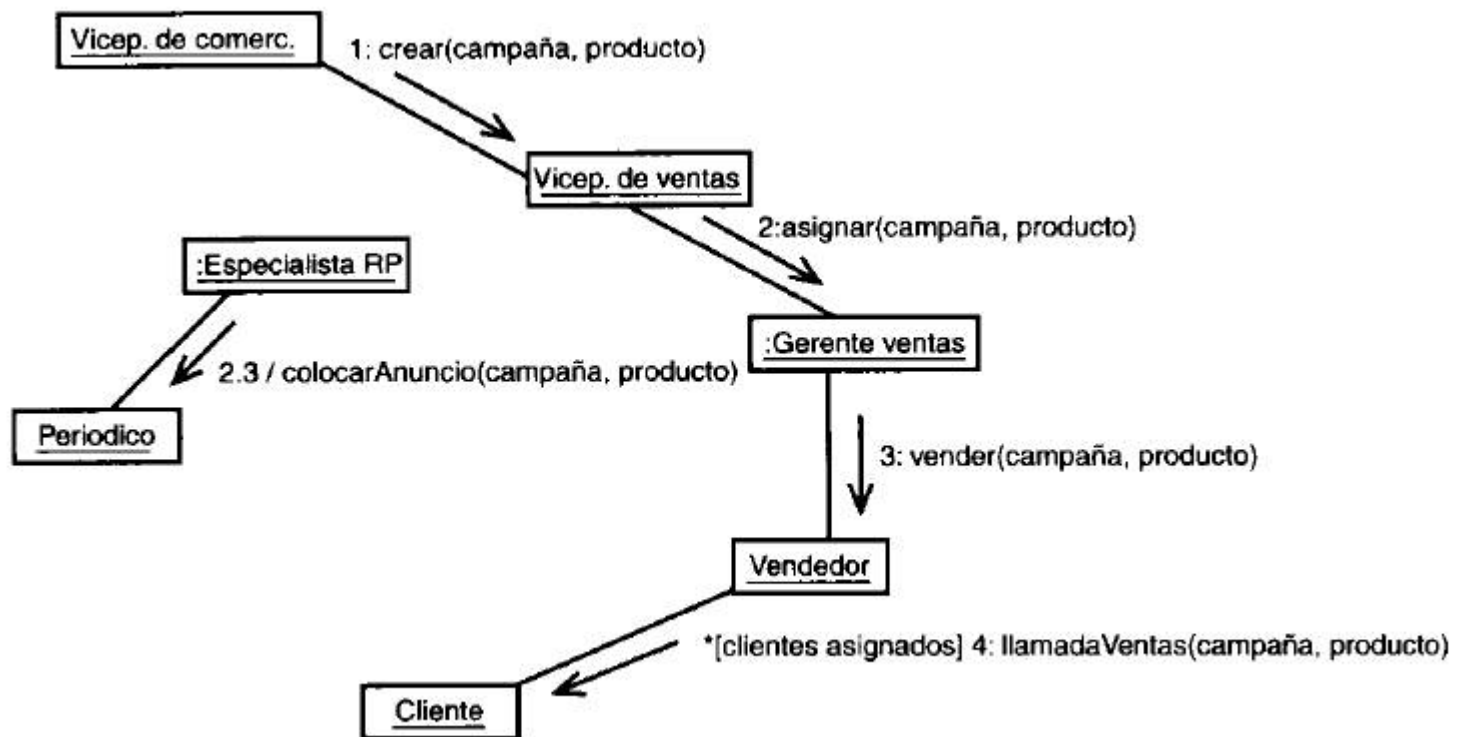
- Por ejemplo, supongamos que los objetos son personas de una empresa y que están trabajando en la campaña de un nuevo producto:
  1. El vicepresidente de comercialización le pide al de ventas que cree una campaña para un producto particular
  2. El vicepresidente de ventas crea la campaña y la asigna al gerente de ventas
  3. El gerente de ventas instruye a un agente de ventas para que venda el producto de acuerdo con la campaña
  4. El agente de ventas hace llamadas para vender el producto a los clientes en potencia

# Diagramas de colaboración

5. Después de que el vicepresidente de ventas ha dado la comisión y el gerente de ventas ha expedido la orden (esto es, cuando se han completado los pasos 2 y 3), un especialista en relaciones públicas de la empresa hará una llamada al periódico local y colocará un anuncio de la campaña

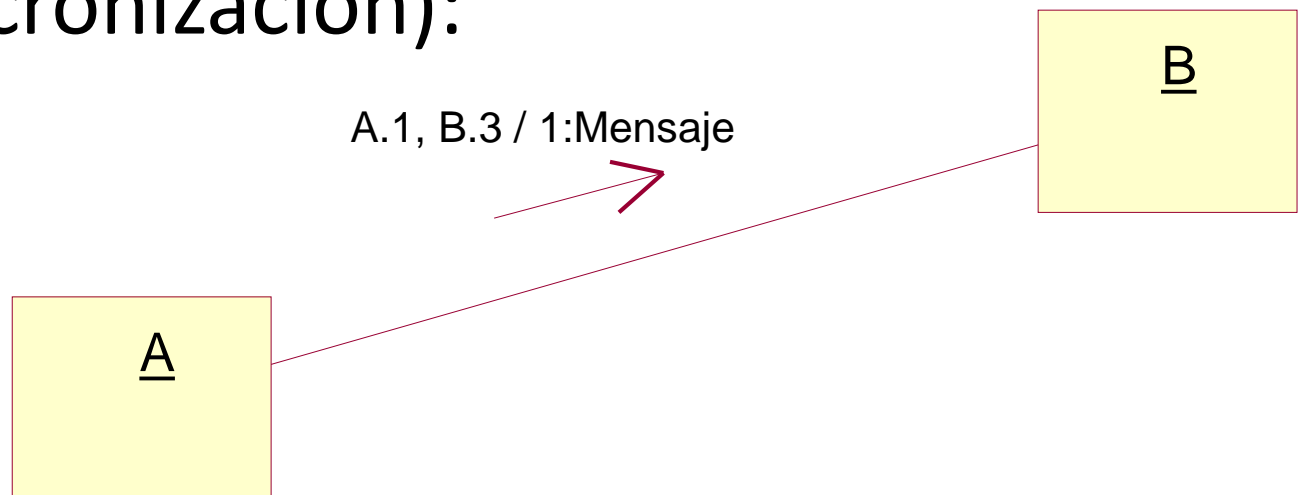


# Diagramas de colaboración



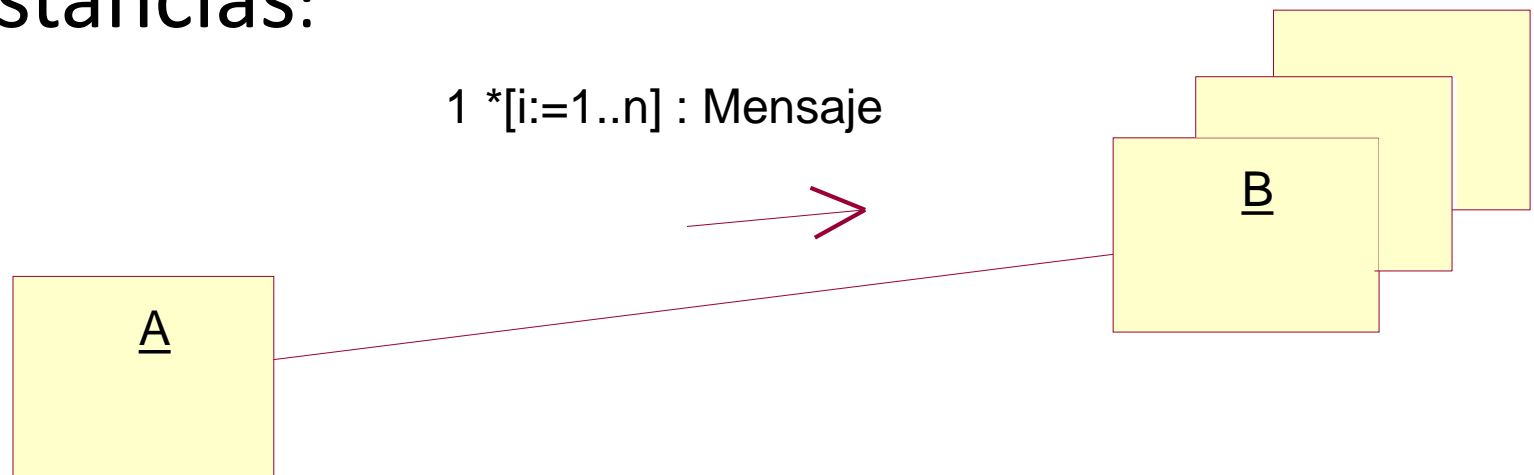
# Diagramas de colaboración

- Un mensaje desencadena una acción en el objeto destinatario
  - Un mensaje se envía si han sido enviados los mensajes de una lista (sincronización):



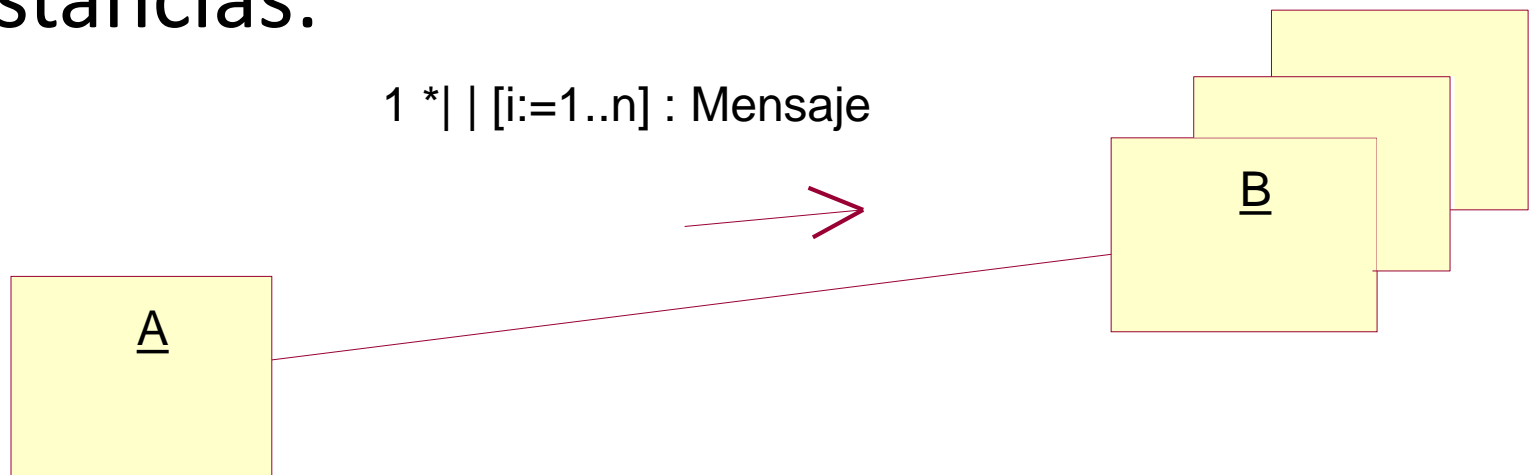
# Diagramas de colaboración

- Un mensaje se envía iterada y secuencialmente a un conjunto de instancias:



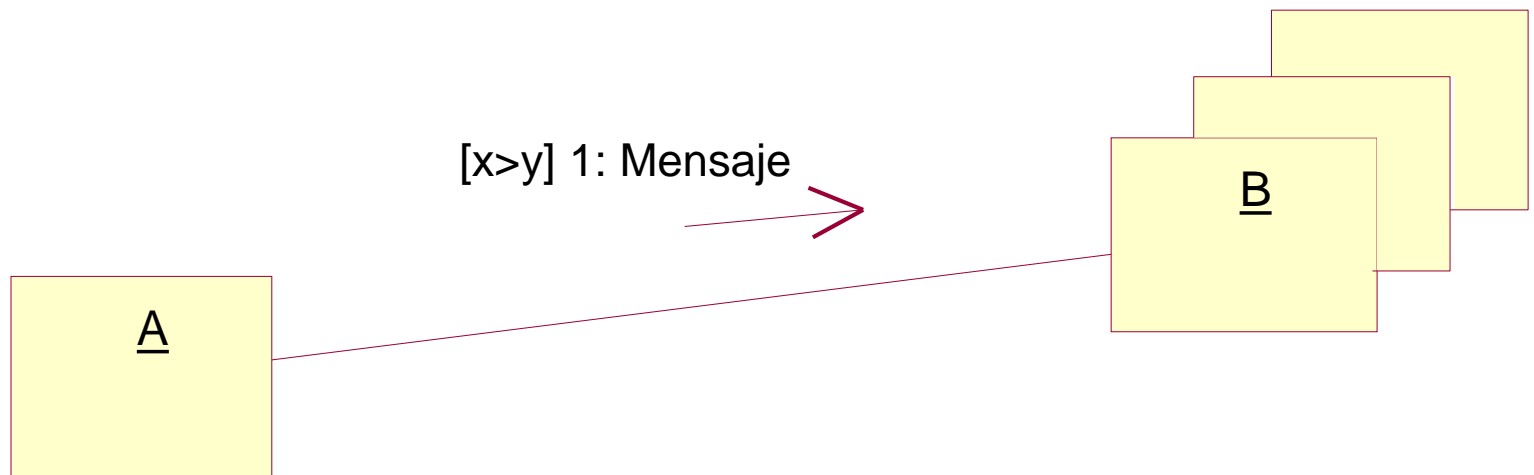
# Diagramas de colaboración

- Un mensaje se envía iterada y concurrentemente a un conjunto de instancias:



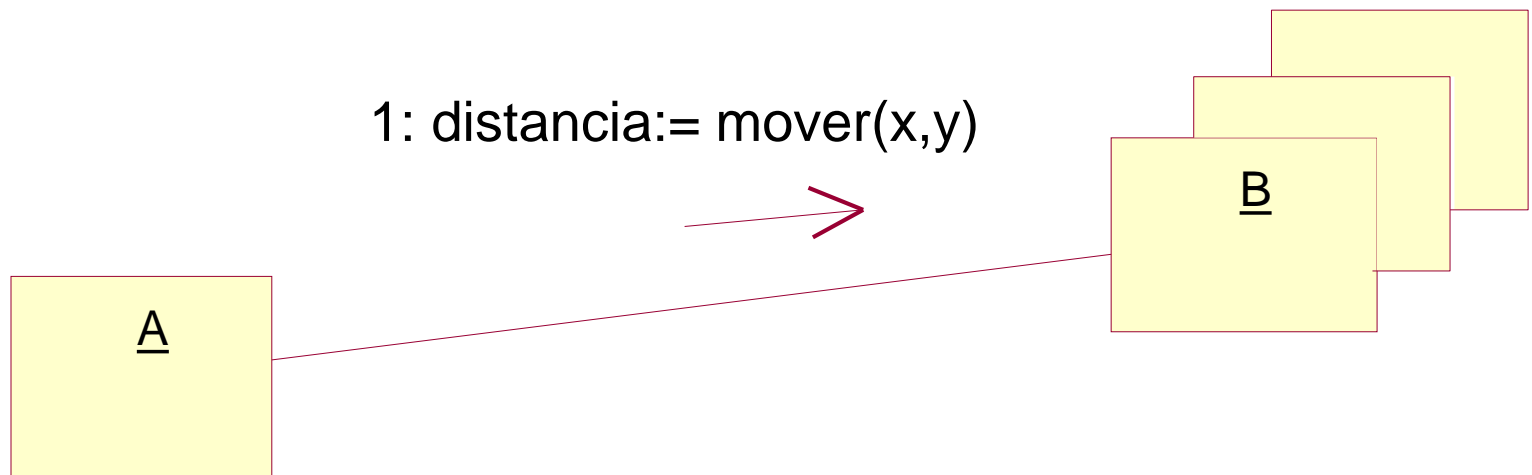
# Diagramas de colaboración

- Un mensaje se envía de manera condicionada:

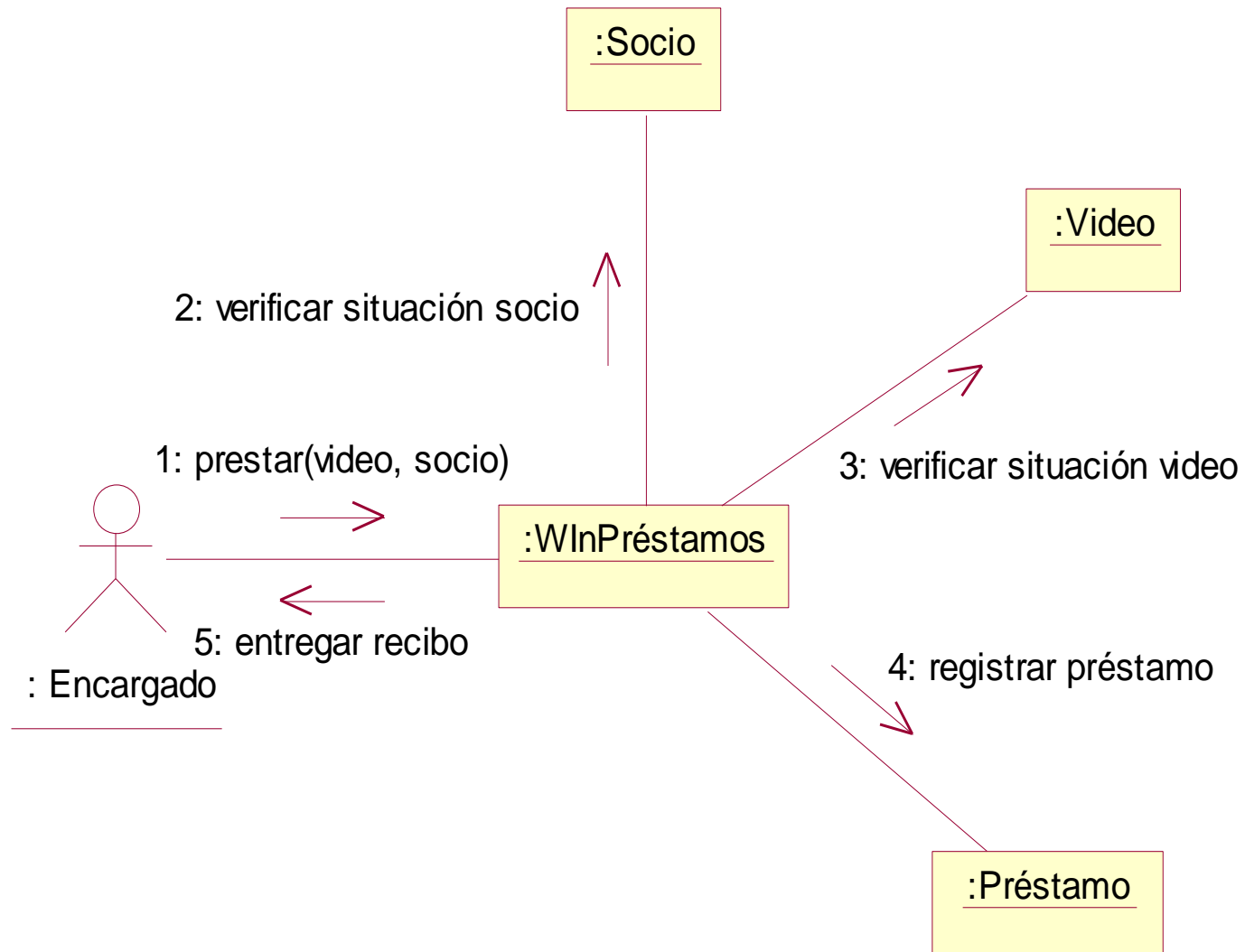


# Diagramas de colaboración

- Un mensaje que devuelve un resultado:



# Diagramas de colaboración



# Ejercicio

- Un usuario desea imprimir un archivo para lo cual le envía la orden al ordenador, el cual a su vez la envía al servidor de impresión siendo éste el encargado de dirigirlo a la impresora. En caso de que la impresora esté ocupada el archivo a imprimir se dirige hacia la cola de impresión, la cual en su momento le indicará al servidor de impresión que tiene el archivo pendiente por imprimir



# Preguntas

1. Un diagrama de secuencia se basa en una representación temporal
  - Verdadero
  - Falso
2. Un diagrama de secuencia requiere la intervención:
  - De las clases
  - De los objetos
3. ¿Puede un objeto mandarse un mensaje a sí mismo?
  - Si
  - No

# Bibliografía

- UML gota a gota. Martin Fowler
- Ingeniería del software. Ian Sommerville
- UML distilled. Martin Fowler
- UML 2. Practique la modelización. Laurent Debrauwer y Naouel Karam