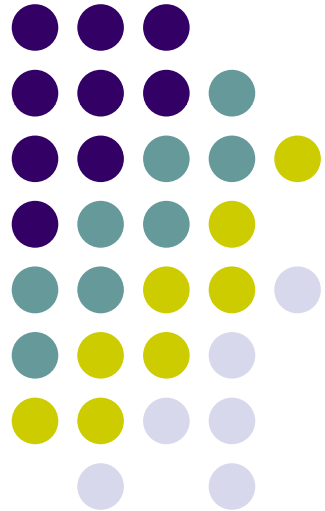
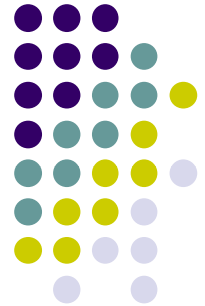
Metodologías de Diseño y Programación

Arquitectura de Software



Contenido



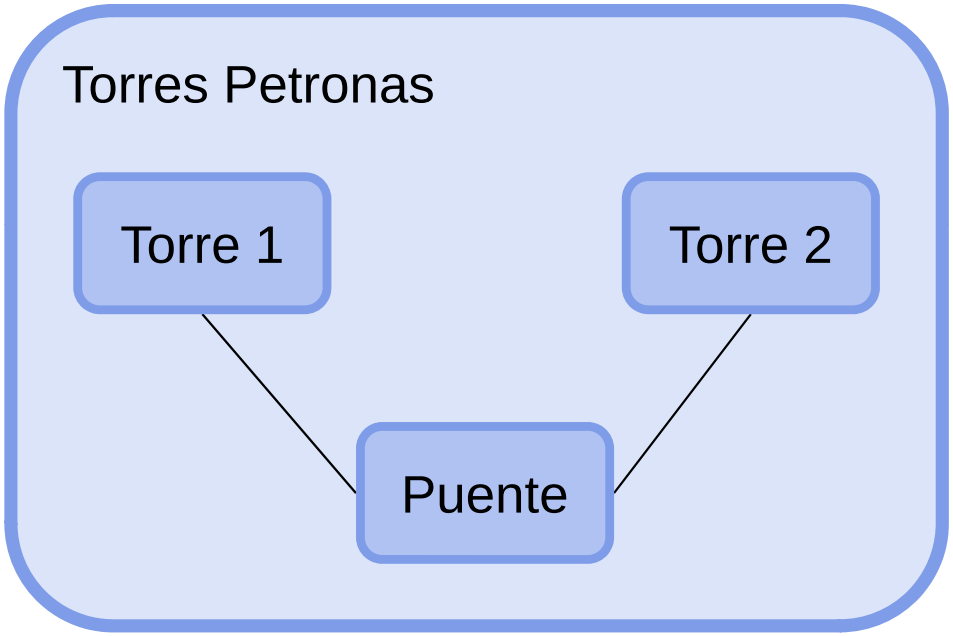
- Introducción
- Arquitectura de Software
- Arquitectura Lógica
- Arquitectura en Capas

Introducción

- La Arquitectura de Software busca expresar la estructura global de una aplicación
- El nivel de abstracción empleado para expresar dicha estructura es mayor que el empleado para detallar la solución al problema de software planteado (diseño)
- El objetivo de la arquitectura no es detallar la solución adoptada sino que es proveer una visión global de la misma para simplificar su comprensión

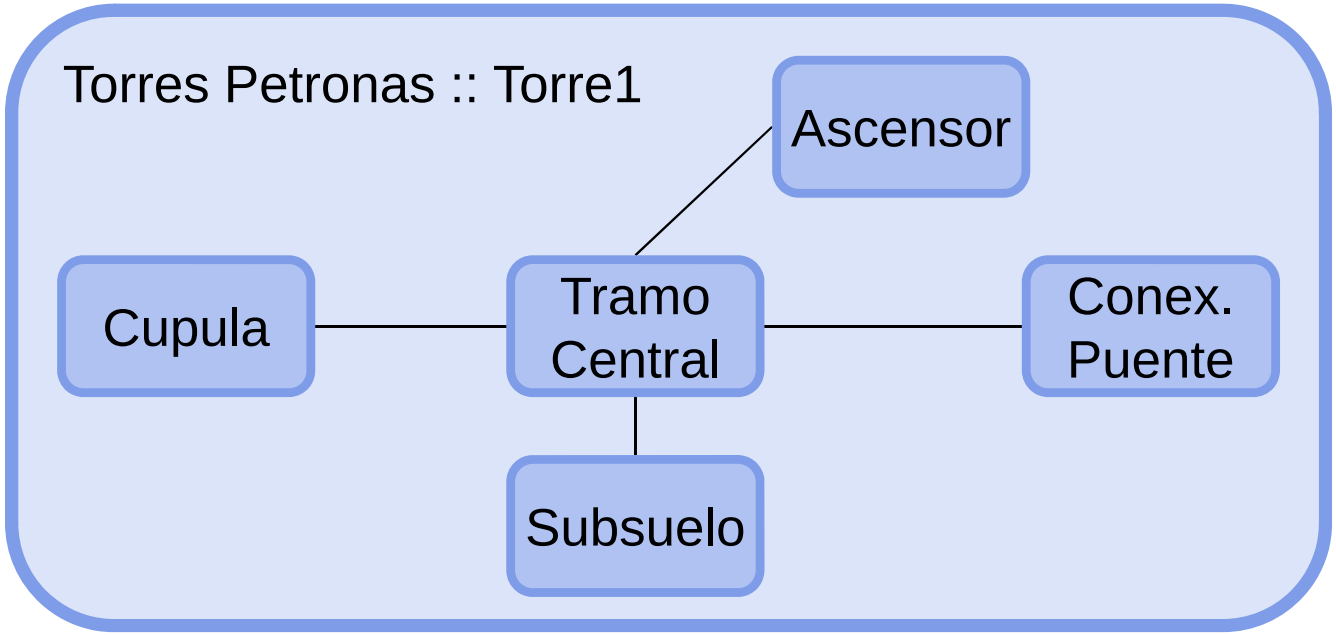
Introducción (2)

- La arquitectura de software permite obtener una visión simplificada pero completa de la solución



Introducción (3)

- La arquitectura se puede refinar sucesivamente mostrando cada vez más detalle
- Ese camino de refinamiento sucesivo conduce al diseño de cada una de las grandes piezas



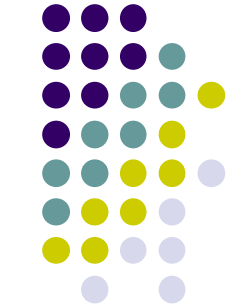
Arquitectura de Software

- La arquitectura no solamente ayuda a comprender mejor un sistema ya terminado
- Realizar la arquitectura antes de diseñar la solución permite llevar el problema a un nivel de abstracción tal que es posible manejar la complejidad de toda la aplicación
- De esta manera un desarrollador puede abarcar todo el sistema sin perderse en los detalles de más bajo nivel de alguna de sus partes
- Puede entenderse como “diseño de alto nivel”

Arquitectura de Software (2)

- La estructura expresada por la arquitectura puede ser estudiada desde diferentes puntos de vista
- Desde la estructura interna de la propia aplicación hasta la estructura informática sobre la cual la aplicación ejecutará
- Usualmente la arquitectura es representada en un artefacto específico (SAD)
  - Su contenido se conforma con partes de los diferentes modelos creados

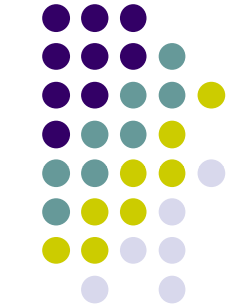
Arquitectura Lógica



- Uno de los puntos de vista desde donde se suele estudiar la estructura de una aplicación es la **estructura interna**
- La arquitectura desde ese punto de vista se denomina **Arquitectura Lógica**
- Esta arquitectura se desarrolla y se presenta utilizando un enfoque top-down

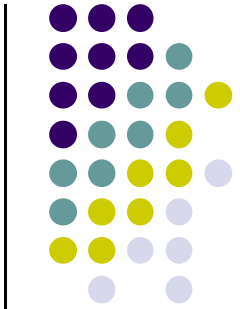
Arquitectura Lógica

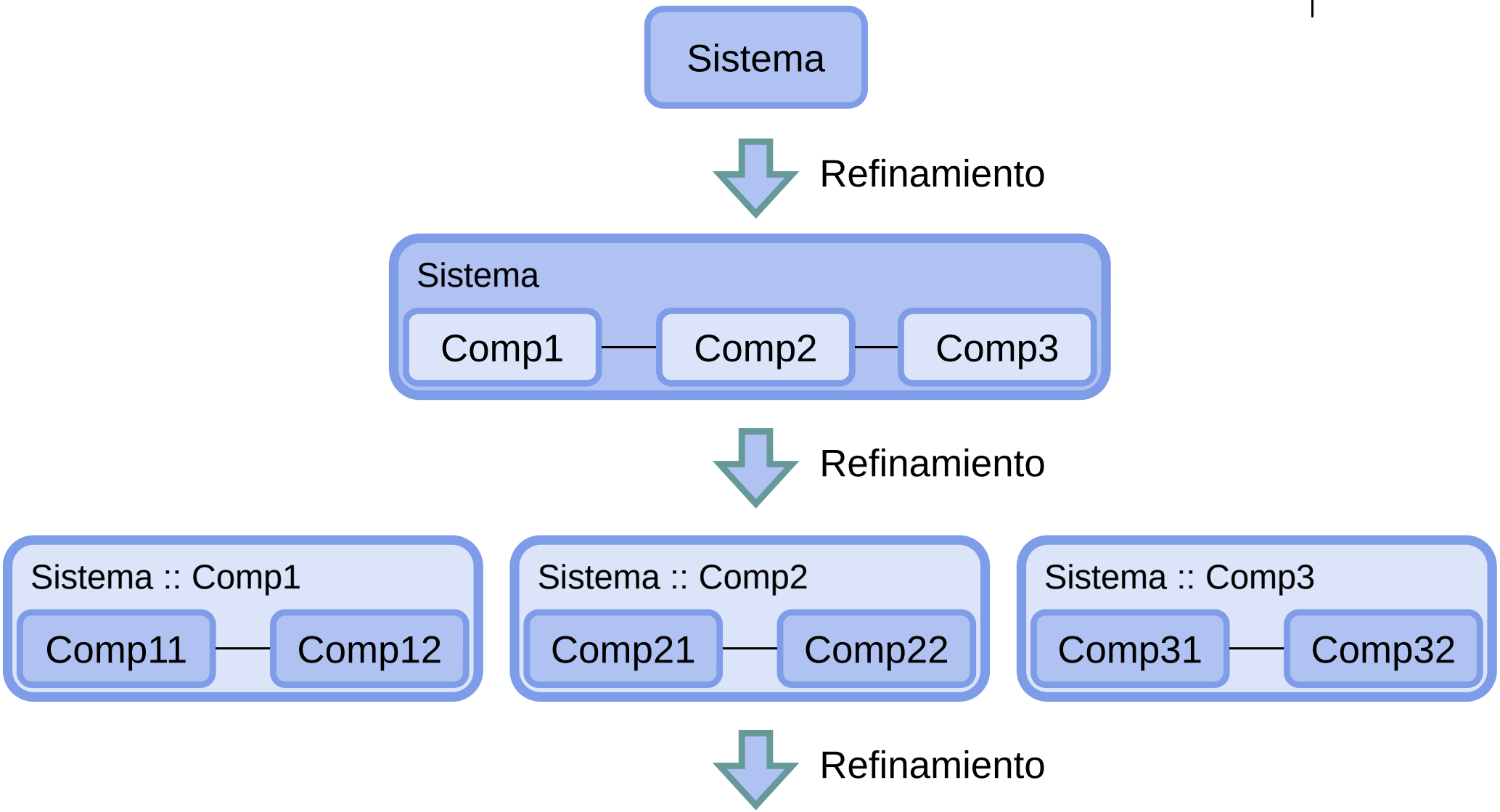
Enfoque Top-Down



- Se parte de una visión del sistema como una unidad atómica identificando componentes internos
- Cada uno de estos será responsable de resolver diferentes aspectos de la aplicación
- Se refina sucesivamente cada uno de dichos componentes en componentes de granularidad más fina

Arquitectura Lógica
Enfoque Top-Down (2)

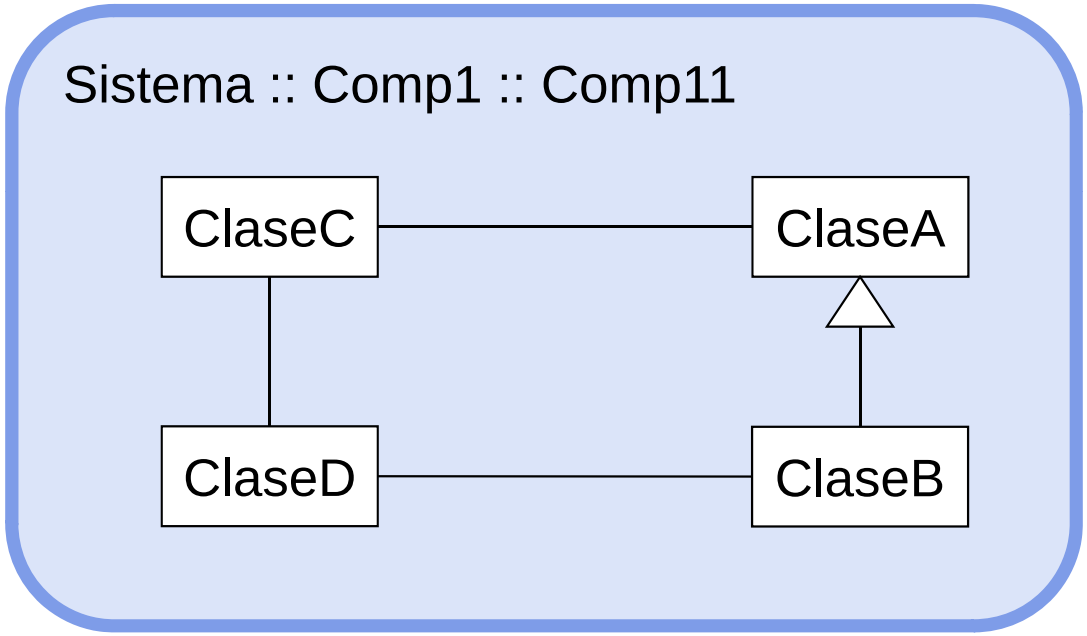




Arquitectura Lógica

Enfoque Top-Down (3)

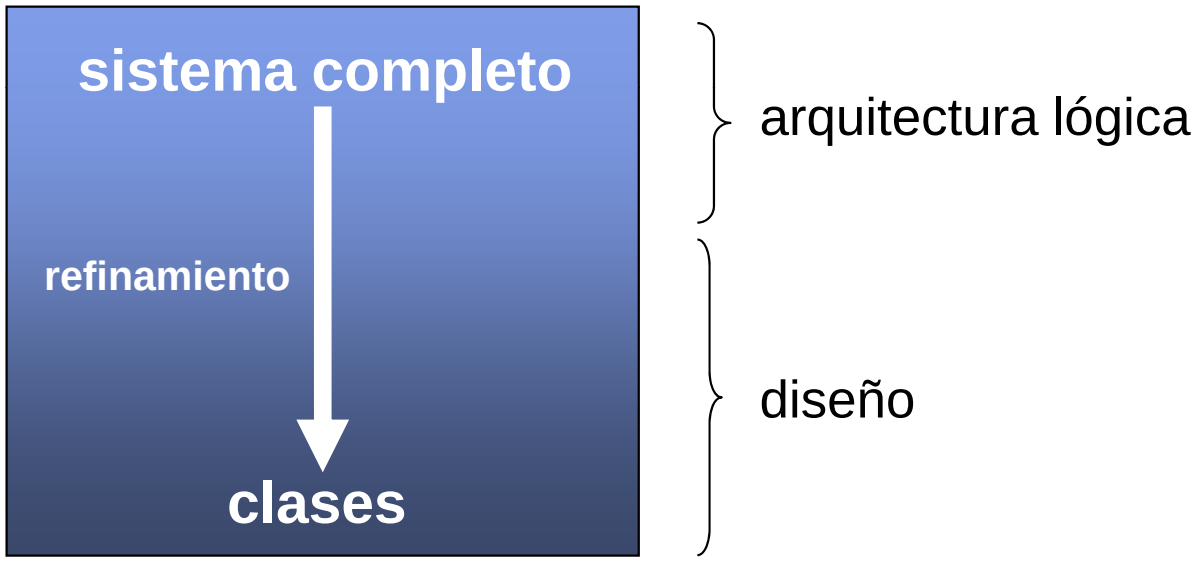
- El refinamiento llega hasta al punto en que se definen las clases que constituyen la solución al problema (diseño)



Arquitectura Lógica

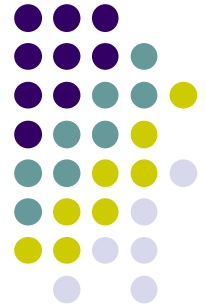
Enfoque Top-Down (4)

- El límite entre Arq. Lógica y Diseño es difuso
- Cuando se alcanza un nivel de detalle que hace al sistema inabarcable en amplitud se considera que comienza el Diseño



Arquitectura Lógica

Enfoque Top-Down (5)



- ¿Cómo se realiza la partición de los componentes en los primeros niveles del refinamiento?
  - ¿Existe algún criterio que indique cuál es una buena partición y cuál no?
  - ¿Siempre se particiona de la misma forma?
  - ¿De qué depende?

Arquitectura Lógica

Enfoque Top-Down (5)

- Existen **guías** de particionamiento
- Cada una de ellas propone
  - Un tipo de partición particular
  - Asignación de responsabilidades a los componentes resultantes
- Se denominan estilos o patrones de arquitectura
- La elección del estilo a aplicar depende del tipo de sistema que se esté construyendo

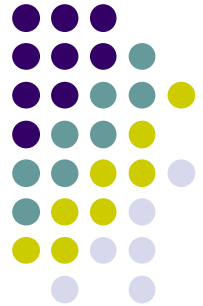
Aspectos de una Aplicación

- El diseño de un sistema de software comprende la resolución de múltiples aspectos de una aplicación
- La forma en que esos aspectos sean resueltos determina la flexibilidad del diseño
- Desde un punto de vista lógico es preferible separar el diseño de aspectos diferentes para
  - Permitir que evolucionen independientemente
  - Simplificar el problema y tener mejor visibilidad de las partes que componen la aplicación

Aspectos de una Aplicación (2)

- Diseñar e implementar en forma conjunta diferentes aspectos
  - Usualmente simplifica la arquitectura, pero
  - Complica el diseño
- La separación de aspectos generalmente
  - Complica la arquitectura (estructura general), pero
  - Simplifica el diseño de cada uno al permitir enfocarse en cada aspecto por separado, y
  - Permite trazar mejor cada aspecto

Aspectos de una Aplicación (3)



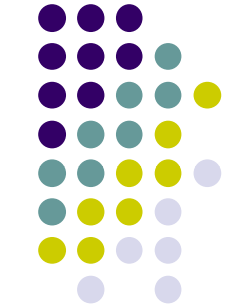
```
class Persona {
      //atributos

      void mostrar() {
            write(atributos);
      }

      void procesar(entrada) {
            //hacer algo con entrada y atributos
      }

      void guardar() {
            write(arch,atributos);
      }
}
```

Aspectos de presentación

Aspectos de lógica de la aplicación

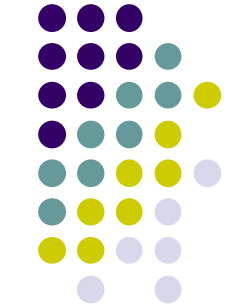
Aspectos de acceso a la persistencia de datos

Aspectos de una Aplicación (4)



- En el ejemplo anterior se detectan fragmentos de código con diferentes propósitos en una misma clase
  - Código para procesar la información existente que implementa la lógica de la aplicación,
  - Código de interacción con el usuario, y
  - Código que sirve para almacenar los datos en un medio persistente

Aspectos de una Aplicación (5)

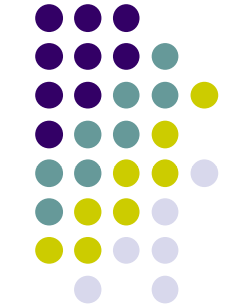


- Esto es común a la mayoría de los sistemas de información interactivos
- Se puede decir que estas aplicaciones abarcan básicamente tres aspectos
  - **Presentación:** incluye todo lo referente a la interacción del sistema con los usuarios en el mundo exterior
  - **Lógica:** se encarga del procesamiento particular que el sistema deba realizar sobre la información que maneja
  - **Persistencia:** consiste en el almacenamiento persistente de dicha información

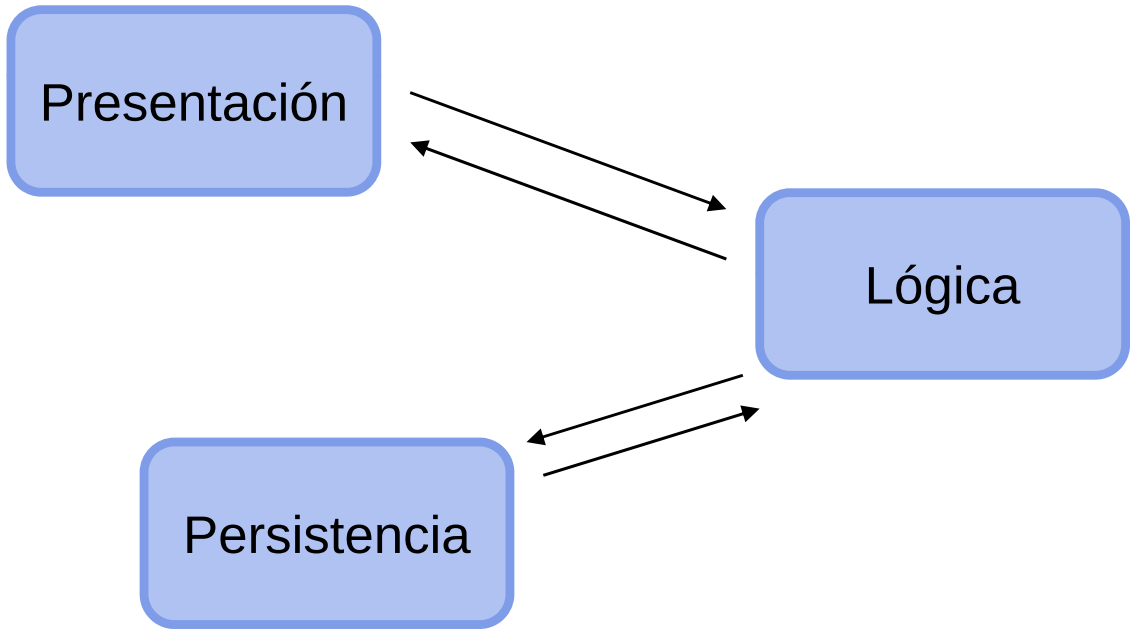
Aspectos de una Aplicación (6)

- Incluir los tres aspectos en una misma clase no resulta flexible
- La clase completa queda dependiente de
  - La forma en que los datos son mostrados u obtenidos del usuario, y además de
  - La forma en que los datos son almacenados
- Es deseable establecer una separación de dichos aspectos
- Es decir, mantener el código referente a cada aspecto en clases separadas

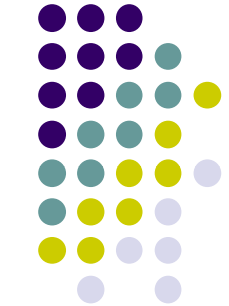
Aspectos de una Aplicación (7)



- Esto sugiere un criterio concreto de partición de componentes

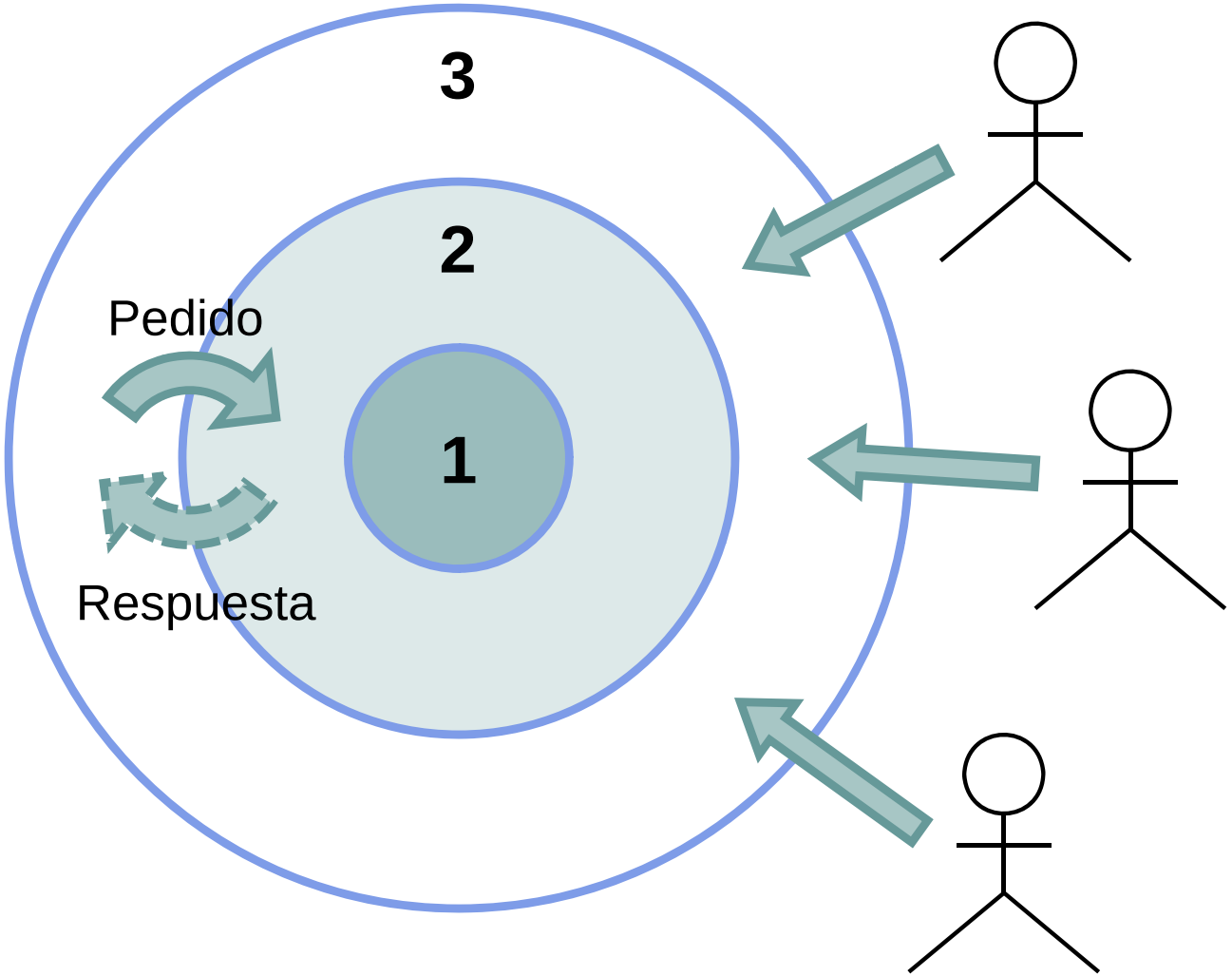


Arquitectura en Capas



- Una Arquitectura en Capas es la arquitectura de un sistema que haya sido particionado según el estilo de Capas
- Estilo de Capas
  - Define diferentes “niveles” de elementos
  - Los elementos de un mismo nivel tienen responsabilidades de abstracción similar
  - Los elementos de un nivel están para atender los pedidos de los elementos del nivel superior

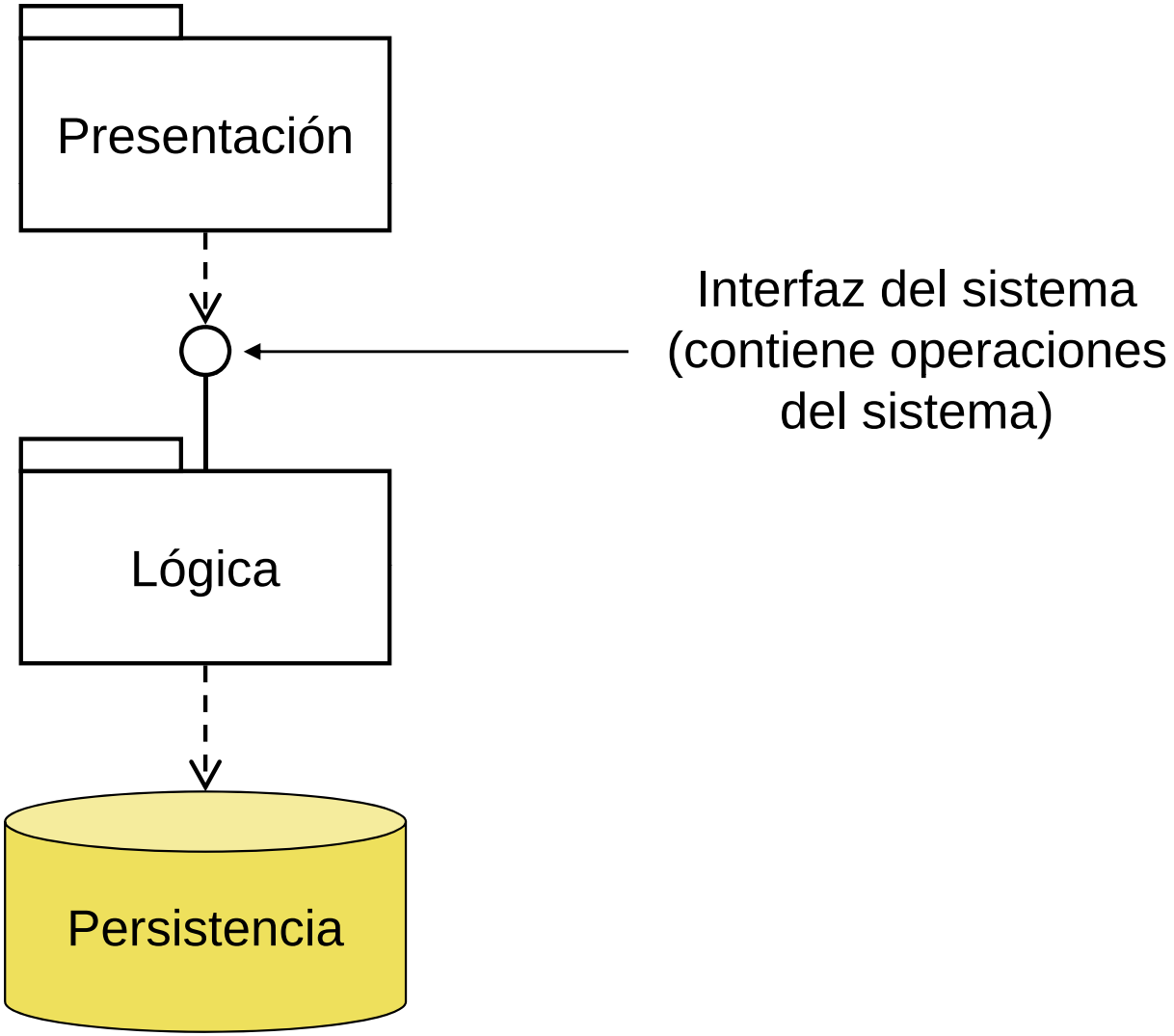
Arquitectura en Capas (2)



Arquitectura en Capas (3)

- La partición definida anteriormente es compatible con el estilo de Capas
- Se definen por lo tanto las siguientes capas
  - Presentación
  - Lógica
  - Persistencia
- Los actores utilizan solamente la capa de presentación
- La capa de persistencia no requiere de los servicios de ninguna otra

Arquitectura en Capas (4)

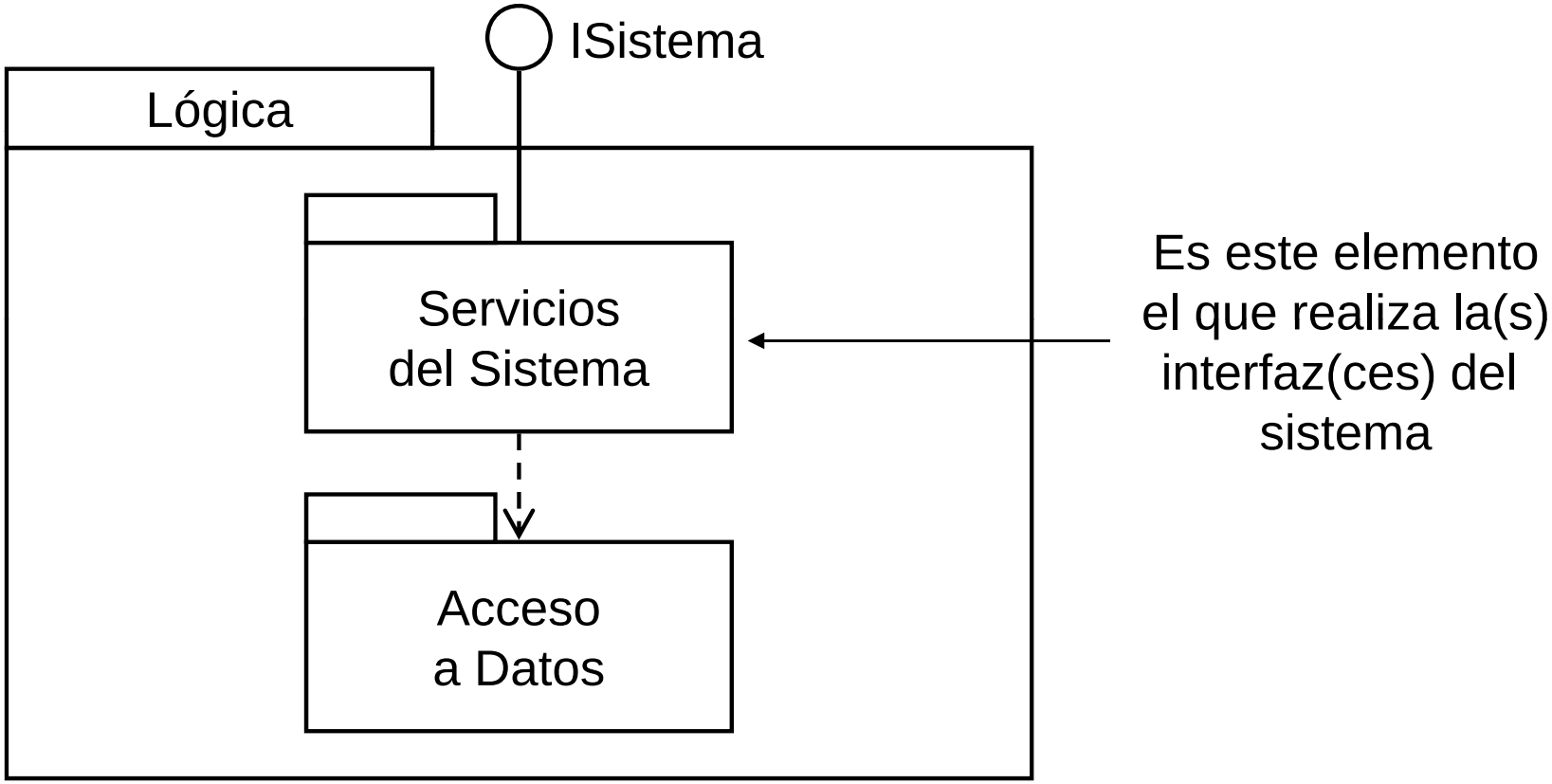


Arquitectura en Capas (5)

- ¿Qué hay en cada capa?
  - Presentación: clases que se encargan de capturar la entrada de los usuarios y mostrar información
  - Lógica:
    - Clases que describen los objetos que procesarán la información para satisfacer los casos de uso del sistema
    - Clases que permiten a las anteriores acceder a los datos
  - Persistencia: datos del sistema que necesiten ser preservados (texto plano, XML, base de datos, etc.)
- Nos enfocaremos en la capa lógica

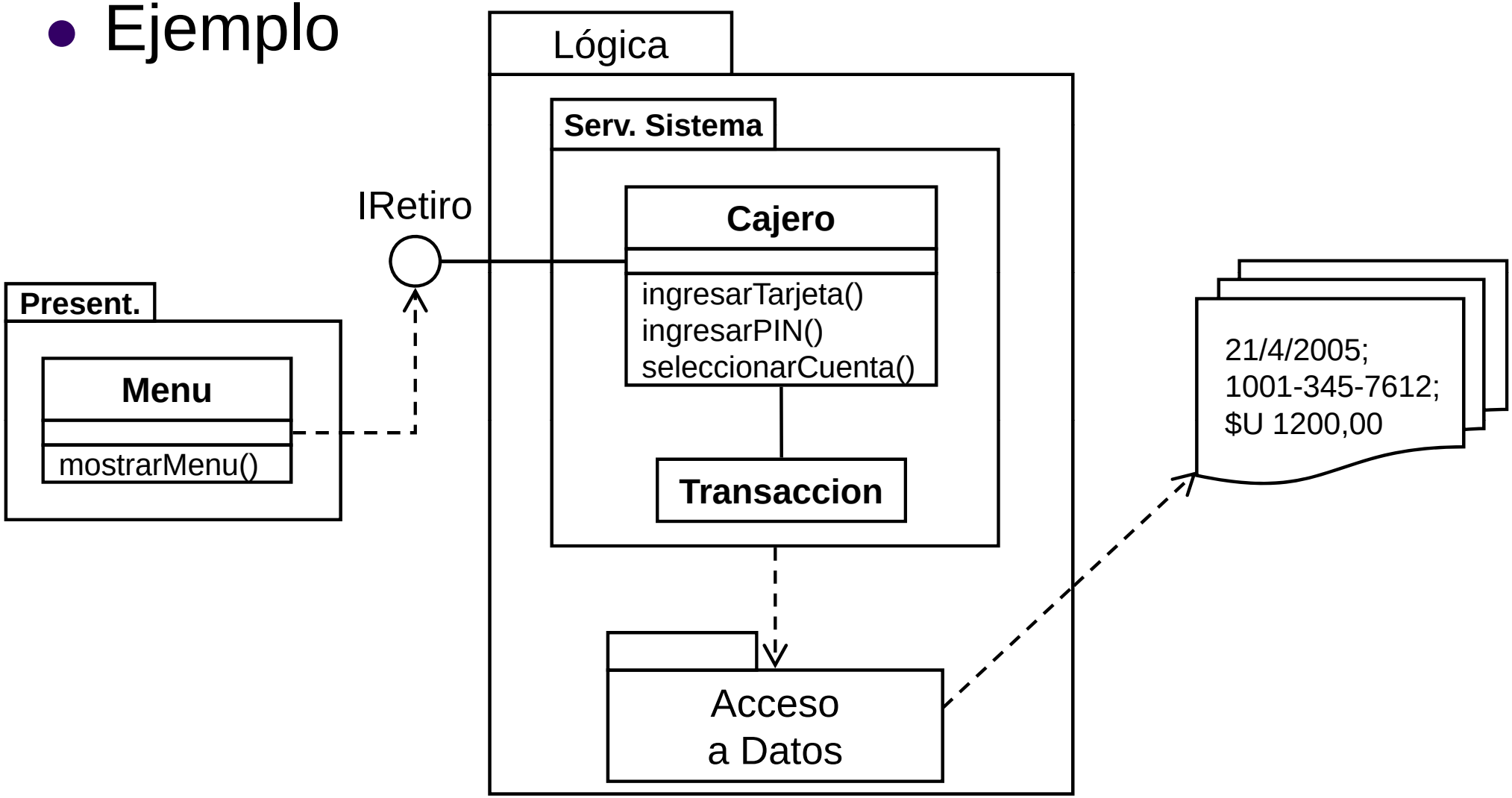
Arquitectura en Capas (6)

- Usualmente la capa lógica es refinada de la siguiente manera



Arquitectura en Capas (7)

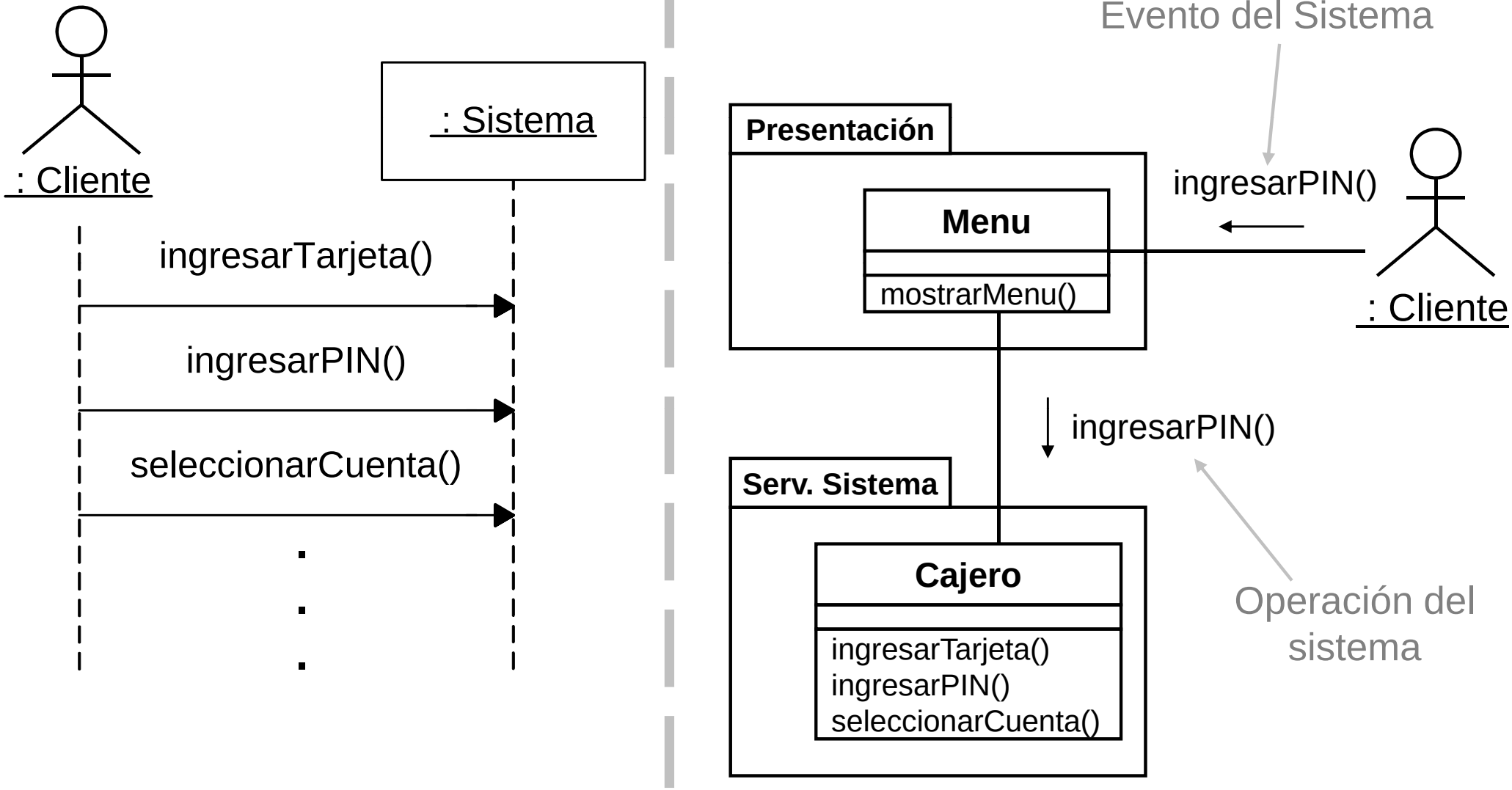
- Ejemplo



Operaciones del Sistema

- Los Diagramas de Secuencia del Sistema ilustran la forma en que los actores realizan “invocaciones” sobre el sistema
- Al estudiar la Arquitectura Lógica es posible profundizar en los detalles de cómo se realizan dichas invocaciones

Operaciones del Sistema (2)



Implementación

```
// pertenece en forma lógica a la Capa de Presentacion
class Menu {
      IRetiro atm;

      void mostrarMenu() {
            // leer en t el número de tarjeta
            atm.ingresarTarjeta(t);
            // leer en p el número de PIN
            atm.ingresarPIN(p);
            // leer en c el número de cuenta
            atm.seleccionarCuenta(c);
            .
            .
}
```

Implementación (2)

```
// pertenecen en forma lógica a la Capa Logica
interface IRetiro {
      void ingresarTarjeta();
      void ingresarPIN();
      void seleccionarCuenta();
}

class Cajero realize IRetiro {
      public void ingresarTarjeta() {...}
      public void ingresarPIN {...}
      public void seleccionarTarjeta() {...}
      .
      .
}
```

Estado de la Sesión

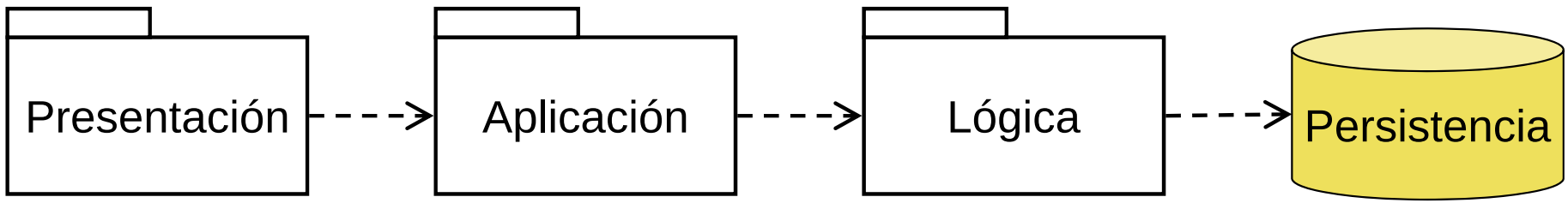
- Generalmente el sistema debe recordar cierta información que un actor le va proveyendo a lo largo de un caso de uso
- De no hacerlo, el actor deberá repetirla en cada mensaje subsiguiente
- Ejemplo
  - El sistema debe recordar el número de cuenta seleccionado para luego consultar su saldo y realizar el débito
- NO es responsabilidad de un habitante de la Capa de Presentación mantener el estado

Estado de la Sesión (2)

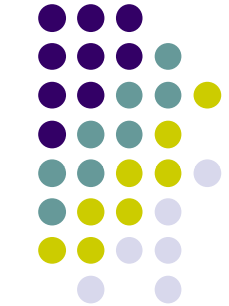
- En el ejemplo anterior es la instancia de Cajero quien mantiene el estado de la sesión
- El enfoque de responsabilizar del estado de la sesión a quien ofrece las operaciones del sistema es en general suficiente
- En algunos casos se separan estas responsabilidades introduciendo una nueva capa: La Capa de Aplicación

Capa de Aplicación

- Los objetos habitantes de esta capa tienen como responsabilidad mantener el estado de la sesión
  - Actúan como mediadores entre la presentación y la lógica

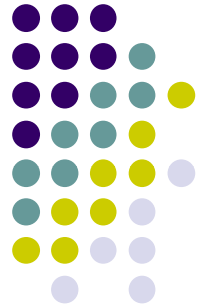


Capa de Aplicación (2)

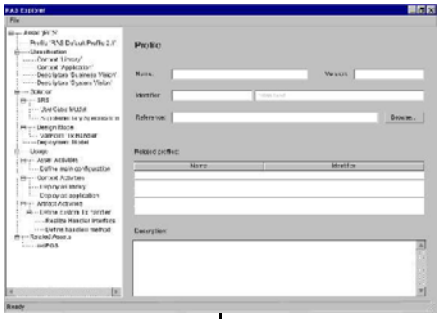


- La Capa de Aplicación es particularmente útil en casos en que
  - Existe más de una Capa de Presentación
  - El sistema es multi-usuario
  - La Capa de Presentación está físicamente separada de la Capa Lógica
- Cada actor llevando adelante un caso de uso tiene asignado un objeto de Capa de Aplicación propio para que lleve el estado de su sesión

Capa de Aplicación (3)



Presentación



info de productos

info de productos

Aplicación

: AppCoord

: AppCoord

procesar venta

procesar venta

Lógica

: Tienda

Capa de Aplicación (4)

- Un objeto de Capa de Aplicación puede además coordinar el diálogo entre la Presentación y la Lógica
- De esta manera el coordinador le indica a la Presentación qué “pantalla” utilizar para capturar los próximos datos necesarios en el caso de uso
- Esta responsabilidad suele ser asignada a la Capa de Presentación (por “simplicidad”)