3.3: Realización de diagramas de secuencia: capas software y patrones GRASP

A. Goñi, J. Ibáñez, J. Iturrioz, J.A. Vadillo

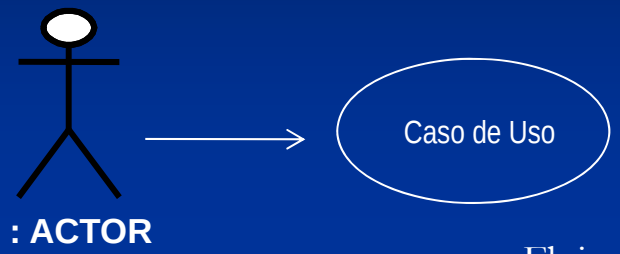
informatika fakultatea — facultad de informática

OCW 2013

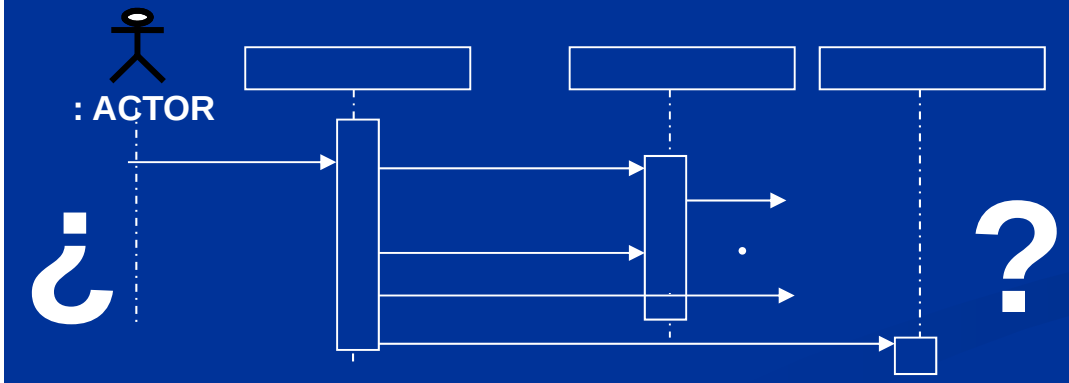
eman ta zabal zazu

Universidad del País Vasco — Euskal Herriko Unibertsitatea

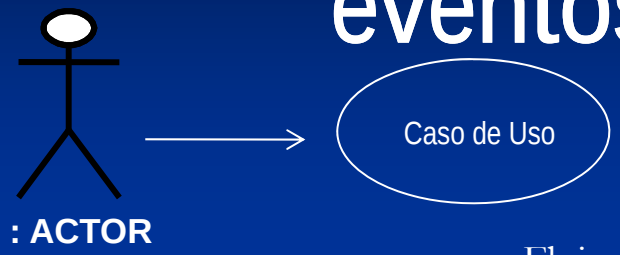
3.3.- ¿ Cómo realizar los diagramas de secuencia a partir de los flujos de eventos de casos de uso?



Flujo de Eventos
-Actor proporciona…
-El sistema …
Flujo de Eventos Alternativo
- Si ocurre X…

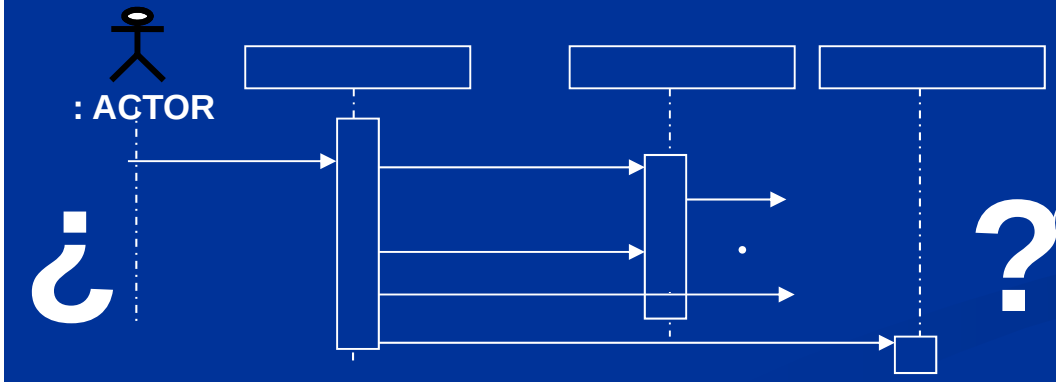


¿ Cómo realizar los diagramas de secuencia a partir de los flujos de eventos de casos de uso?



Flujo de Eventos
-Actor proporciona…
-El sistema …
Flujo de Eventos Alternativo
- Si ocurre X…

Usando PATRONES de DISEÑO

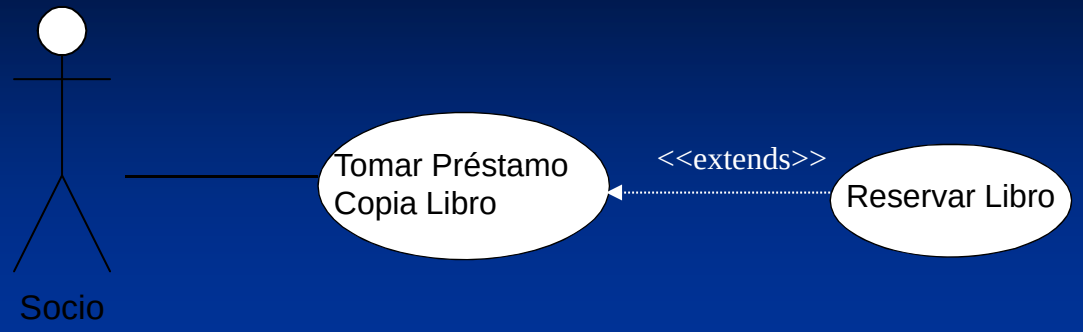


¿ ? Aplicando una ARQUITECTURA SOFTWARE en varios niveles/capas

Realización de diagramas de secuencia

- Diseñaremos un diagrama de secuencia a partir del flujo de eventos del caso de uso Tomar Préstamo Copia Libro (TPCL)
  - aplicando una arquitectura software en varios niveles
    - diferenciando los niveles/capas de presentación (interacción con el usuario), lógica del negocio o acceso a datos.
  - aplicando patrones GRASP cuando lo consideremos necesario

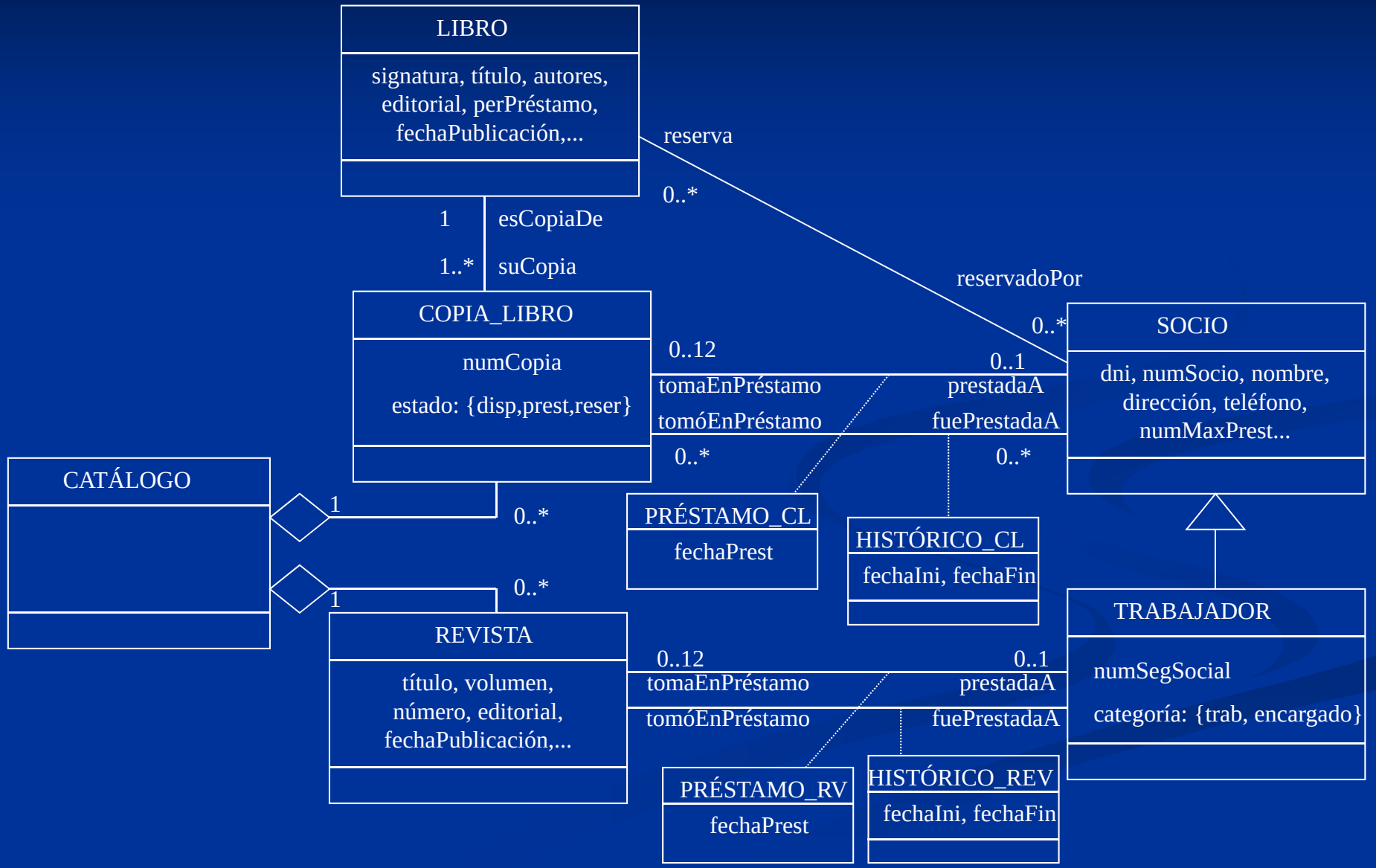
CU: Tomar Préstamo Copia Libro



Flujo de eventos:

- El socio proporciona su número de socio y la signatura del libro que quiere tomar en préstamo
- El sistema comprueba si existe alguna copia no prestada de dicho libro
- Si no hay copias disponibles:
  EXTENDS RESERVAR LIBRO
- Se comprueba que el socio no se pasa de su número máximo de libros en préstamo
- Se registra el nuevo préstamo con la fecha actual

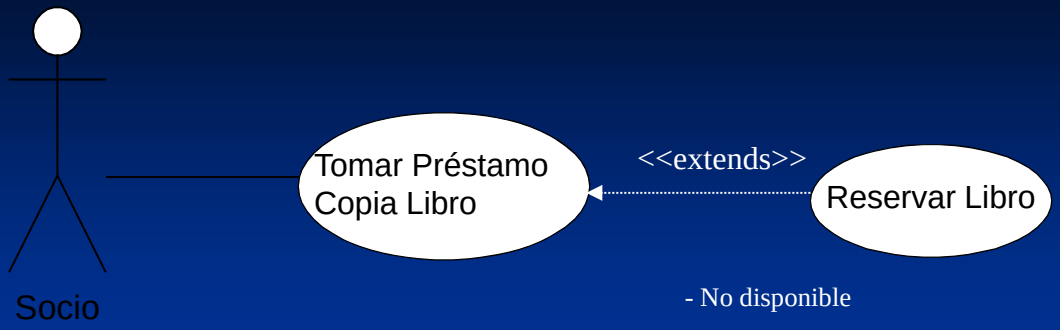
Modelo del Dominio



3.3.1- Capas lógicas en el software: Presentación, lógica del negocio y acceso a datos

- En los flujos de eventos hay acciones de diferente naturaleza o relacionadas con el/la:
  - 3.3.1.1.- Presentación
    - interfaces de usuario y la interacción con el mismo
  - 3.3.1.2.- Lógica del negocio
    - resolver los problemas del negocio
    - implementar las reglas propias del negocio
  - 3.3.1.3.- Acceso a datos
    - Recuperar, insertar, actualizar y borrar objetos del dominio
      - Se dará persistencia a los objetos del modelo del dominio utilizando una BDOO

EJ: CU: Tomar Préstamo Copia Libro



PRESENTACIÓN
LÓGICA DEL NEGOCIO
ACCESO A DATOS

Flujo de eventos:

- El socio proporciona su número de socio y la signatura del libro que quiere tomar en préstamo
- El sistema comprueba si existe alguna copia no prestada de dicho libro.
- Si no hay copias disponibles/ copias disponibles EXTENDS RESERVAR LIBRO
- Se comprueba que el socio no se pasa de su número máximo de libros en préstamo
- Se registra el nuevo préstamo con la fecha actual

3.3.2- Patrones de responsabilidad GRASP

- 3.3.2.1.- Patrón CONTROLADOR
- 3.3.2.2.- Patrón EXPERTO
- 3.3.2.3.- Patrón CREADOR
- 3.3.2.4.- Patrón BAJO ACOPLAMIENTO
- 3.3.2.5.- Patrón ALTA COHESIÓN

¿Qué es un patrón?

- Un PATRÓN es una SOLUCIÓN para un PROBLEMA que se repite
- Idea propuesta en 1979 por Christopher Alexander, profesor de arquitectura
  - *"Cada patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez."*

¿Qué es un patrón de diseño?

- La idea de PATRÓN aplicada al DESARROLLO SOFTWARE
- Un PATRÓN de DISEÑO es una SOLUCIÓN a un PROBLEMA de DISEÑO
- Un patrón debe ser
  - EFECTIVO: ha servido para resolver problemas similares
  - REUTILIZABLE: aplicable a diferentes problemas de diseño

Patrones GRASP

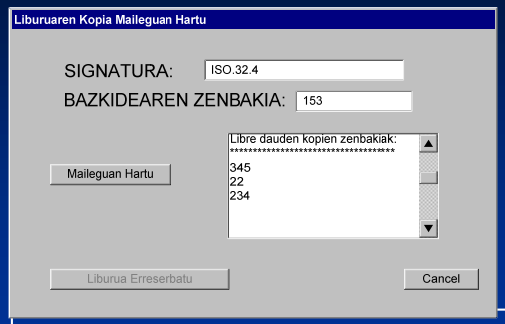
- GRASP: **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns
- Los patrones GRASP describen los principios fundamentales del diseño de objetos y sus responsabilidades
- GRASP: significa “entender”, “comprender”
- El nombre GRASP sugiere la importancia de COMPRENDER (GRASP) los principios fundamentales para diseñar software orientado a objetos de manera correcta
  - **Los 5 primeros patrones GRASP son: EXPERTO, CREADOR, ALTA COHESIÓN, BAJO ACOPLAMIENTO y CONTROLADOR**

3.3.1.1.- ¿Cómo se representa la interfaz de usuario en un diagrama de secuencia?

- Se define un objeto de una clase INTERFAZ (también llamada clase FRONTERA) que está especializada en comunicarse con el ACTOR (sabe aceptar eventos de entrada del usuario y mostrar resultados de salida)
- Ese objeto representará a una interfaz de usuario gráfico habitualmente (interfaz AWT/Swing, página HTML, JSP, ASP.NET,…)

Los objetos de clases INTERFAZ/FRONTERA son los que implementan la capa/nivel de PRESENTACIÓN

Diagrama de secuencia TPCL (1)



: ACTOR

obj1: IU_CU_TPCL

tiempo

1: dar código libro

2: dar código socio

3: pulsar botón

4: ¿A qué MÉTODO de qué OBJETO debe llamar?

Invocación a la LÓGICA DEL NEGOCIO

3.3.1.2.- Invocación a Lógica del Negocio
3.3.2.1.- Patrón CONTROLADOR

Nombre: CONTROLADOR

PROBLEMA: ¿A quién se le asigna la responsabilidad de recibir o manejar eventos de entrada al sistema?

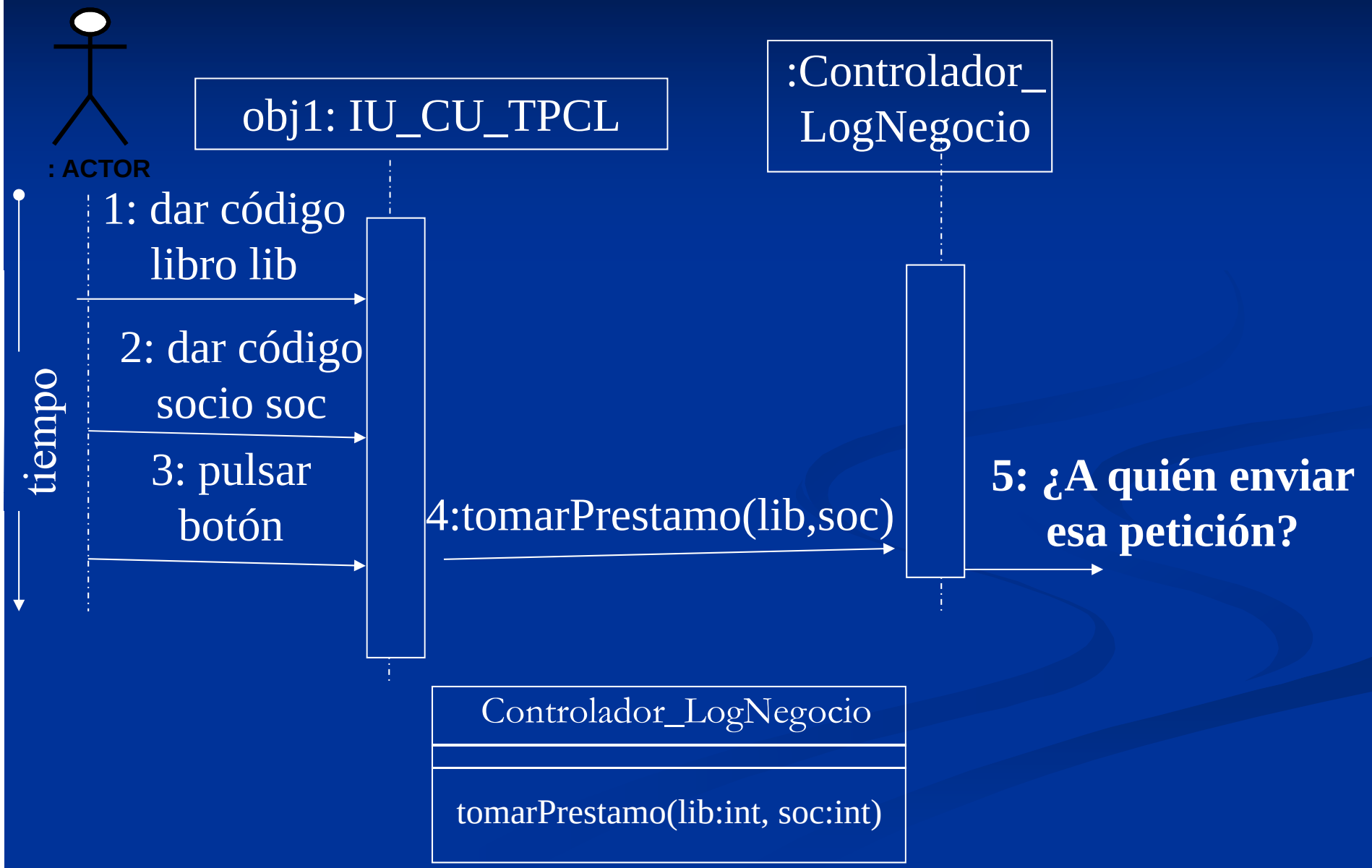
[Un evento de entrada al sistema es un evento generado por un actor externo, que se asocia con una operación del sistema.]

SOLUCIÓN: A un objeto (clase) que representa al sistema global, dispositivo o subsistema .

Es un único OBJETO para todo el sistema donde se colocan TODAS LAS OPERACIONES DEL SISTEMA. También se suele llamar objeto "FACADE" (o FACHADA).

El CONTROLADOR es el que ofrecerá las operaciones de la LÓGICA DEL NEGOCIO

Diagrama de secuencia TPCL (2)



3.3.1.3.- Acceso al nivel de datos / objetos del modelo del dominio

- ¿A quién le pide el controlador la ejecución de esa operación, en quién delega? ¿Qué objeto/s puede/n ejecutar esa operación?
- ¿Qué necesita hacer el método?
  - ENCONTRAR ALGÚN DATO/INFORMACIÓN (Patrón EXPERTO)
  - CREAR NUEVOS OBJETOS (Patrón CREADOR)

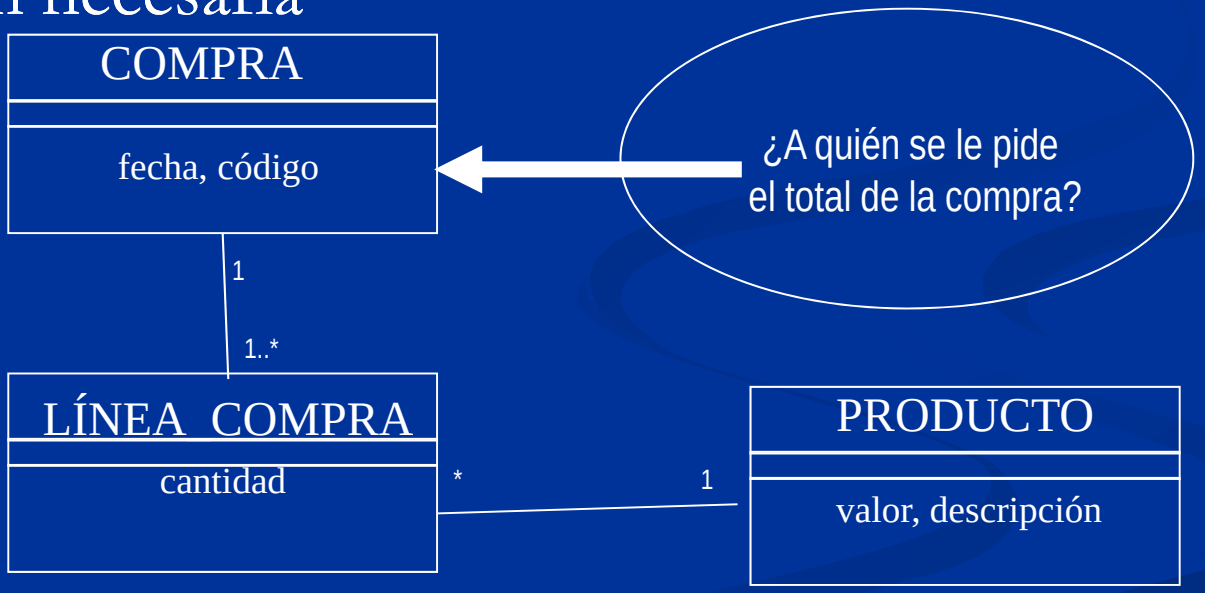
El CONTROLADOR de la LÓGICA del NEGOCIO puede necesitar acceder al nivel o capa de DATOS

3.3.2.2.- Patrón EXPERTO

Nombre: EXPERTO

PROBLEMA: ¿A quién se le pide que busque un determinado dato o genere información?

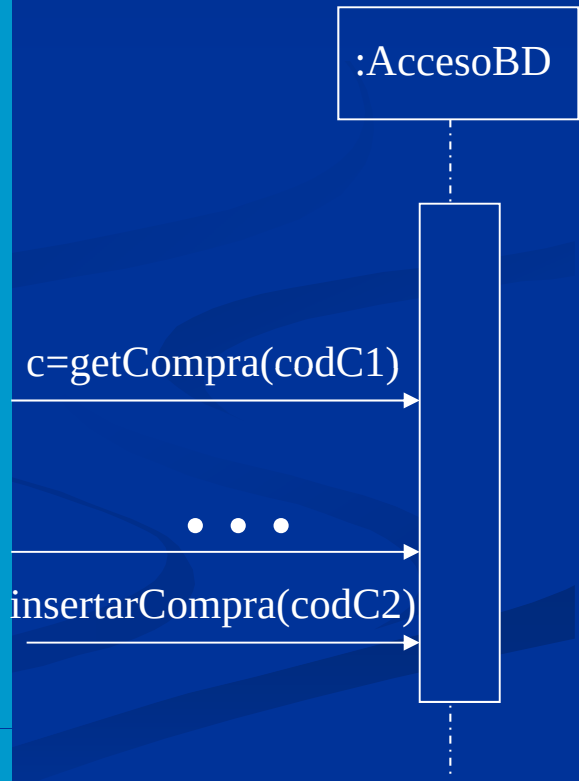
SOLUCIÓN: Al objeto (clase) que cuenta con los datos o información necesaria



NOTA IMPORTANTE: si es un objeto del dominio, hay que asegurarse de que se encuentra cargado en la memoria principal. En ese caso, ese objeto del dominio sería el EXPERTO.

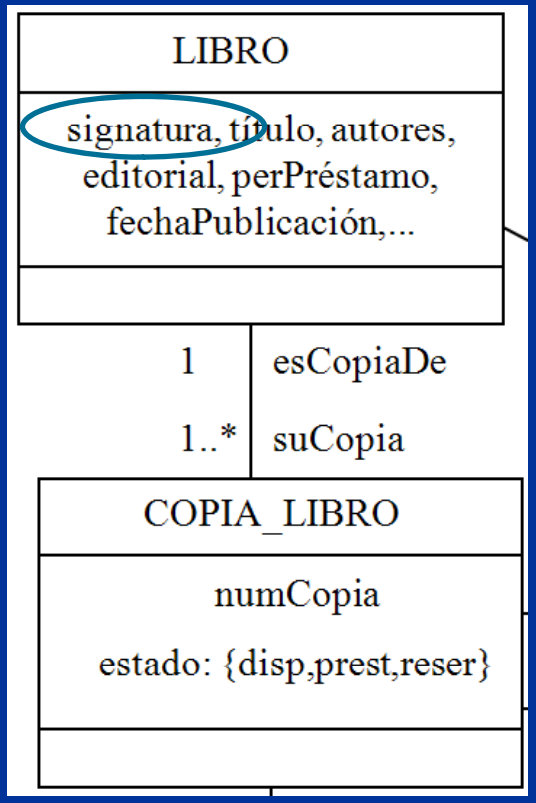
Acceso al nivel de datos para cargar los objetos del dominio

Los objetos del dominio que queramos invocar deben estar cargados en la memoria principal. Lo haremos usando un objeto de una clase (AccesoBD) que será: 1) **experto** en acceder a la BDOO, o bien 2) **controlador** de la BDOO porque también se encargará de las operaciones de inserción, borrado y modificación, además de las operaciones de recuperación.



Ejemplo de aplicación de patrón EXPERTO

El sistema comprueba si hay alguna copia libre de ese libro



¿Qué objeto puede saber si hay alguna copia libre del libro con la signatura proporcionada?

El OBJETO de la clase LIBRO

¡Pero no está cargado en la memoria principal! ¡No sabemos su referencia!

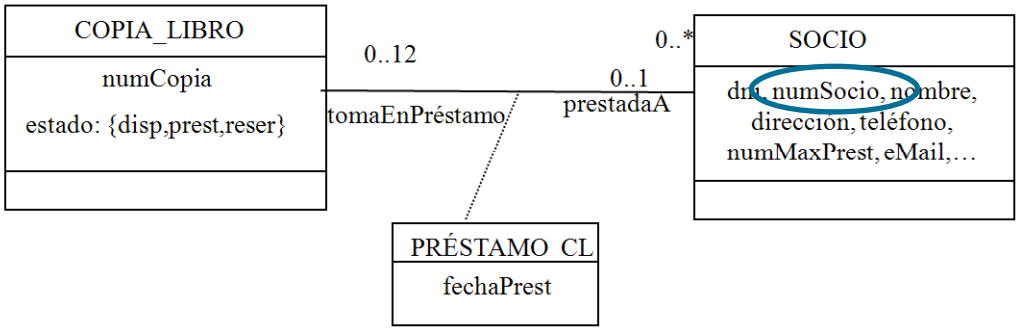
Ejemplo de aplicación de patrón EXPERTO

Se comprueba que el socio no se pasa de su número máximo de libros en préstamo

¿Qué objeto puede saber si el socio con el número de socio proporcionado no se pasa de su número máximo de préstamos?

El OBJETO de la clase SOCIO

¡Pero no está cargado en la memoria principal! ¡No sabemos su referencia!



Ejemplo de aplicación de patrón EXPERTO/CONTROLADOR

¿Qué objeto puede obtener el objeto LIBRO y el objeto SOCIO, conocidas la signatura y el número de socio?

El OBJETO que gestiona la BASE de DATOS (AccesoBD)

Diagrama de secuencia TPCL (3)

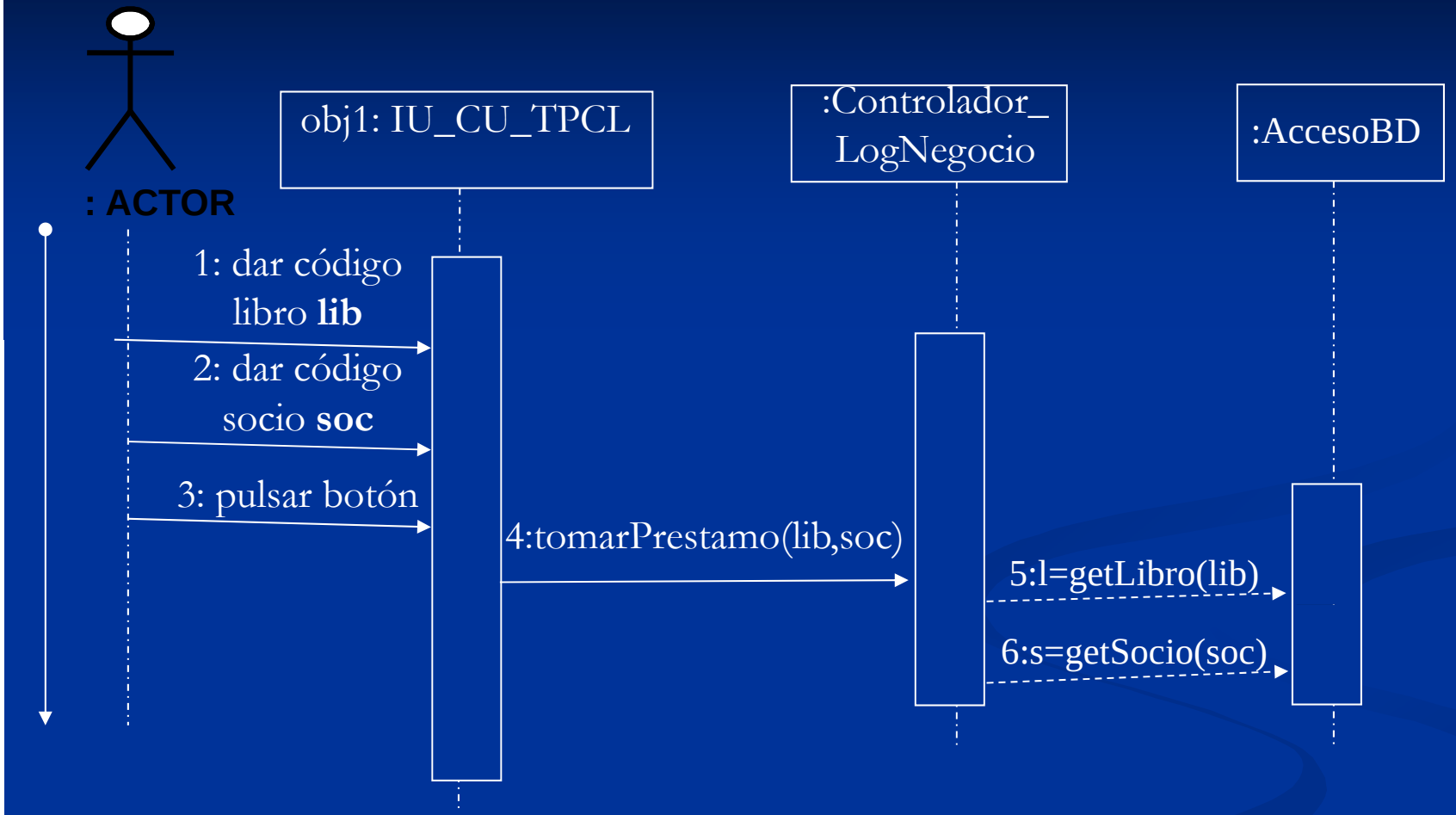
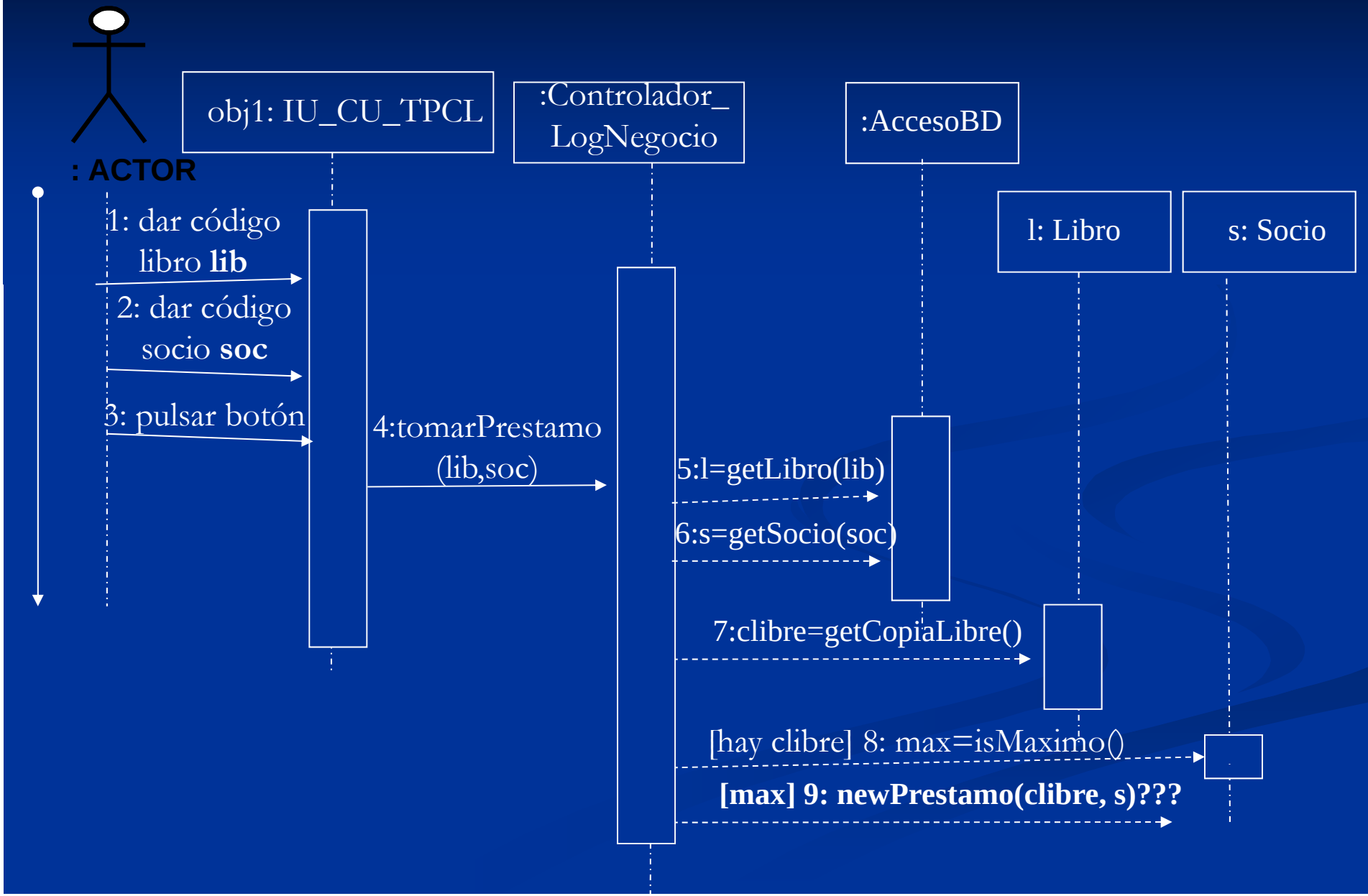


Diagrama de secuencia TPCL (4)



3.3.2.3.- Patrón CREADOR

Nombre: CREADOR

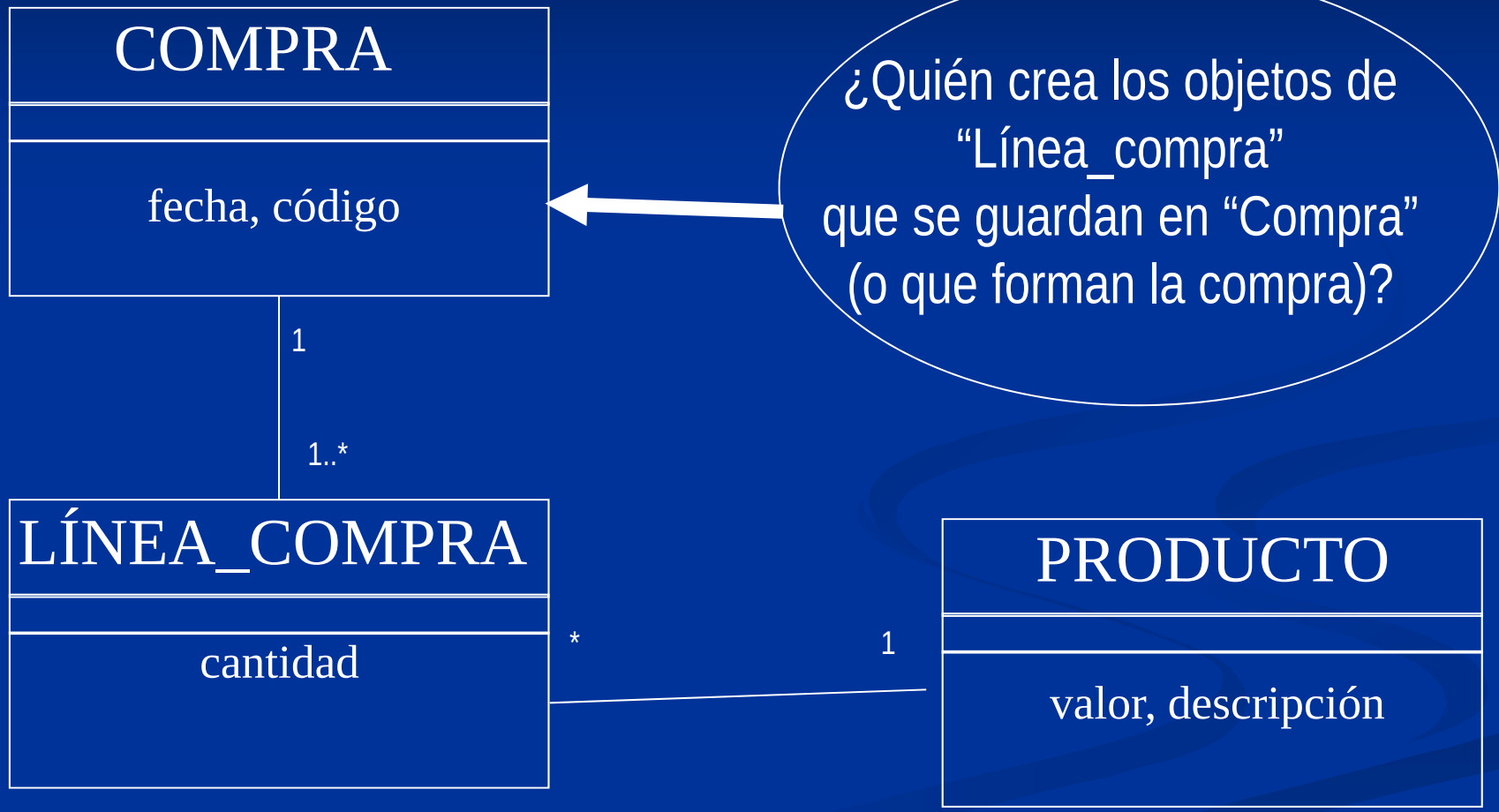
PROBLEMA: ¿Quién debe crear los objetos de una clase A?

SOLUCIÓN: Esa responsabilidad se le añadirá a la clase B si se cumple alguna de estas condiciones:

- La clase B guarda los objetos de la clase A
- La clase B está formada por objetos de A (AGREGACIÓN/COMPOSICIÓN)
- Cuando hay que crear un objeto de A, B tiene todos los datos de inicialización necesarios

  (B es un EXPERTO en la creación de A)

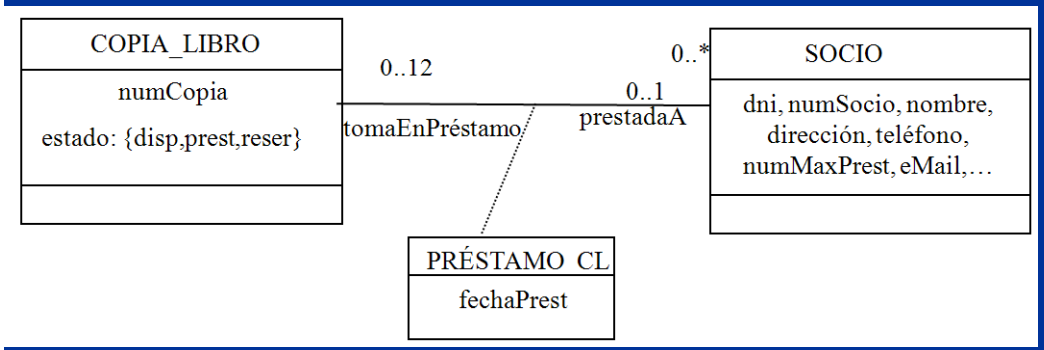
3.3.2.3.- Patrón CREADOR



Ejemplo de aplicación de patrón CREADOR

Se registra el nuevo préstamo con la fecha actual

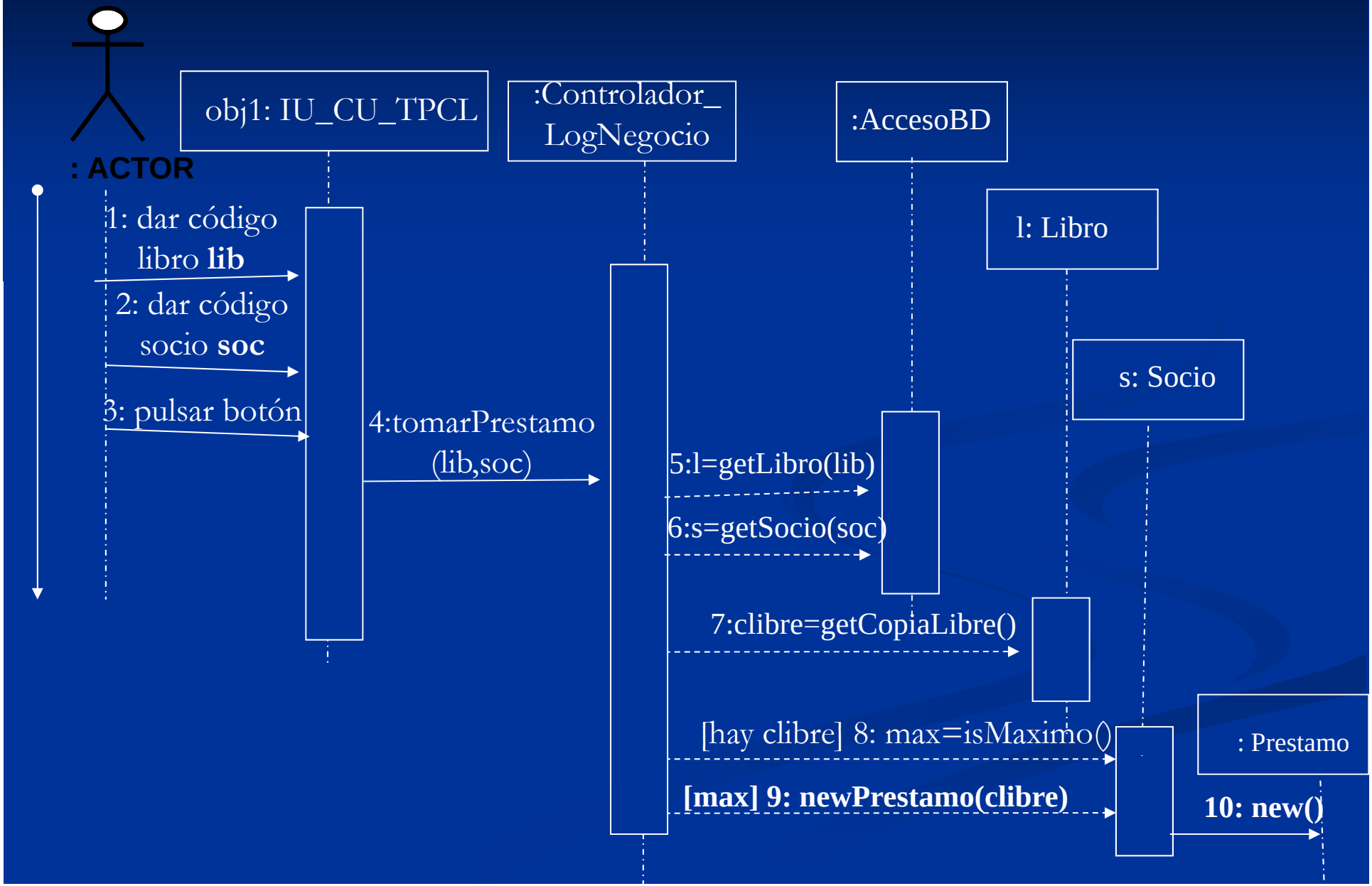
¿Qué objeto guarda los préstamos?



El objeto de la clase SOCIO

Nota: también podría ser el objeto de la clase COPIA_LIBRO

Diagrama de secuencia TPCL (5)



Ejemplo de aplicación de patrón CREADOR (continuación)

Se registra el nuevo préstamo con la fecha actual

¿Qué objeto guarda los préstamos?

El objeto de la clase SOCIO

¡¡¡Pero el objeto de AccesoBD también, ya que debe dar persistencia al nuevo préstamo !!!!

Diagrama de secuencia TPCL (6) (alternativa 1)

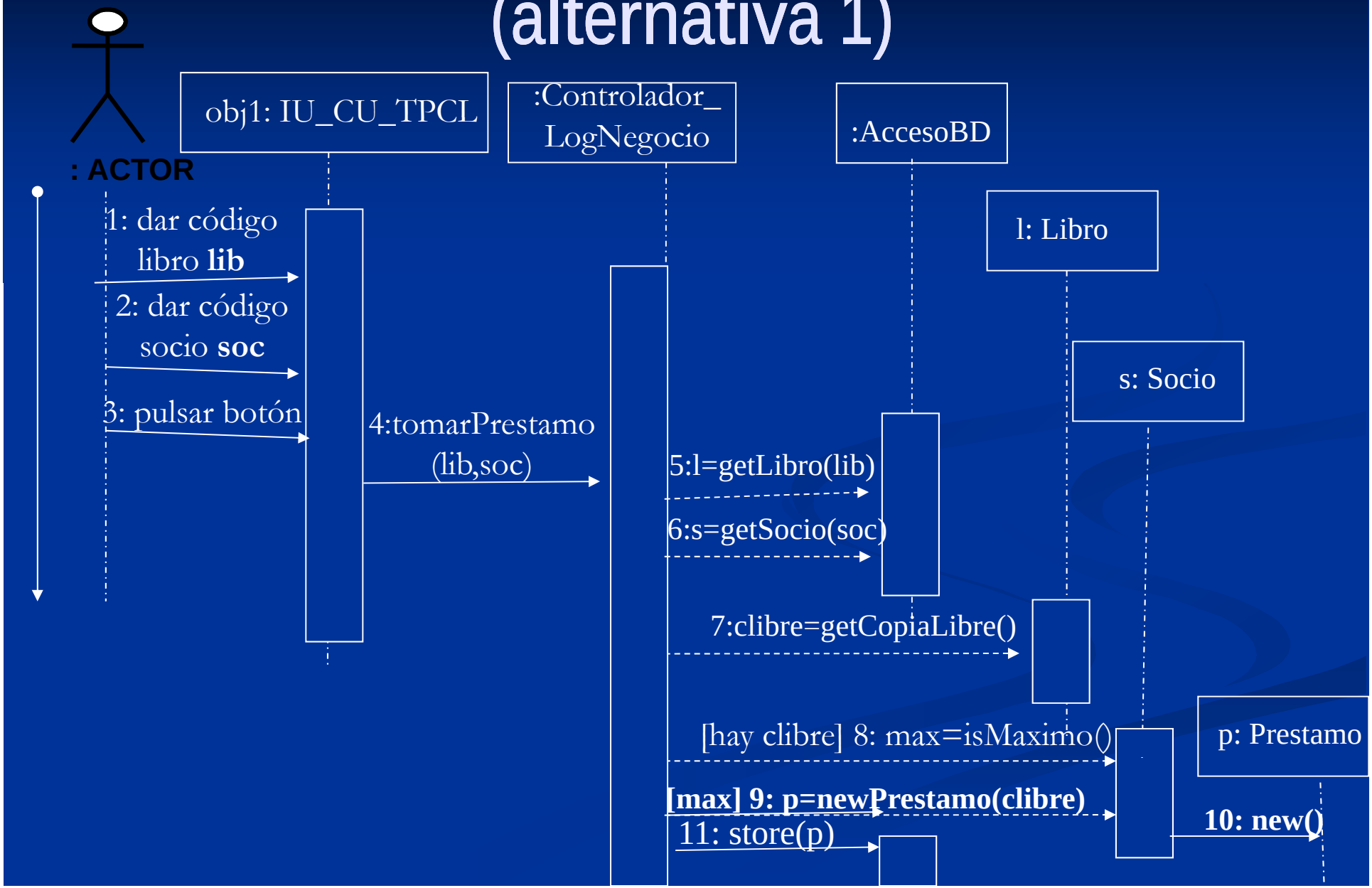
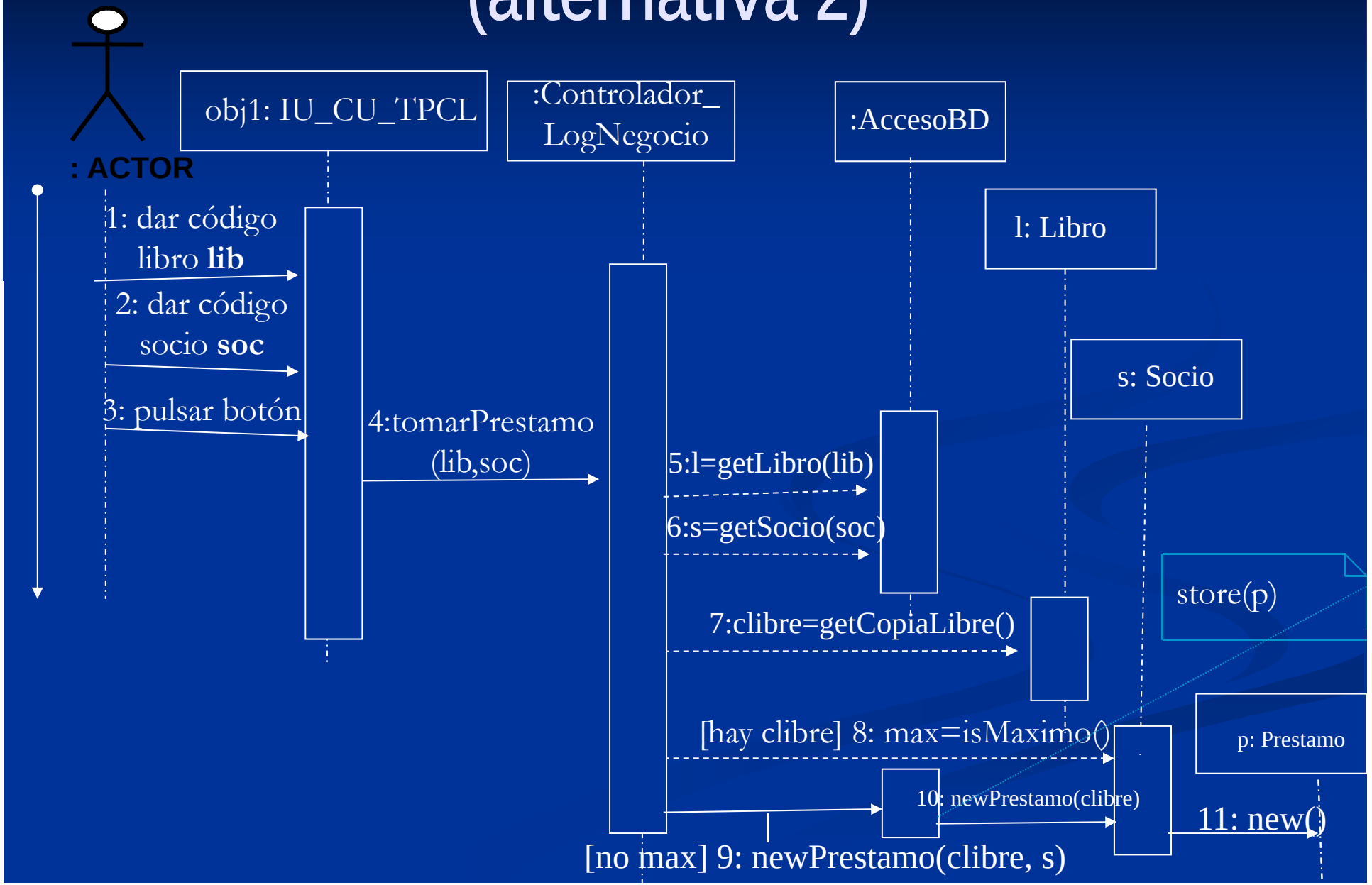
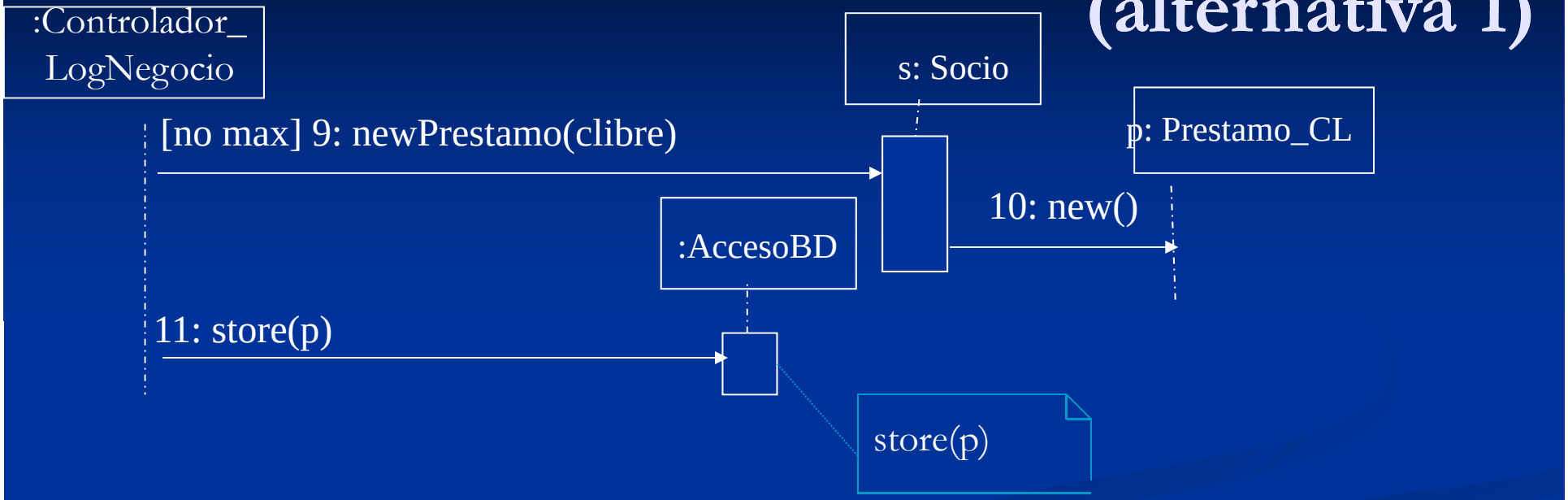


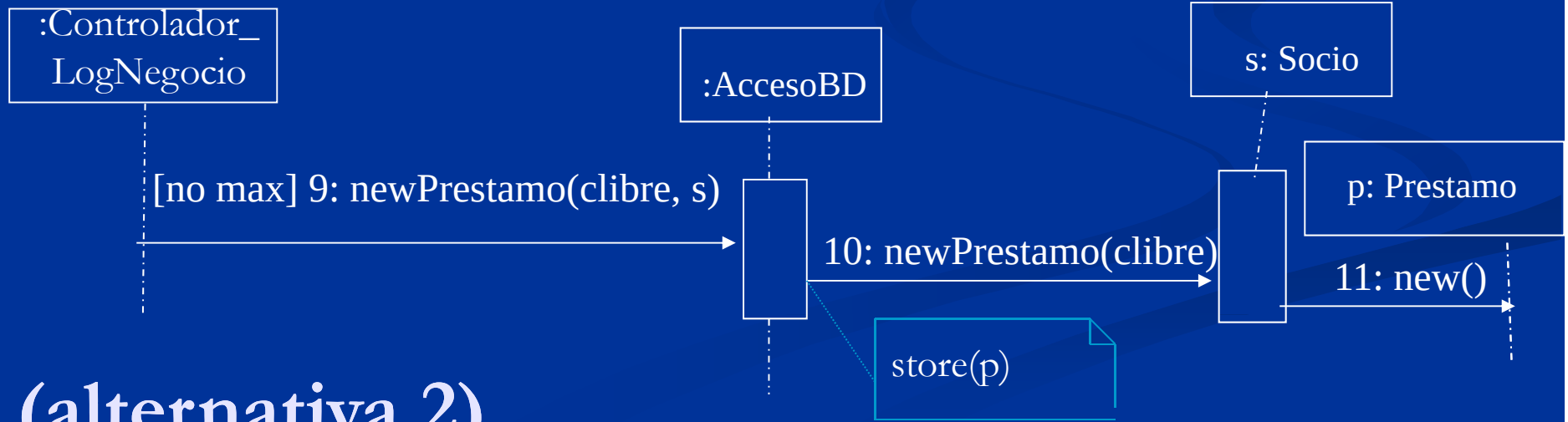
Diagrama de secuencia TPCL (6) (alternativa 2)



¿Qué alternativa de diseño escoger?

(alternativa 1)





(alternativa 2)

3.3.2.4.- Patrón BAJO ACOPLAMIENTO

Nombre: BAJO ACOPLAMIENTO

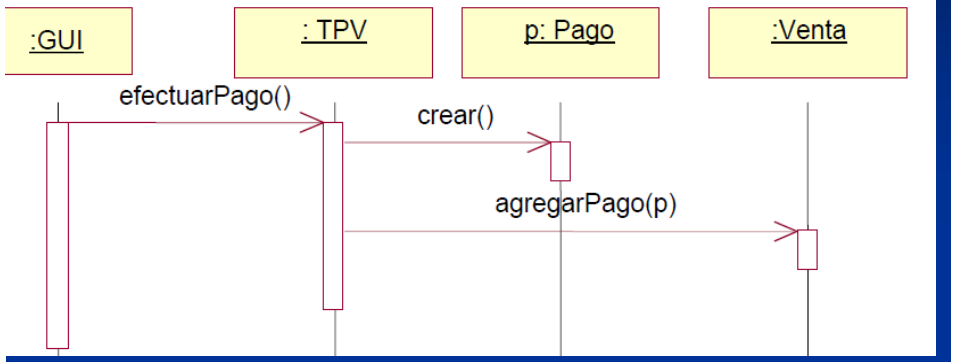
PROBLEMA: ¿Cómo reducir las dependencias entre clases?

SOLUCIÓN: Asignar la responsabilidad de manera que el acoplamiento permanezca bajo

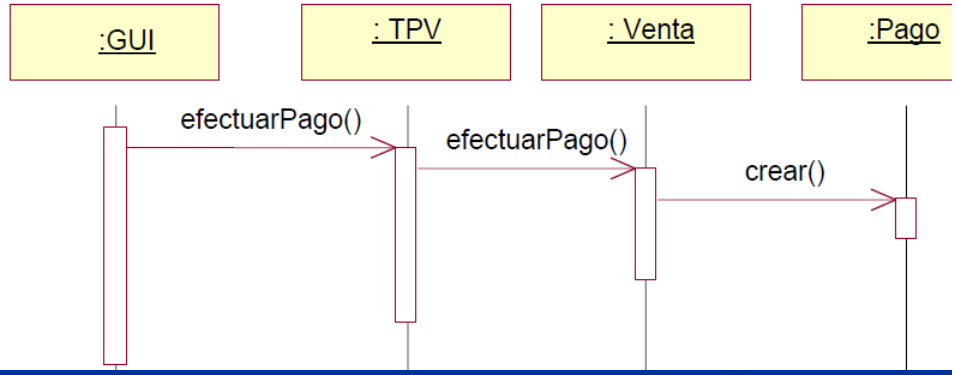
Existe acoplamiento entre A y B si A usa B (A tiene atributo del tipo B, o bien un método que usa o devuelve B,…)

Ejemplo: BAJO ACOPLAMIENTO

(1)



(2)



El diseño (2) tiene acoplamiento más bajo:

- En ambos Venta está acoplada a Pago
- En (1) TPV está acoplada a Pago y a Venta
- En (2) TPV está acoplada a Venta, ¡pero no a Pago !

Nota: el nivel de acoplamiento no se puede considerar de manera aislada a otros patrones como el EXPERTO y el ALTA COHESIÓN

3.3.2.5.- Patrón ALTA COHESIÓN

Nombre: ALTA COHESIÓN

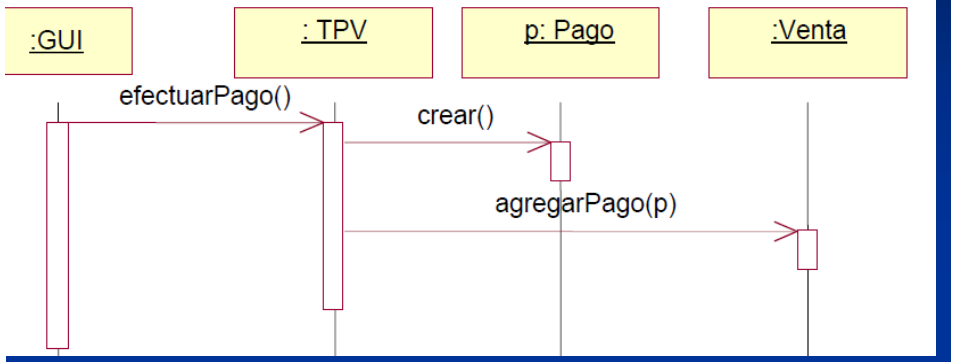
PROBLEMA: ¿Cuánto están relacionadas las responsabilidades de una clase? ¿Cómo mantener la complejidad manejable?

SOLUCIÓN: Asignar una responsabilidad de manera que la cohesión permanezca alta.

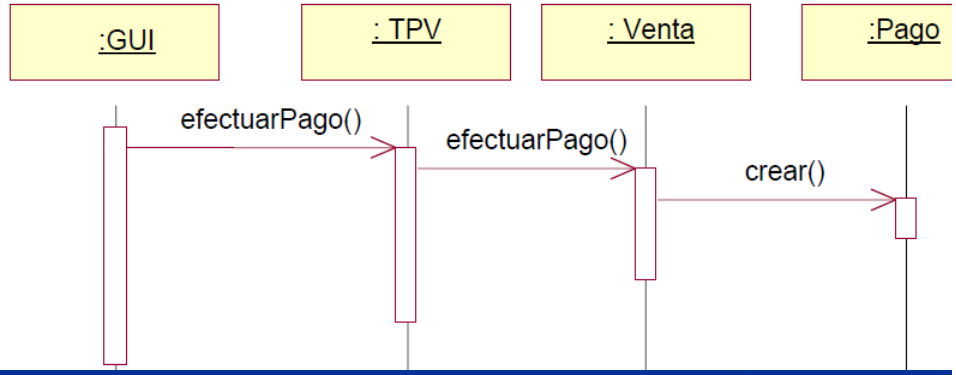
Una clase tiene baja cohesión, si es una clase que tiene muchas responsabilidades no relacionadas, que hace demasiado trabajo, que no delega. Son clases difíciles de entender, reutilizar, mantener,…

Ejemplo: ALTA COHESIÓN

(1)



(2)



El diseño (1) tiene una cohesión más baja:

- Tiene una clase (TPV) que se encarga de crear el Pago, no ha delegado la creación del Pago en Venta.

Nota: el nivel de cohesión no se puede considerar de manera aislada a otros patrones como el EXPERTO y el BAJO ACOPLAMIENTO

¿Qué alternativa escoger? (1)

- Teniendo en cuenta únicamente los patrones ALTA COHESIÓN /BAJO ACOPLAMIENTO parece que la mejor alternativa es la (2)
- Sin embargo, conviene tener en cuenta quién tiene la responsabilidad de proporcionar la PERSISTENCIA DE OBJETOS

¿Qué alternativa escoger? (2)

- **La alternativa (1) sería más interesante si quisiéramos que el nivel de ACCESO A DATOS no conociera las CLASES PROVINIENTES DEL MODELO DEL DOMINIO, que simplemente recuperara o diera persistencia a objetos.**
  - **En ese caso la cohesión de la clase AccesoBD sería mayor, ya que tendría menos responsabilidades, pero la cohesión de la clase ControladorLogNegocio sería menor ya que tendría que ocuparse también de la persistencia de los objetos (pidiéndoselo a AccesoBD)**
- La alternativa (2) sería más interesante si quisiéramos que el nivel de ACCESO A DATOS conociera el MODELO DEL DOMINIO, y fuera un experto en RECUPERAR, INSERTAR ACTUALIZAR y BORRAR objetos de la BD).
  - La cohesión de la clase AccesoBD sería menor, ya que tendría más responsabilidades, pero a cambio, la cohesión del controlador aumentaría, ya que las operaciones que implicaran actualizar, insertar, borrar y algunas de las de recuperar en la BD se las pediría a la clase AccesoBD y no a las clases del dominio.

Decisión de diseño para el Acceso a Datos

- Cualquier operación que IMPLIQUE un ACCESO a la BD (RECUPERACIÓN, INSERCIÓN, ACTUALIZADO O BORRADO de un objeto de la BD) se enviará al objeto de la clase ACCESOBD, que será la EXPERTA en comunicación con el SGBDOO.
- Solamente cuando el controlador de la lógica del negocio conozca la referencia exacta del objeto del dominio al que quiera preguntar (no modificar, insertar o borrar) podrá solicitarle la ejecución de algún método

(alternativa 2)

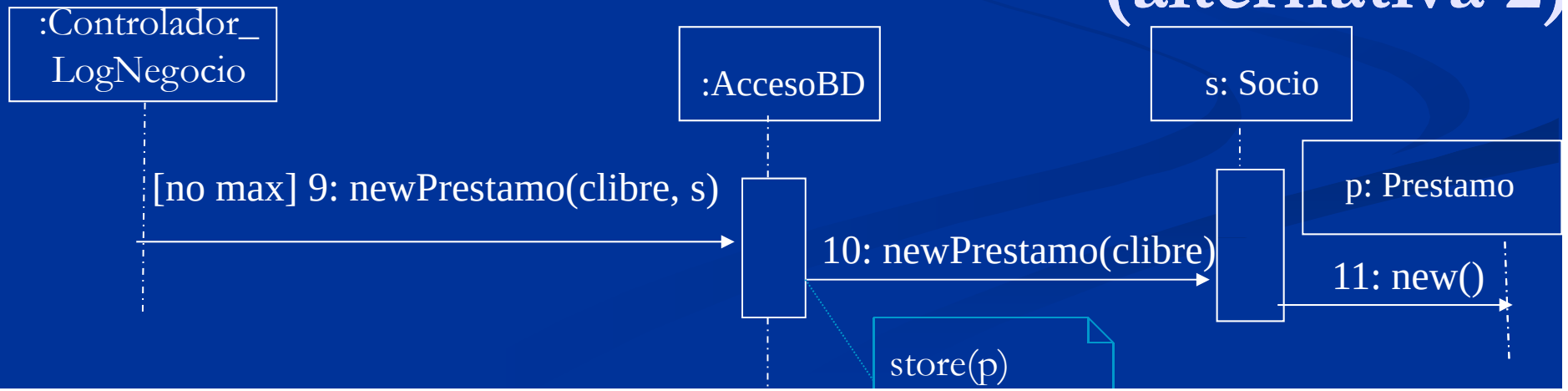


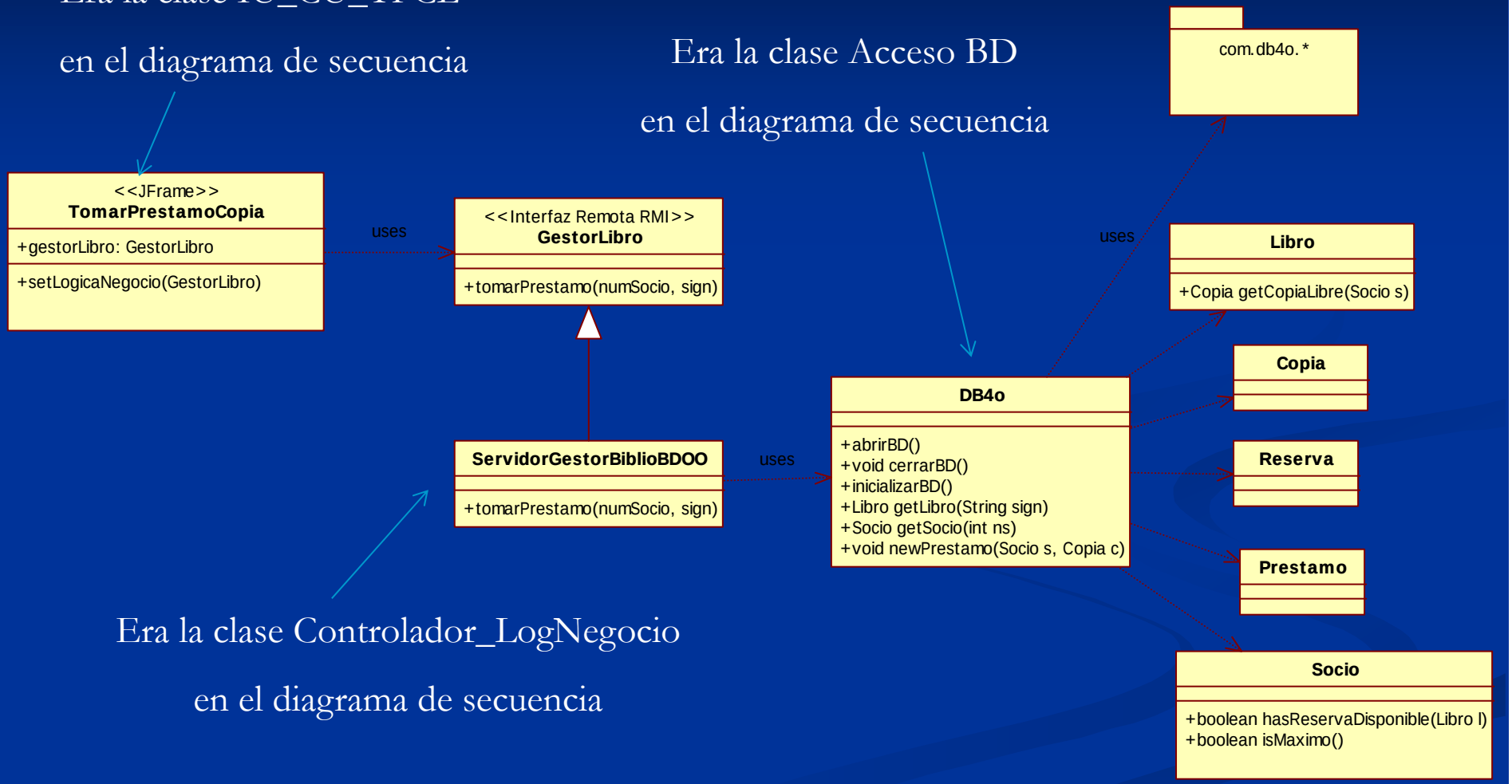
Diagrama de clases para el caso de uso

Era la clase IU_CU_TPCL

en el diagrama de secuencia

Era la clase Acceso BD

en el diagrama de secuencia



Era la clase Controlador_LogNegocio

en el diagrama de secuencia