Programación III

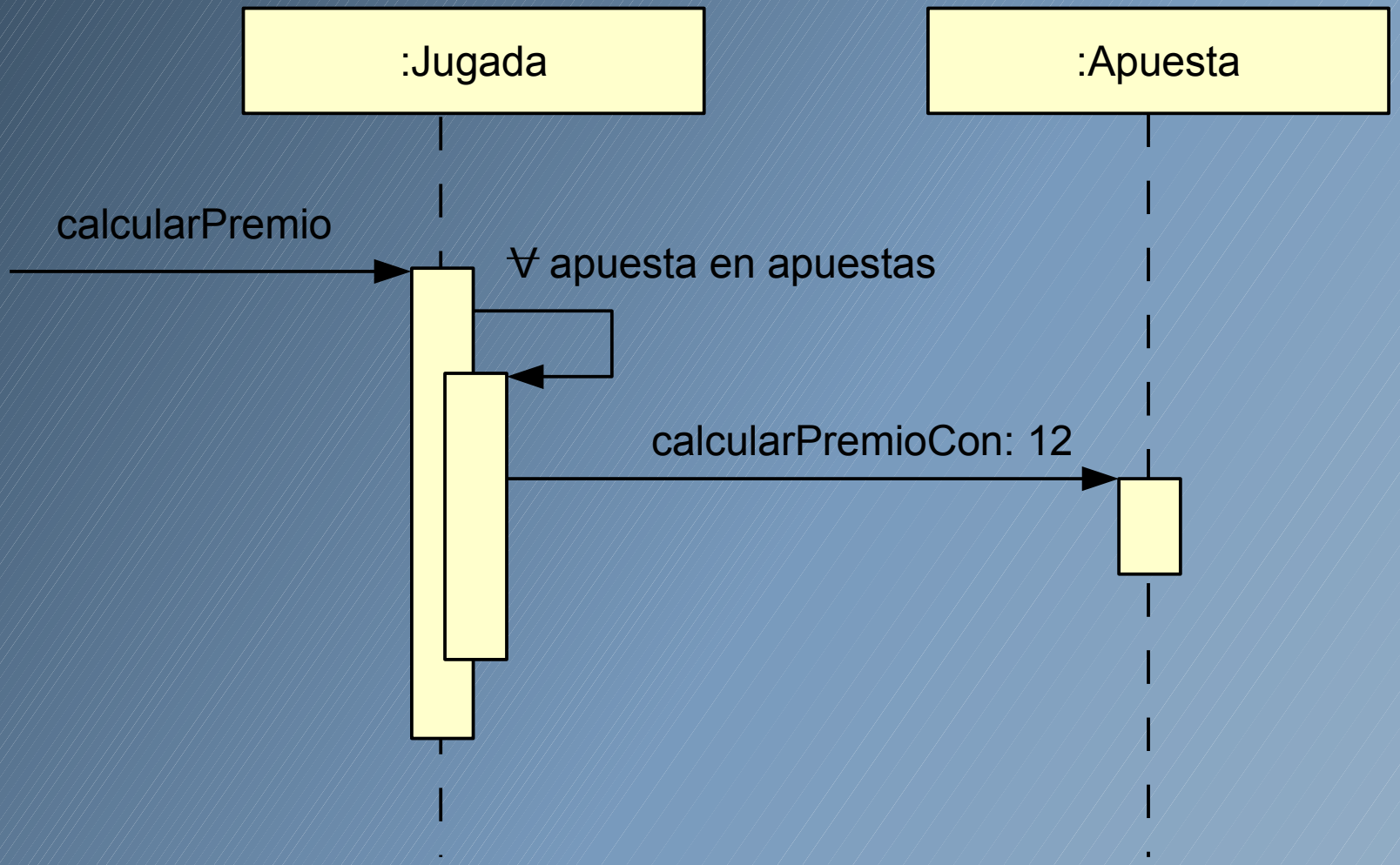
Clase XX

Andrés Fortier

Recordemos la ruleta

- Una jugada poseía un número ganador y una colección de apuestas.
- Teníamos una jerarquía de apuestas.
- ¿Cómo realizamos un diagrama de interacción para Jugada>>calcularPremio?

Recordemos la ruleta

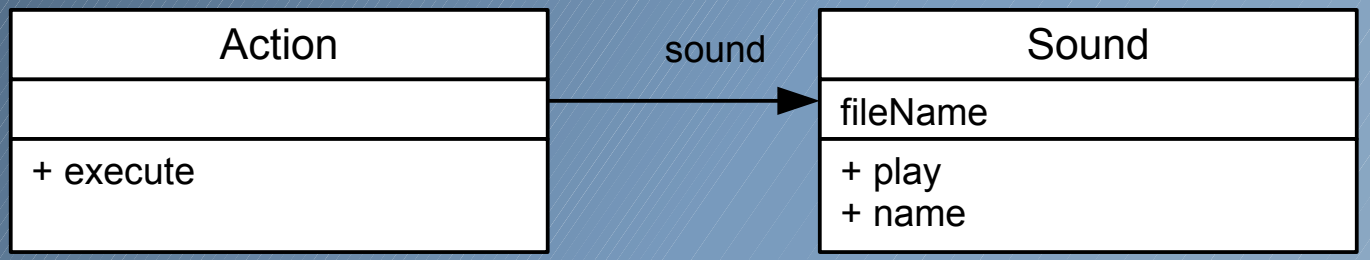


Patrones de Diseño

- Soluciones reutilizables a problemas recurrentes.
- No son diseños terminados o implementaciones concretas.
- Son descripciones que pueden utilizarse en diferentes dominios de problema.

Sonidos en el S.O.

- Supongamos que hay una clase Sound que entiende los mensajes:
  - #play
  - #name



play

“Interactúa con el SO para ejecutar el archivo de sonido ”

name

^fileName.

Sonidos en el S.O.

- Problema: ¿cómo manejamos el caso que no haya sonido?
- Al realizar la acción

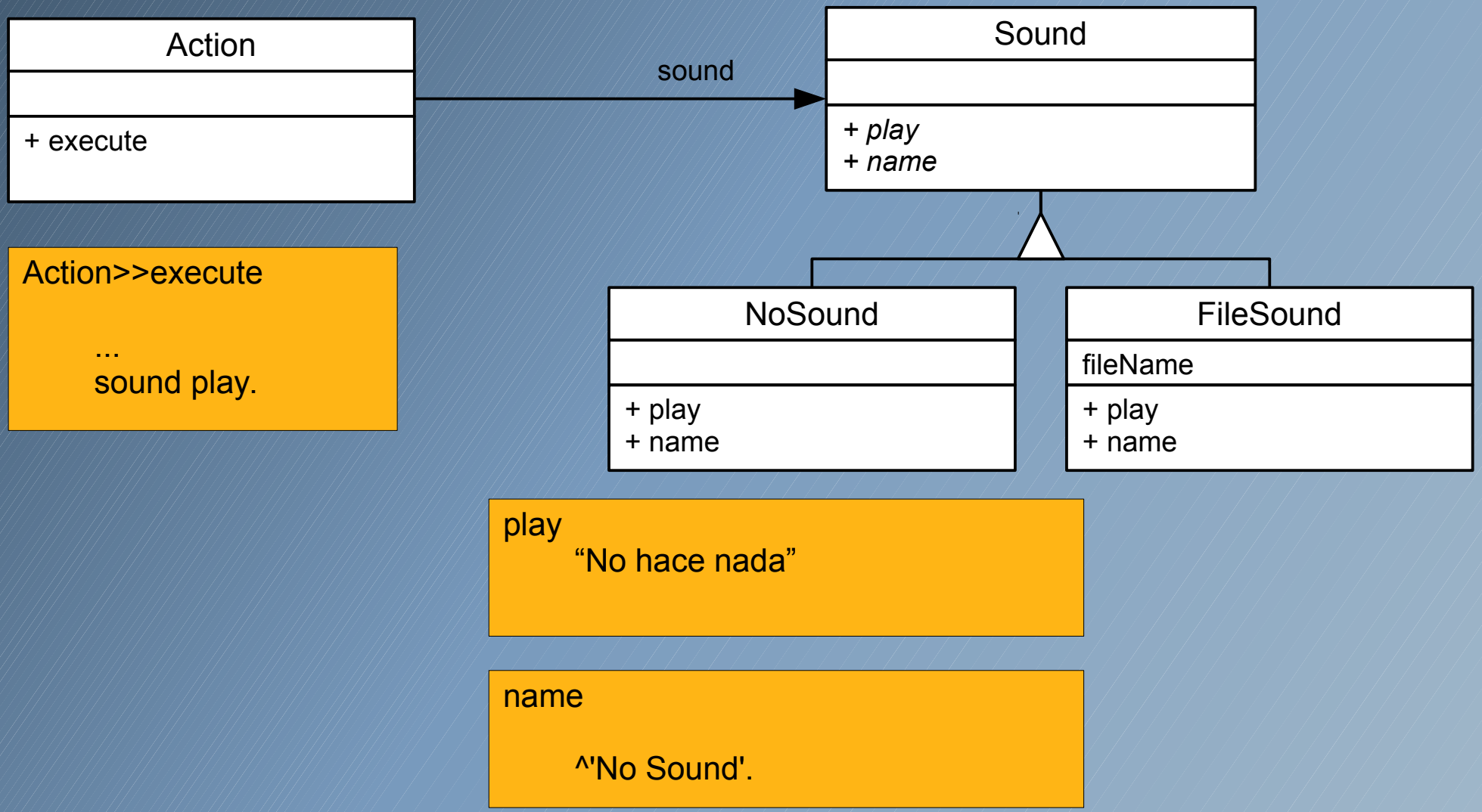
```
Action>>execute

	...
	sound ~= nil ifTrue: [sound play].
```

- Al mostrar la pantalla de configuración

```
	...
	action sound = nil
		ifTrue: [ 'No sound' ]
		ifFalse: [ action sound name ].
	...
```

Sonidos en el S.O.



Null object

- Motivación
  - En la mayoría de los lenguajes OO una referencia a un objeto puede ser nula (***nil***, ***null***, etc).
  - Enviar un mensaje a una referencia nula suele generar un error.
- Objetivo
  - Encapsular una referencia nula en un objeto que tenga el mismo protocolo que el objeto esperado.
  - Implementar los métodos en dicho objeto “haciendo nada” (esto depende del dominio de problema).

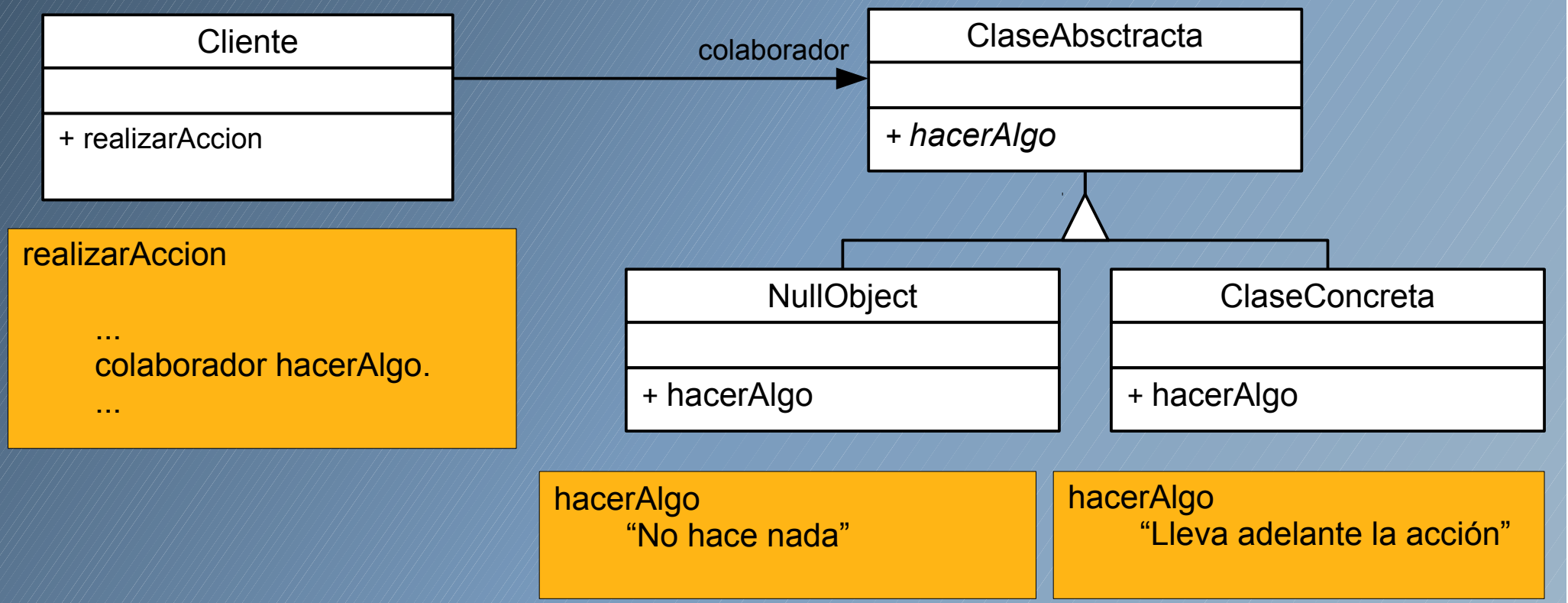
Null object

- Si
  - La presencia de un objeto es optativa.
  - Necesitamos enviar un mensaje a dicho objeto y debemos verificar su presencia.
  - El comportamiento en el caso que el objeto no esté presente es no hacer nada o utilizar un valor predefinido.
- Entonces
  - Implementar un objeto con el mismo protocolo que el esperado, cuya implementación sea no hacer nada o utilizar los valores predefinidos.
  - Usar una instancia de dicho objeto en lugar de una referencia nula.

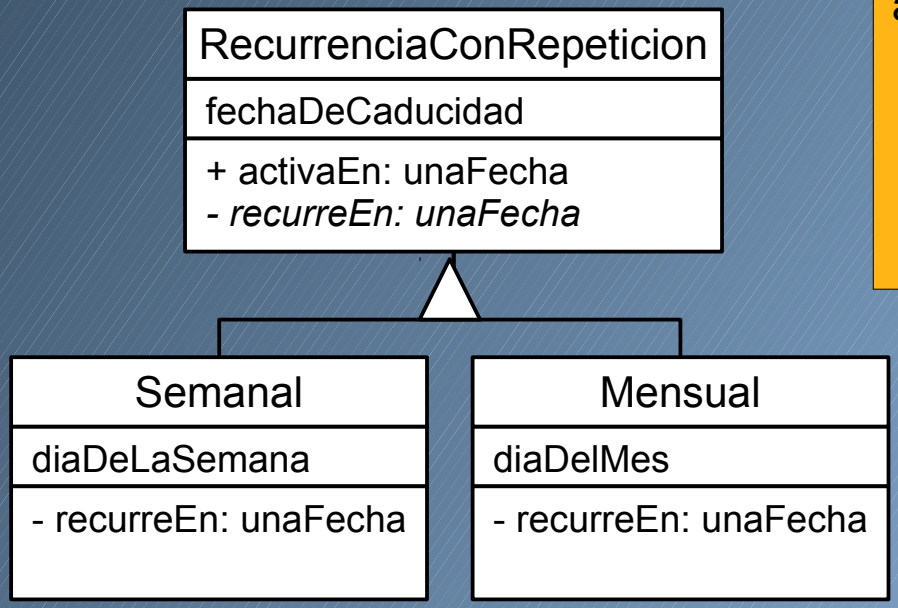
Null object

- Participantes
  - Cliente (quien envía el mensaje).
  - Clase abstracta (define el protocolo).
  - Clase concreta (implementa el comportamiento deseado).
  - Null object (implementa el comportamiento nulo).

Null object


Cliente
+ realizarAccion
colaborador
ClaseAbsctracta
+ hacerAlgo
realizarAccion
...
colaborador hacerAlgo.
...
NullObject
+ hacerAlgo
ClaseConcreta
+ hacerAlgo
hacerAlgo
“No hace nada”
hacerAlgo
“Lleva adelante la acción”

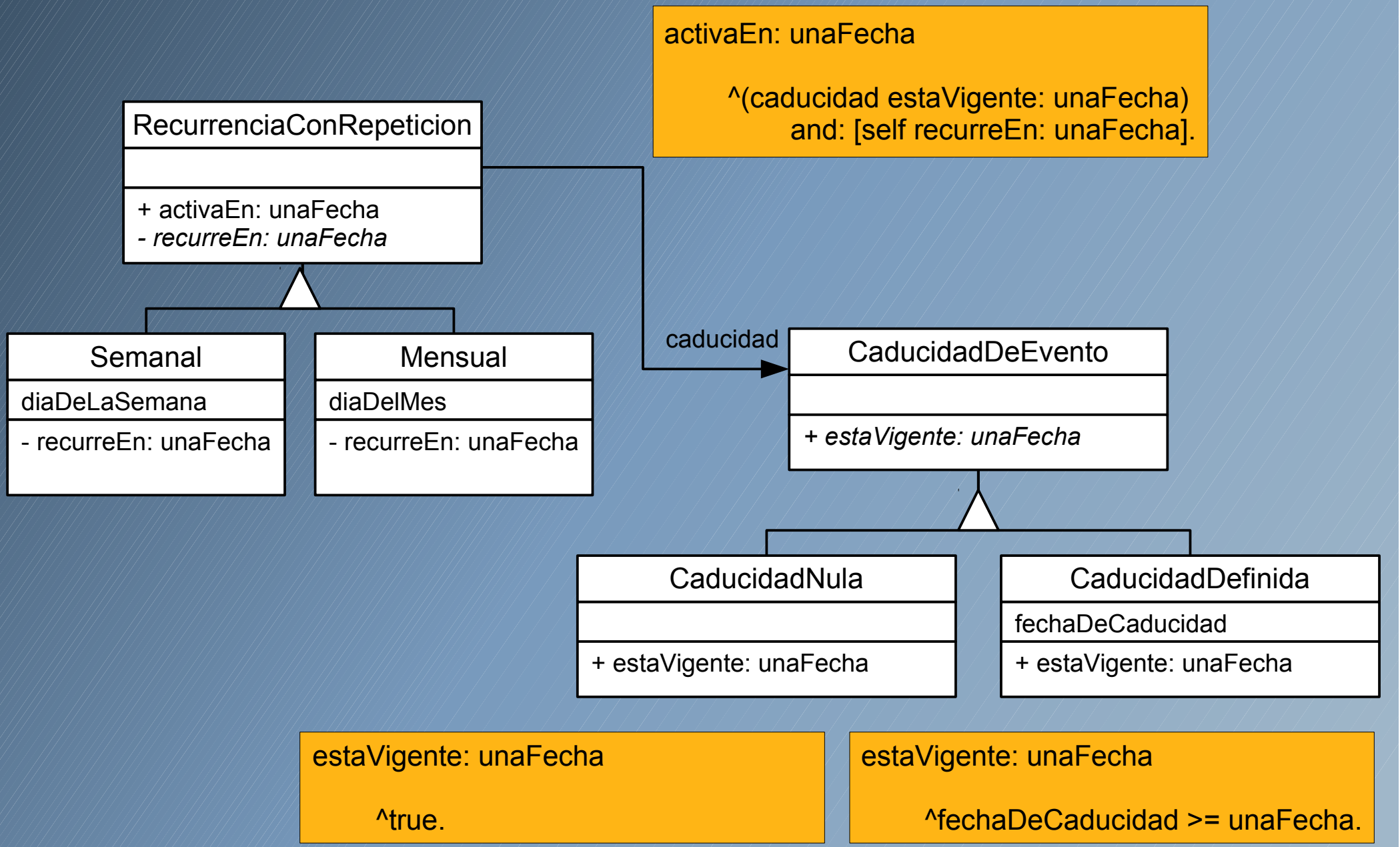
Recordemos el caso de la fecha optativa



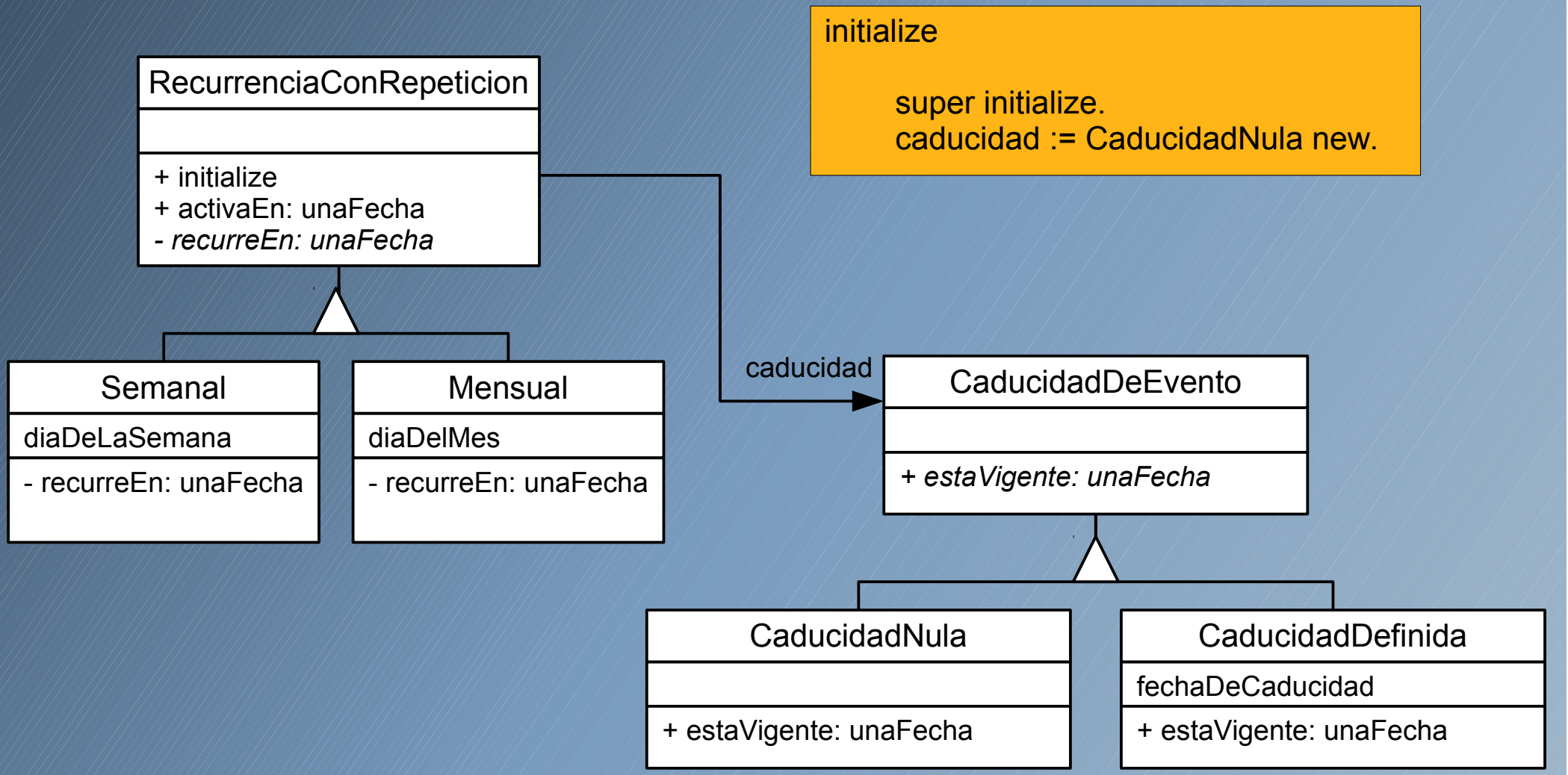

```
activaEn: unaFecha

  ^(fechaDeCaducidad ~= nil)
     ifTrue: [(fechaDeCaducidad >= unaFecha)
                    & (self recurreEn: unaFecha)]
     ifFalse: [self recurreEn: unaFecha].
```

Recordemos el caso de la fecha optativa



Caducidad optativa - inicialización



Caducidad optativa - asignación