

# Diagramas de secuencia

## Interacciones básicas

# Para qué sirven los diagramas de secuencia?



# Para qué sirven los diagramas de secuencia?

- En UML 2 existen 13 tipos de diagramas distintos para representar aspectos de estructura o de comportamiento de un sistema que existe o que está en construcción.

# Para qué sirven los diagramas de secuencia?

- Entre los diagramas para representar comportamiento, están los diagramas que hacen énfasis en aspectos de interacción entre los objetos. Estos incluyen diagramas de comunicación, interacción, secuencia y tiempo.

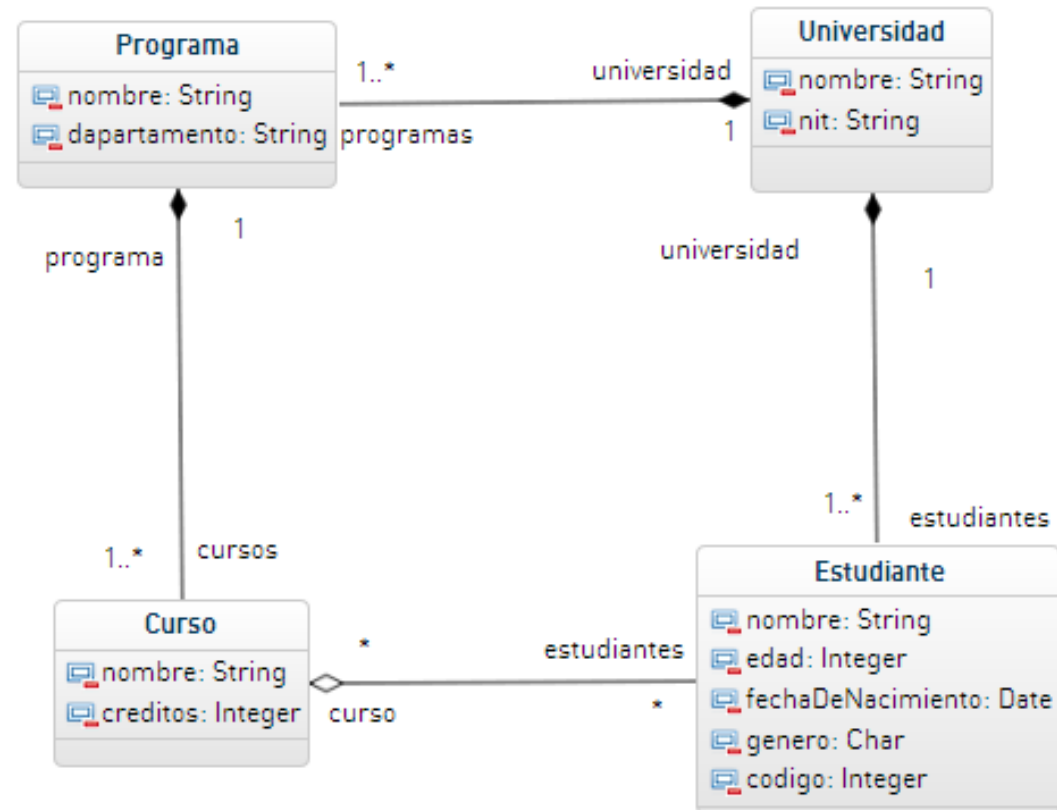
# Para qué sirven los diagramas de secuencia?

- En este módulo vamos a trabajar con los diagramas de secuencia que permiten modelar la secuencia de interacciones entre distintos objetos para lograr alguna tarea ya sea un escenario de un caso de uso, la lógica de un método o la lógica de un servicio.

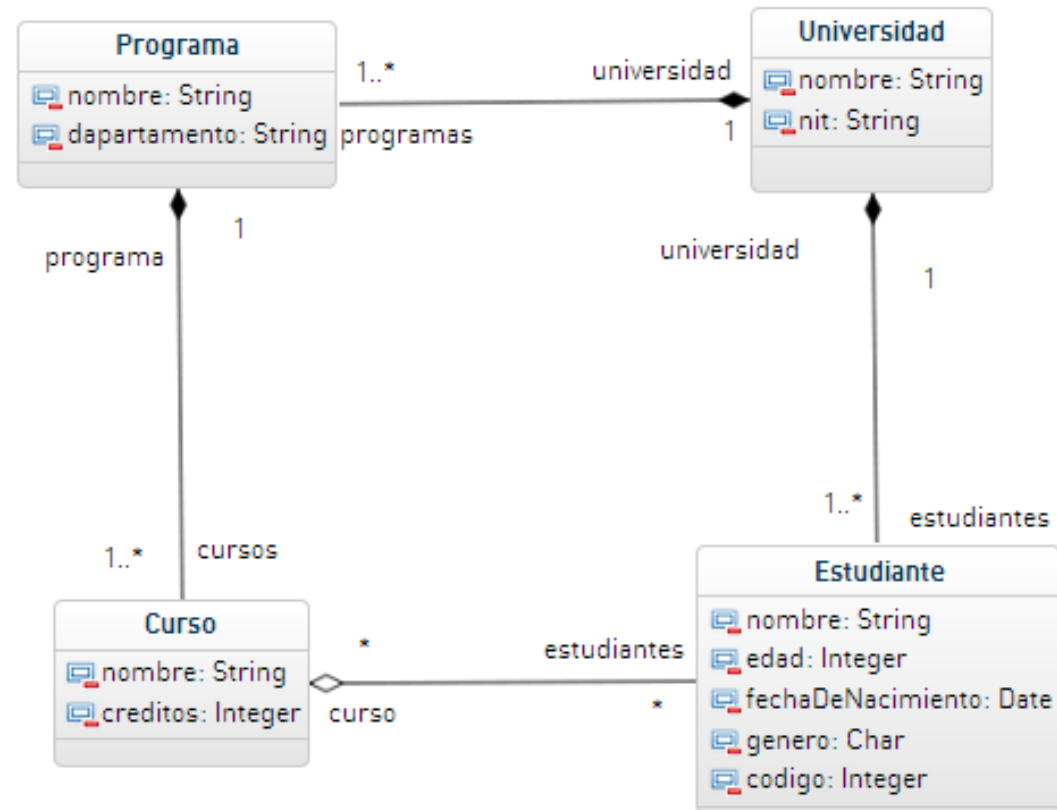
# Para qué sirven los diagramas de secuencia?

- Para entender el propósito de los diagramas de secuencia vamos a empezar con un ejemplo.

- En la figura del ejemplo, tenemos el caso “Universidad”
- Aquí podemos ver que hay 4 clases Universidad, Programa, Curso, Estudiante y que estas clases están relacionadas entre sí.

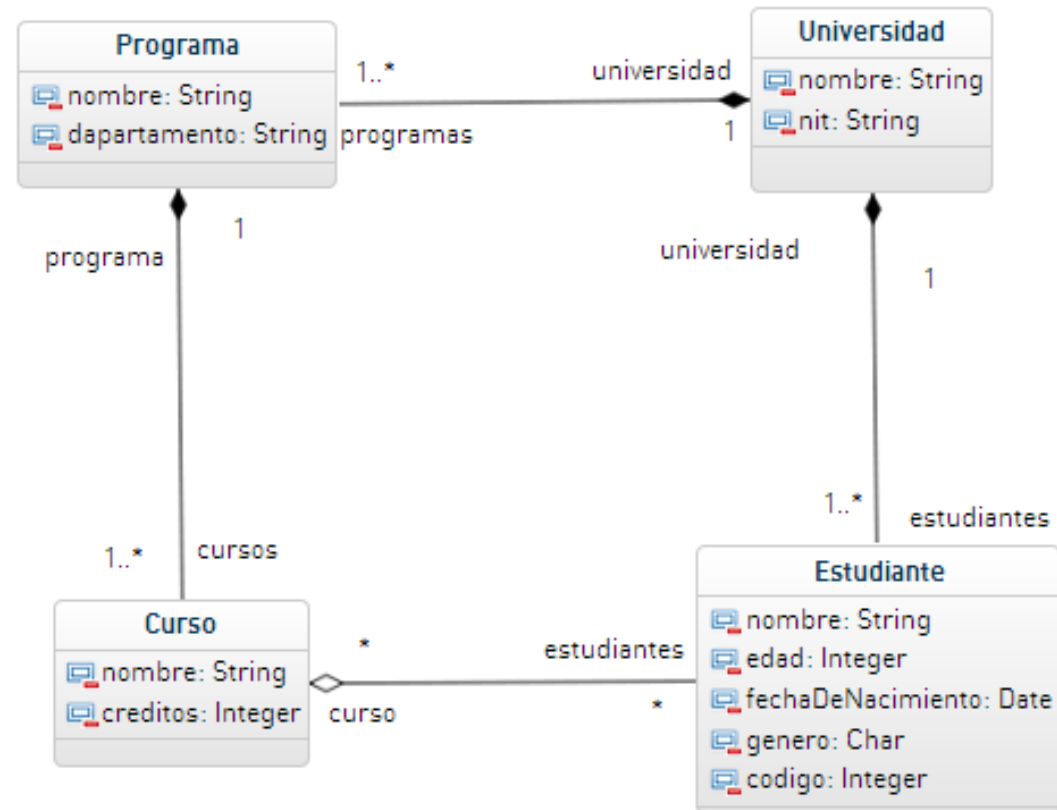



- Este diagrama de clases es una representación estática del sistema.





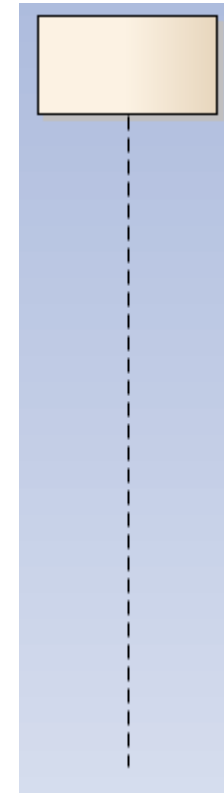
- Ahora estamos interesados en mostrar cómo interactúan objetos de estas clases.
- Es decir, estamos interesados en modelar el comportamiento del sistema



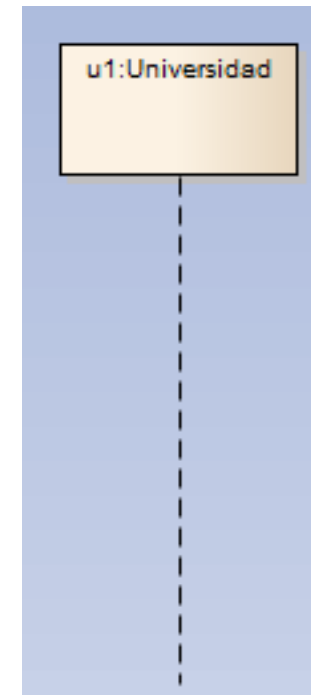


En un diagrama de secuencia  
se representan objetos y no  
clases.

Un objeto se representa con una caja rectangular y una línea punteada que sale de la caja hacia abajo.

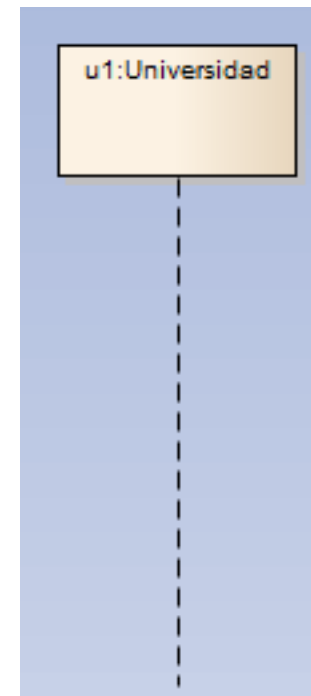


En el ejemplo, el objeto se llama **u1** y es una instancia de la clase Universidad.



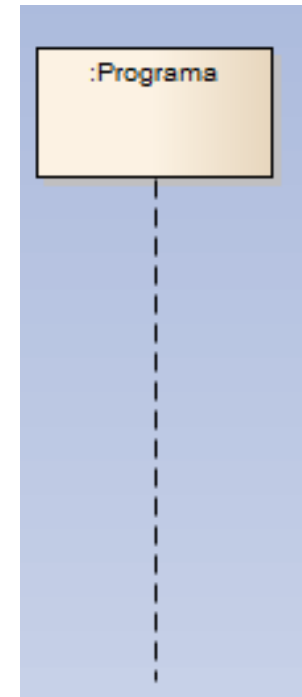
La línea punteada que sale de la caja se llama la línea de la vida del objeto (*lifeline*).

Esta línea establece un orden en las acciones que realiza el objeto.

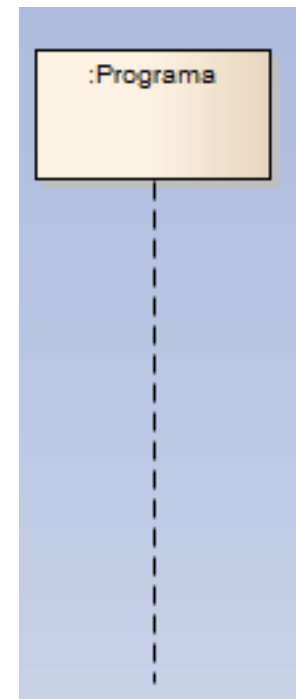


Es importante indicar siempre a qué clase pertenece el objeto.  
De qué clase es instancia.

Como en este ejemplo, el nombre del objeto puede omitirse, si dentro de lo que se está describiendo no se necesita precisarlo, pero siempre debe tener el nombre de la clase.  
Aquí estamos diciendo que tenemos un objeto instancia de la clase Programa

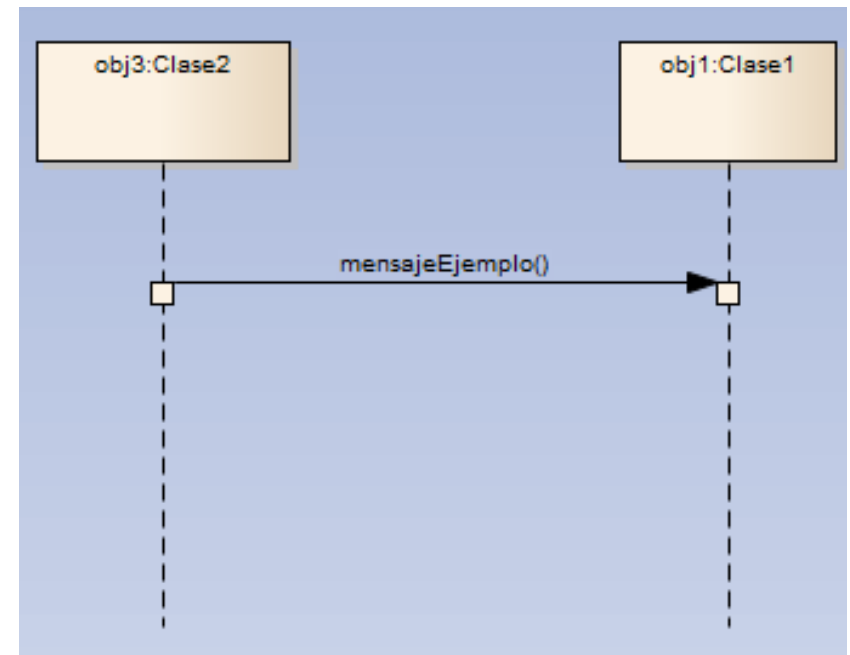


Cuando un objeto aparece en un diagrama de secuencia significa que éste está activo en ejecución, es decir que otros objetos se pueden comunicar con él.



La interacción entre los objetos se realiza a través de mensajes que se envían entre ellos.

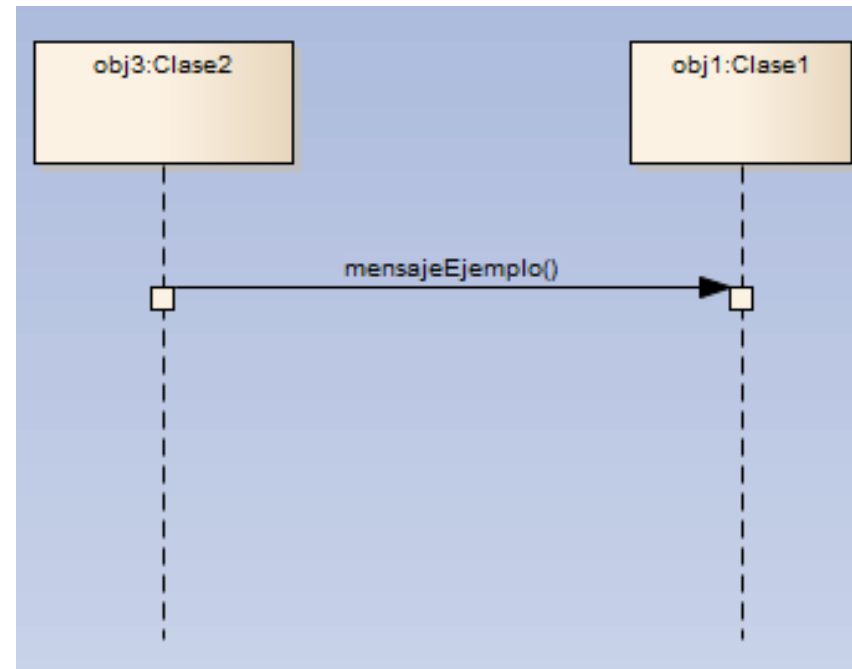
En la figura, el **obj3** de la clase Clase2 envía un mensaje llamado “mensajeEjemplo” al objeto **obj1** instancia de la clase Clase1





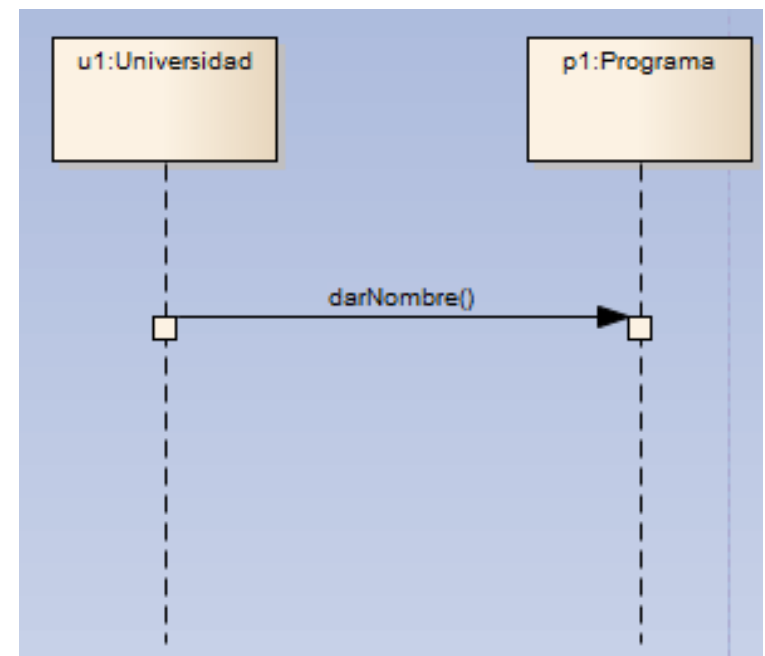
Los mensajes pueden ser de distintos tipos: síncronos o asíncronos, perdidos, encontrados, llamados o señales.

Por ahora vamos a focalizar en **mensaje síncronos** que corresponden a llamados de métodos.

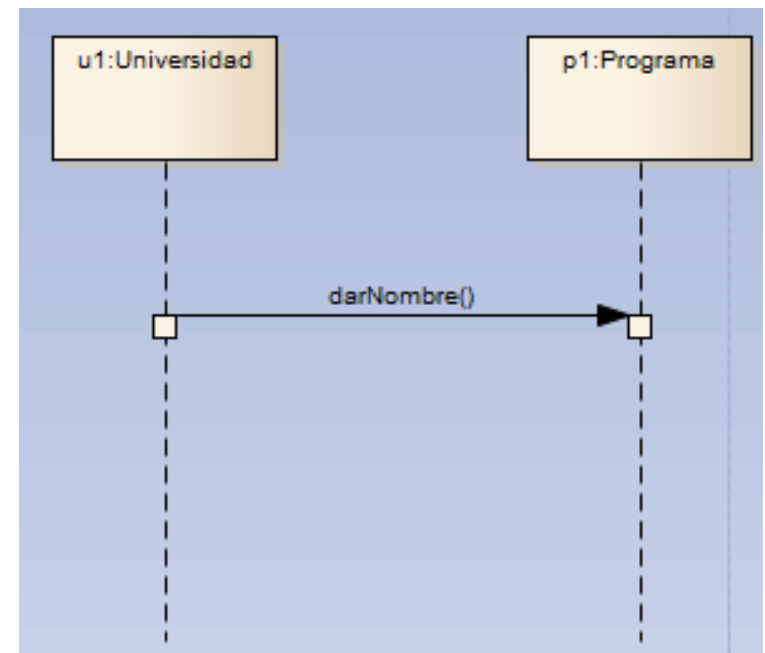


De nuestro ejemplo, decimos que el objeto **u1** de la clase Universidad le envía un mensaje al objeto **p1** de la clase Programa.

En este ejemplo, el mensaje corresponde al llamado del método **darNombre()**.

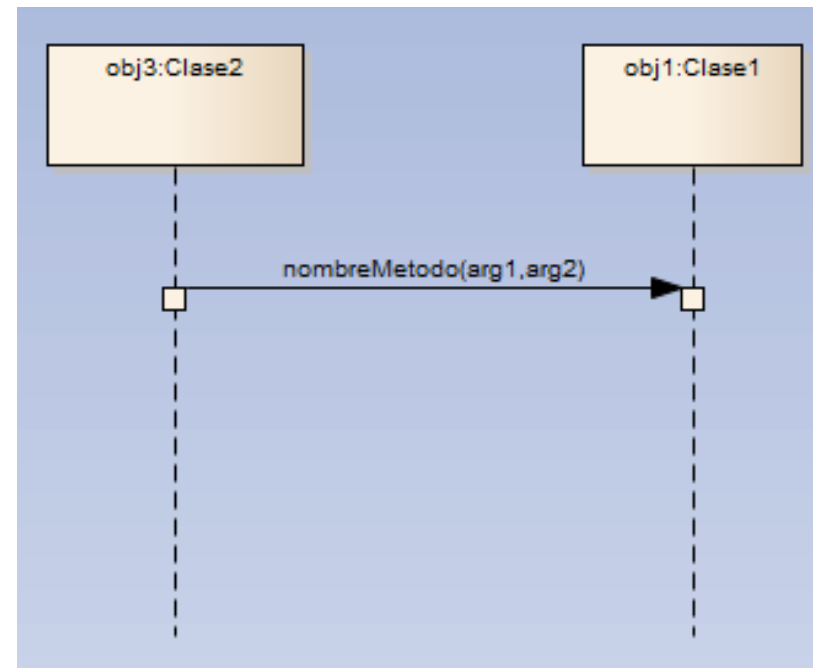


Este es un mensaje síncrono. Lo que significa que una vez que se ejecute el método darNombre en el objeto **p1**, la secuencia (el control) regresa al objeto quien hizo el llamado, en este caso el objeto **u1**.



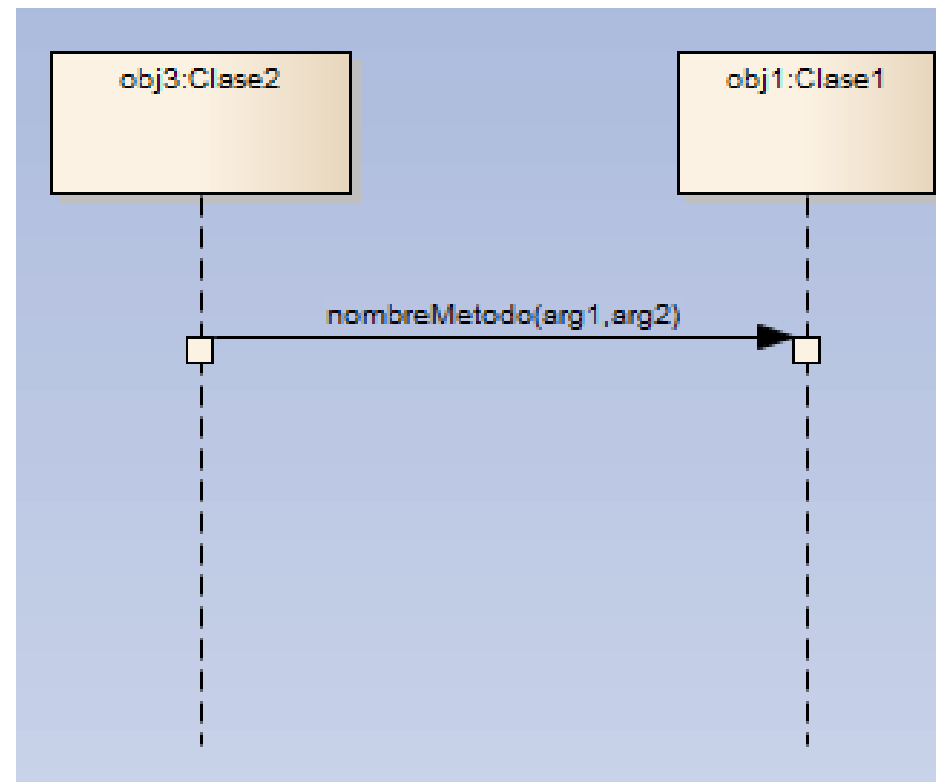
En este ejemplo, el **obj3** de la clase Clase2 envía un mensaje al objeto **obj1** de la clase Clase1.

Este mensaje es el llamado del método “nombreMetodo” con sus argumentos arg1 y arg2.

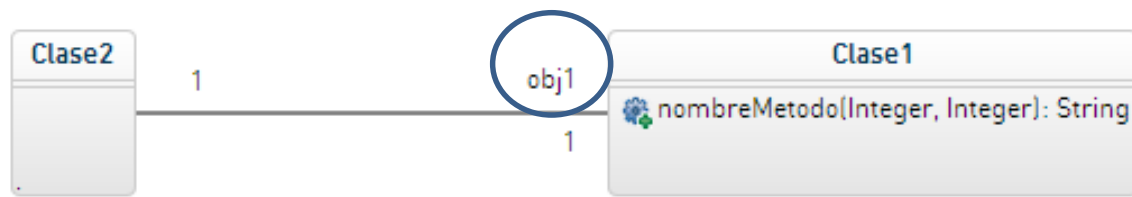
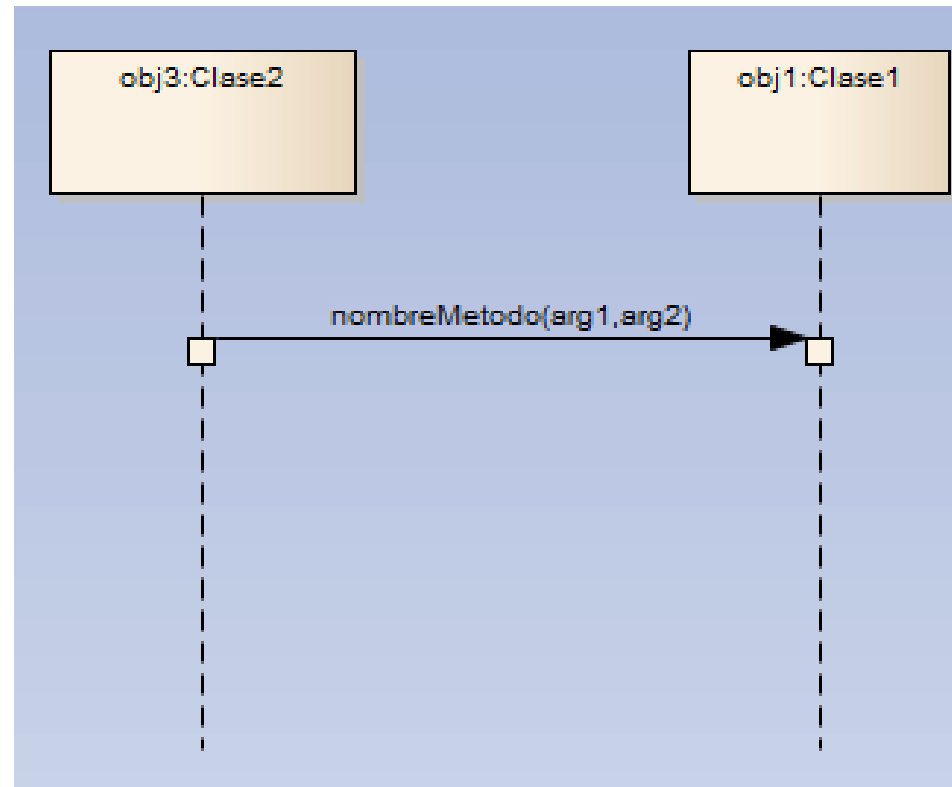


Note que para que este mensaje sea válido, se tienen que cumplir dos cosas:

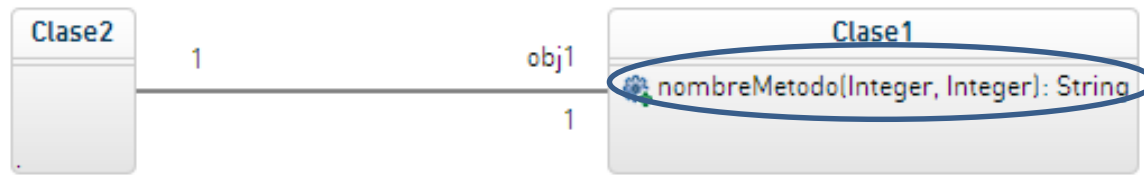
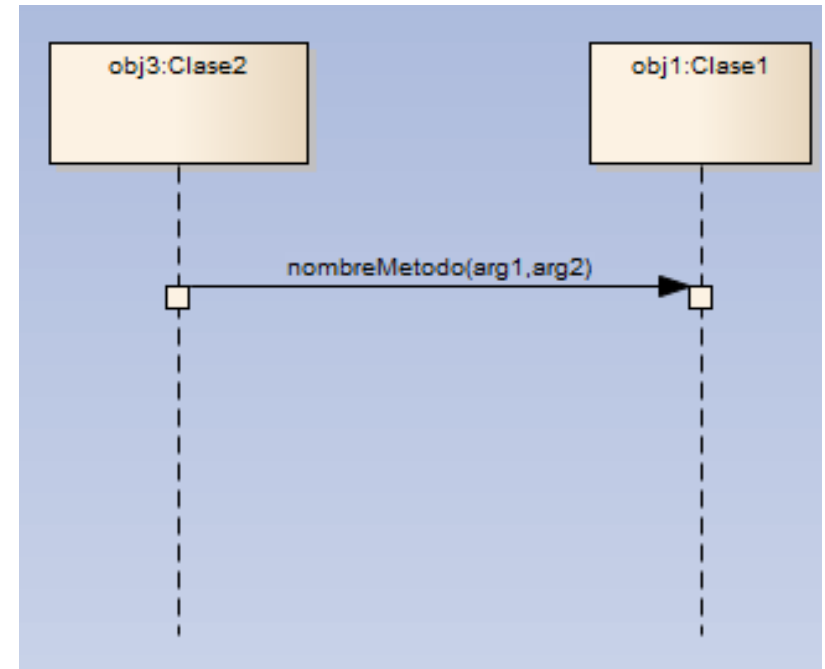
1. El objeto **obj3** “conoce” al objeto **obj1**.
2. El método *nombreMetodo* está definido en la clase Clase1 con los parámetros correspondientes para que el llamado sea válido



En el diagrama de clases ejemplo, todos los objetos de la clase Clase2 tienen una asociación con un objeto de Clase1 llamado obj1. Así, el obj3 conoce al obj1.

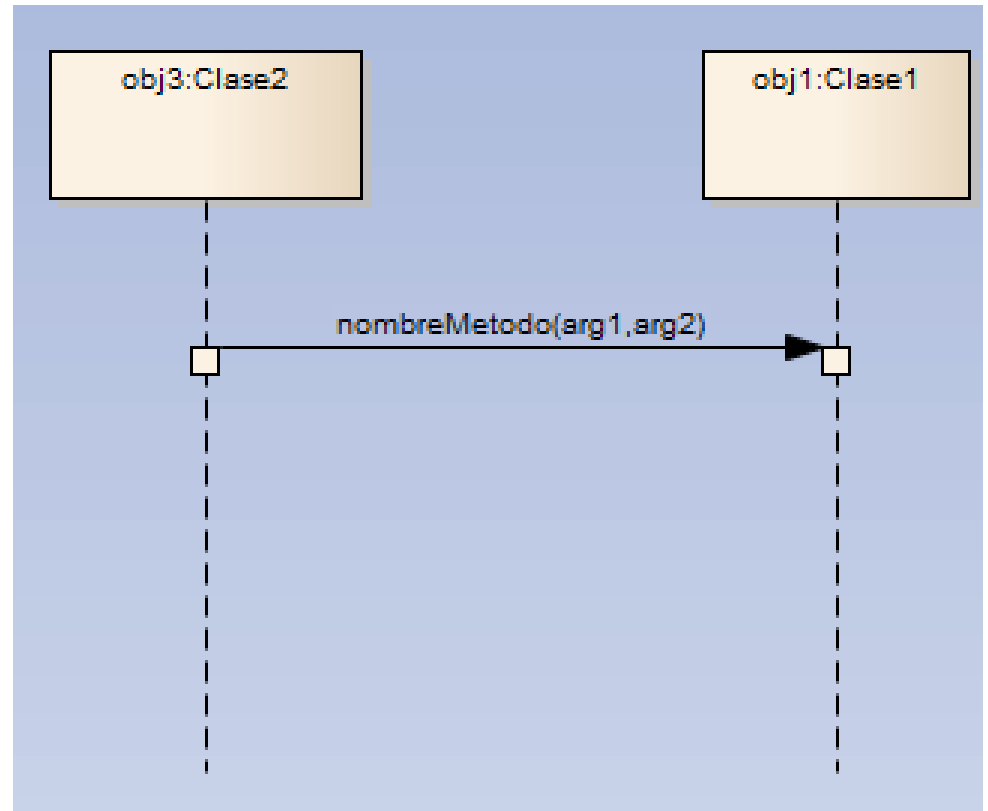


La segunda condición  
corresponde al nombre del  
método. Aquí podemos  
observar que el método está  
efectivamente definido en la  
Clase1



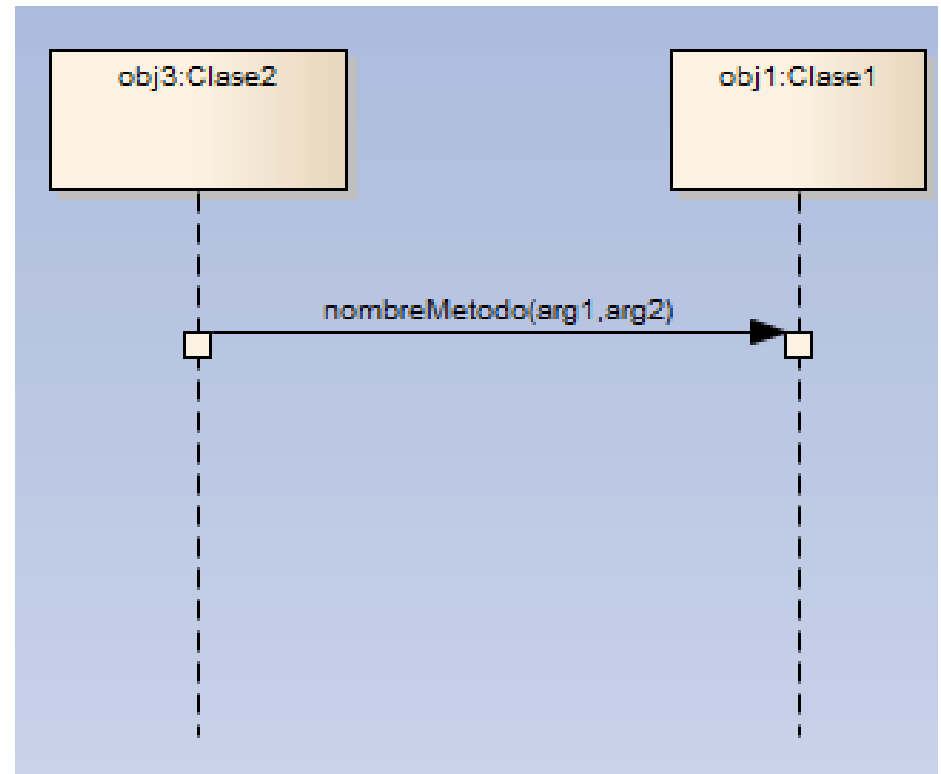
Más sobre la notación:

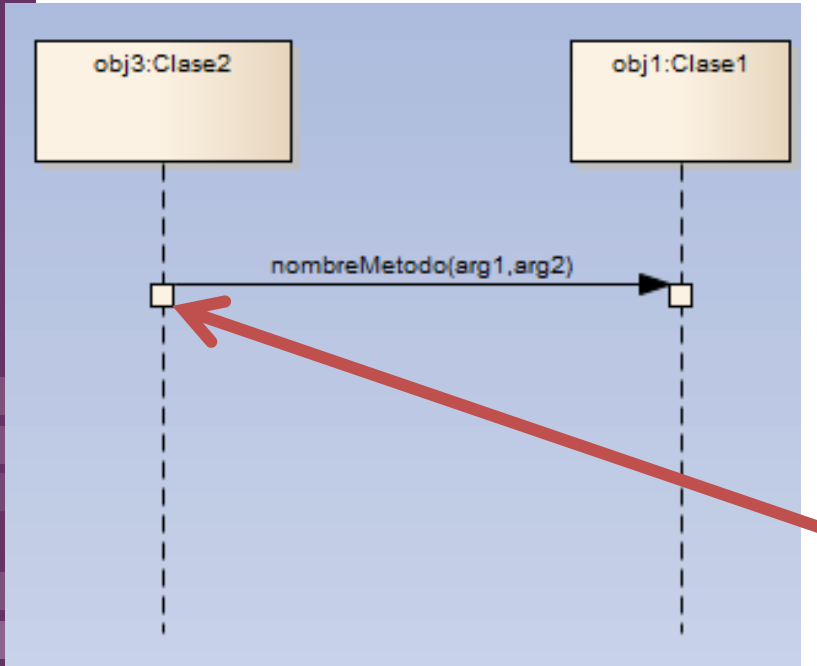
El rectángulo delgado en cada uno de los objetos representa una ocurrencia de ejecución o activación de un foco de control.



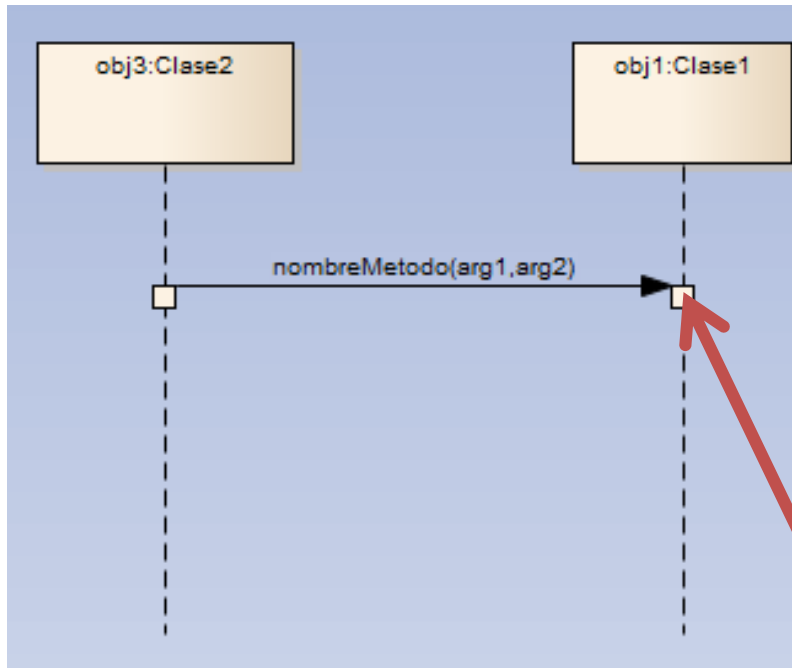


En este diagrama hay dos ocurrencias de ejecución o dos rectángulos delgados sobre las líneas de vida de los objetos.

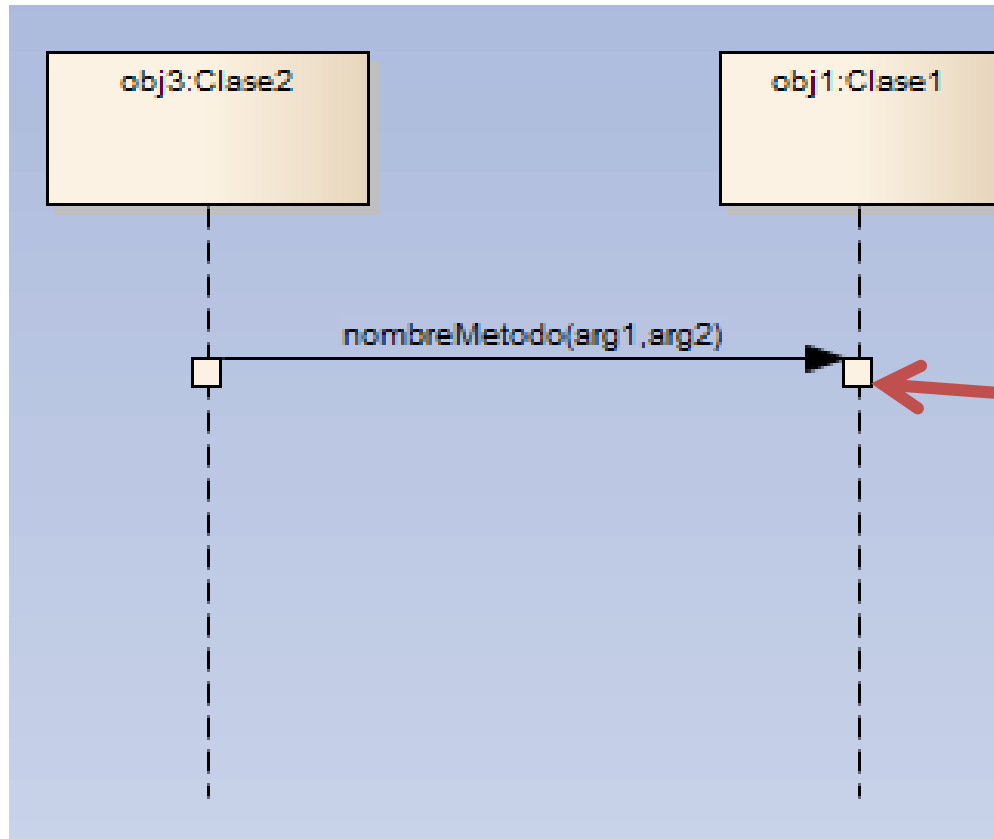




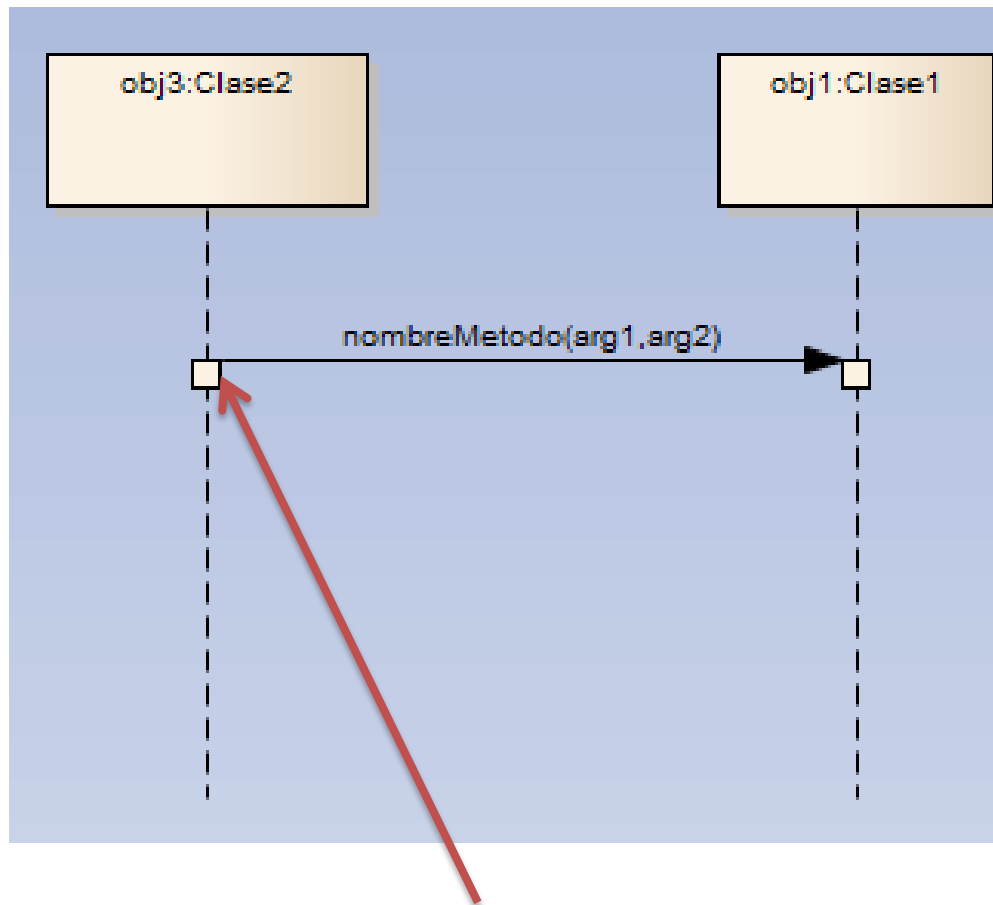
En el primer caso se está representando que la invocación al método ocurrió durante ese foco de control.



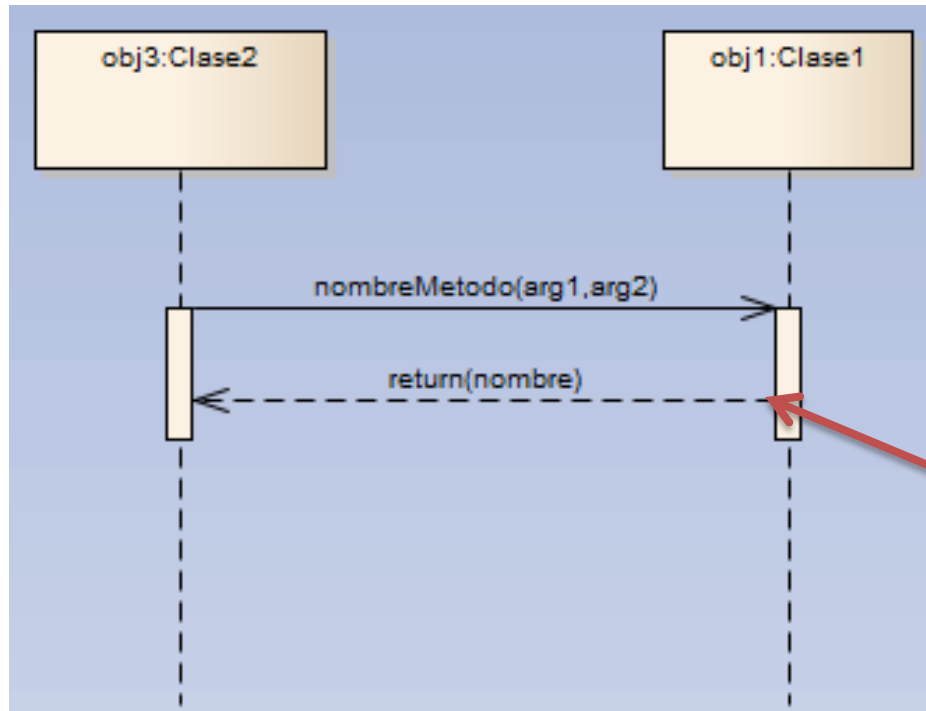
En el segundo caso, se está representando que como consecuencia de la invocación, se creó una ocurrencia de ejecución para el objeto obj1.



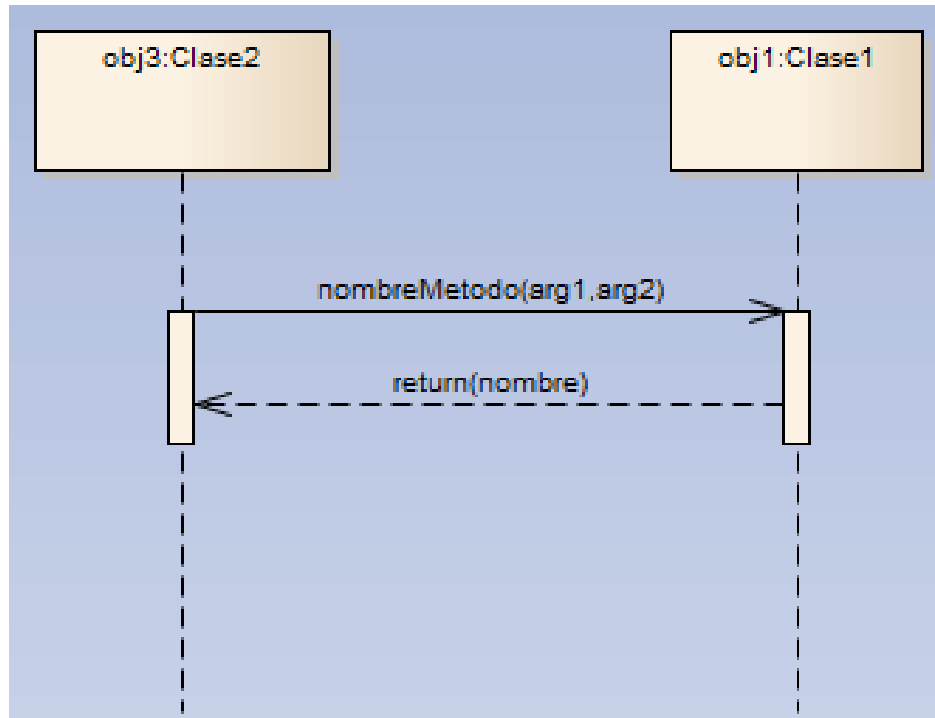
Cuando se termina la ejecución del método en obj1, se acaba la ocurrencia de ejecución (como si el objeto ahora estuviera de nuevo en reposo listo para que otros objetos se comuniquen con él).



Dado que en nuestro ejemplo se trata de un llamado síncrono, una vez que se termina la ejecución del método, el control vuelve al objeto **obj3** después del llamado (donde indica la línea roja)



Si el llamado del método fuera asíncrono, se debe indicar el regreso del llamado.



En la notación, la diferencia entre síncrono y asíncrono está en la flecha del mensaje.



síncrono




asíncrono

# Diagramas de secuencia

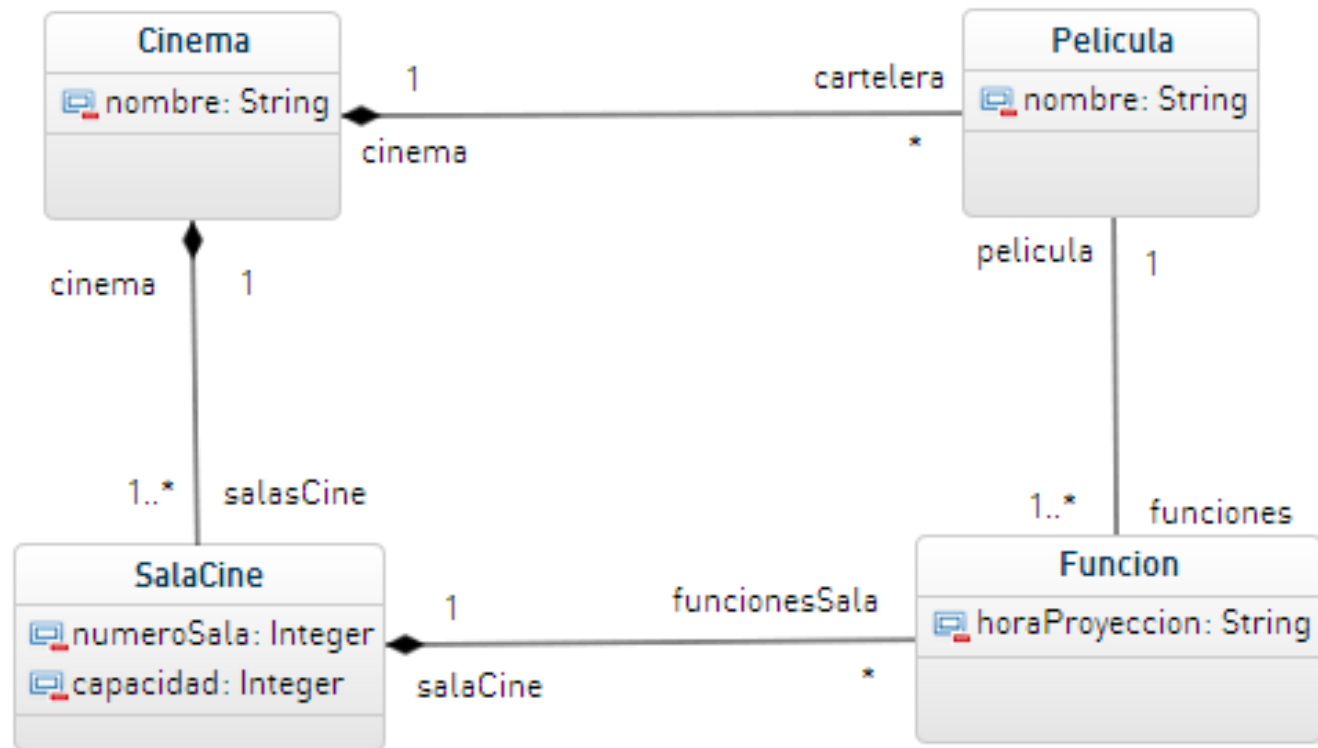
Interacciones básicas- Ejemplo  
Patrón Experto



- 
- Como ya mencionamos, los diagramas de secuencia se pueden utilizar para :
  - Modelar el comportamiento de los escenarios de caso de uso
  - Modelar la lógica de un método
  - Modelar la lógica de un servicio
  - En el siguiente ejemplo modelaremos escenarios de caso de uso y a través de este ejemplo reforzaremos los conceptos de los patrones GRASP y de los diagramas de secuencia.

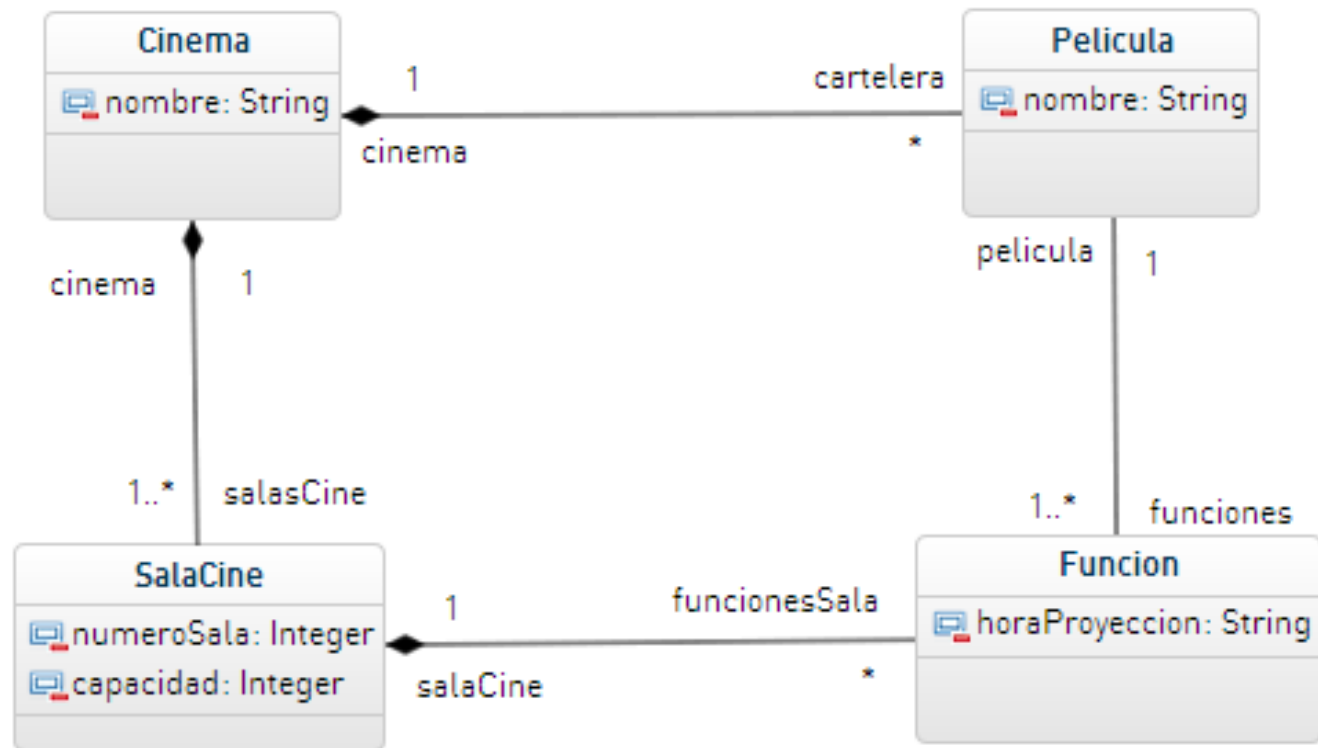
# Ejemplo

- Suponga que tenemos el siguiente diagrama de clases sobre Cinema.



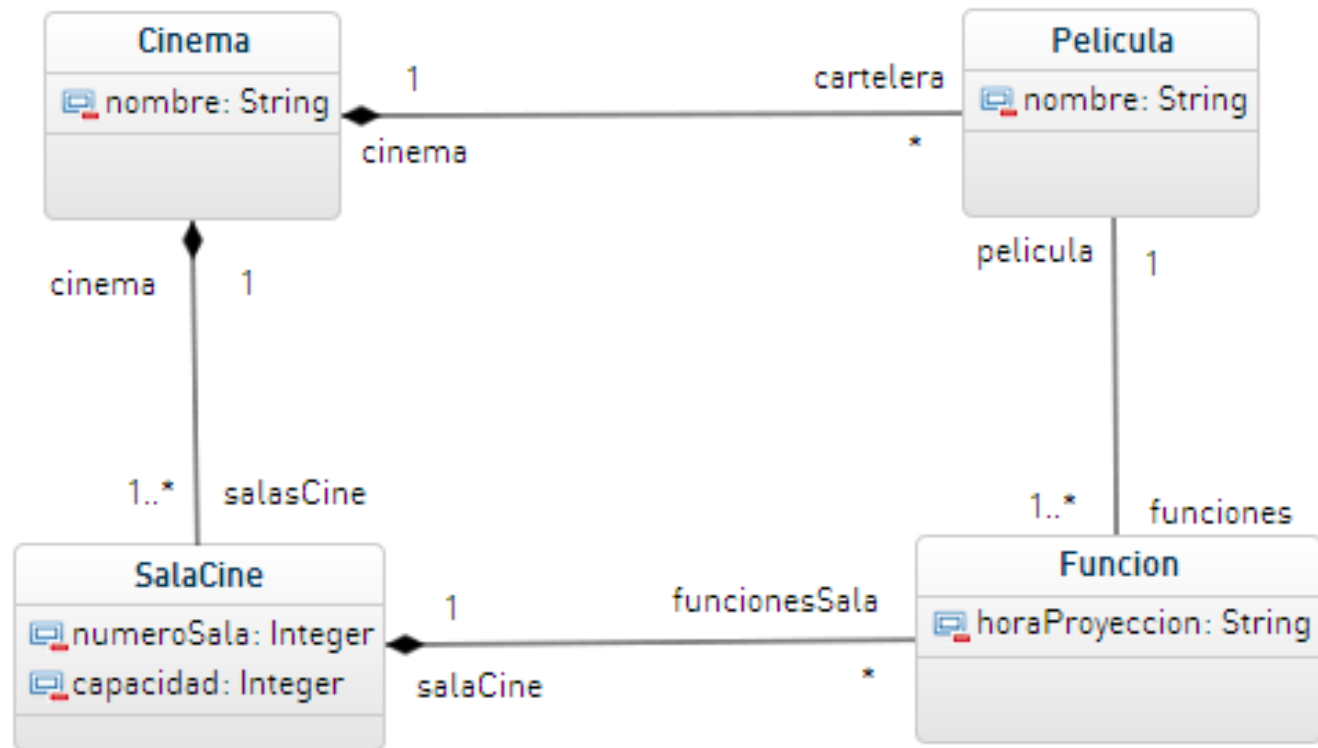
# Ejemplo

- Este diagrama modela los elementos estructurales de un cinema: Cinema, SalaCine, Pelicula, Funcion
- El Cinema se compone de salas de cine, cada una con una capacidad y un número de sala.



# Ejemplo

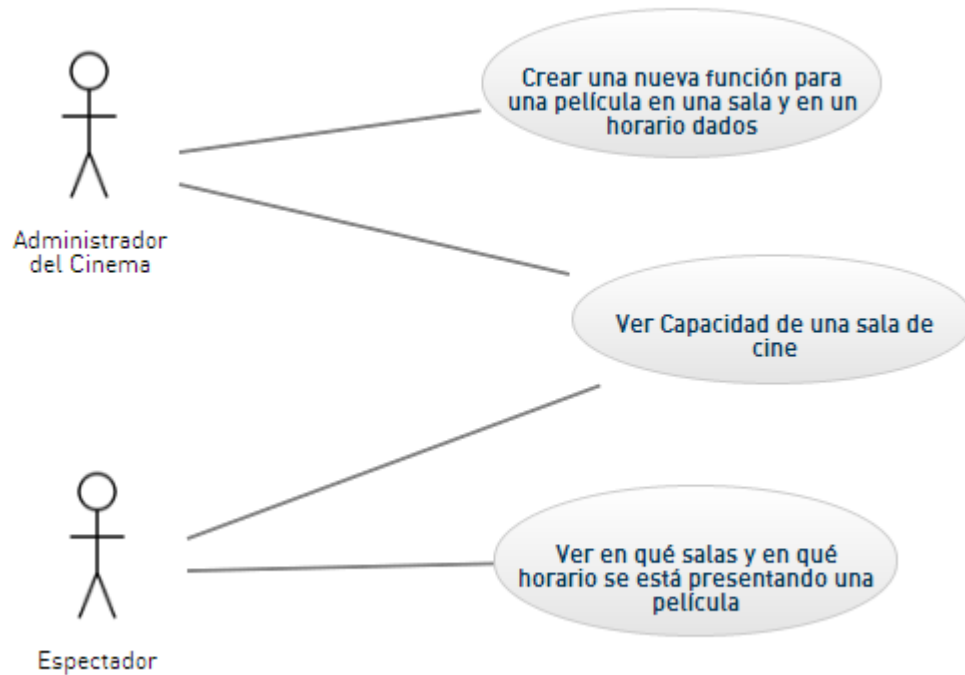
- El Cinema es el dueño de su cartelera que está compuesta por películas y tenemos también, el concepto Función que relaciona en cuál sala de cine y a qué hora se proyecta una película en particular.



# Ejemplo

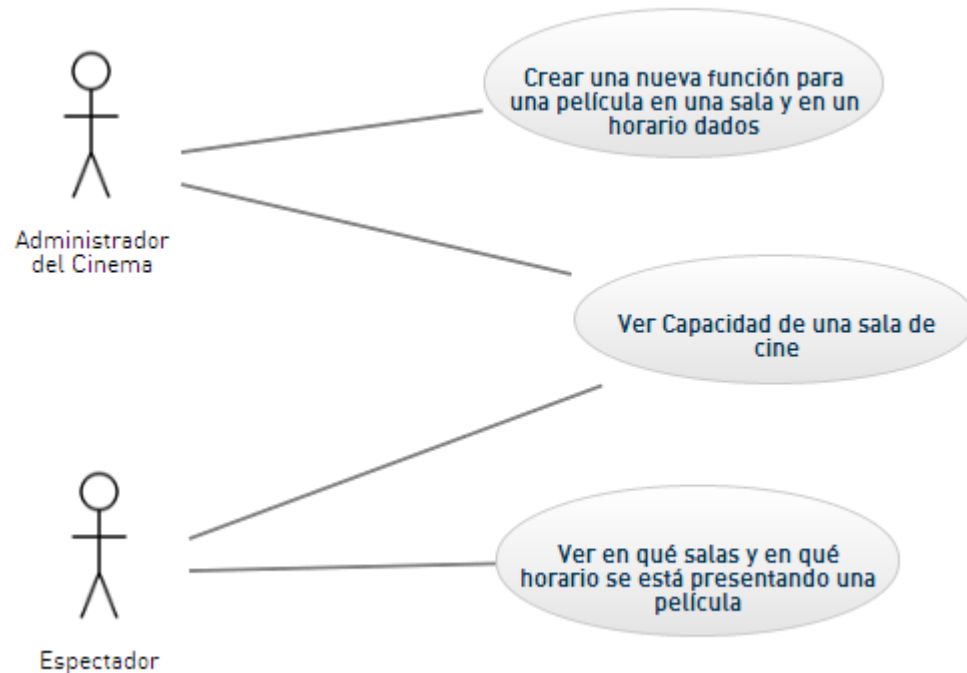
- Suponga que para este mundo del Cinema nos interesa resolver los siguientes casos de uso:

# Ejemplo



# Ejemplo

- Cuando tenemos los casos de uso, nos interesa verificar si la información que tenemos en el diagramas de clases es suficiente para poder resolverlo.

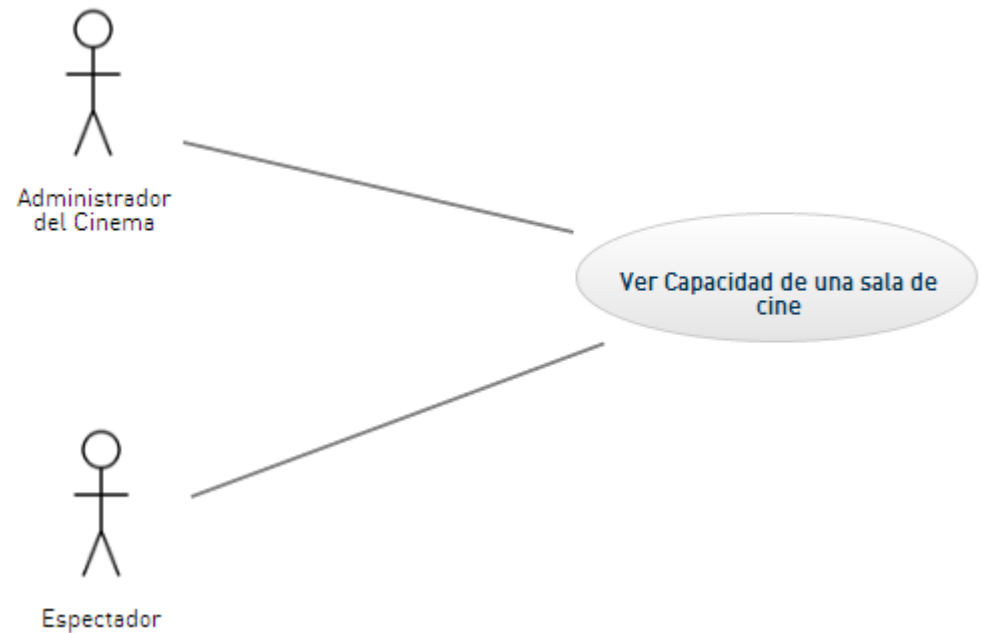


Para esto utilizamos los patrones GRASP y los diagramas de secuencia.

Como estamos modelando comportamiento, vamos a tener que preguntarnos “de quién es la responsabilidad de conocer o hacer algo?” y vamos a ir tomando decisiones asignando métodos a las clases, eventualmente cambiando o agregando relaciones, en la medida en que vamos desarrollando el diagrama de secuencia.

# Ejemplo

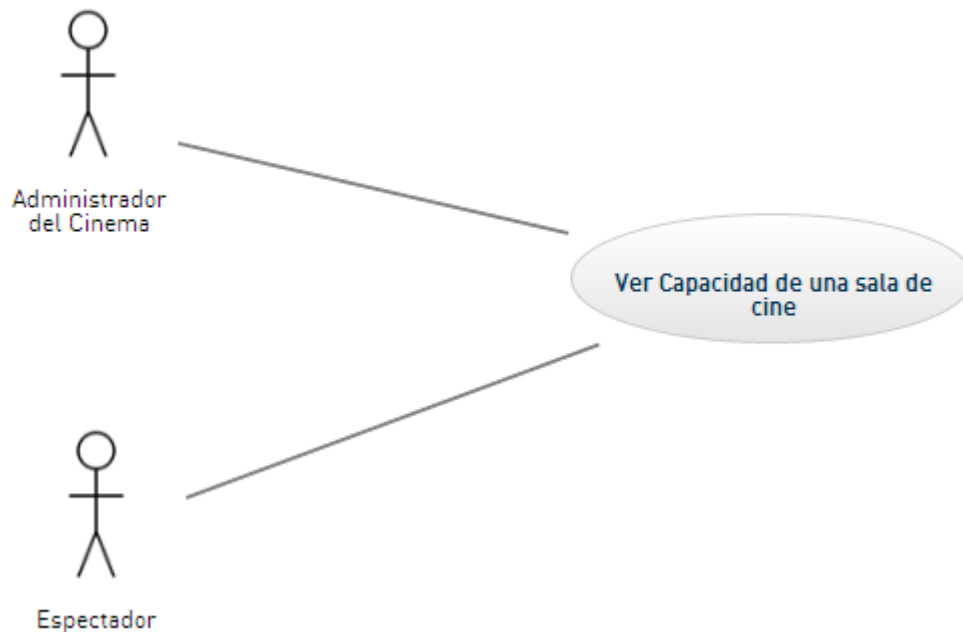
- Vamos a modelar el diagrama de secuencia del caso de uso ver capacidad de la sala de cine.





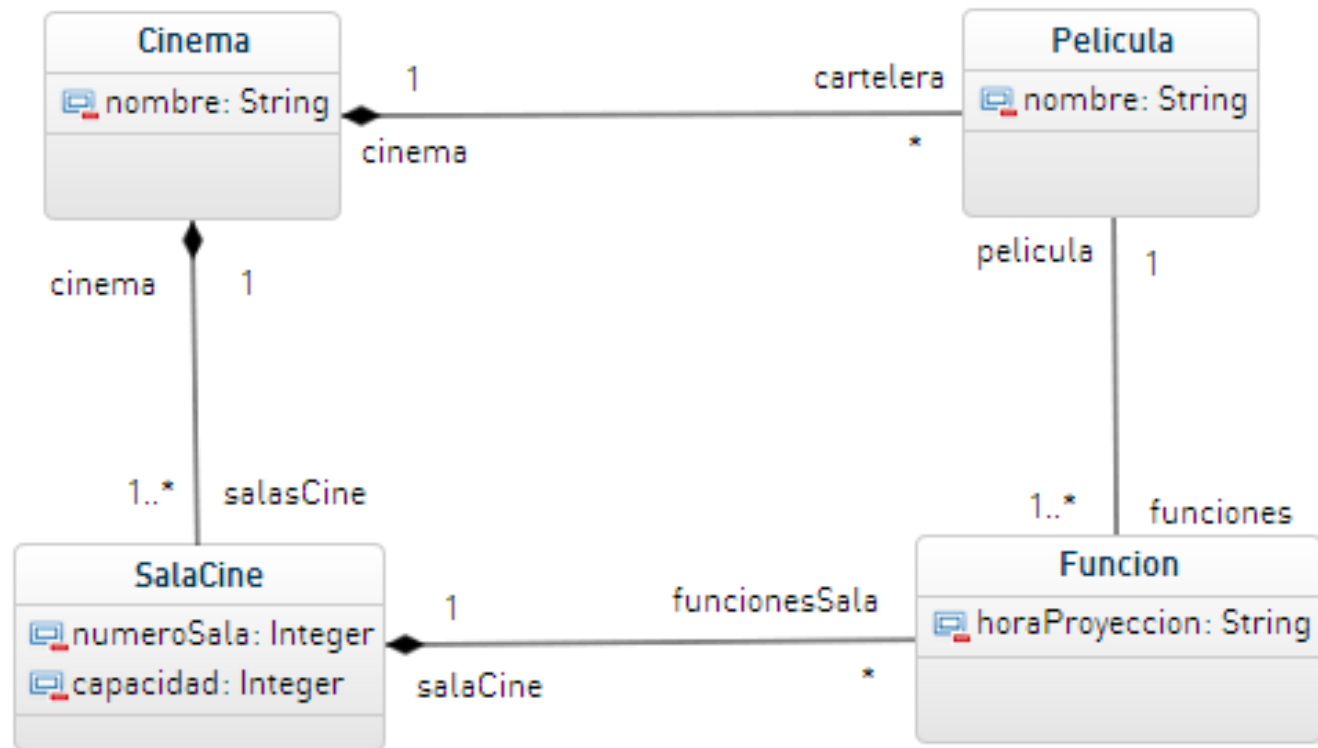
# Ejemplo

- La primera pregunta que debemos hacernos es: quién es el dueño de la información de la capacidad de la sala de cine?



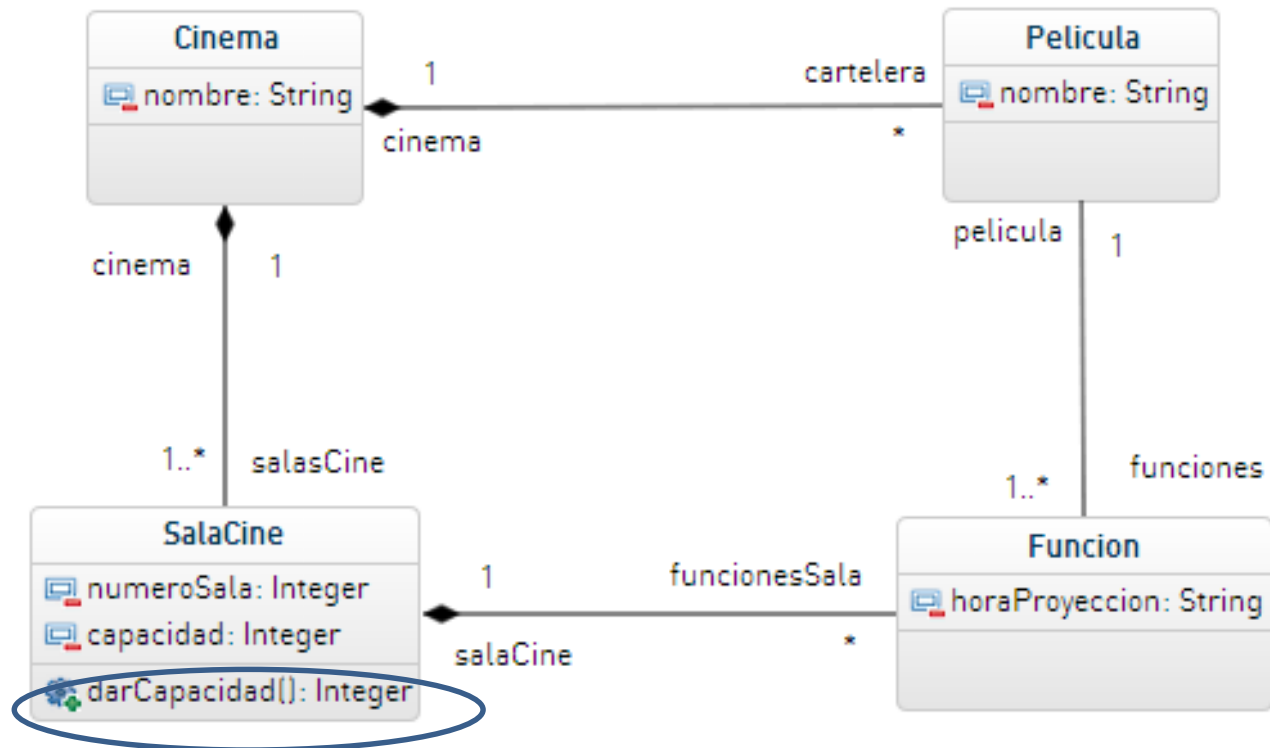
# Ejemplo

- Si observamos el diagrama de secuencia, la clase dueña de esta información es la SalaCine



# Ejemplo

- Siguiendo el patrón Experto, la clase SalaCine debe tener un método darCapacidad() que devuelva el valor del atributo capacidad.



# Ejemplo

Entonces tenemos que dado un objeto instancia SalaCine, este objeto es responsable de dar la información sobre su capacidad.

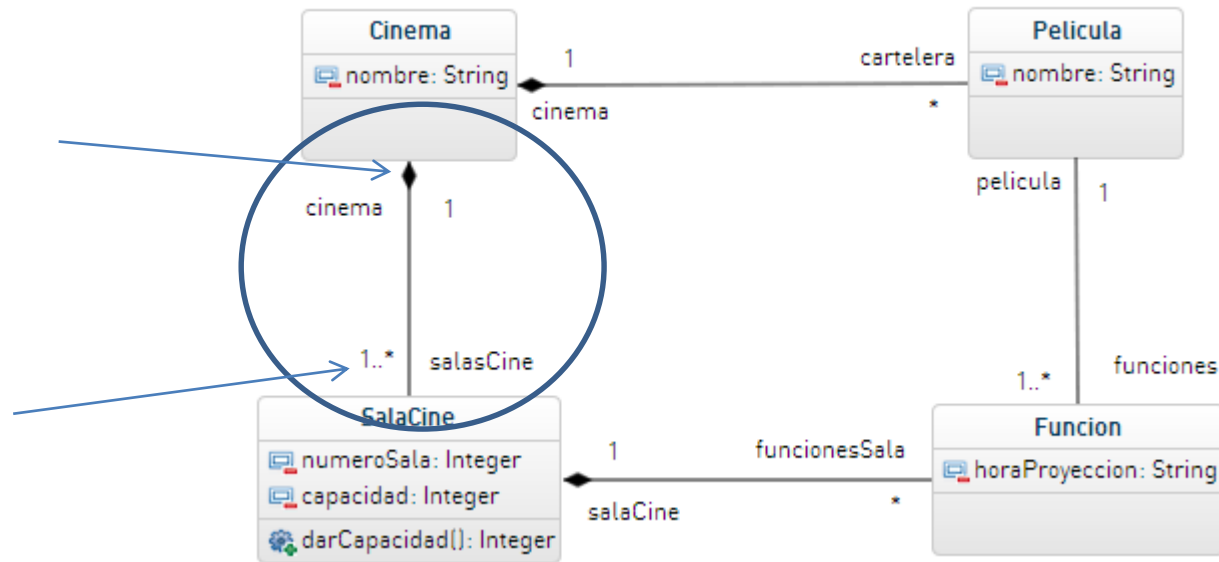
Pero para resolver el caso de uso, debemos preguntarnos: cuál es ese objeto SalaCine?

Ese objeto es el correspondiente a un número de sala dado por el actor del caso de uso. Entonces podemos precisar la pregunta anterior por:

**De quién es la responsabilidad de encontrar el objeto SalaCine que corresponde a un número dado?**

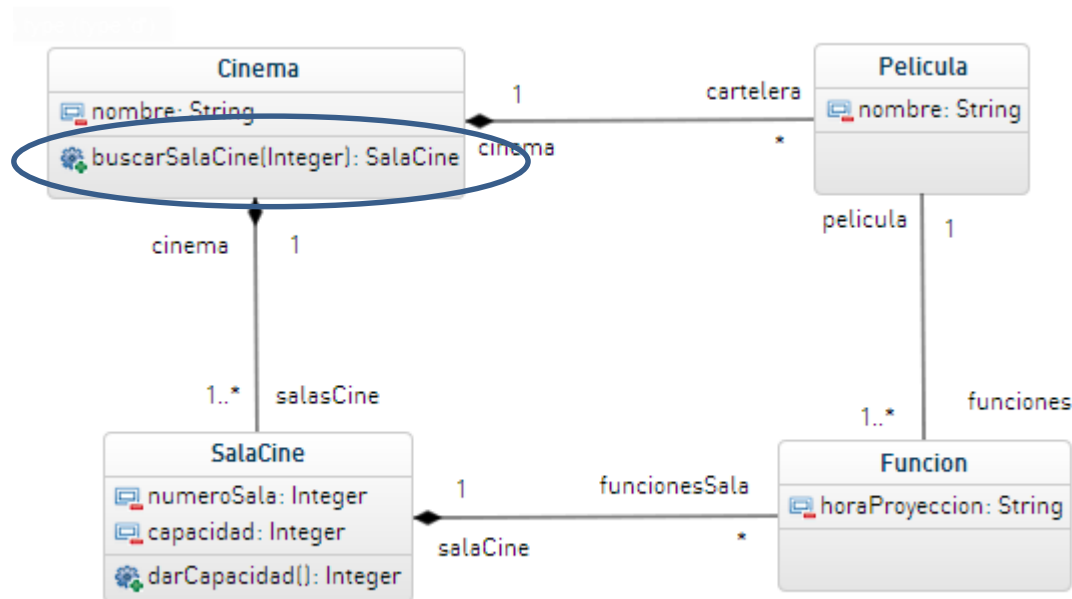
# Ejemplo

- Siguiendo con el patrón Experto, vemos que la clase Cinema es la dueña de la colección de salasCine.



# Ejemplo

- Así, agregamos el método `buscarSalaCine(int)` quién recibe el número de la sala deseado y retorna un objeto `SalaCine`.



# Ejemplo

- Para modelar el caso de uso, necesitamos un responsable de coordinar las tareas:
  1. Recibir desde afuera (afuera significa algún tipo de interfaz con el usuario actor del caso de uso) del cinema la petición de dar capacidad de una sala de cine dado un número
  2. buscar el objeto SalaCine que corresponde a ese número
  3. Pedirle a ese objeto su capacidad.

# Ejemplo

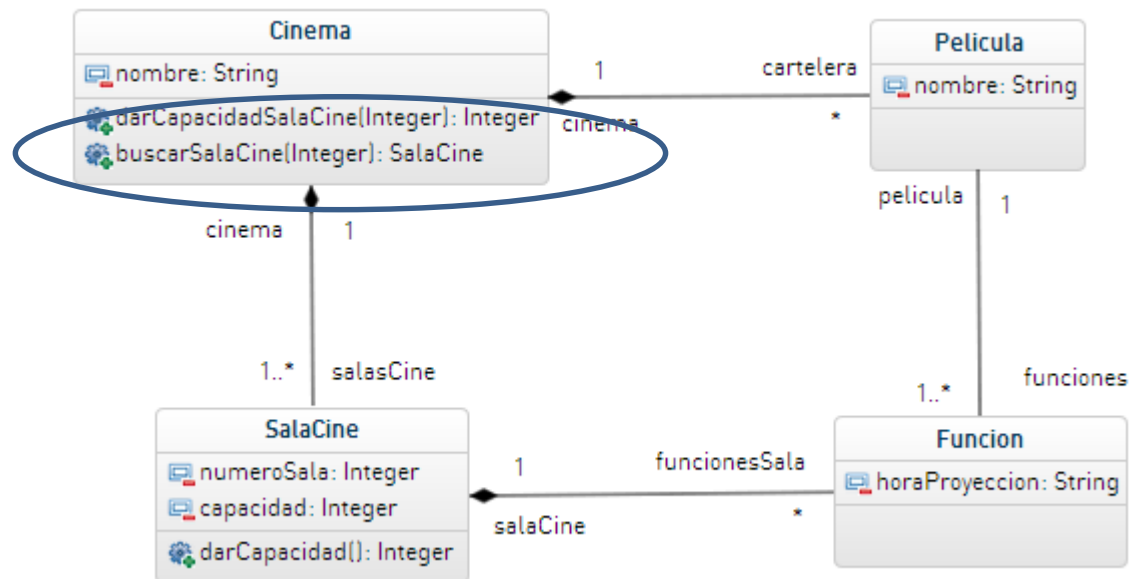
Nos falta contestar la pregunta: quién es el responsable de recibir las peticiones desde afuera ?

La respuesta es la clase Cinema quién está aquí jugando un rol de principal o de controlador del sistema.



# Ejemplo

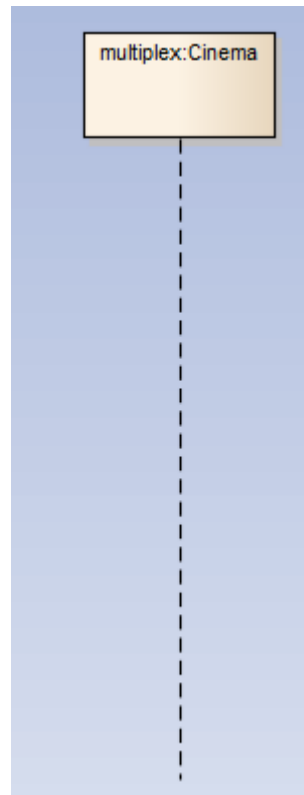
Agregamos entonces un nuevo método a la clase Cinema  
darCapacidadSalaCine(int) que recibe como argumento el numero de la sala deseada.



# Ejemplo

- Ahora podemos modelar el comportamiento de este caso de uso mostrando la secuencia de las interacciones entre los objetos.

# Ejemplo

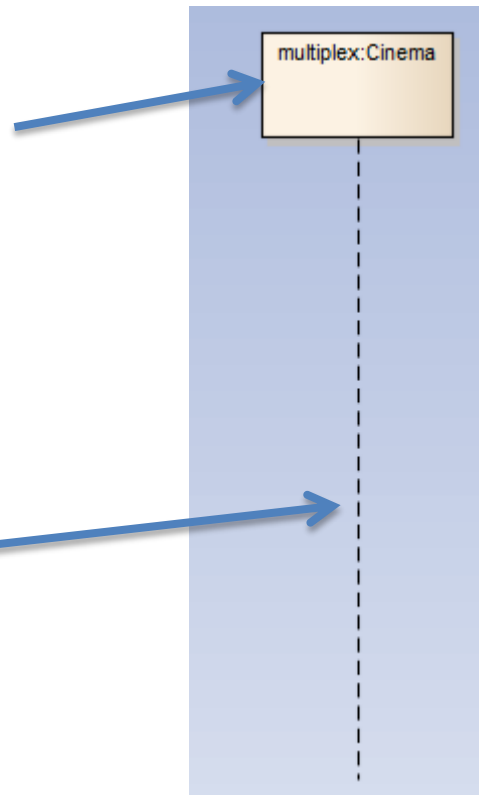


- Vamos a suponer que ya hay en ejecución un objeto de la clase Cinema que va a recibir los mensajes de los actores (las peticiones) de los casos de uso.
- Para propósitos del ejemplo este objeto lo hemos llamado “multiplex”

# Ejemplo

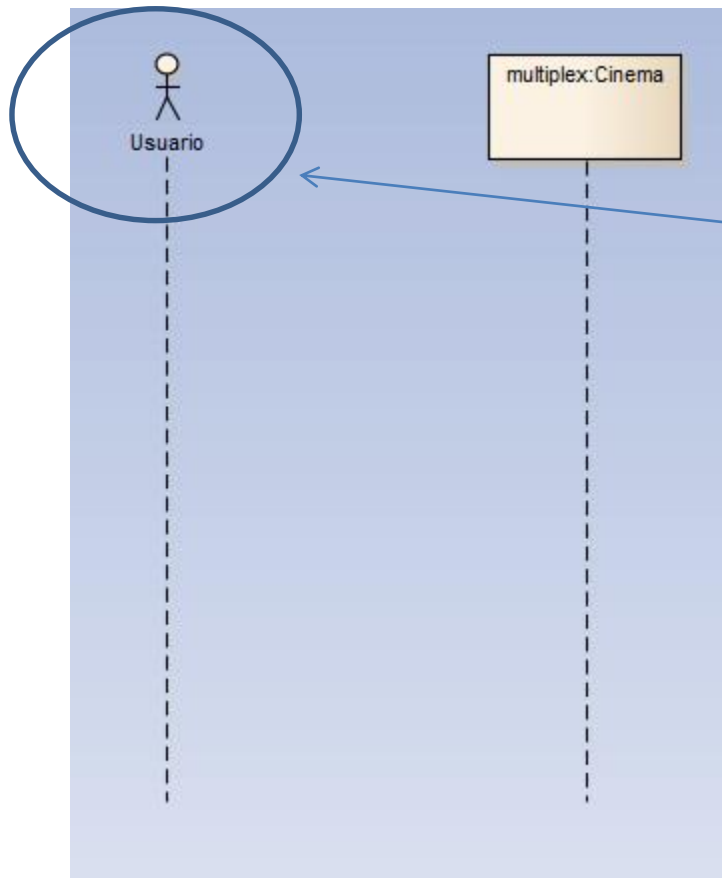
multiplex es el nombre del objeto  
Cinema es la clase a la que pertenece el objeto

La línea punteada representa la línea de vida del objeto.  
El tiempo va de abajo hacia arriba. Esto es importante para entender el orden de la secuencia de interacción.



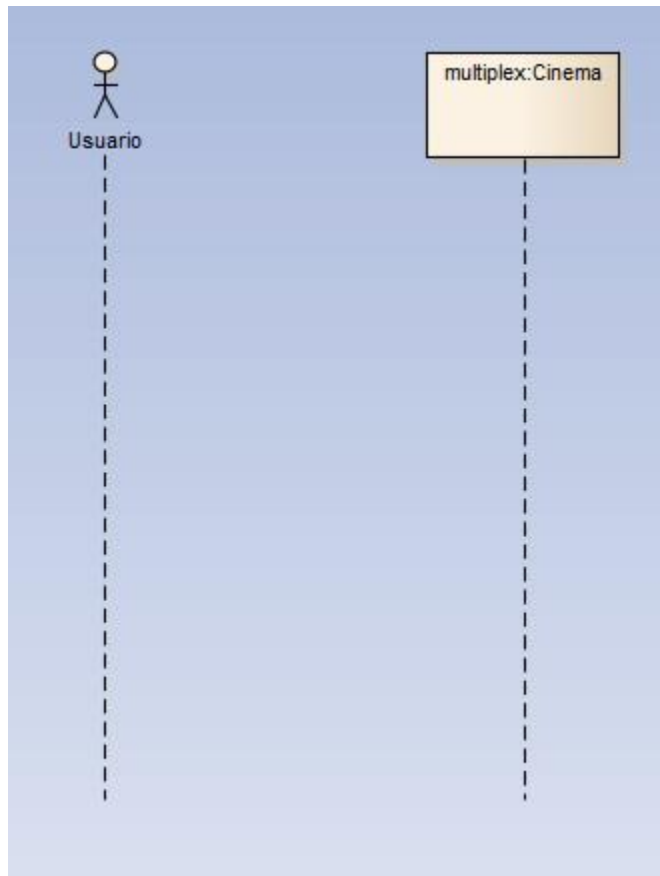
- Para recordar la sintaxis, note que el objeto se representa con el rectángulo que contiene el nombre del objeto y la clase a la que pertenece y su línea de vida.

# Ejemplo



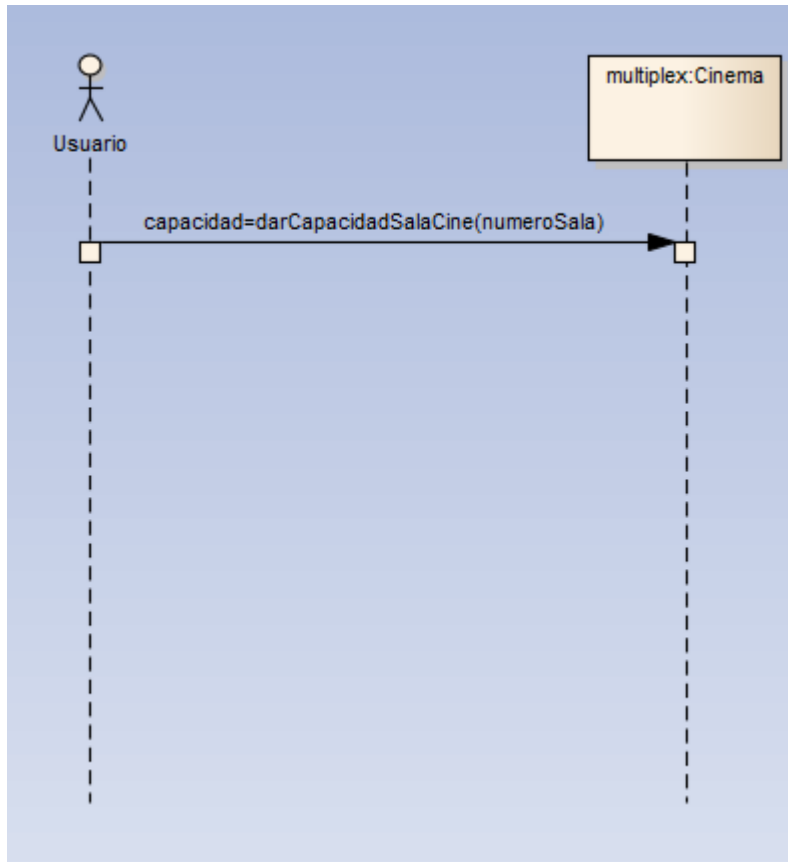
- El objeto multiplex recibe un mensaje de un usuario externo al sistema que le solicita la capacidad de una sala de cine y da el número de la sala .
- Note que aquí no estamos modelando la interfaz con el usuario, estamos suponiendo que “algo externo” se comunica con el objeto multiplex.

# Ejemplo



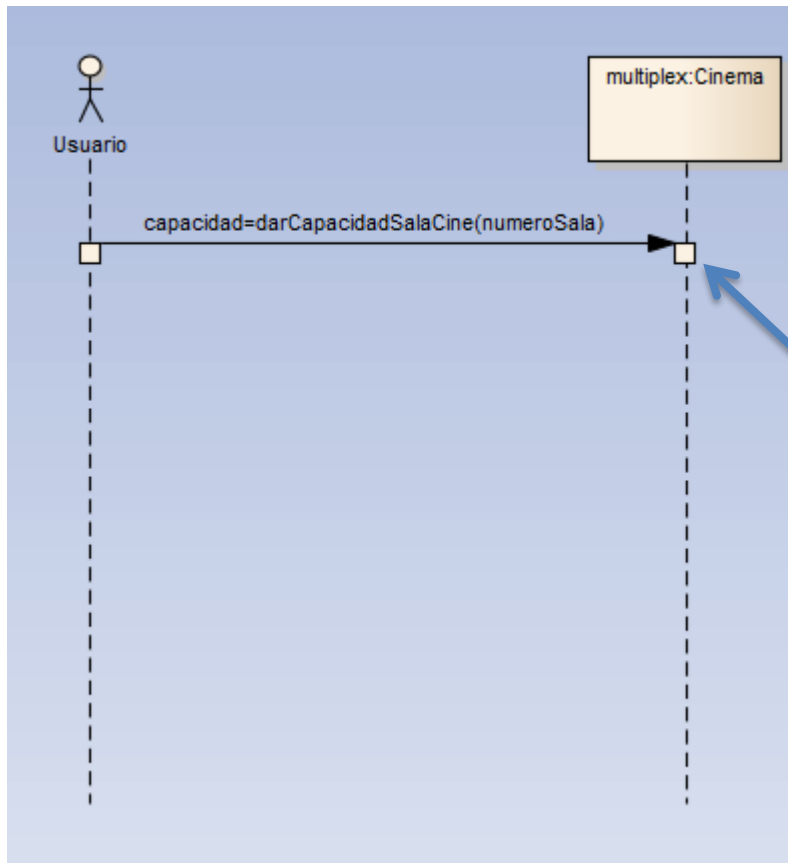
- El objeto multiplex recibe un mensaje que le solicita la capacidad de una sala de cine y da el número de la sala .

# Ejemplo



- Es un llamado síncrono
- El método ya está definido en la clase Cinema y retorna un valor entero que en este diagrama lo vamos a recoger en una variable llamada “capacidad”

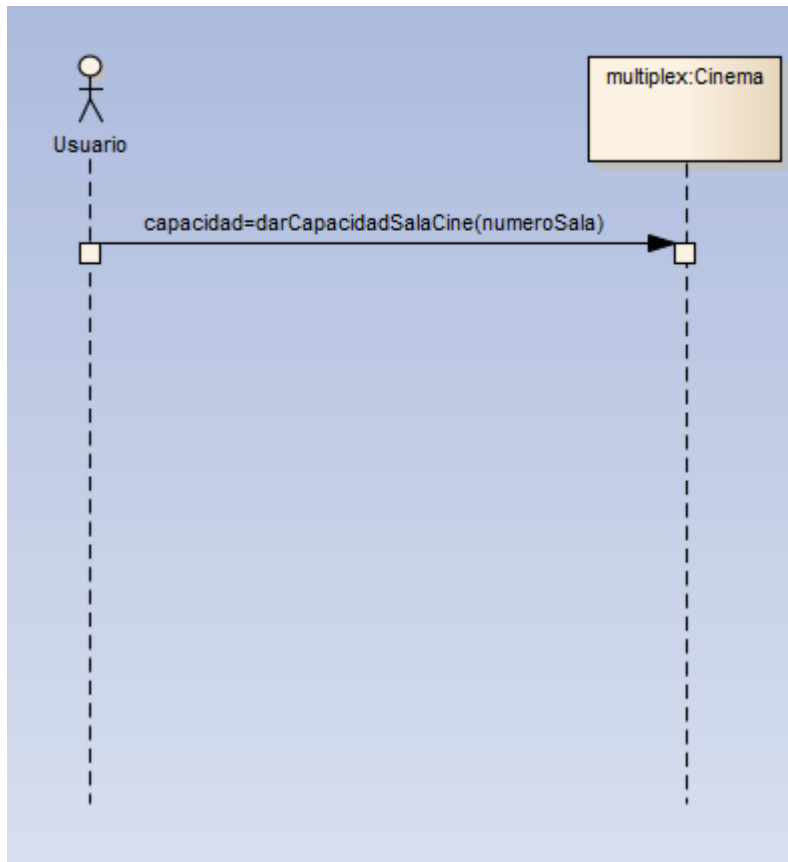
# Ejemplo



- La recepción del mensaje por el objeto multiplex, inicia una ocurrencia de ejecución del método `darCapacidadSalaCine(int)`.

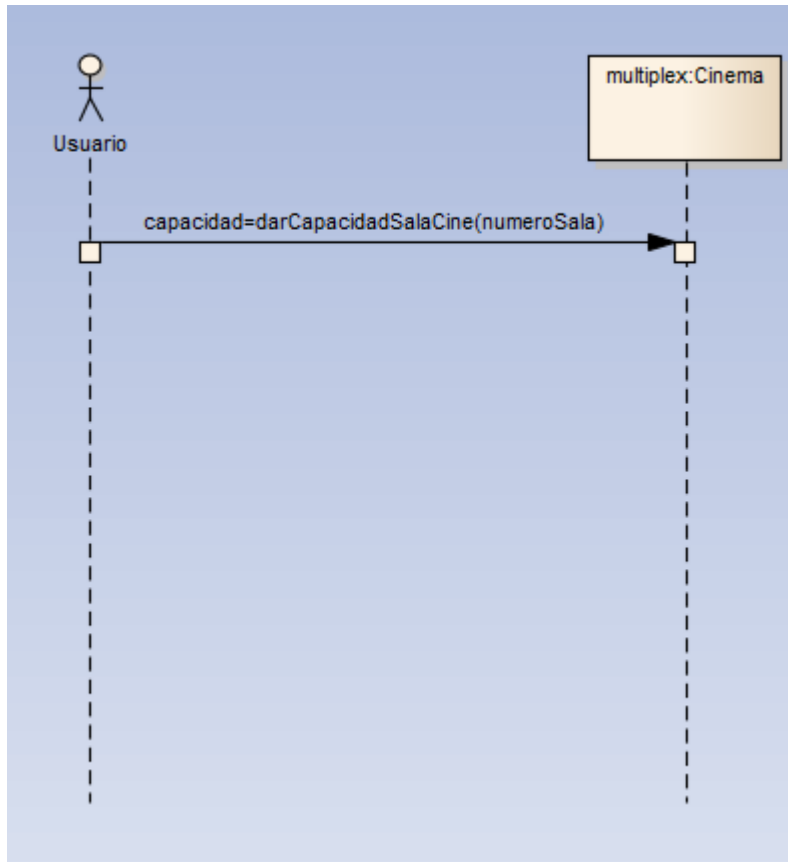


# Ejemplo



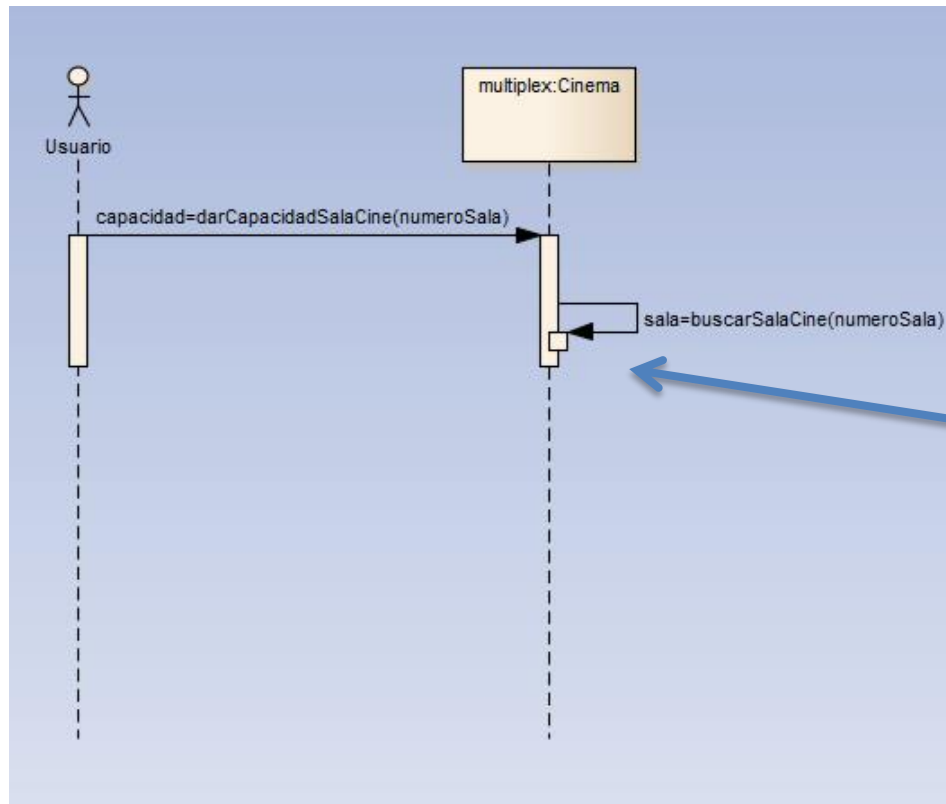
- Ahora modelamos el comportamiento del método `darCapacidadSalaCine(int)`, es decir, lo que pasa a partir del inicio de la ocurrencia de ejecución.

# Ejemplo



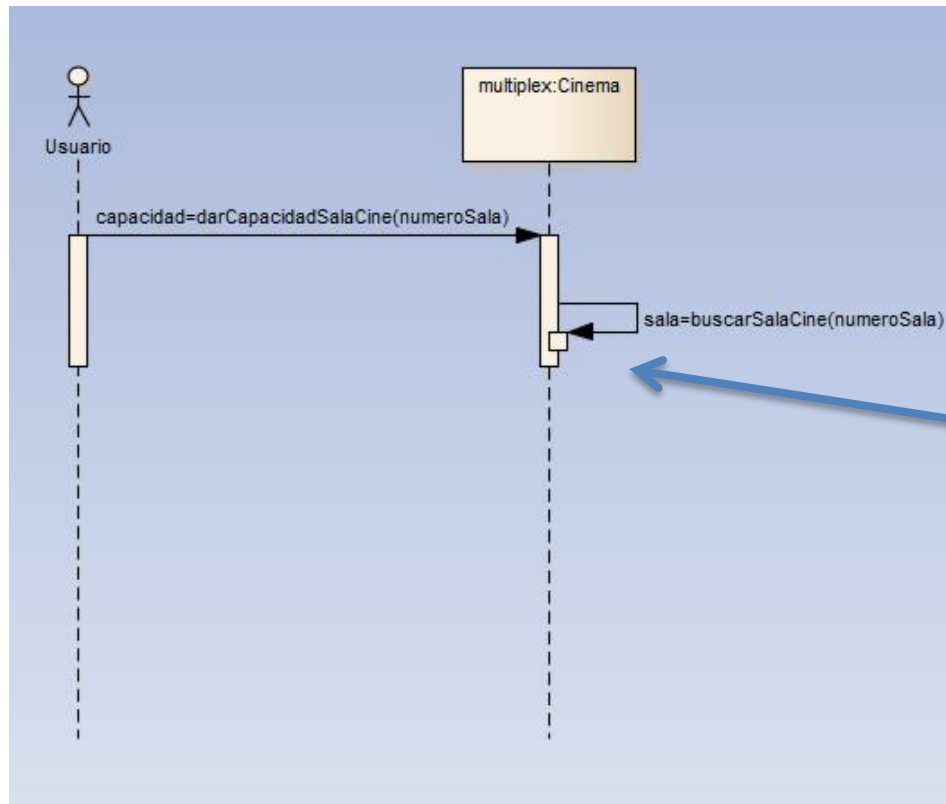
- Como ya lo dijimos informalmente los pasos por hacer son:
- Buscar el objeto sala que corresponde al número dado
- Pedirle a este objeto su capacidad

# Ejemplo



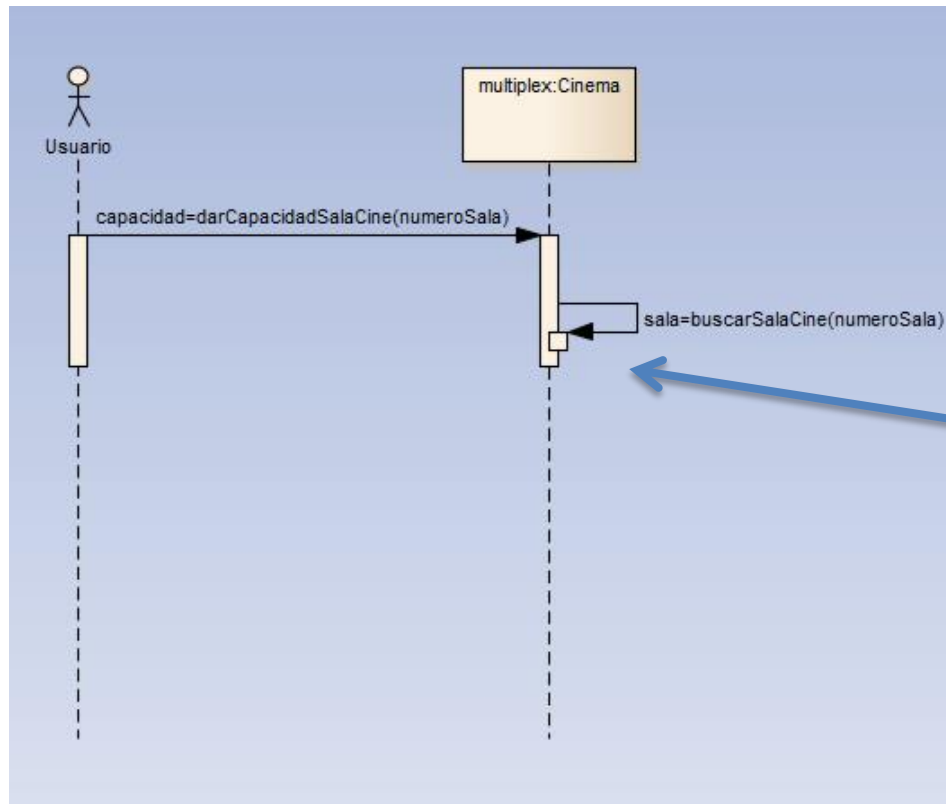
- Siguiendo la ejecución del método en el objeto multiplex:
  - 1. Invoca el método `buscarSalaCine(int)` sobre si mismo para obtener el objeto sala de cine correspondiente al número dado.

# Ejemplo



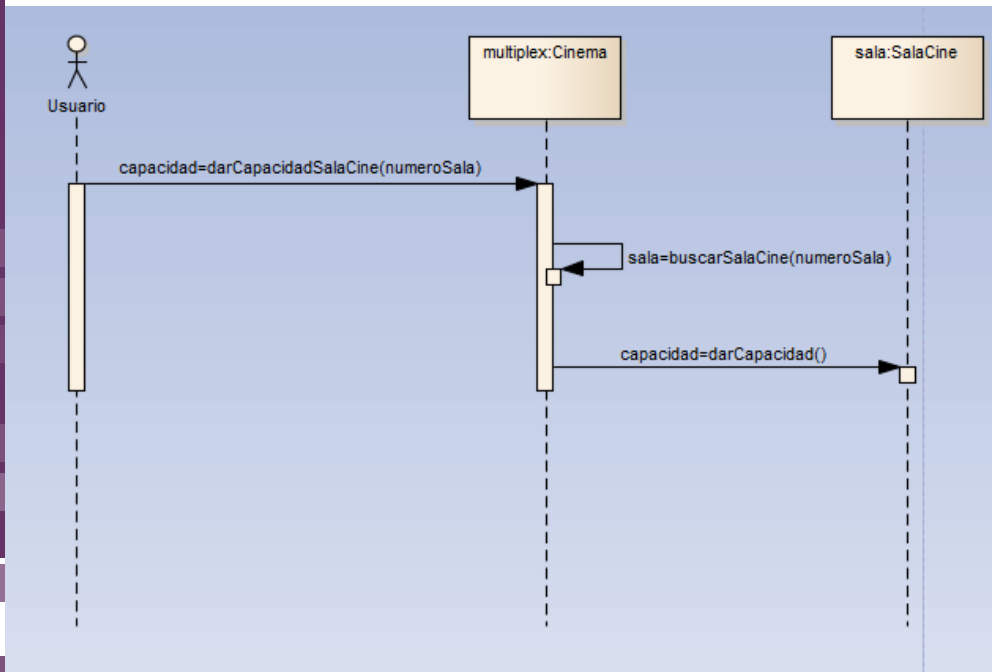
- Note que el rectángulo pequeño indica la ocurrencia de ejecución del método buscar.
- Este método devuelve un objeto llamado *sala* que es una instancia de la clase *SalaCine*

# Ejemplo



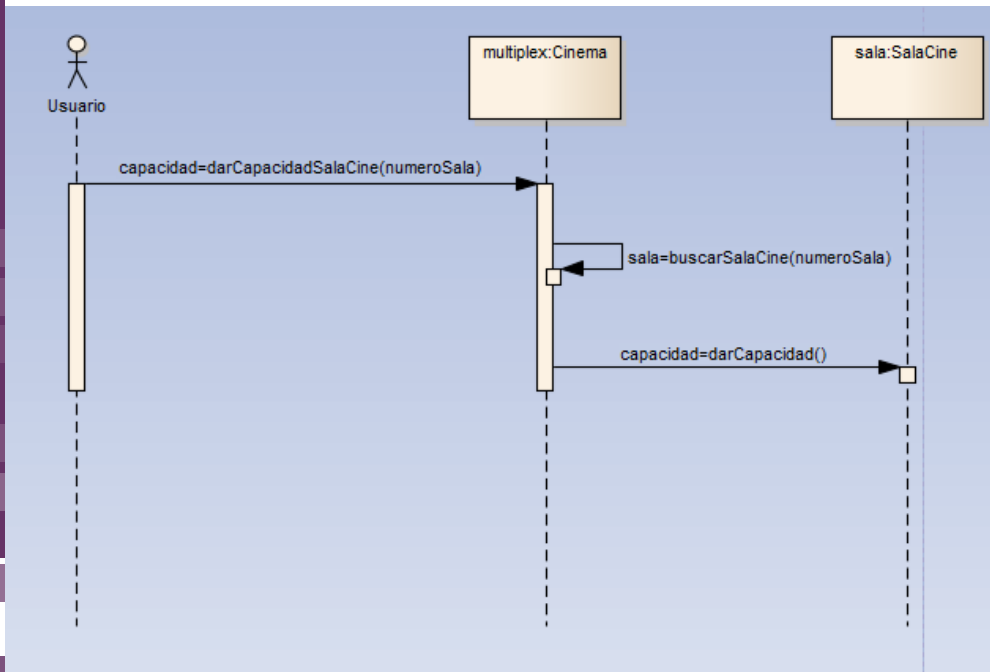
- Siguiendo la ejecución del método en el objeto multiplex:
  - 2. El objeto multiplex ahora puede interactuar con el objeto sala.

# Ejemplo



- Para esto le envía un mensaje `darCapacidad()` que debe retornar la capacidad de la sala.

# Ejemplo



- Como todos los llamados son síncronos, una vez que se termina la ejecución de `darCapacidad()` en el objeto `sala`, el control regresa al objeto `multiplex` quien termina la ejecución del método `darCapacidadSalaCine(int)` y el valor solicitado se retorna en la variable `capacidad`.

# Diagramas de secuencia


Interacciones básicas- Ejemplo  
Patrón Creador



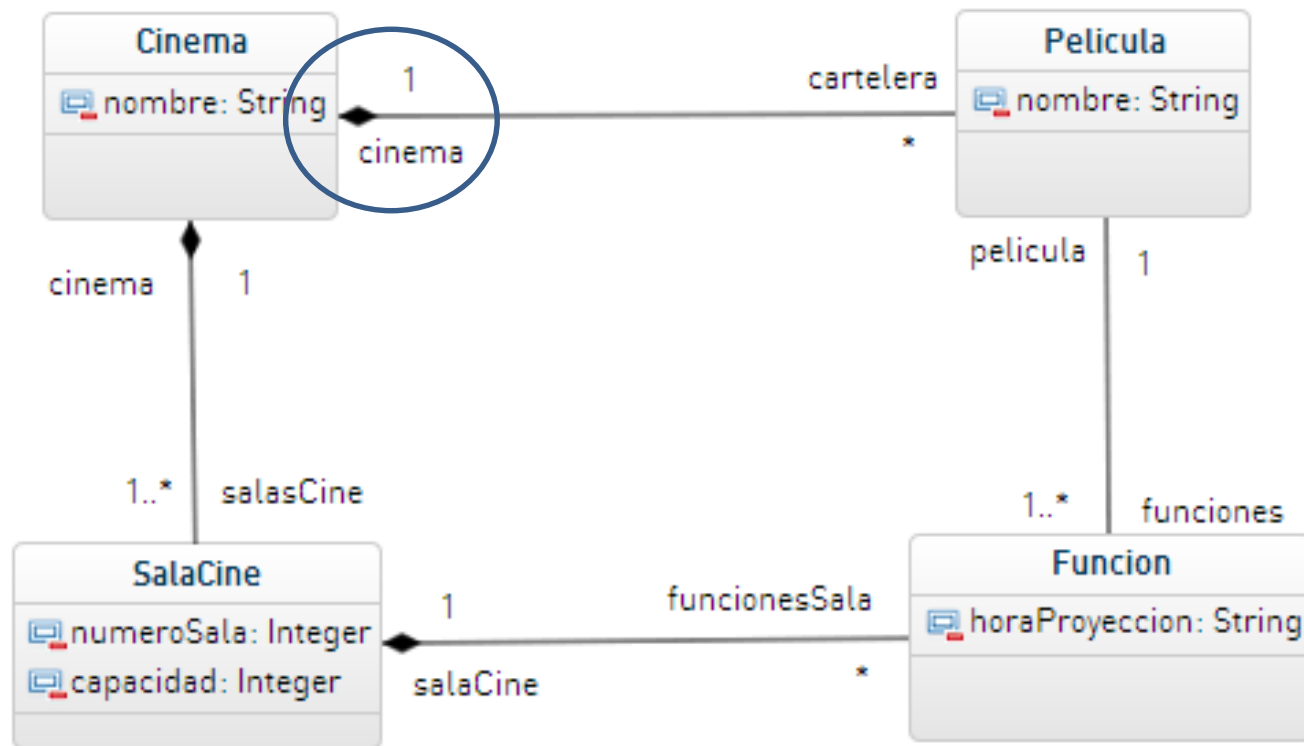
# Ejemplo

- Vamos a continuar con el ejemplo del Cinema modelando el comportamiento del caso de uso “Crear Película”

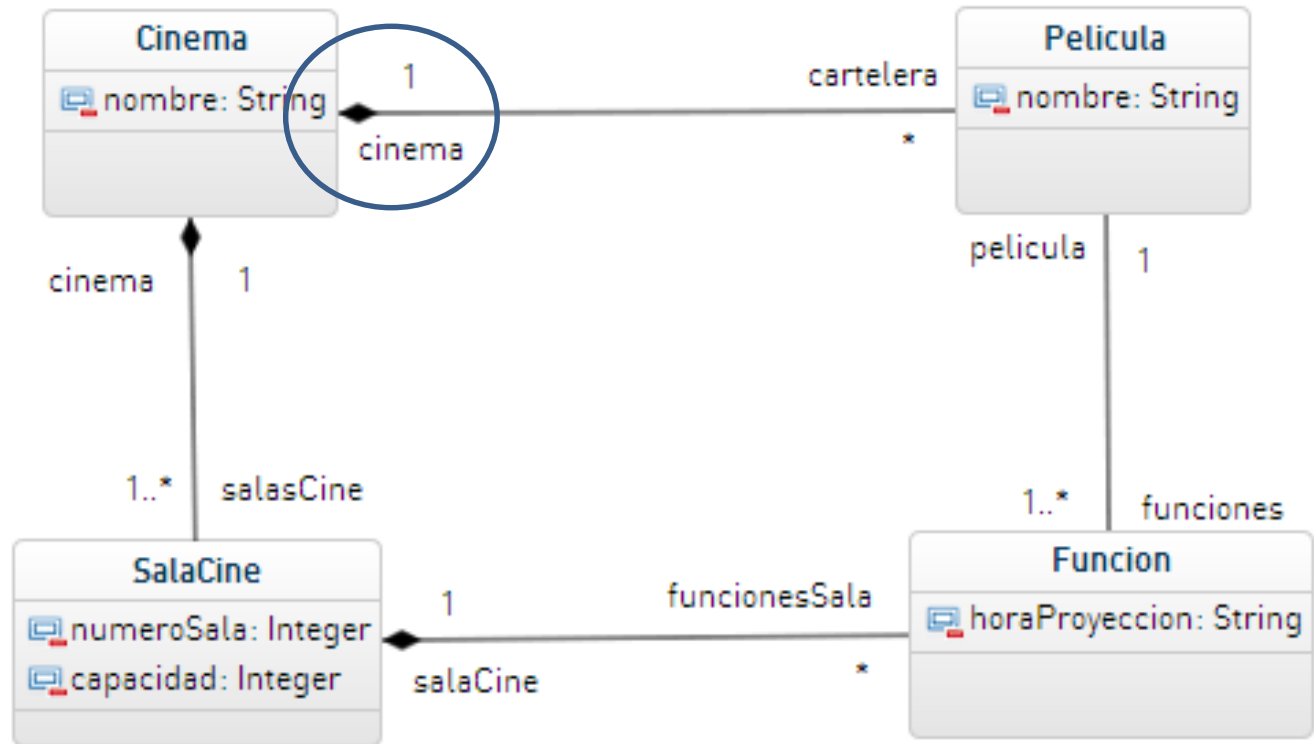


- 
- La primera pregunta que debemos hacernos es: de quién es la responsabilidad de crear una película?

Revisando el diagrama de clases podemos identificar que hay una relación de composición entre la clase Cinema y la clase Película.

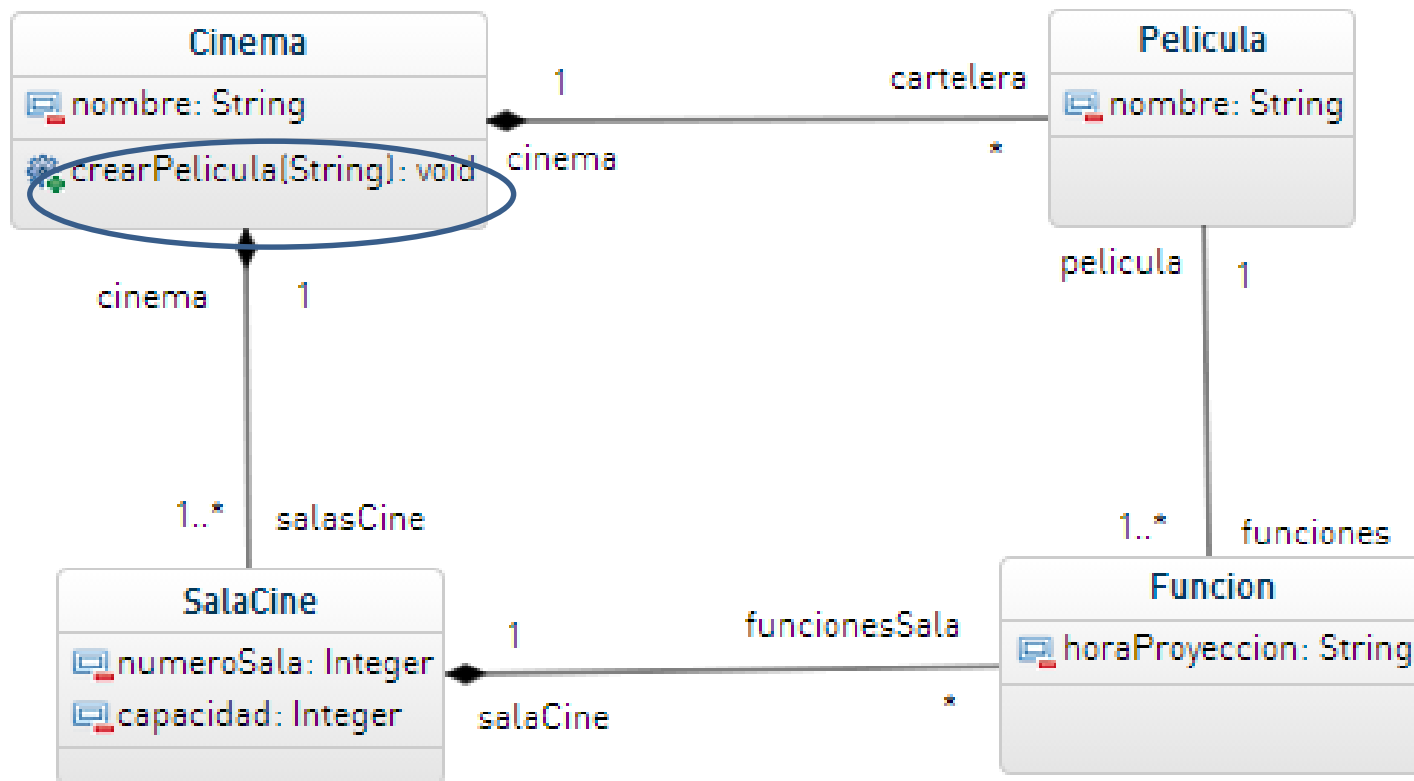


La clase Cinema es la dueña de la cartelera que contiene las películas que se exhiben en el cinema.



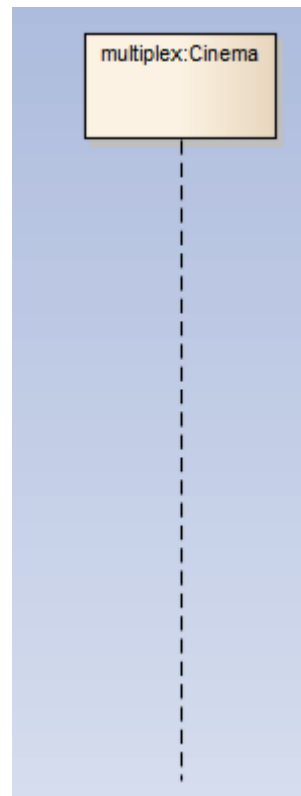
Siguiendo la guía del patrón Creador, el responsable de crear las películas es el cinema.

Para esto agregamos el método crearPelícula(nombre) en la clase Cinema.



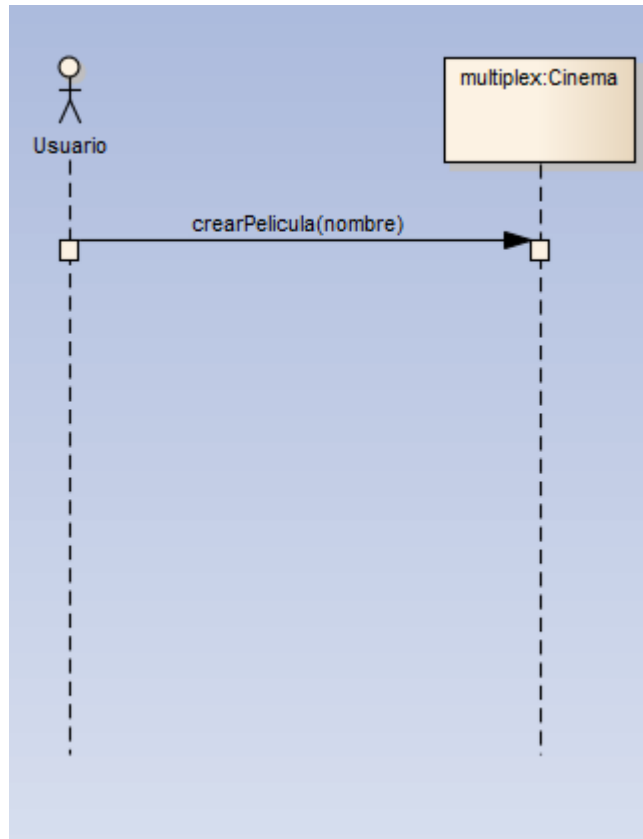
- Ahora vamos a modelar la interacción de los objetos para crear una película utilizando un diagrama de secuencia

# Ejemplo



- Vamos a suponer que ya hay en ejecución un objeto de la clase Cinema que va a recibir los mensajes de los actores (las peticiones) de los casos de uso.
- Para propósitos del ejemplo este objeto lo hemos llamado “multiplex”

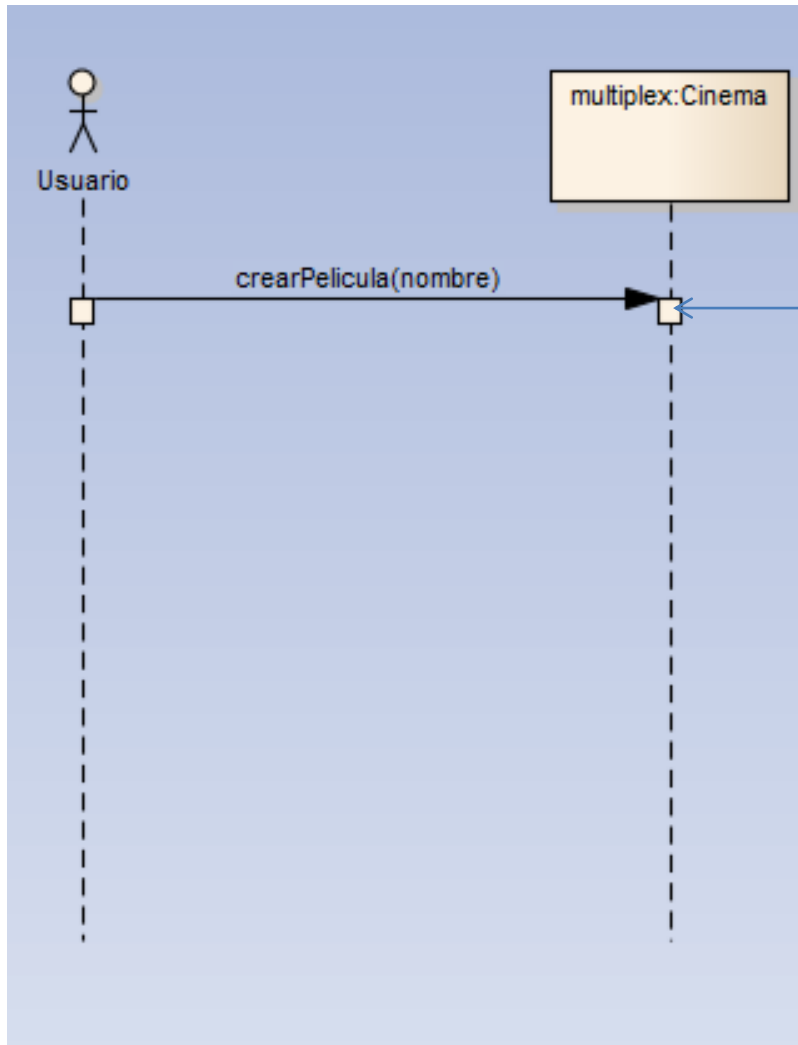
# Ejemplo



- Un actor externo llamado **usuario** envía el mensaje *crearPelicula(nombre)* al objeto **multiplex**.

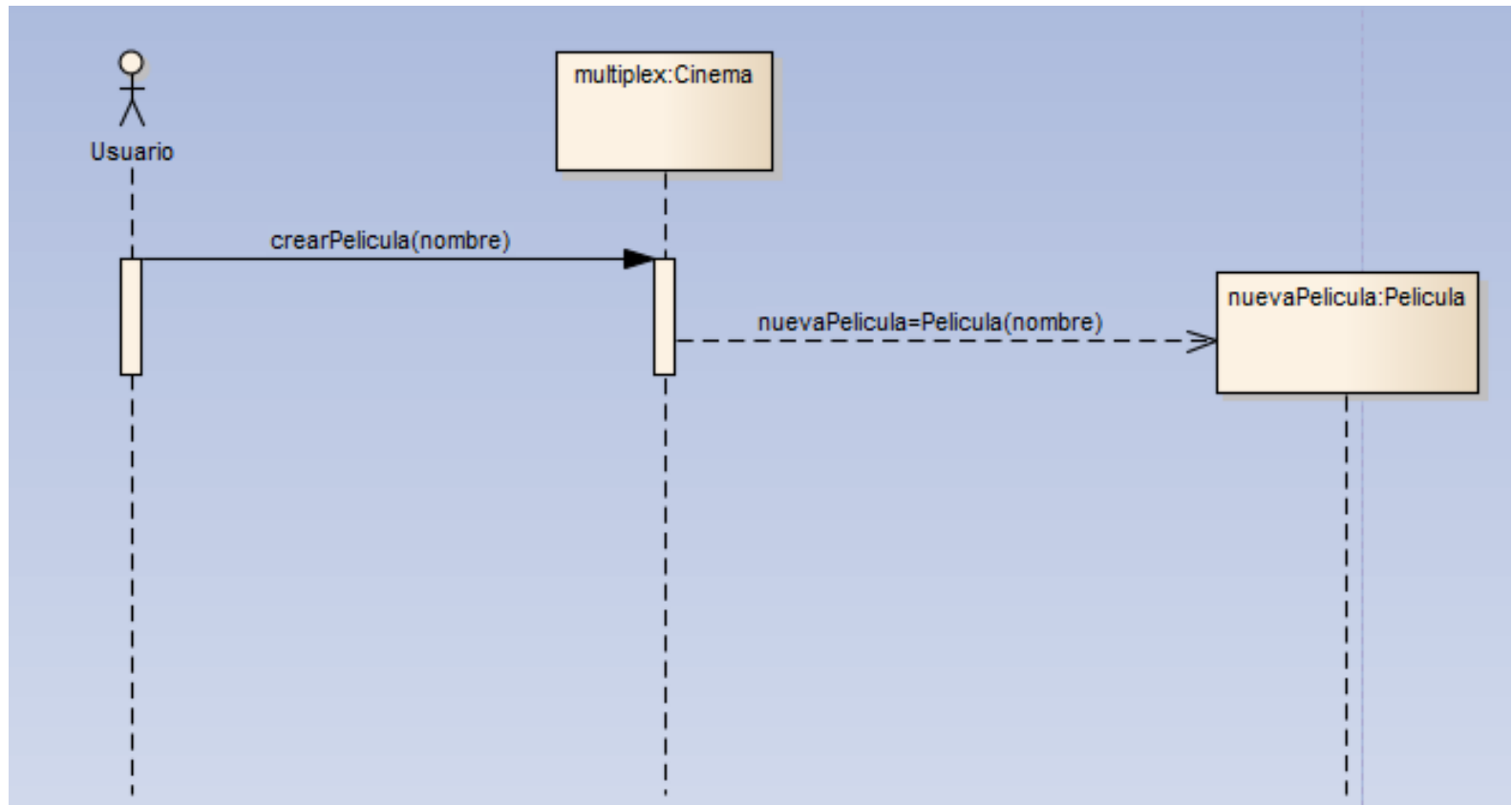


# Ejemplo



- Note la creación de la ocurrencia de ejecución en el objeto multiplex para el método crearPelícula

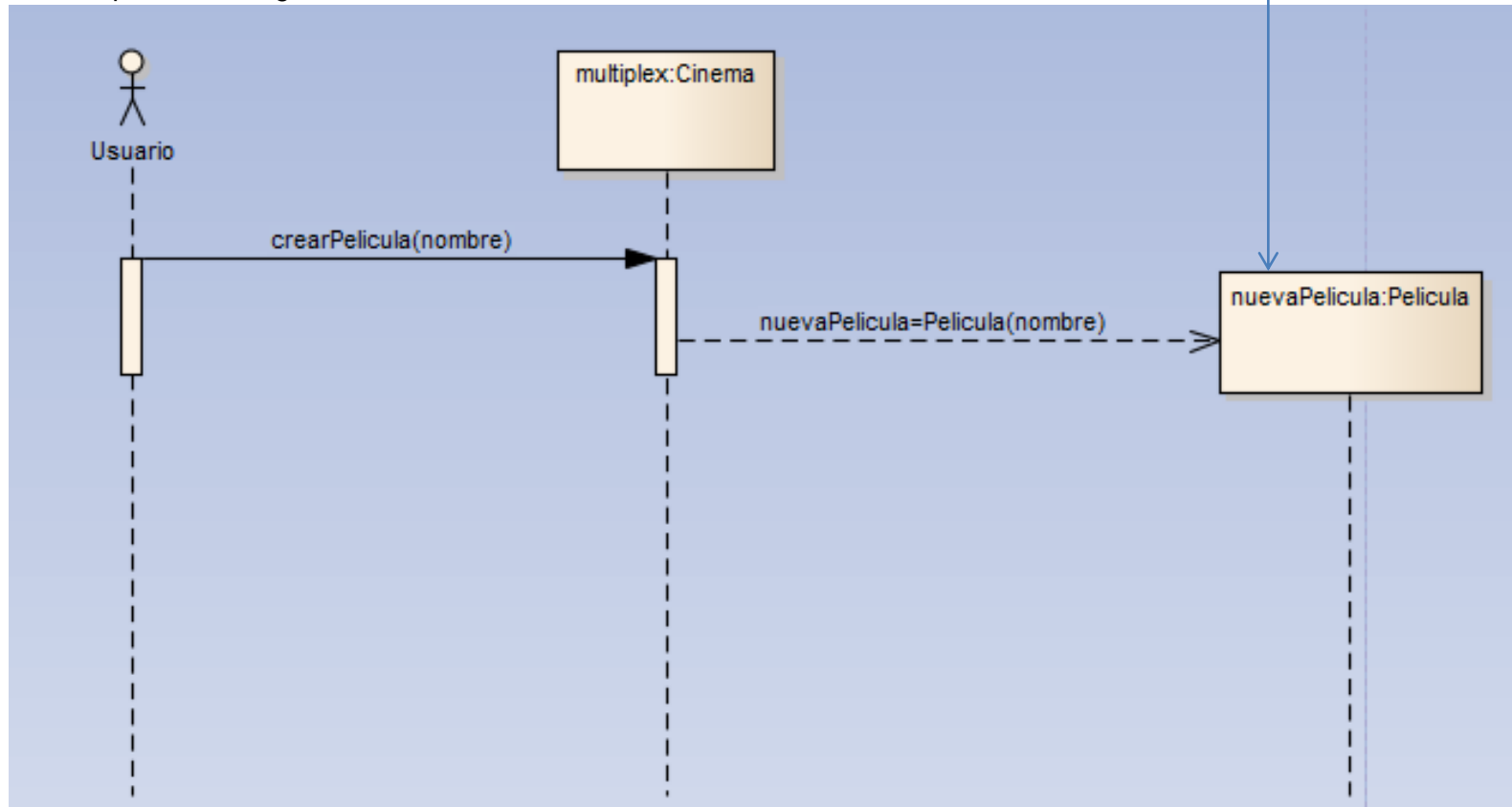
# Ejemplo



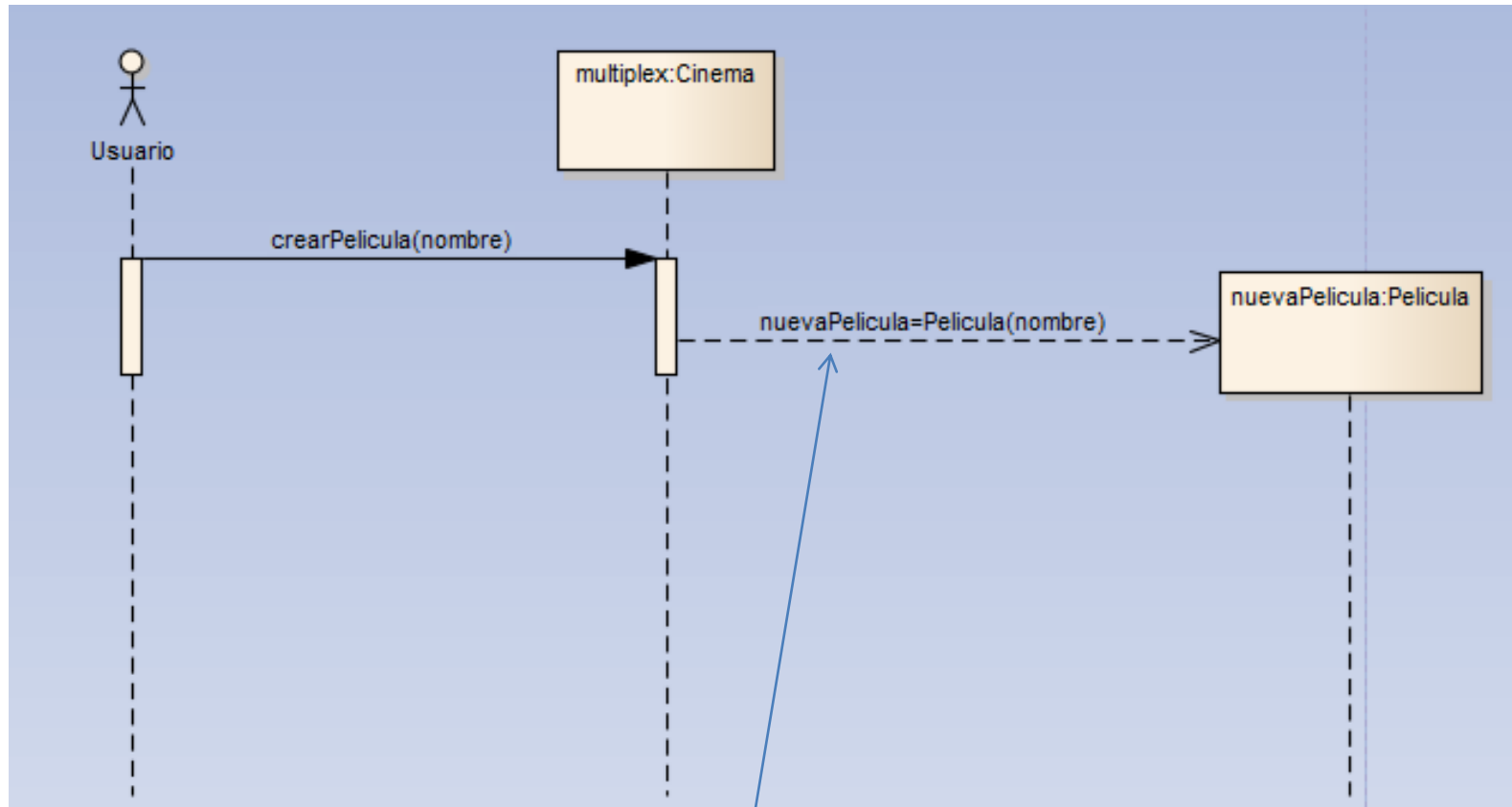
El objeto **multiplex** crea una nueva instancia de tipo Película.

# Ejemplo

Note que en la sintaxis del diagrama de secuencia, cuando se trata de la creación de un objeto este aparece en el punto donde se llamó el constructor y no en la parte superior del diagrama

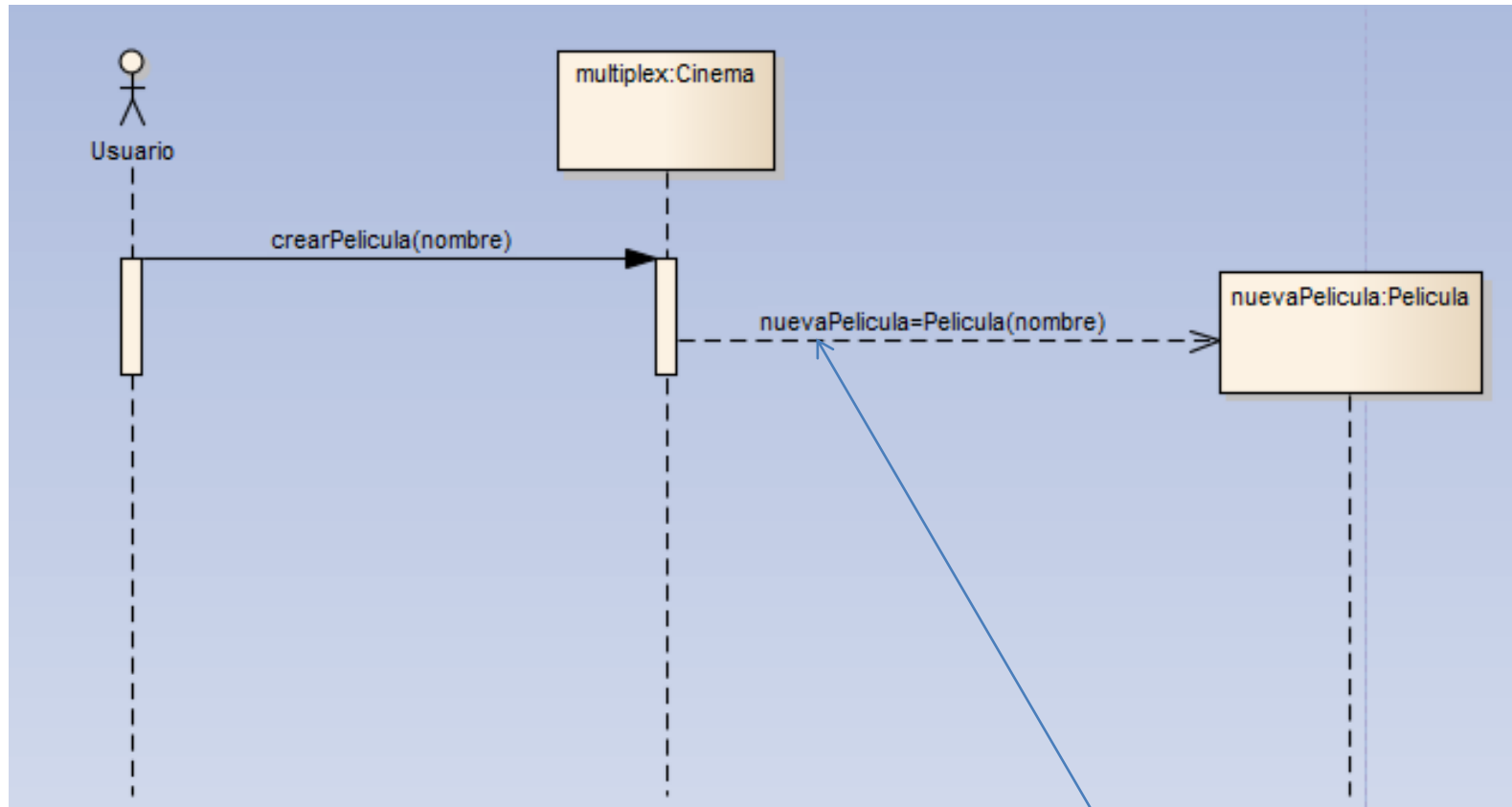


# Ejemplo



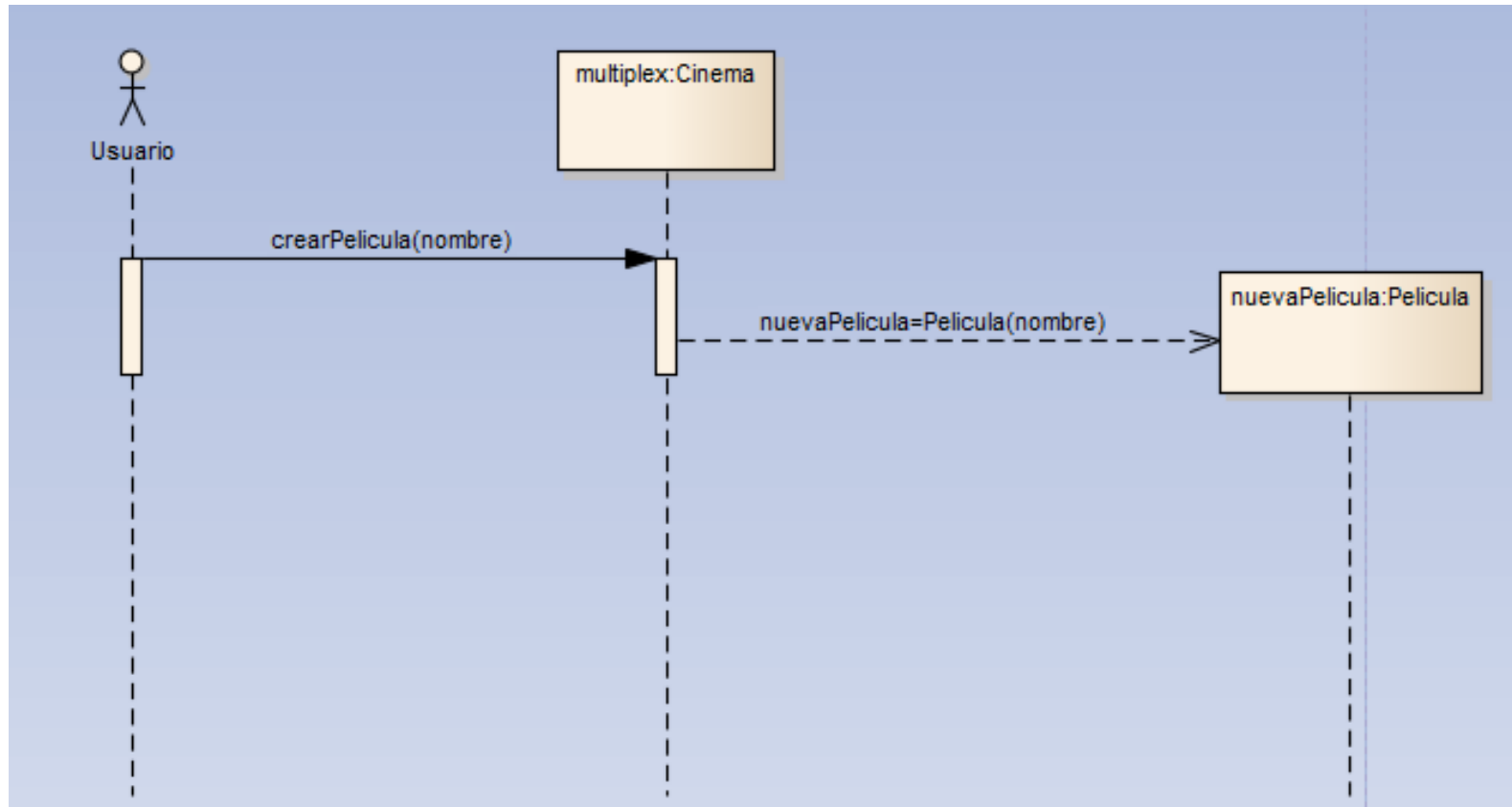
Note también que la línea interacción es una línea punteada.  
Hay una variable llamada nuevaPelicula que contendrá el nuevo objeto creado

# Ejemplo



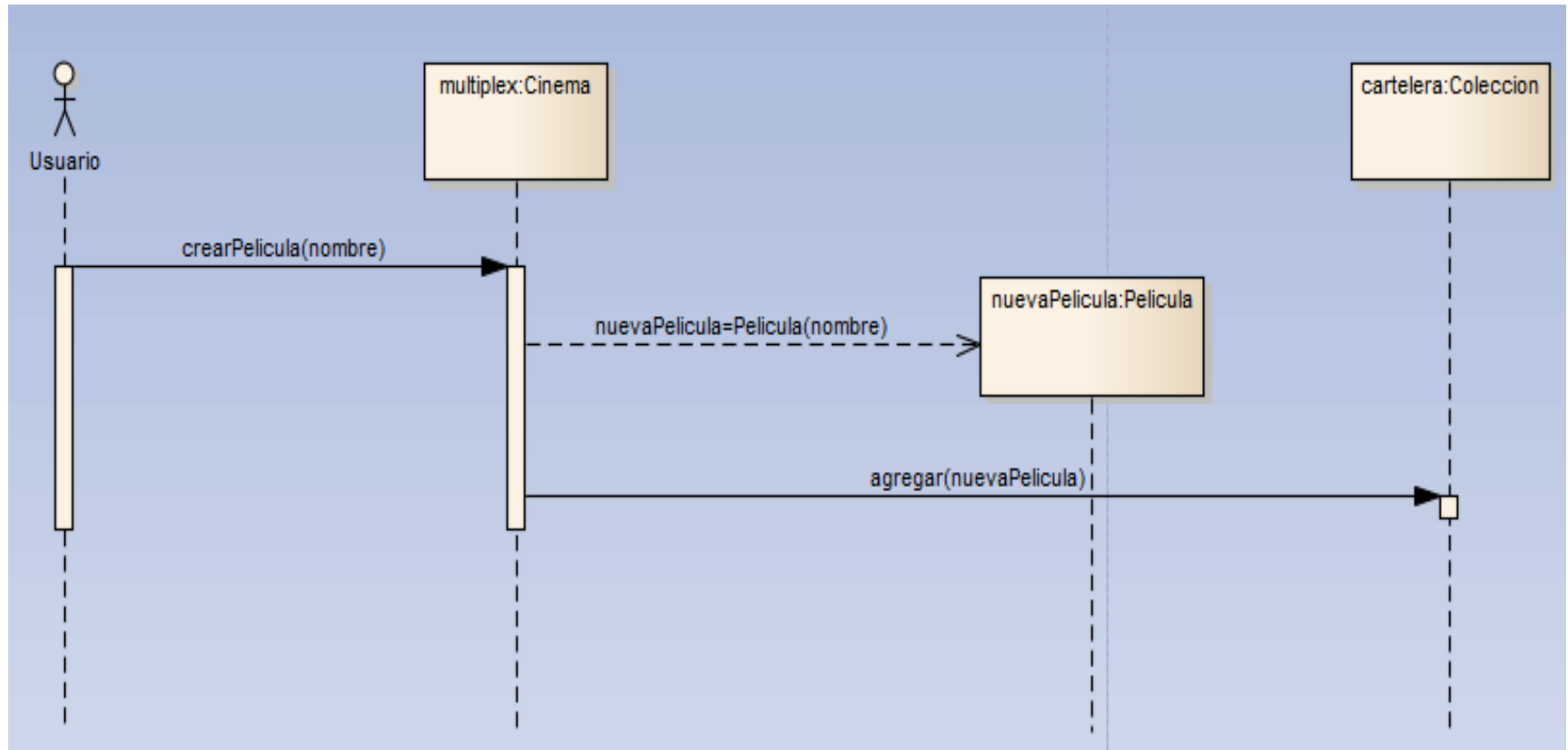
Hay una variable llamada nuevaPelicula que contendrá el nuevo objeto creado

# Ejemplo



Al constructor de la clase Película se encarga de que el nuevo objeto tenga en su atributo privado nombre el nombre que se pasó como argumento.

# Ejemplo



Finalmente, agregamos el objeto **nuevaPelícula** a la colección cartelera

# Ejemplo



¿Qué pasaría si la película  
que se va a crear ya existe?