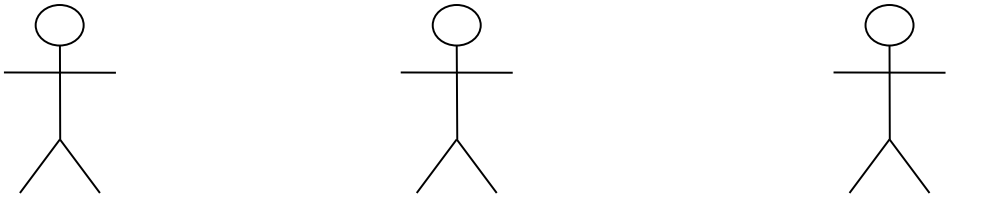
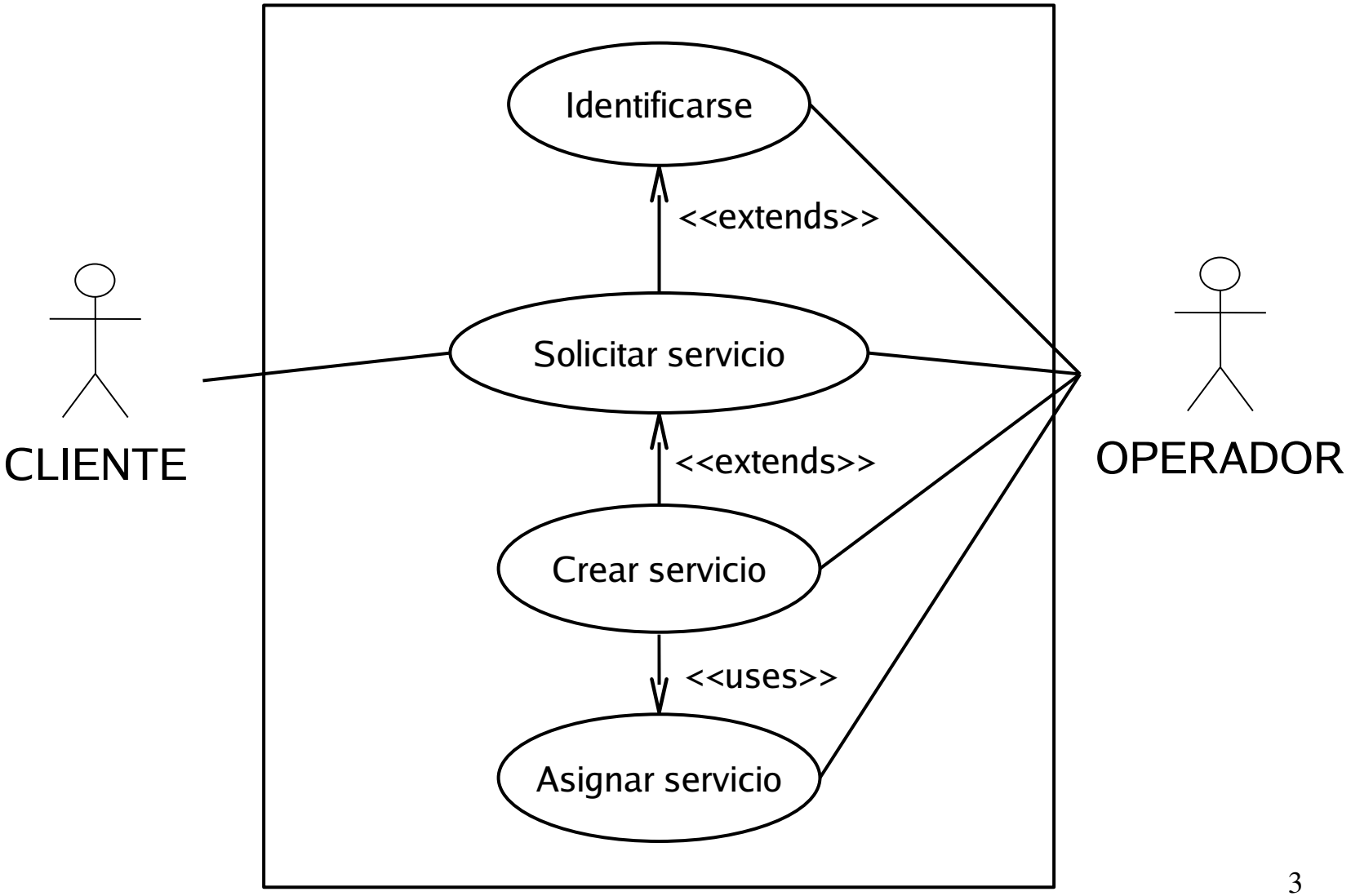
Solución Examen Junio 2007 (a)

- Ejercicio GeoTaxi (1h 20 min.)
  - Diagrama de Casos de Uso y
  - Casos de uso (2,5 puntos)
  - Modelo de Dominio (1,5 puntos)

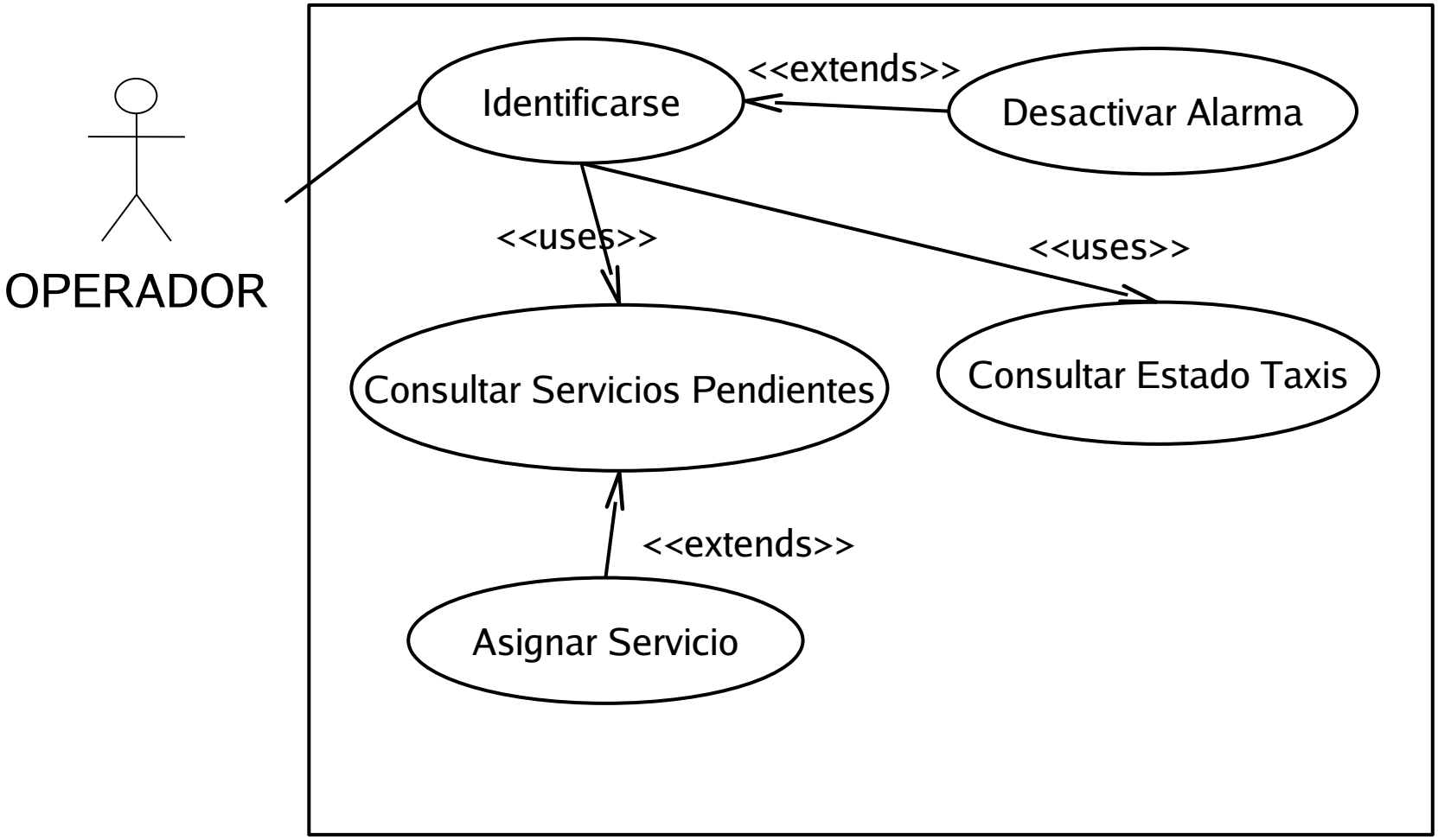
Actores



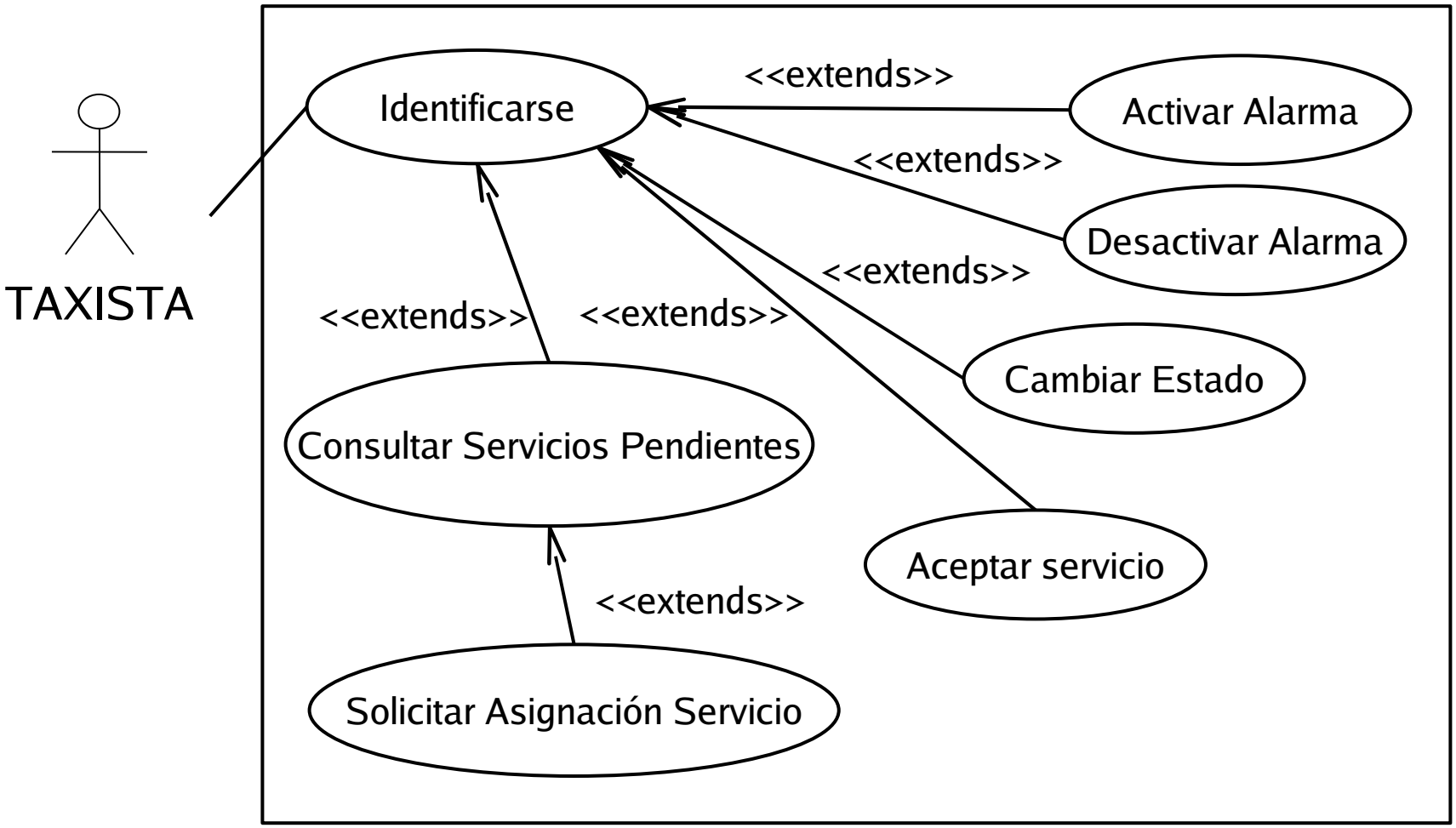
Actor Cliente



Actor Operador



Actor Taxista



Casos de uso de alto nivel (1)

Caso de uso: **Identificarse**

Actores: OPERADOR, TAXISTA

Descripción: Cada usurio (Operador o Taxista) debe identificarse para acceder al sistema GeoTaxi.

Caso de uso: **Solicitar Servicio**

Actores: CLIENTE, OPERADOR

Descripción: Un Operador recibe una llamada de un Cliente solicitando un servicio. Cada operador dispone de un terminal donde aparecen las solicitudes pendientes de asignar y el plano de la ciudad. Sobre el plano aparece la localización de los distintos taxis y su estado: libre, ocupado o fuera de servicio. Los taxis libres aparecen destacados.

Casos de uso de alto nivel (2)

Caso de uso: **Crear servicio**

Actores: OPERADOR

Descripción: Para asignar un taxi a un servicio, el operador introduce la dirección origen del servicio.

Caso de uso: **Asignar servicio**

Actores: OPERADOR

Descripción: El sistema localiza los taxis libres más cercanos. El operador selecciona uno de ellos. El sistema informa al taxista presentando un mensaje a su terminal.

Casos de uso de alto nivel (3)

Caso de uso: **Consultar Servicios Pendientes**

Actores: OPERADOR

Descripción: El sistema muestra al Operador el plano de la ciudad, las solicitudes pendientes de asignar y las solicitudes de asignación de servicios de los taxis libres.

Caso de uso: **Consultar Estado Taxis**

Actores: OPERADOR

Descripción: El sistema muestra al Operador sobre el plano de la ciudad la localización de los distintos taxis y su estado: libre, ocupado, fuera de servicio o alarma. Los taxis libres aparecen destacados. También aparecen destacados los taxis en situación de alarma.

Casos de uso de alto nivel (4)

Caso de uso: **Aceptar Servicio**

Actores: TAXISTA

Descripción: El sistema informa al Taxista de que el Operador le ha asignado un servicio. El sistema presenta un mensaje en el terminal del Taxista. El Taxista siempre puede aceptar o rechazar el servicio. Si lo acepta el estado pasa a estar ocupado.

Caso de uso: **Consultar Servicios Pendientes**

Actores: TAXISTA

Descripción: El sistema muestra al Taxista las solicitudes que llevan más de 5 minutos pendientes de asignar.

Casos de uso de alto nivel (5)

Caso de uso: **Solicitar Asignación Servicio**

Actores: TAXISTA

Descripción: Cualquier taxi libre puede solicitar la asignación de un servicio que lleva más de 5 minutos sin asignar.

Caso de uso: **Cambiar Estado**

Actores: TAXISTA

Descripción: El terminal del taxi permite modificar su estado: libre, ocupado o fuera de servicio.

Casos de uso de alto nivel (6)

Caso de uso: **Activar Alarma**

Actores: TAXISTA

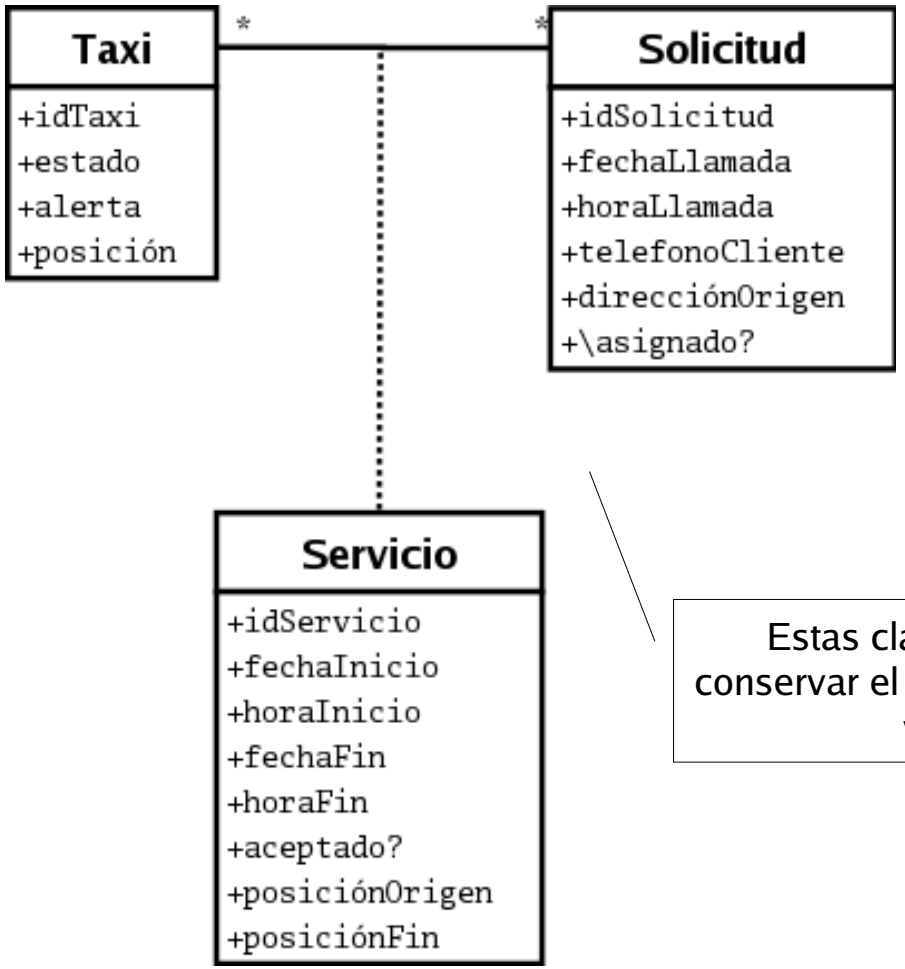
Descripción: El terminal del taxi dispone de un botón de alarma, para situaciones de riesgo. Al ser pulsado, los operadores y los demás taxis verán el mensaje de alerta y conocerán exactamente su posición.

Caso de uso: **Desactivar Alarma**

Actores: TAXISTA, OPERADOR

Descripción: El terminal del taxi y los operadores disponen de una opción para desactivar la alarma.

Modelo de dominio

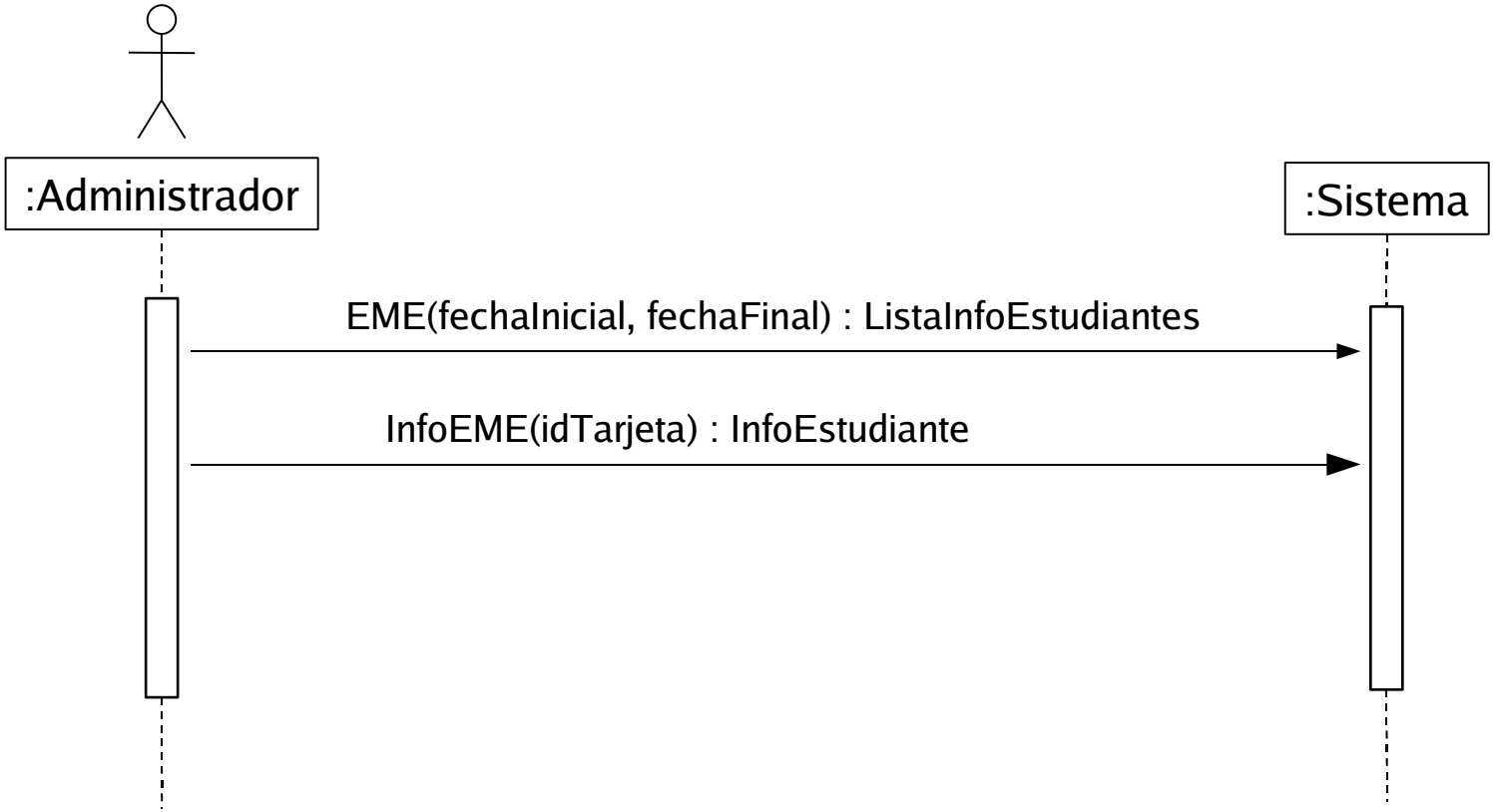


Estas clases nos permiten conservar el historial de solicitudes y servicios

Solución Examen Junio 2007 (b)

- Ejercicio Estudiante más Estudioso (1h 20 min.)
  - Análisis (1,5 puntos): Diagrama Secuencia Sistema + Contratos
  - Diseño (2,5 puntos): Diagramas de Secuencia

Diagrama secuencia sistema



Contrato operación EME

- **Name**:EME(fechaInicio, fechaFin) : ListaInfoEstudiantes
- **Responsabilities**

  Dadas una fecha de inicio y fin de un periodo, obtener los estudiantes ordenados descendentemente por tmd
- **Preconditions**

  Las fechas fechaInicio y fechaFin son válidas, fechaInicio<=fechaFin
- **Postconditions**
- **Salida**

  ListaInfoEstudiantes = Lista(<idUsuario, nombre, idTarjeta, tmd, tt>) donde

  tmd = tiempo medio diario permanencia en la Biblioteca en el periodo

  tt = tiempo total de permanencia en la Biblioteca en el periodo

Contrato operación InfoEstudiante

- **Name**:InfoEstudiante(idTarjeta) : InfoEstudiante
- **Responsabilities**

  Muestra para cada dia del periodo seleccionado, las horas de entrada y salida de la Biblioteca y el tiempo diario total de permanencia en la Biblioteca.
- **Preconditions**

  Se dispone de fechaInicio, fechaFin del periodo

  idTarjeta es válido
- **Postconditions**
- **Salida**

  InfoEstudiante = lista(<fecha, tiempoTotal, lista(<fecha, hora, E/S>)>)

Diagrama de Secuencia EME (1)

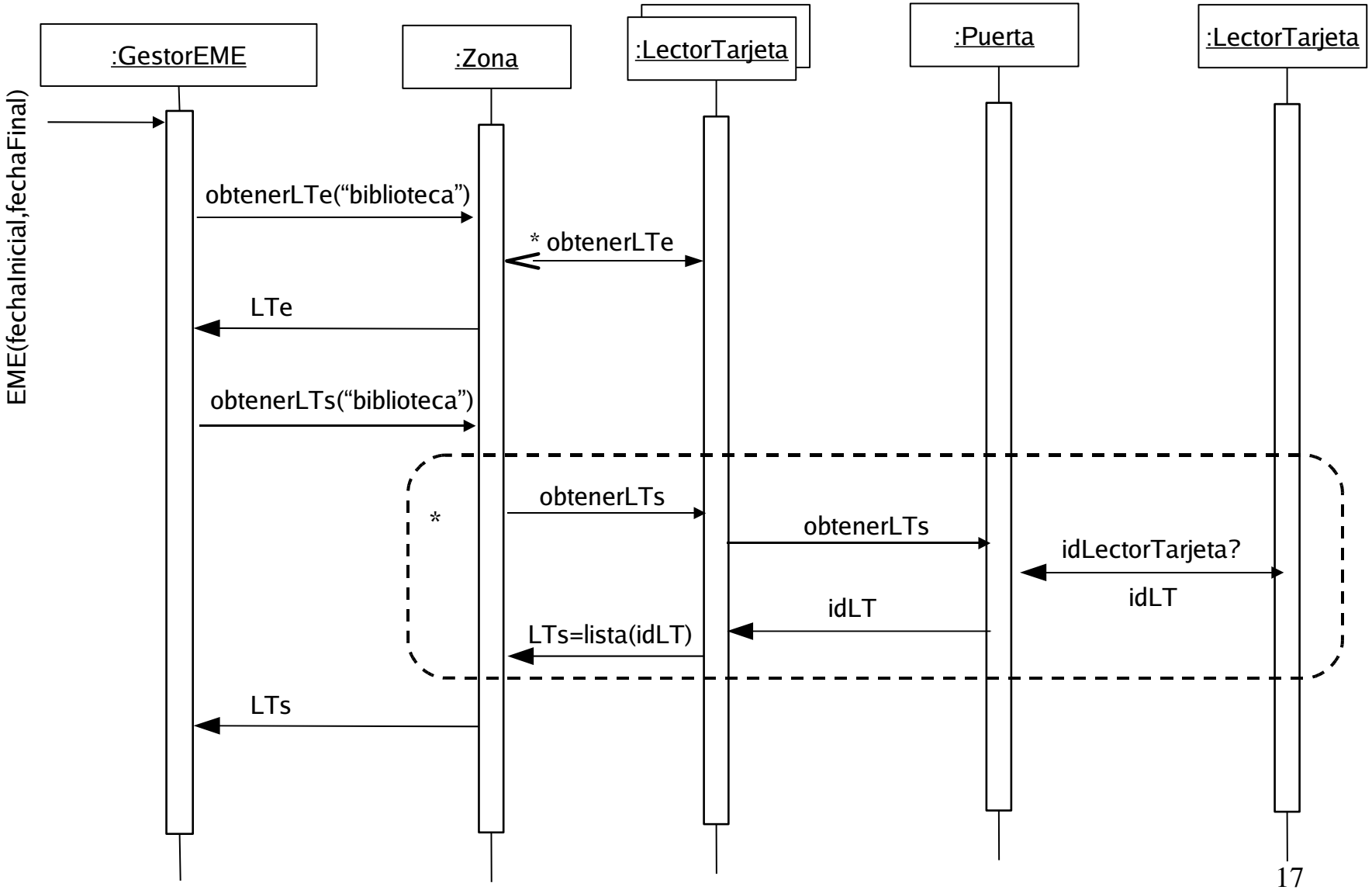
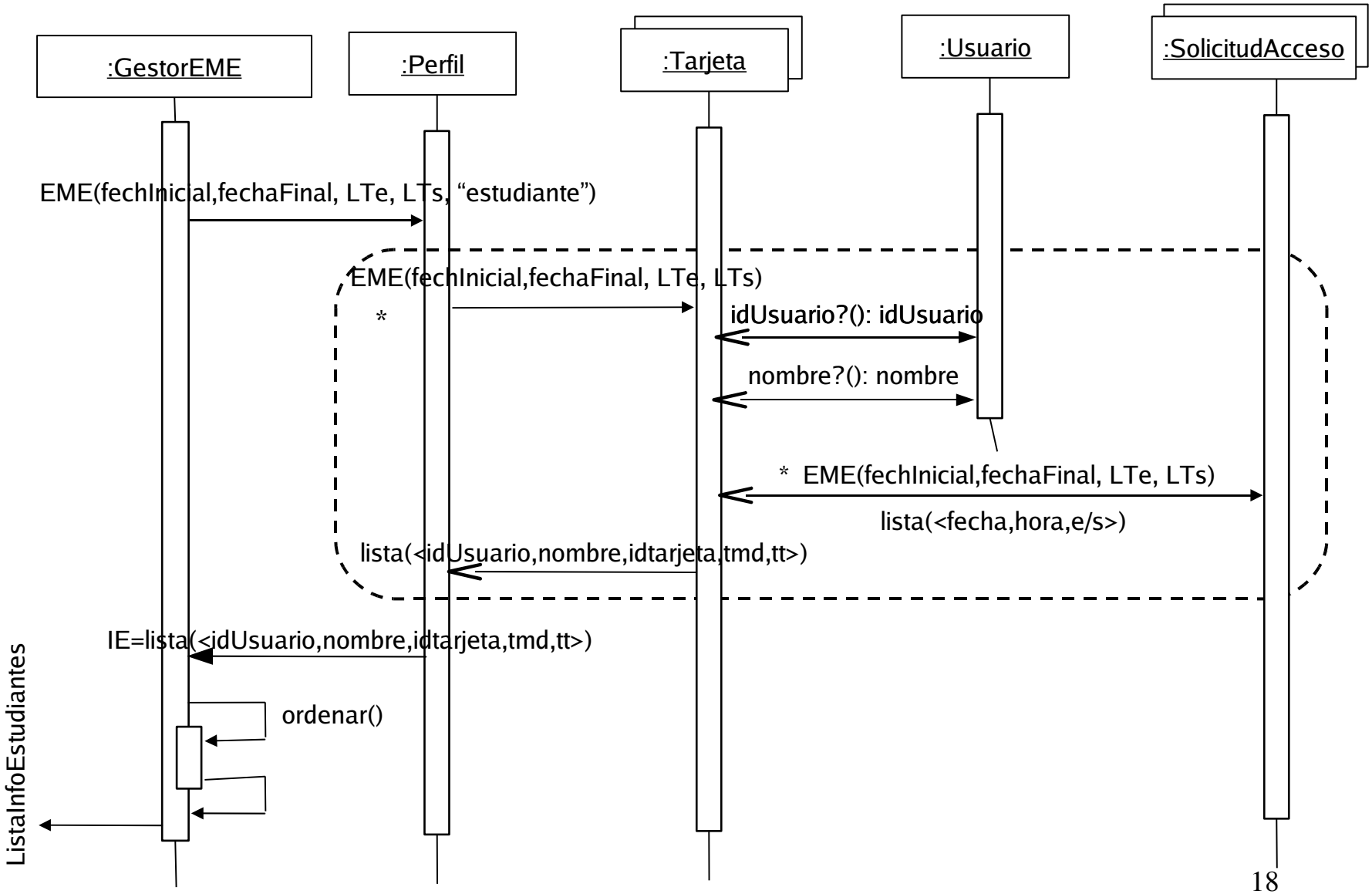


Diagrama de Secuencia EME (2)



1) Escogemos el patrón controlador para gestionar el evento externo EME. Aunque otras opciones son posibles, a falta de más información al tratarse de modelar un caso de uso, seleccionamos el controlador de caso de uso: GestorEME. Además, esta clase artificial tiene acceso directo a todas las zonas y perfiles y gestiona las estructuras auxiliares LTe (Lectores de Tarjeta Entrada) y LTs (Lectores de Tarjeta Salida). Con ello pretendemos un diseño global con alta cohesión y bajo acoplamiento.

2) Por el patrón experto, el método ObtenerLTe de GestorEME selecciona los lectores de tarjeta de entrada a la Biblioteca. Se aplica el mismo patrón experto para ObtenerLTe de Zona.

3) Por el patrón experto, el método ObtenerLTs de GestorEME selecciona los lectores de tarjeta de salida de la Biblioteca. Se aplica el mismo patrón experto para ObtenerLTs de Zona. Como cada Puerta conoce sus lectores de entrada y salida, aplicamos el mismo patrón experto, para a través de Puerta acceder a los identificadores de los Lectores de Tarjeta de Salida.

4) Ambas listas de identificadores de Lectores de Tarjeta de entrada y salida de la Biblioteca son almacenados en LTe y LTs, respectivamente.

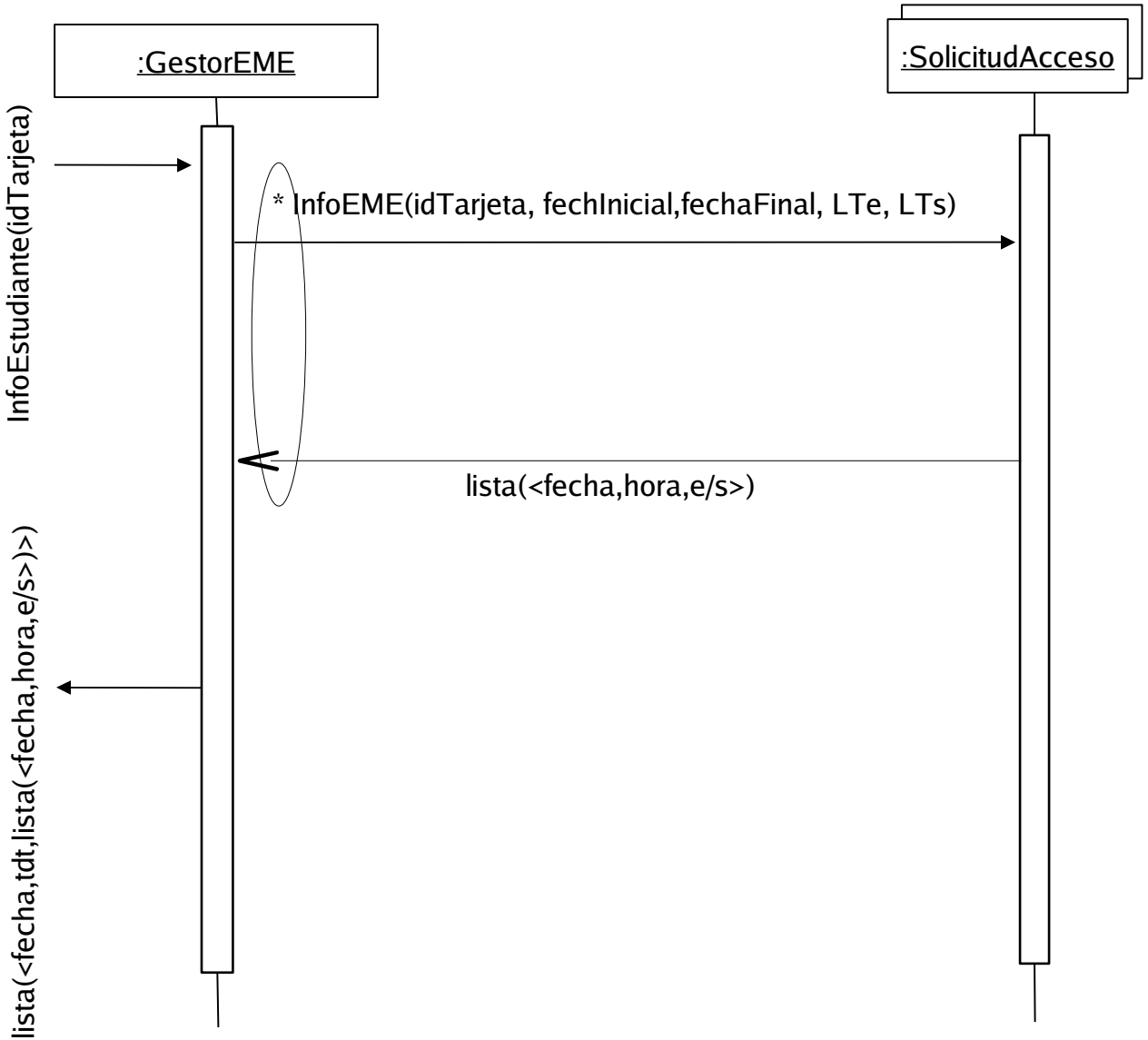
5) Por el patrón experto, el método EME de GestorEME selecciona el perfil estudiante. Ahora tendremos que acceder a Usuario para saber su nombre y recorrer todas las tarjetas de estudiate para obtener todas las Solicitudes de Acceso permitidas entre las fechas solicitadas y que pertenezcan a uno de los identificadores de Lectores de Tarjeta de la Biblioteca (ya sean de entrada o de salida).

6) Por el patrón experto, el método EME de Perfil recorrerá todas las tarjetas de estudiantes. Asímismo, se obtiene el identificador y nombre de cada usuario.

7) Por el patrón experto, el método EME de Tarjeta recorrerá todas las solicitudes de acceso correspondientes a la tarjeta de estudiante. Se obtiene fecha y hora de las que hayan sido permitidas entre fechaInicio y fechaFin cuyo lector de tarjeta corresponda a uno de entrada o salida de la Biblioteca.

8) Por el patrón experto, el método EME de Perfil ordena las horas de entrada y salida y restando la hora de salida a la hora de entrada correspondiente, acumula en tt, para cada tarjeta, el tiempo total de permanencia en la Biblioteca en el periodo y calcula en tmd, el tiempo medio diario permanencia en la Biblioteca en el periodo.

9) Finalmente, se ordenan la información de cada estudiante descendentemente por tmd.

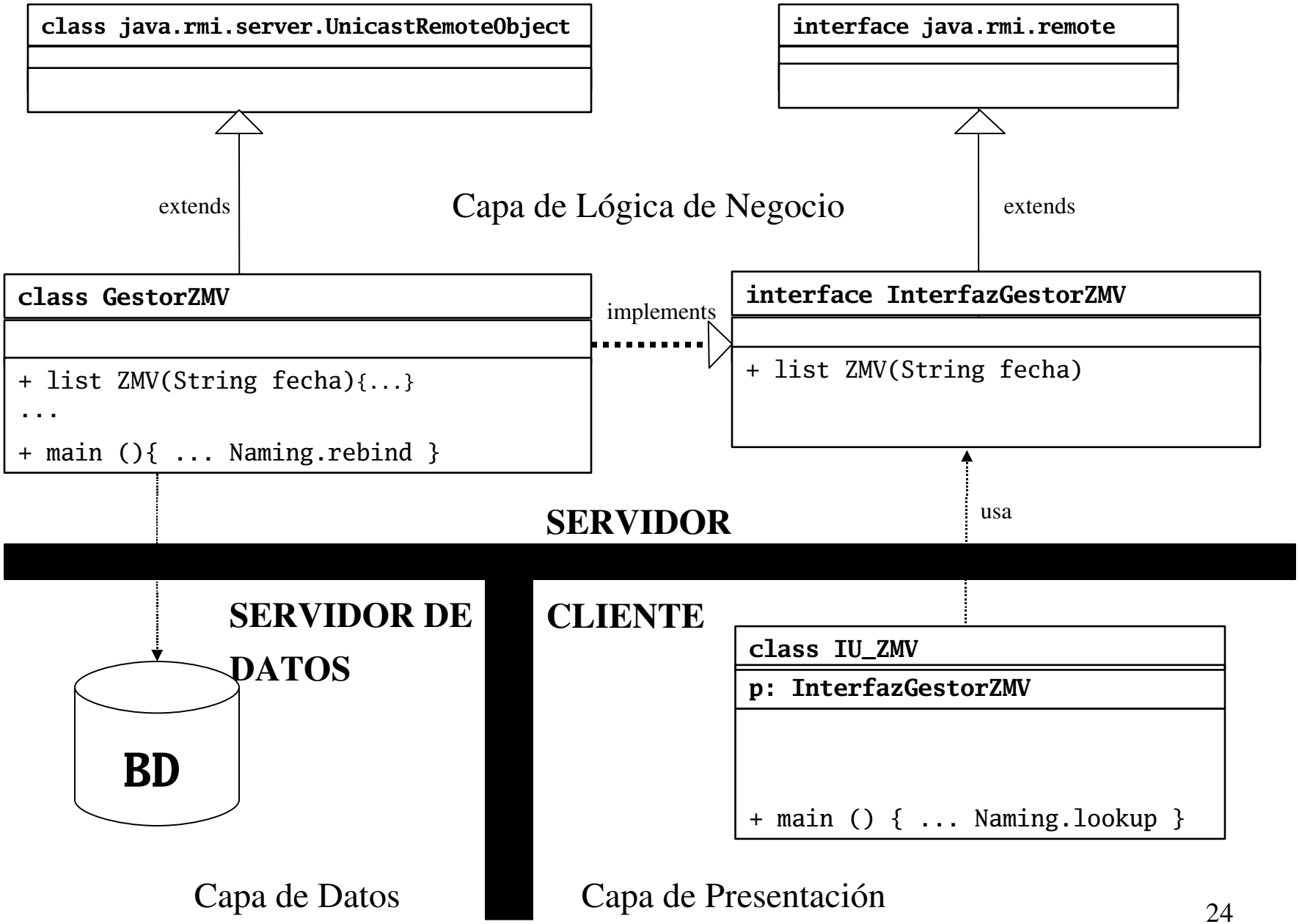

:GestorEME
:SolicitudAcceso
InfoEstudiante(idTarjeta)
* InfoEME(idTarjeta, fechInicial,fechaFinal, LTe, LTs)
lista(<fecha,hora,e/s>)
lista(<fecha,tdt,lista(<fecha,hora,e/s>)>)

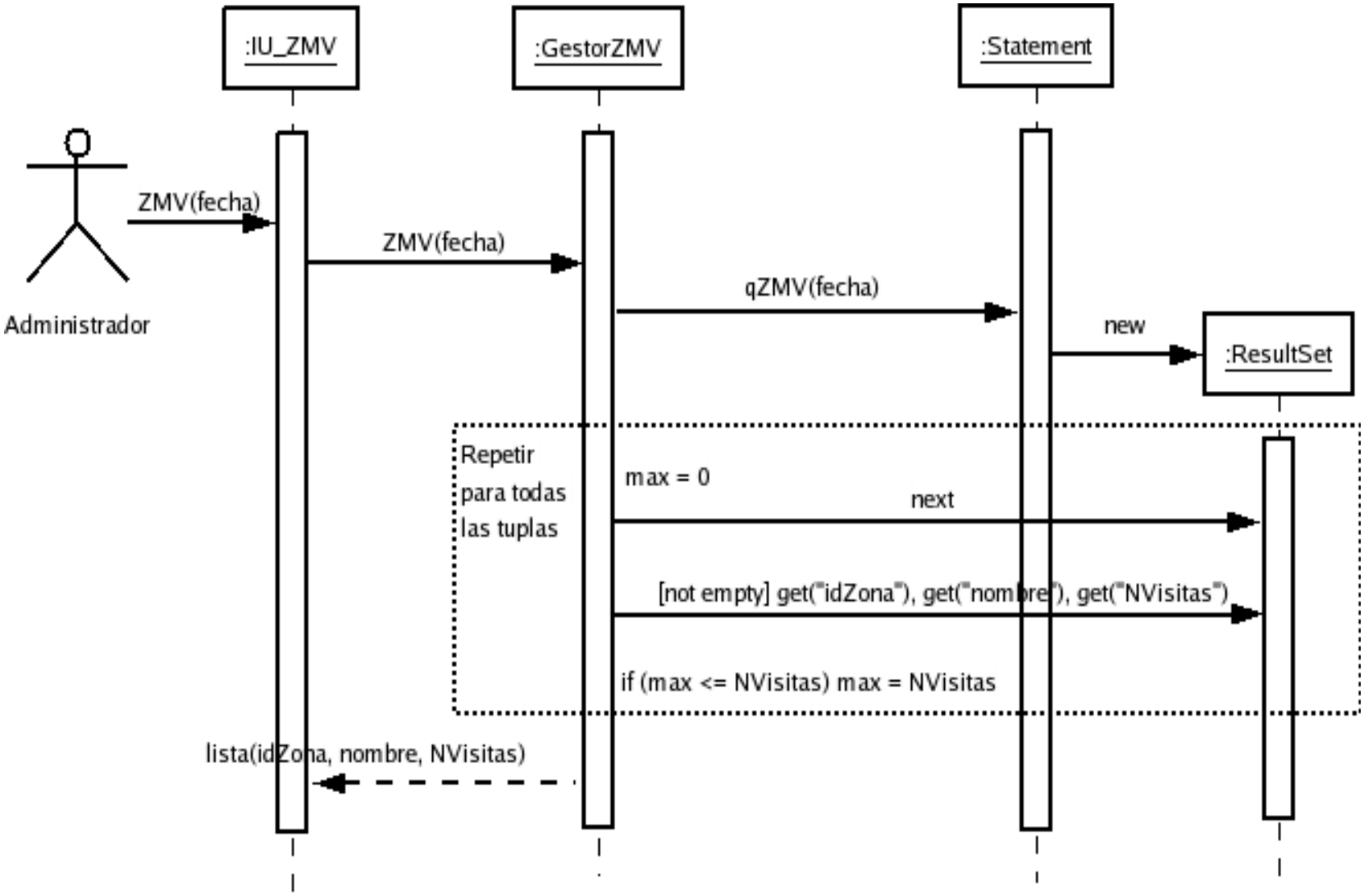
Escogemos el patrón controlador para gestionar el evento externo InfoEstudiante Aunque otras opciones son posibles, a falta de más información al tratarse de modelar un caso de uso, seleccionamos el mismo el controlador de caso de uso que para la operación anterior: GestorEME. Además, esta clase artificial agrupa también todas las Solicitudes de Acceso. Con ello pretendemos un diseño global con alta cohesión y bajo acoplamiento.

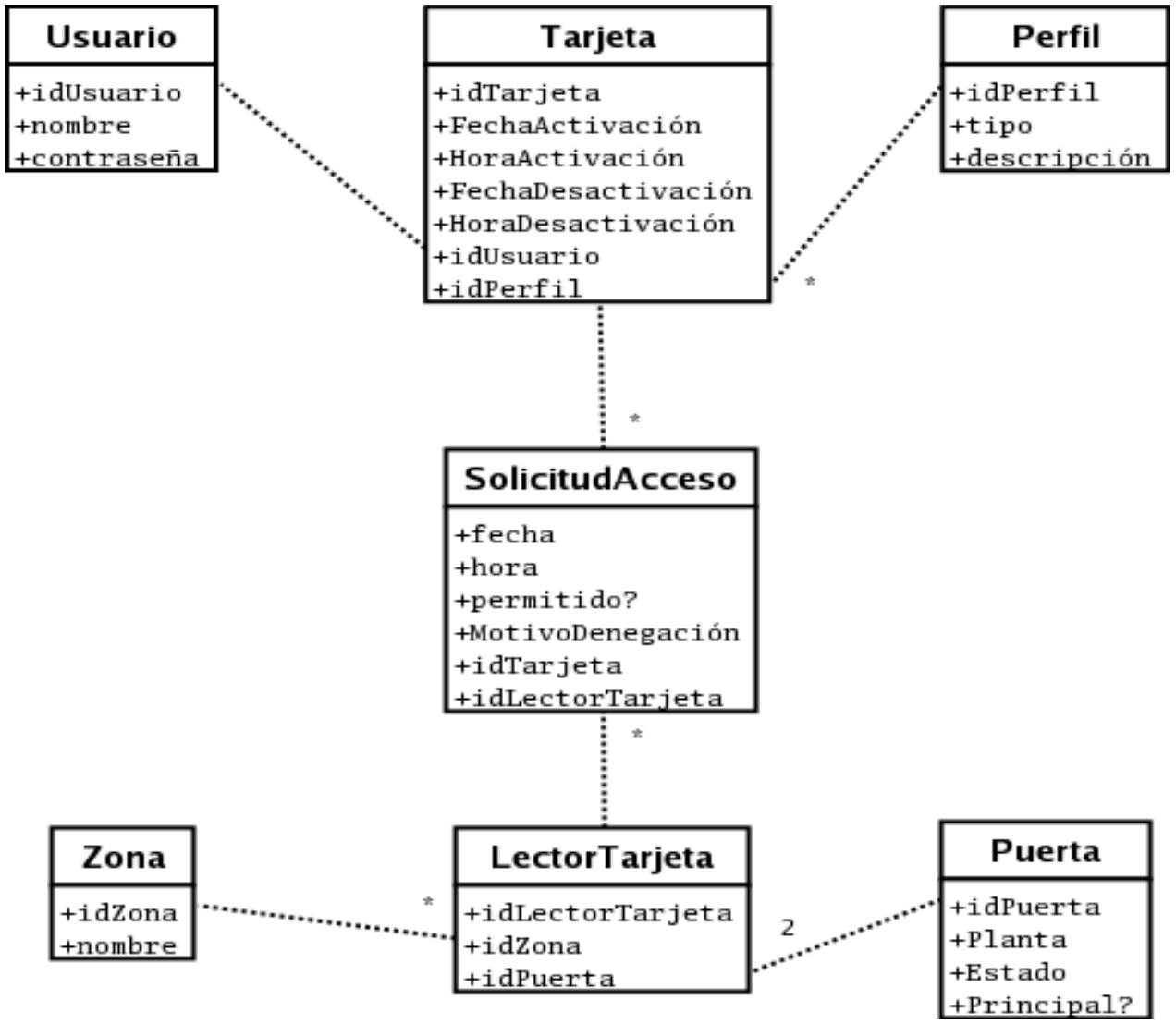
Por el patrón experto, el método InfoEstudiante ordena las horas de entrada y salida, y restando la hora de salida a la hora de entrada correspondiente, acumula en tdt, el tiempo total de permanencia diaria en la Biblioteca.

Solución Examen Junio 2007 (c)

- Ejercicio Implementación (1h.)
  - Realizar el diagrama de clases y la separación física (2 puntos)
  - Completar la consulta SQL qEMV (2 puntos)
  - Implementar el método EMV de la clase Gestor EMV (2 puntos)
  - Responder a las preguntas (4 puntos)


class java.rmi.server.UnicastRemoteObject
interface java.rmi.remote
extends
Capa de Lógica de Negocio
extends
class GestorZMV
implements
interface InterfazGestorZMV
+ list ZMV(String fecha){...}
...
+ main (){ ... Naming.rebind }
+ list ZMV(String fecha)
SERVIDOR
usa
SERVIDOR DE DATOS
CLIENTE
BD
class IU_ZMV
p: InterfazGestorZMV
+ main () { ... Naming.lookup }
Capa de Datos
Capa de Presentación


:IU_ZMV
:GestorZMV
:Statement
Administrador
ZMV(fecha)
ZMV(fecha)
qZMV(fecha)
new
:ResultSet
Repetir
para todas
las tuplas
max = 0
next
[not empty] get("idZona"), get("nombre"), get("NVisitas")
if (max <= NVisitas) max = NVisitas
lista(idZona, nombre, NVisitas)


Usuario
+idUsuario
+nombre
+contraseña
Tarjeta
+idTarjeta
+FechaActivación
+HoraActivación
+FechaDesactivación
+HoraDesactivación
+idUsuario
+idPerfil
Perfil
+idPerfil
+tipo
+descripción
*
*
SolicitudAcceso
+fecha
+hora
+permitido?
+MotivoDenegación
+idTarjeta
+idLectorTarjeta
*
Zona
+idZona
+nombre
*
LectorTarjeta
+idLectorTarjeta
+idZona
+idPuerta
2
Puerta
+idPuerta
+Planta
+Estado
+Principal?

Esta solución distingue las múltiples visitas de un mismo estudiante a una zona en una misma fecha.

```
SELECT Z.idZona,Z.nombre,COUNT(DISTINCT SA.hora) AS NVisitas
FROM zona AS Z
INNER JOIN LectorTarjeta AS LT ON Z.idZona=LT.idZona
INNER JOIN SolicitudAcceso AS SA ON LT.idLectorTarjeta=SA.idLectorTarjeta
INNER JOIN tarjeta as T ON SA.idTarjeta=T.idTarjeta
INNER JOIN perfil AS P ON T.idPerfil=P.idPerfil
WHERE SA.fecha='+fecha+' AND SA.Permitido=1 AND P.tipo="estudiante"
GROUP BY Z.idZona
ORDER BY NVisitas DESC
```

Esta solución NO distingue las múltiples visitas de un mismo estudiante a una zona en una misma fecha (p.e. dos visitas de un mismo estudiante a la biblioteca en una misma fecha sólo cuenta una vez).

```
SELECT Z.idZona,Z.nombre,COUNT(DISTINCT SA.idTarjeta) AS NVisitas
FROM zona AS Z
INNER JOIN LectorTarjeta AS LT ON Z.idZona=LT.idZona
INNER JOIN SolicitudAcceso AS SA ON LT.idLectorTarjeta=SA.idLectorTarjeta
INNER JOIN tarjeta as T ON SA.idTarjeta=T.idTarjeta
INNER JOIN perfil AS P ON T.idPerfil=P.idPerfil
WHERE SA.fecha='+fecha+' AND SA.Permitido=1 AND P.tipo="estudiante"
GROUP BY Z.idZona
ORDER BY NVisitas DESC
```

```
public list ZMV (String fecha) throws RemoteException {
          String SQL = qZMV;
          list l = new LinkedList();
          int max   = 0;
          int NVisitas        = 0;
          boolean continue    = true;
          try {
                    Statement s = c.createStatement(); // c anteriormente definida
                    ResultSet r = s.executeQuery(SQL);
                    while (r.next() && continue) {
                              NVisitas = r.getInt("NVisitas");
                              if (max <= NVisitas) {
                                        list e = new linkedList();
                                        e.add(r.getString("idZona"));
                                        e.add(r.getString("nombre"));
                                        e.add(r.getInt("NVisitas"));
                                        l.add(e);
                                        max = NVisitas;
                                        continue = true
                              } else {
                                        continue = false
                              }
                    }
          }
          catch (Exception ex) {
                    System.out.println(ex.getMessage());
                    list e = new linkedList(); e.add("error"); e.add("null"); e.add(0);
                    l.add(e);
          }
          return l;
}
```

a) ¿Cómo son las instrucciones para cargar el driver del SGBD? ¿Cómo son las instrucciones que establecen la conexión con la base de datos?

Class.forName("org.gjt.mm.mysql.Driver");

Connection c = DriverManager.getConnection("jdbc:mysql://hiper.controla2.es/controla2");

b) ¿Qué instrucciones registran el servicio remoto? ¿En qué método de qué clase se encuentran?

GestorZMV g = new GestorZMV();

Naming.rebind("//localhost:1099/GestorZMV", g);

Se encuentran en el método main de la clase GestorZMV.

c) ¿Qué instrucciones buscan el objeto remoto? ¿En qué método de qué clase se encuentran?

g = (GestorZMV)Naming.lookup("rmi://super.controla2.es:1099/GestorZMV");

Se encuentran en el método main de la clase IU_ZMV.