Introducción a UML 2.0 y Herramientas de Modelado

Código del Curso: CY450

Versión 5.0

Volumen 2

Modelado Avanzado

Modelado Estructural Avanzado

Objetivos de Aprendizaje

Al finalizar esta unidad ud. debería ser capaz de:

- Describir los tipos de clasificadores.
- Explicar acerca de interfaces, roles y paquetes.
- Describir instancias y diagramas de objetos.
- Construir diagramas de objetos.

Clasificadores

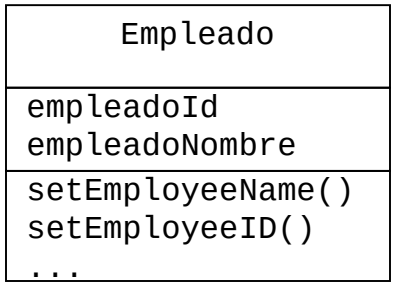
- Un clasificador es un bloque de construcción general de UML.
- Para describir las características estructurales y de comportamiento, el UML usa clasificadores.
- Aquellos elementos del modelo que tienen instancias son llamados clasificadores.
- Las clases son un tipo de clasificadores.

Otros Clasificadores

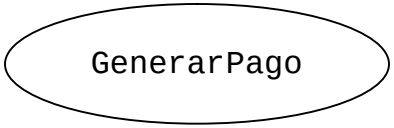
- Interfaz.
- Componente.
- Nodo.
- Caso de uso.
- Subsistema.
- Tipo de Dato.
- Señal.

Notaciones para Clasificadores

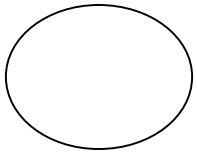
Clase



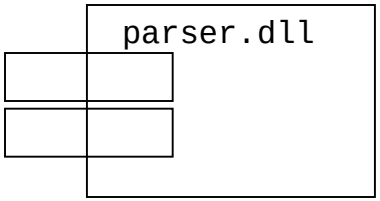
Caso de Uso



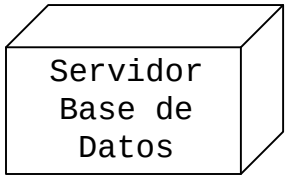
Interfaz



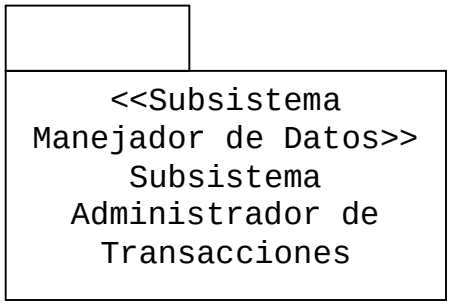
Componente Ejecutable



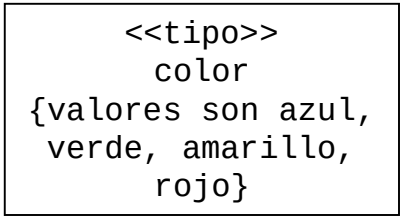
Nodo



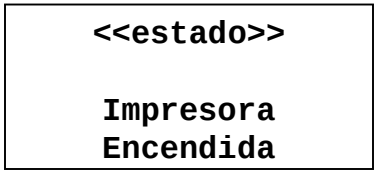
Subsistema



Tipo de Dato



Señal



Elementos Abstractos

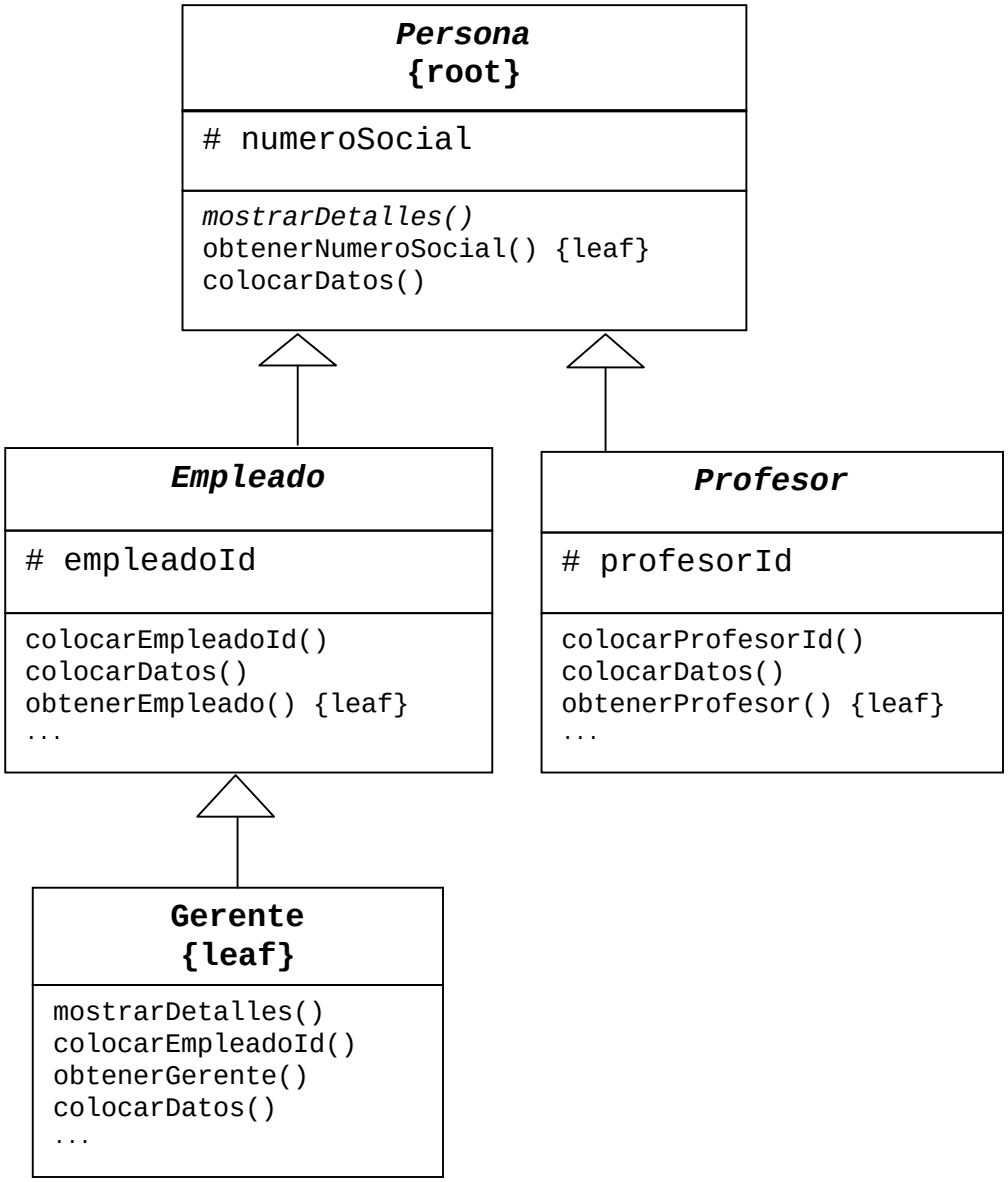
- Existen algunas clases en el modelo cuyas instancias no pueden ser creadas.
- Típicamente, las clases abstractas contienen una o más operaciones que no son implementadas.

Operaciones en Clases Abstractas

Las operaciones en una clase abstracta pueden ser de tres tipos:

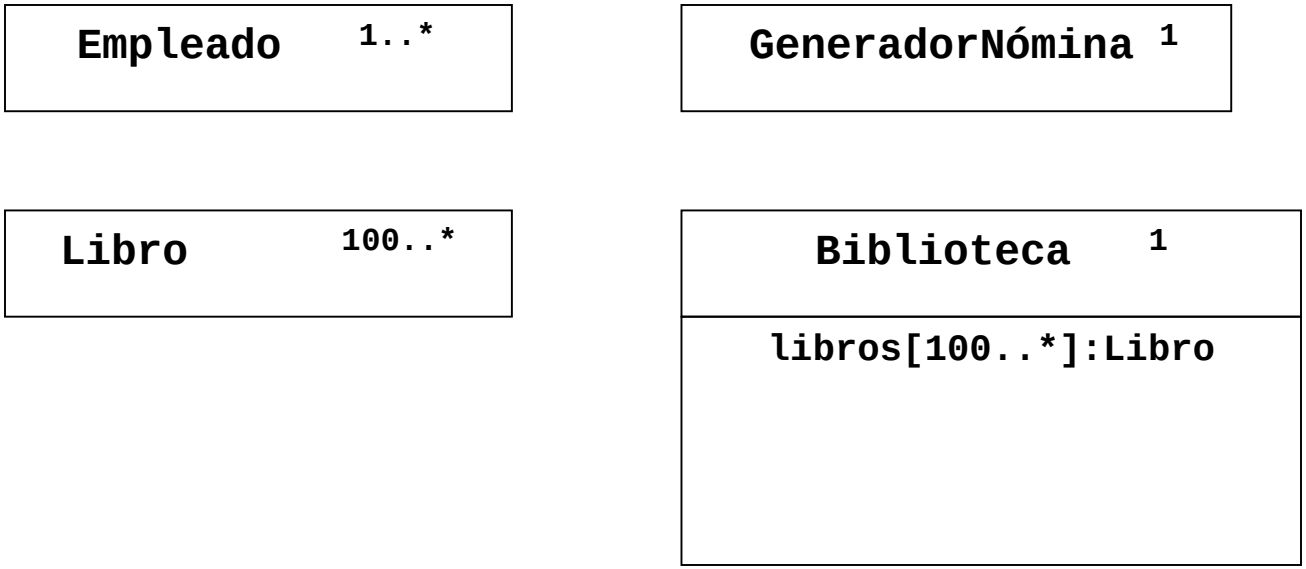
- **Operaciones Abstractas**
  - Deben ser implementadas por una subclase o clase hija.
- **Operaciones Polimórficas**
  - Estas operaciones son aquellas que pueden ser sobreescritas (overriden) en una clase hija.
  - El comportamiento del padre es sustituido en la clase hijo.
- **Operaciones de Hoja {leaf}**
  - Son implementadas en la clase abstracta, la cual puede ser heredada por la clase hija, dependiendo de su visibilidad.

Ejemplos de Elementos Abstractos



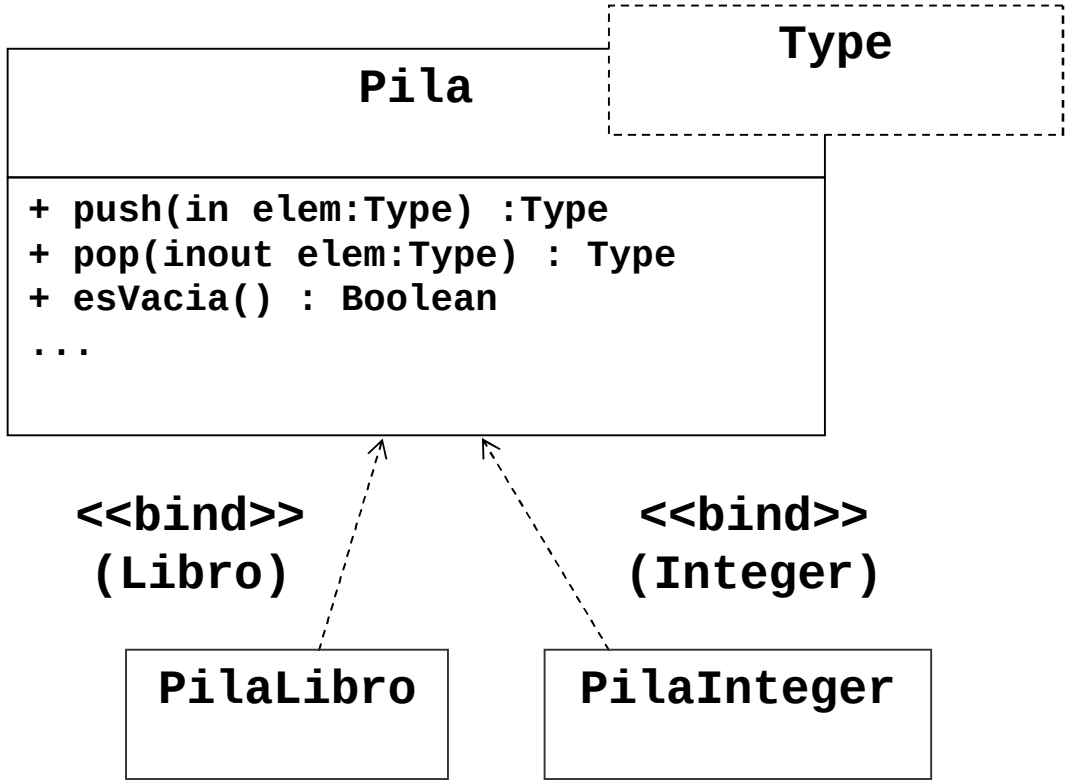
Multiplicidad

- Permite especificar el número de objetos que pueden ser creados para una clase.
- Se especifica la multiplicidad en el extremo derecho contra el nombre de la clase.



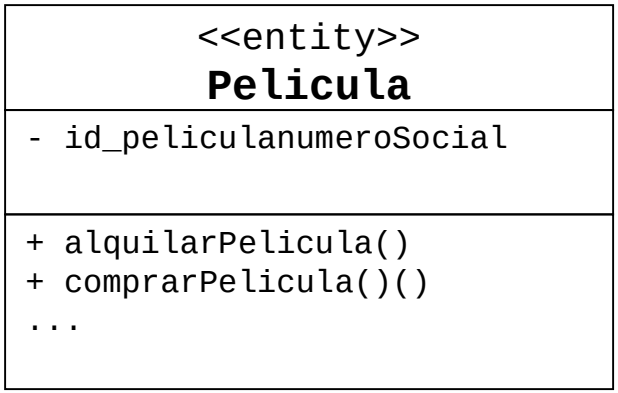
Clases Plantillas (Template)

- Un template o plantilla constituye un elemento parametrizado. Define una familia de clases.
- La clase template se representa por el parámetro dentro de un rectángulo punteado.

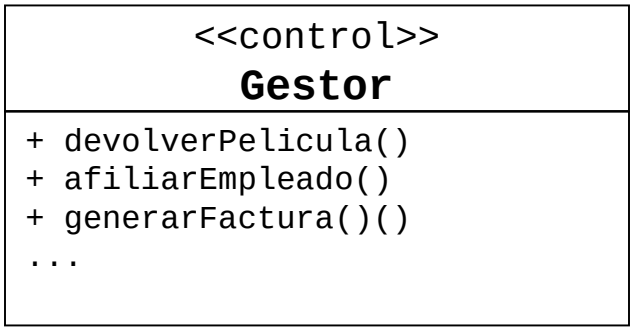


Estereotipos Aplicados a las Clases

- Entity: Solo tiene conocimiento sobre sí mismo y sus relaciones inmediatas.

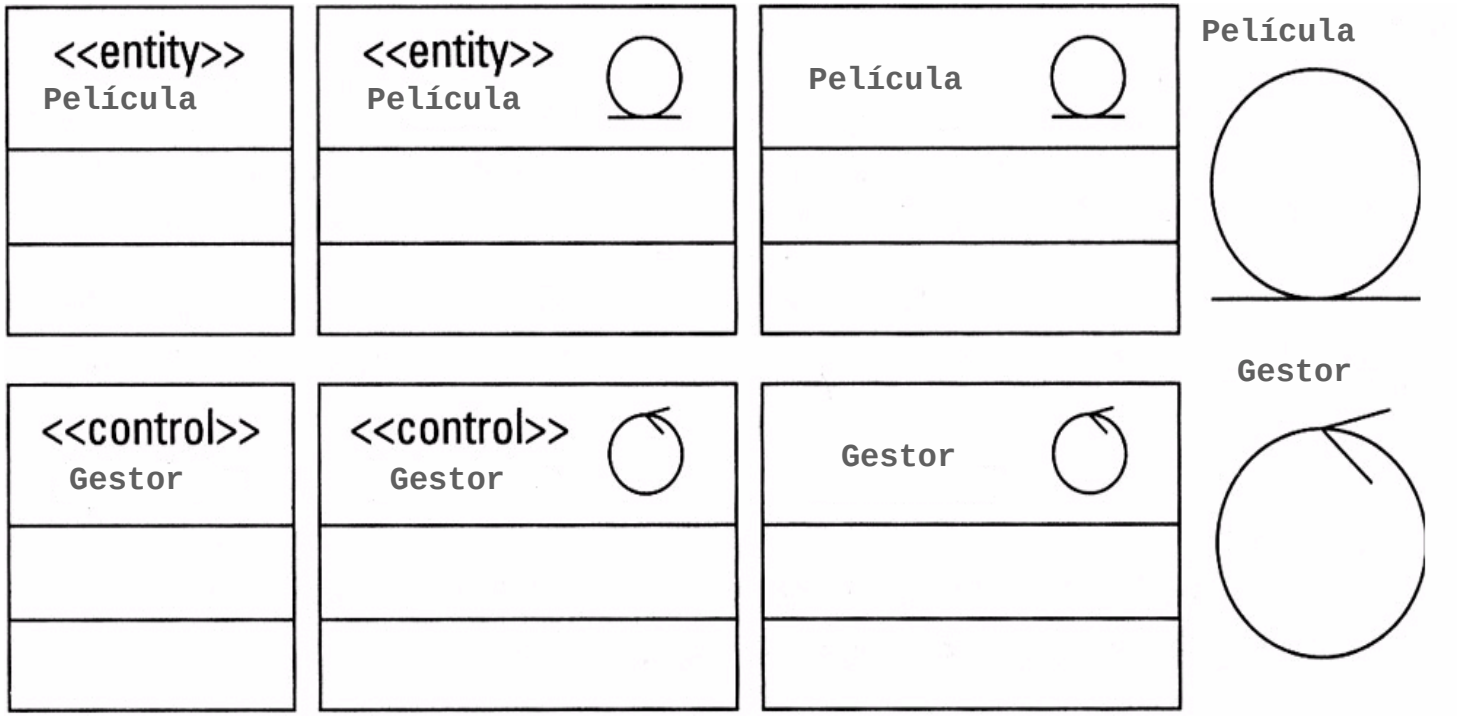


- Control: Representa un comportamiento, más que un recurso.



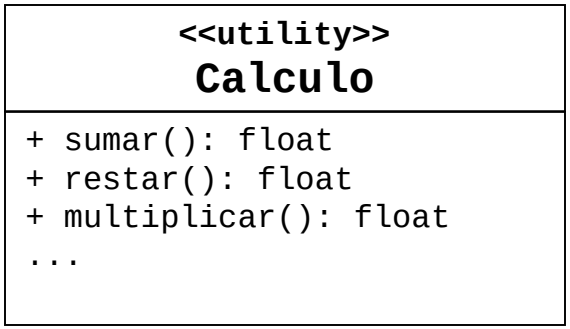
Estereotipos Aplicados a las Clases…1

Distintas maneras de representar los estereotipos <<entity>> y <<control>>

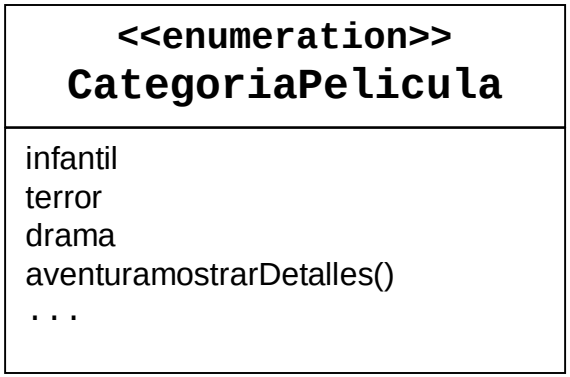


Estereotipos Aplicados a las Clases…2

- Utility: Permite que los demás objetos del sistema accedan a las operaciones y atributos directamente con el nombre de la clase, sin crear instancias de la clase.

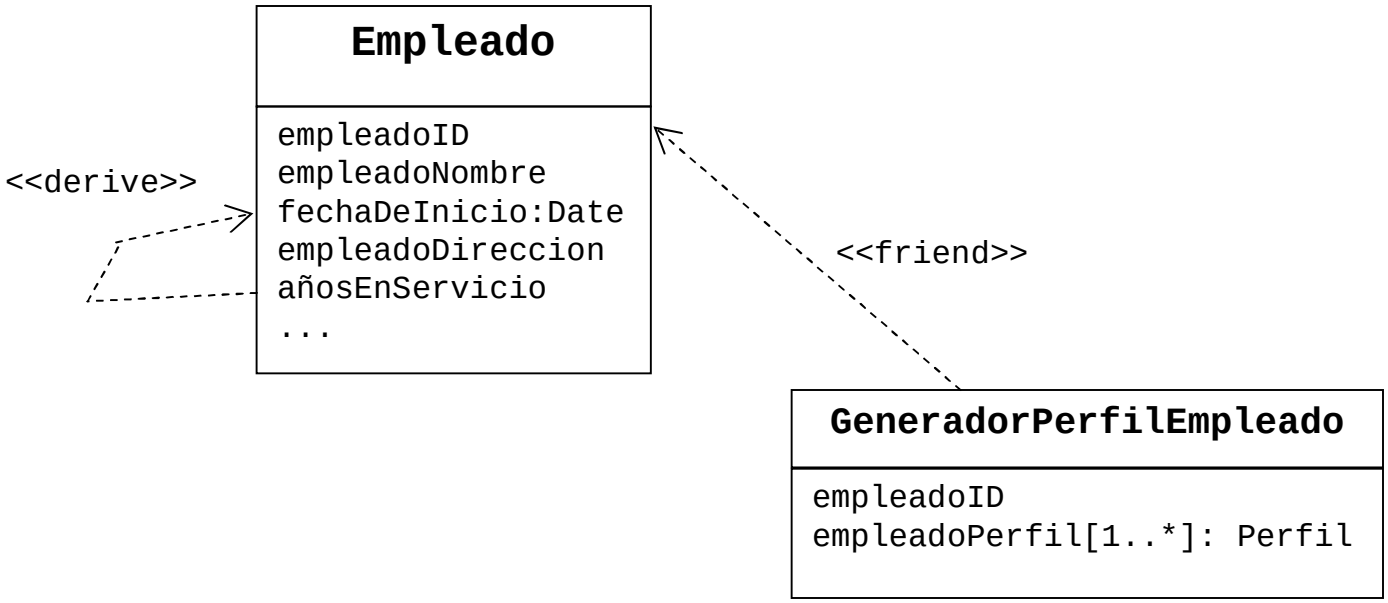


- Enumeration: Son tipos de datos definidos por el usuario, que definen las constantes de un sistema.



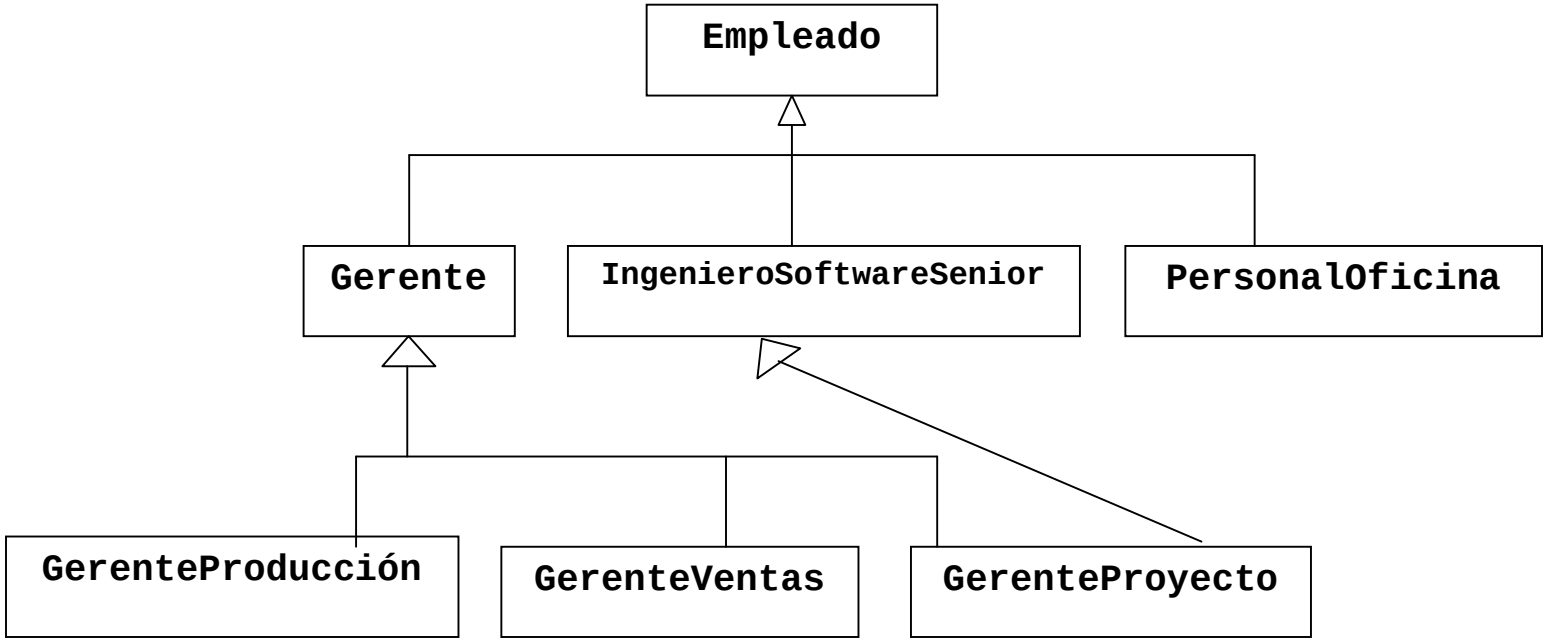
Relación de Dependencia y Estereotipos

- Bind
- Derive
- Friend
- Refine
- Use



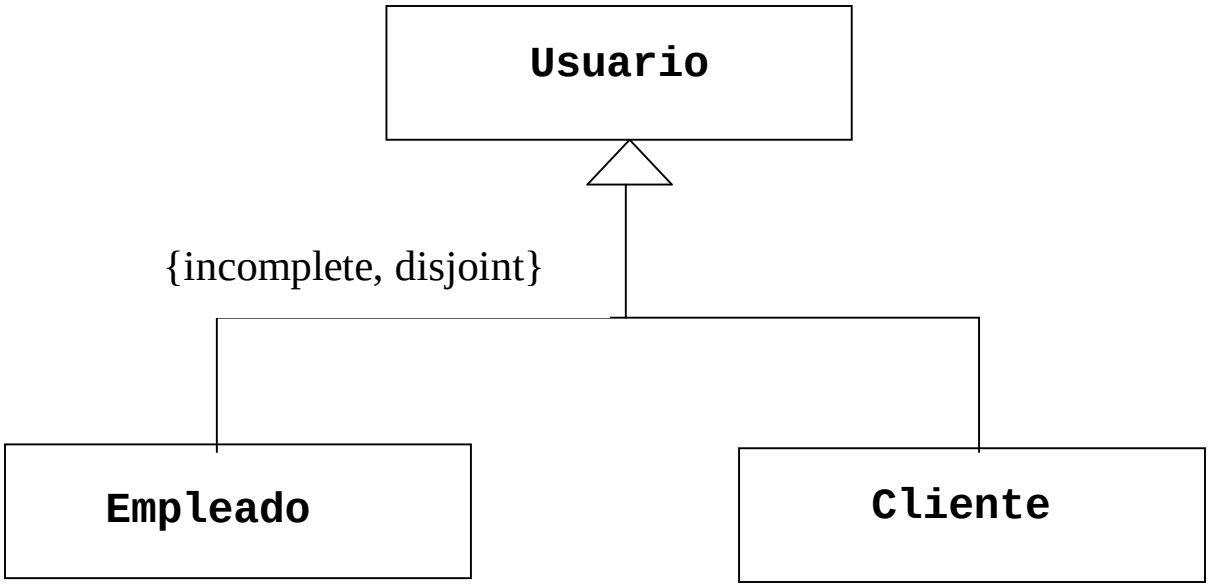
Relación de Generalización

- La jerarquía de herencia especifica la relación “es un” entre la subclase y la súper clase.
- También es posible para una subclase ser una super clase de otra clase.



Relación de Generalización y Estereotipos

- Complete.
- Incomplete.
- Disjoint.
- Overlapping.

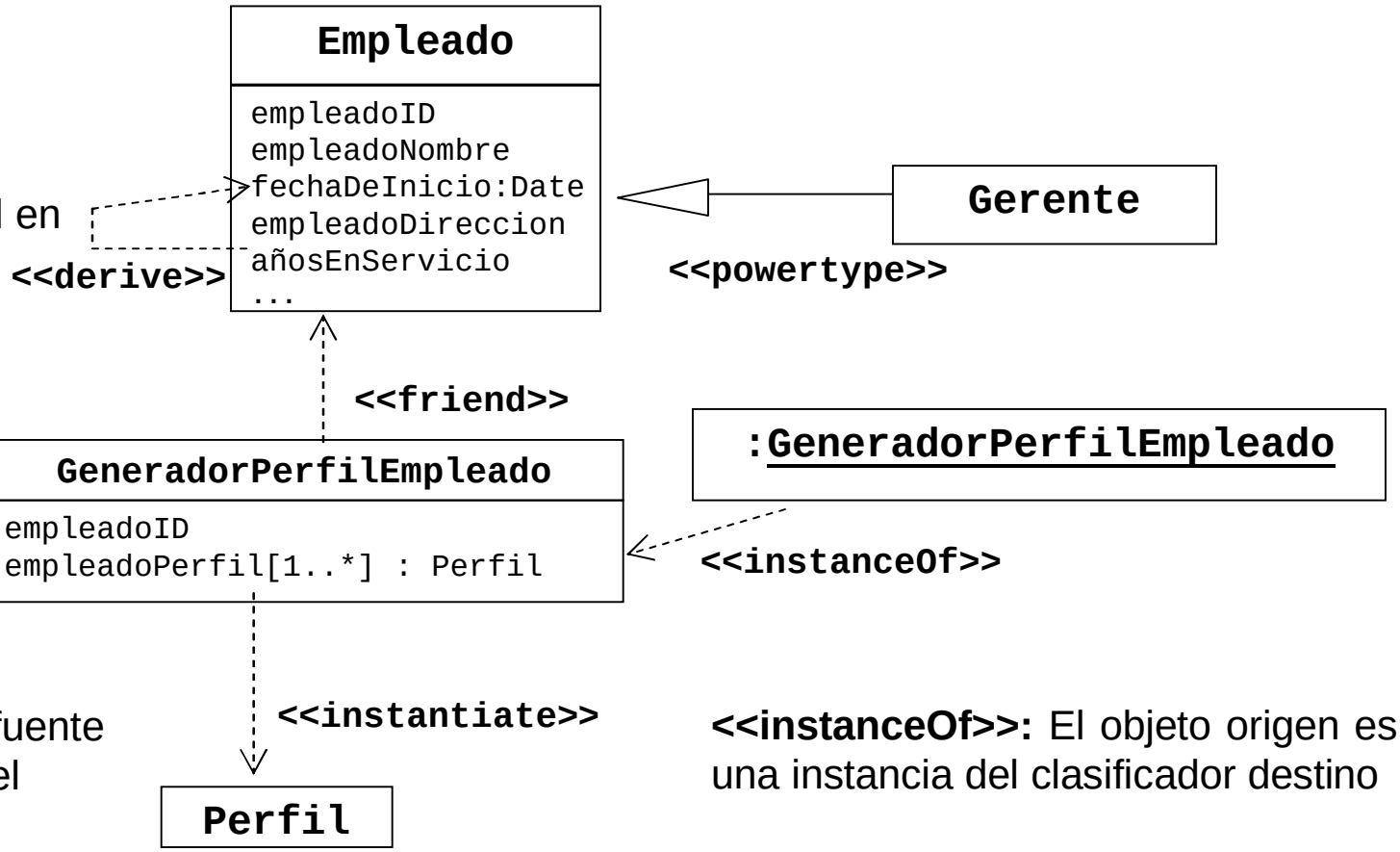


Estereotipos Aplicados a Clases y Objetos

<<derive>>: se usa entre dos atributos o dos asociaciones, la fuente se calcula en base a un valor en el destino

<<powertype>>: especifica un clasificador cuyos objetos son todos hijos de un determinado padre

<<friend>>: proporciona al clasificador origen visibilidad especial en los atributos y operaciones del clasificador destino

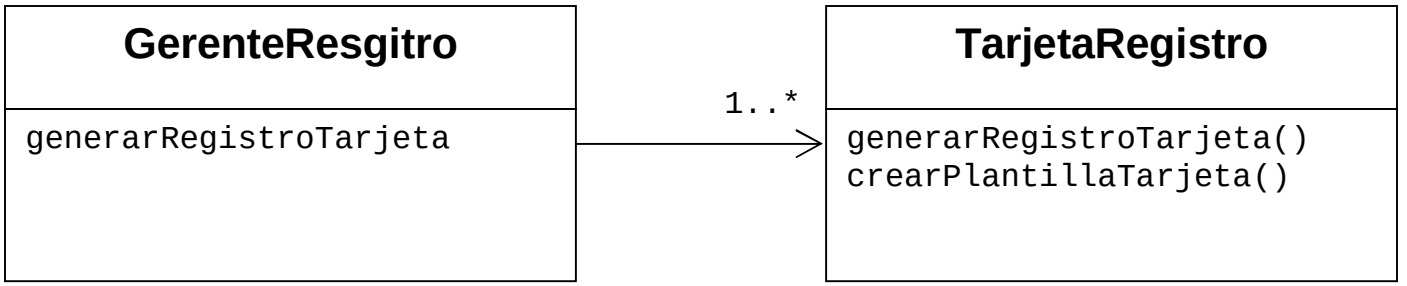


<<instantiate>> establece que la fuente crea instancias del destino

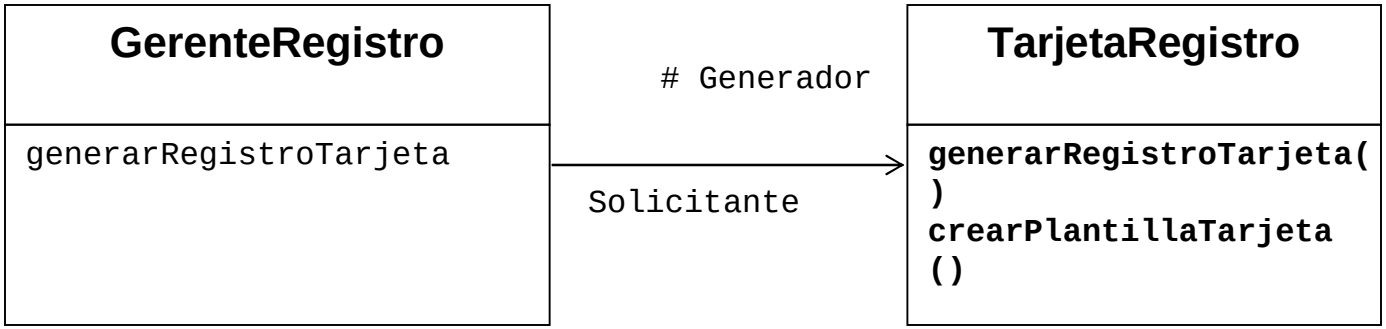
<<instanceOf>>: El objeto origen es una instancia del clasificador destino

Relación de Asociación

- Navegación: Muestra la dirección de la asociación.

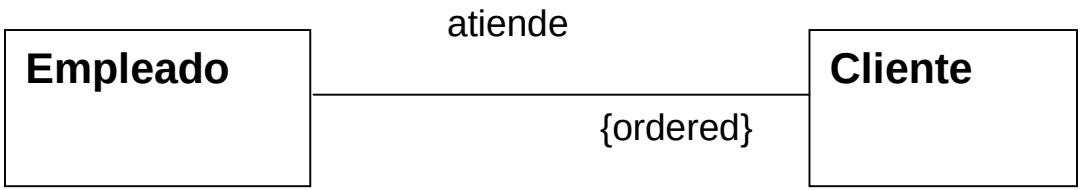


- Visibilidad: Se usa para mostrar si un objeto en una asociación es visible a otros objetos o no.



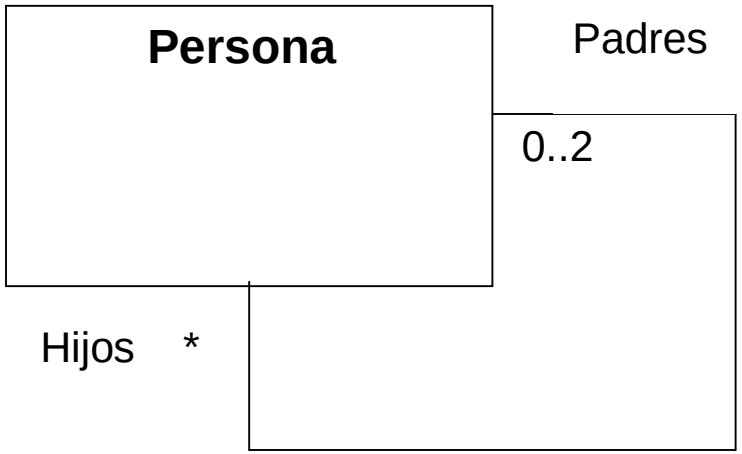
Relación de Asociación y Restricciones

- Ordenación (Ordered)
- Or exclusivo (Exclusive or)
- Subconjunto (subset)



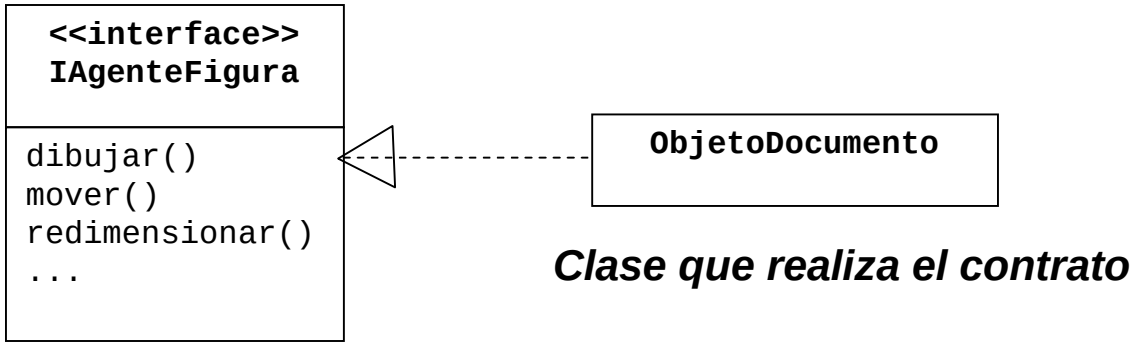
Asociación Reflexiva

Es una clase que tiene una asociación con ella misma.



Interfaces y Realización

- Una interfaz es un conjunto de operaciones que una clase o un componente especifica como su servicio.
- Este es el contrato que la clase o componente necesita llevar a cabo.
- El nombre de la interfaz puede ser un nombre de forma simple o un nombre de ruta. Un nombre de ruta es aquel en el cual la interfaz es prefijada por el nombre del paquete.

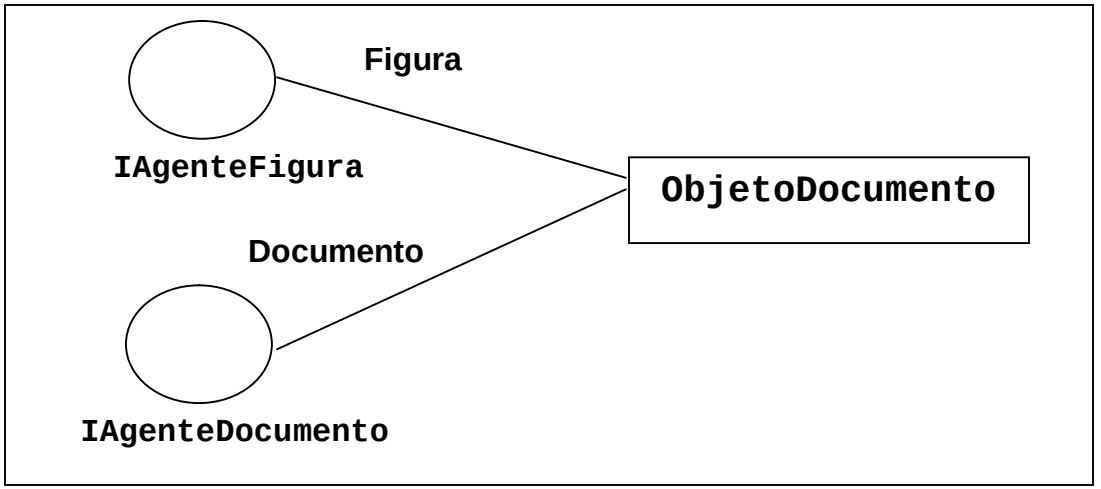


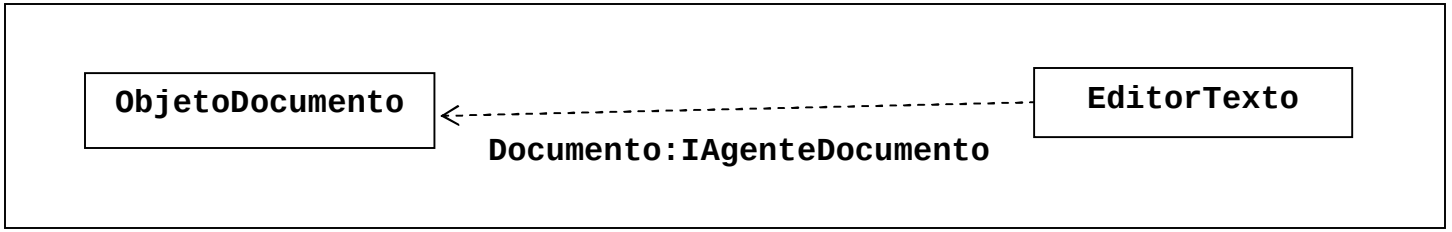
Clase que realiza el contrato

Especifica el Contrato

Roles de las Interfaces

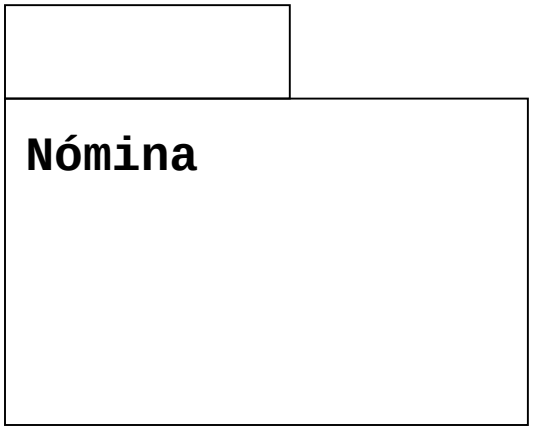
- Una clase puede llevar a cabo más de una interfaz, una instancia de dicha clase soporta el contrato especificado en todas las interfaces que su clase se ha comprometido.
- El Rol es una forma por la cual una abstracción se presenta al mundo.

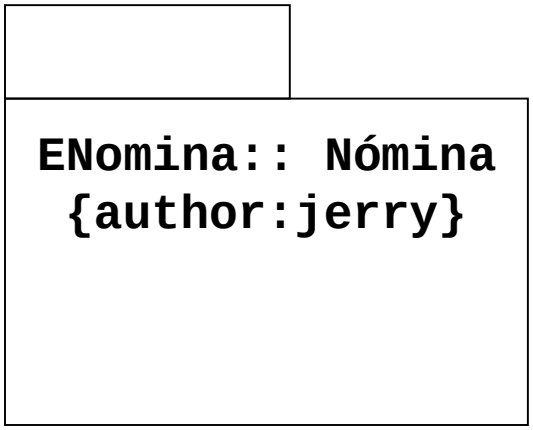




Paquetes

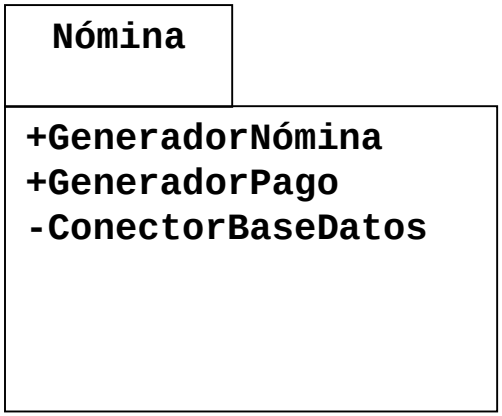
- Es un mecanismo para agrupar elementos. Normalmente los elementos cohesivos están organizados en paquetes.
- Cada paquete tiene un nombre que puede ser Simple o con Nombre de Ruta.
- Los paquetes pueden ser adornados con valores etiquetados y tener compartimientos adicionales.



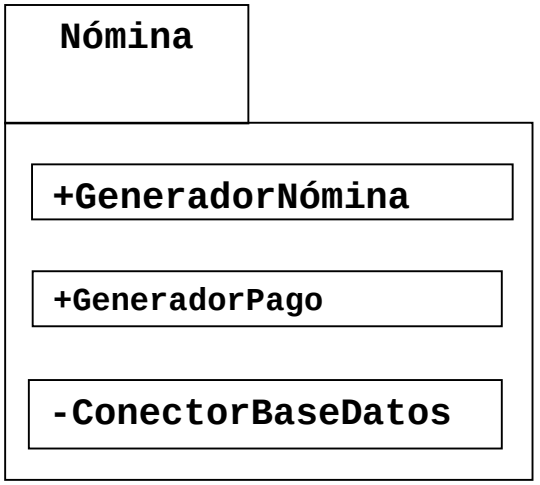


Anidamiento Textual y Gráfico de Paquetes

- Se pueden revelar las clases o componentes dentro del paquete, junto con su visibilidad, en dos formas: **Anidamiento Textual** ó **Anidamiento Gráfico.**



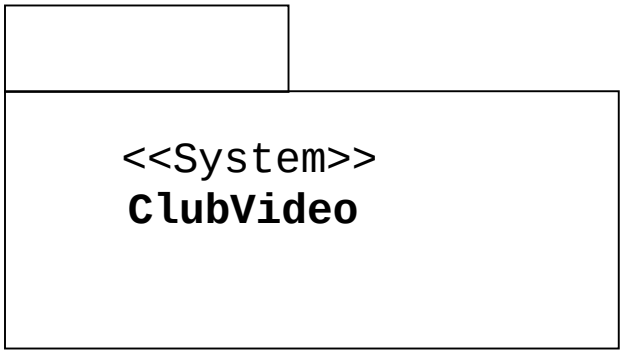
Anidamiento textual

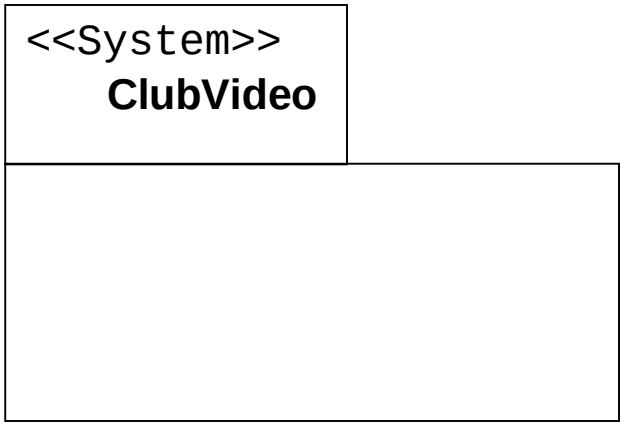


Anidamiento gráfico

Estereotipos de Paquetes

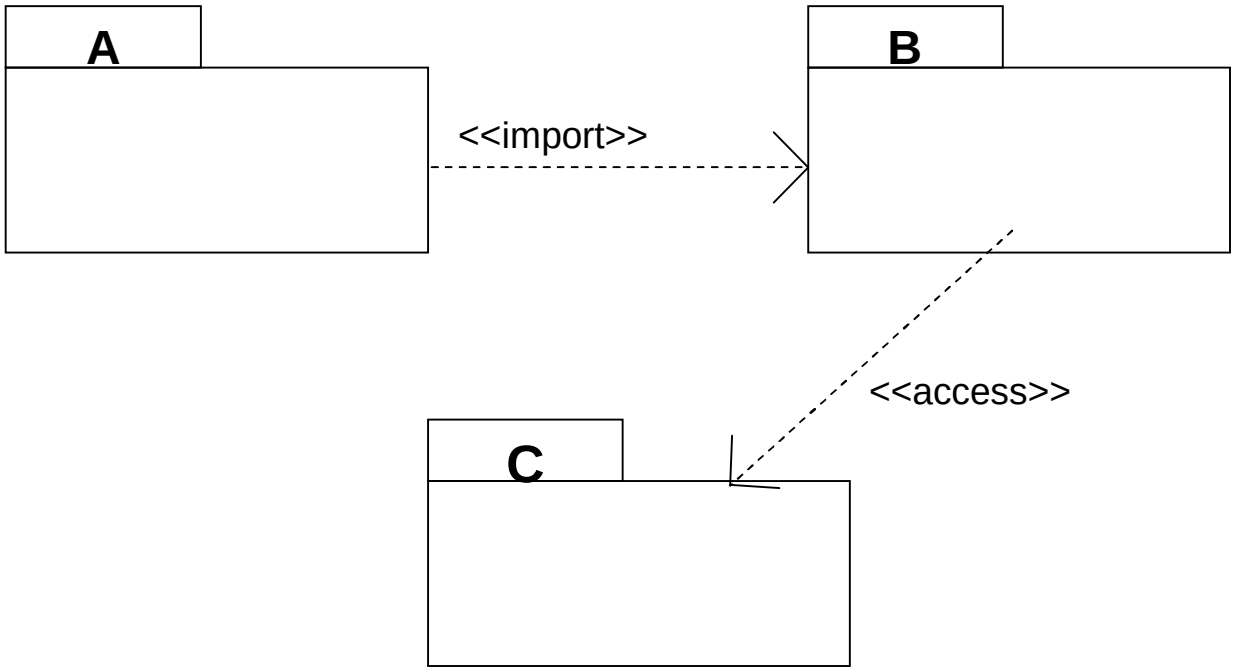
- Facade
- Framework
- Sub-system
- System





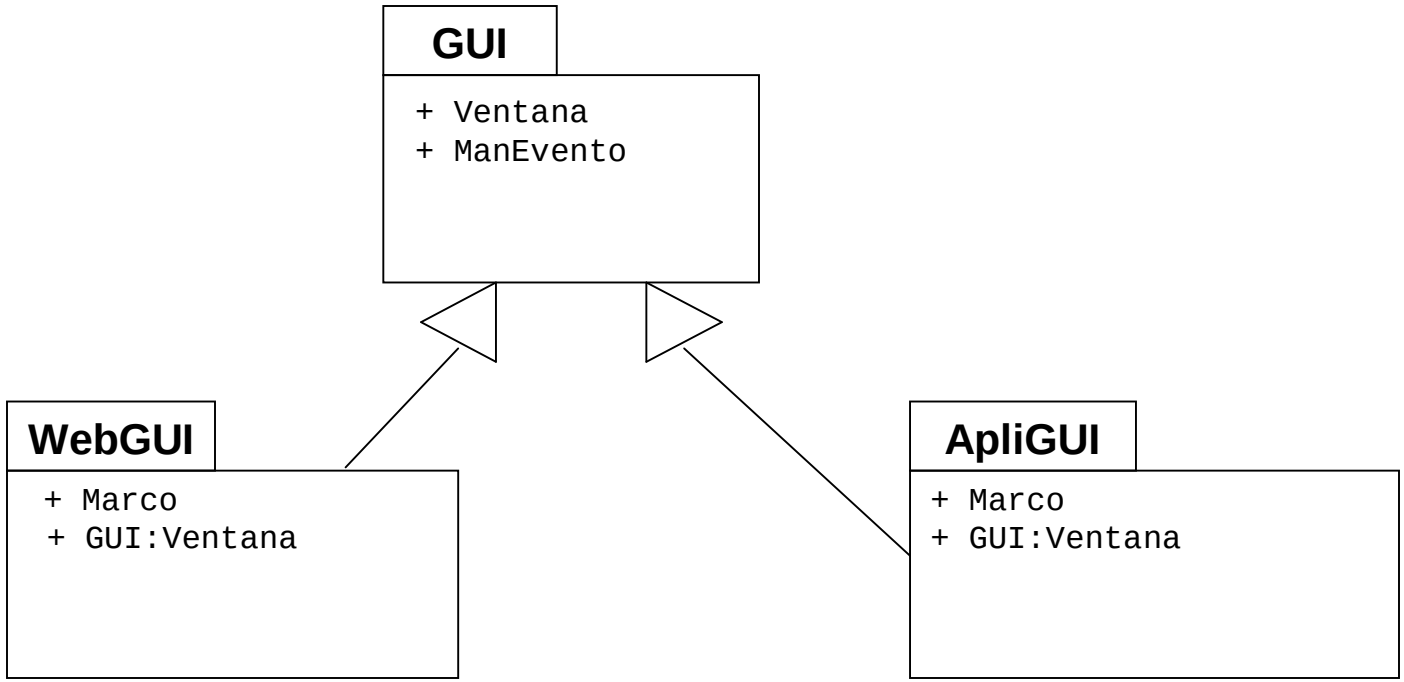
Estereotipos en Relación de Dependencia con Paquetes

- **Import:** Una clase contenida dentro de un paquete puede también aparecer en otro paquete en la forma de un elemento importado.
- **Access:** La dependencia de acceso indica que el contenido del paquete del proveedor puede aparecer en referencias efectuadas por los elementos del paquete cliente.

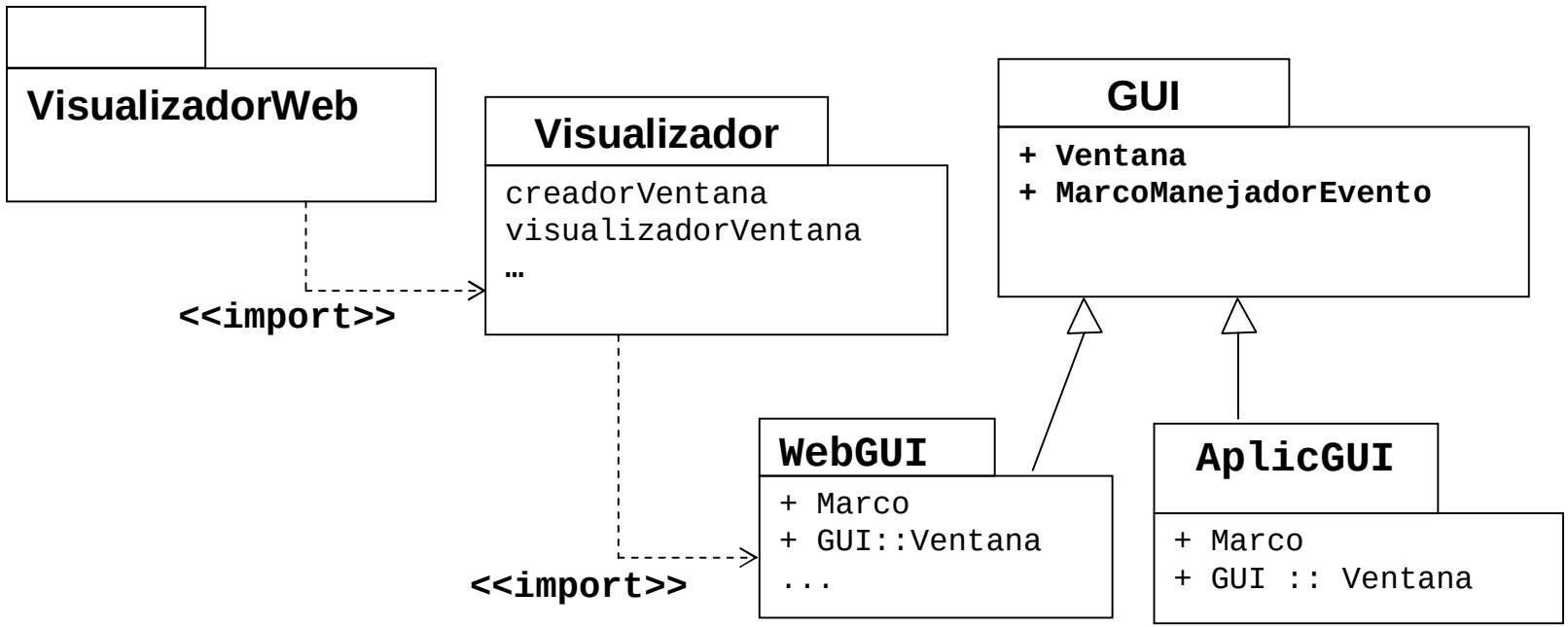


Herencia entre Paquetes

- Los paquetes pueden heredar de un paquete existente. Ellos siguen los mismos principios que las clases.

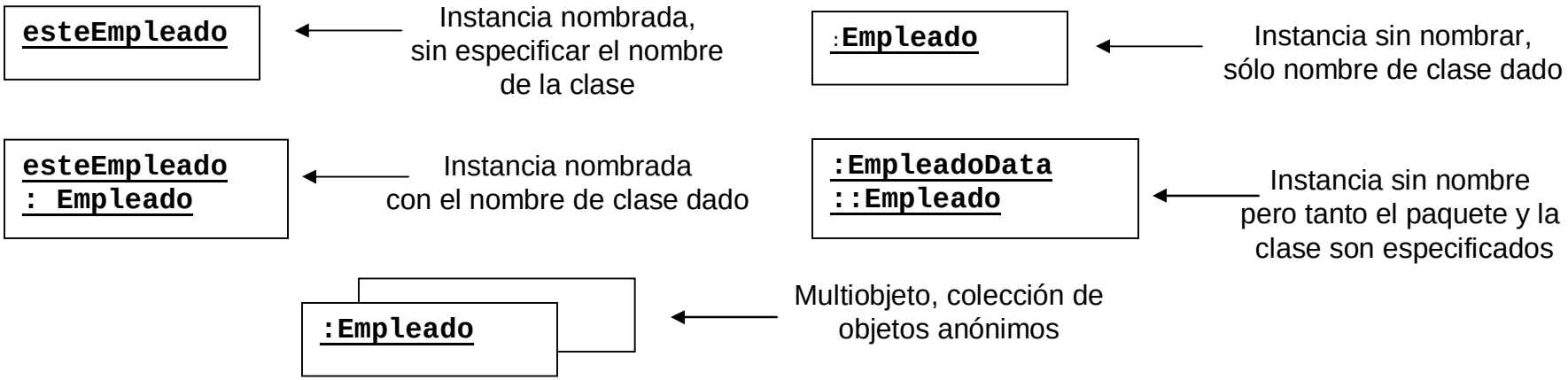


Import, Export y Generalización de Paquetes



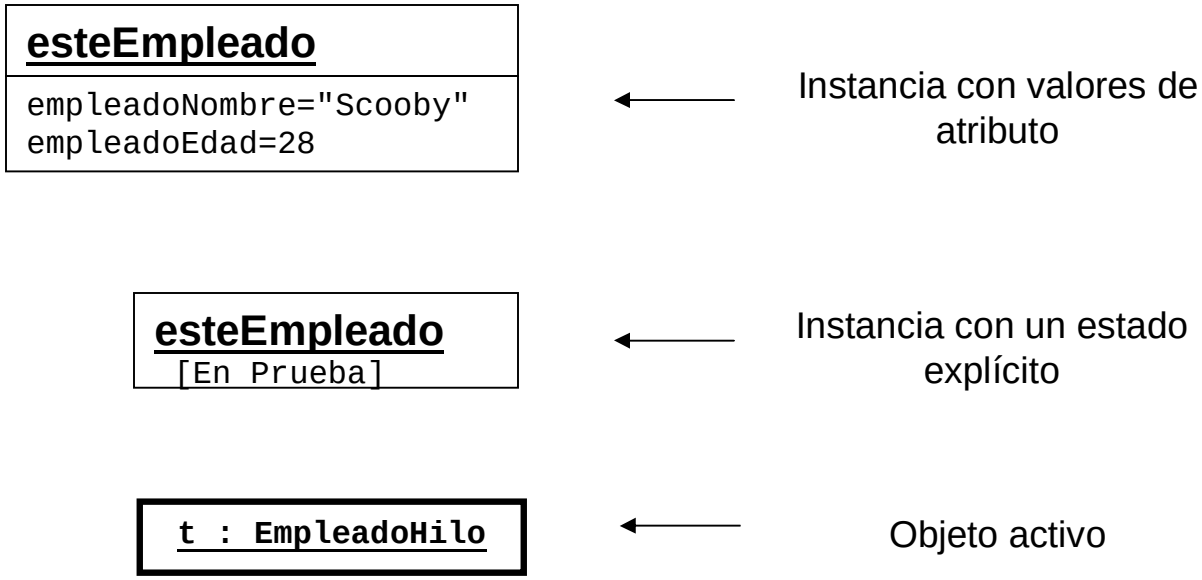
Instancias

- En UML una instancia es descrita como una clase con el nombre subrayado. Los nombres de las instancias pueden ser:
  - **Nombradas**: Proporciona explícitamente su nombre.
  - **Anónima**: No hay un nombre explícito para un objeto, sólo se da el nombre de la clase o paquete.
  - **Huérfana**: Es una instancia cuya abstracción no es conocida, es útil cuando los objetos se cargan dinámicamente en memoria.



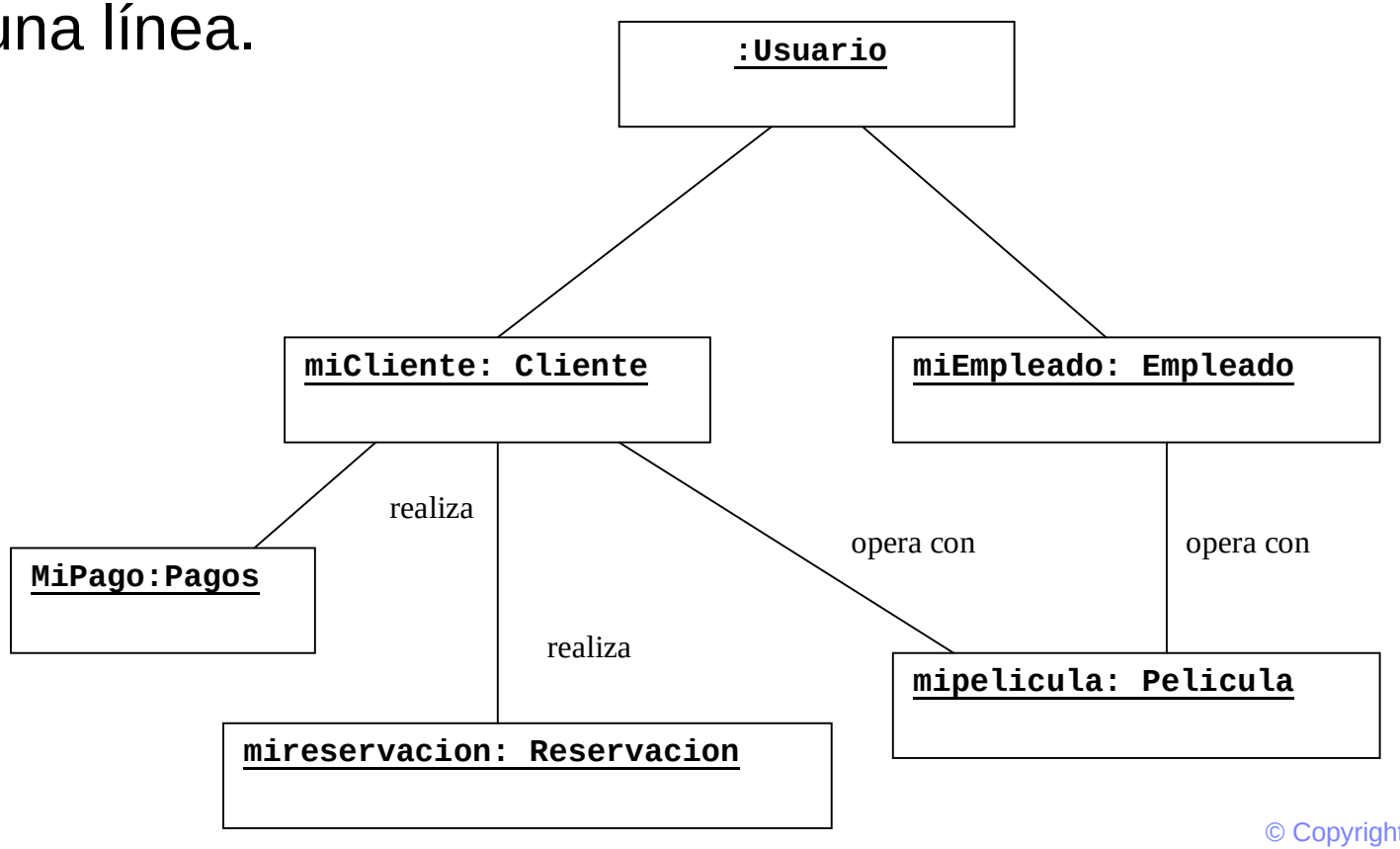
Objeto Activo

- **Objeto activo** es aquel que tiene su propio flujo de control, por ejemplo los hilos y los procesos.
- Se representa igual que un objeto pero con el contorno del rectángulo oscuro.



Diagramas de Objetos

- Un Diagrama de Objetos ilustra un conjunto de objetos y sus relaciones para un contexto o escenario dado.
- Un enlace es la conexión semántica entre objetos, es una instancia de una asociación y se muestra como una línea.



Caso de Estudio: Club de Video

Diagrama de clases que modela como se reserva una película.

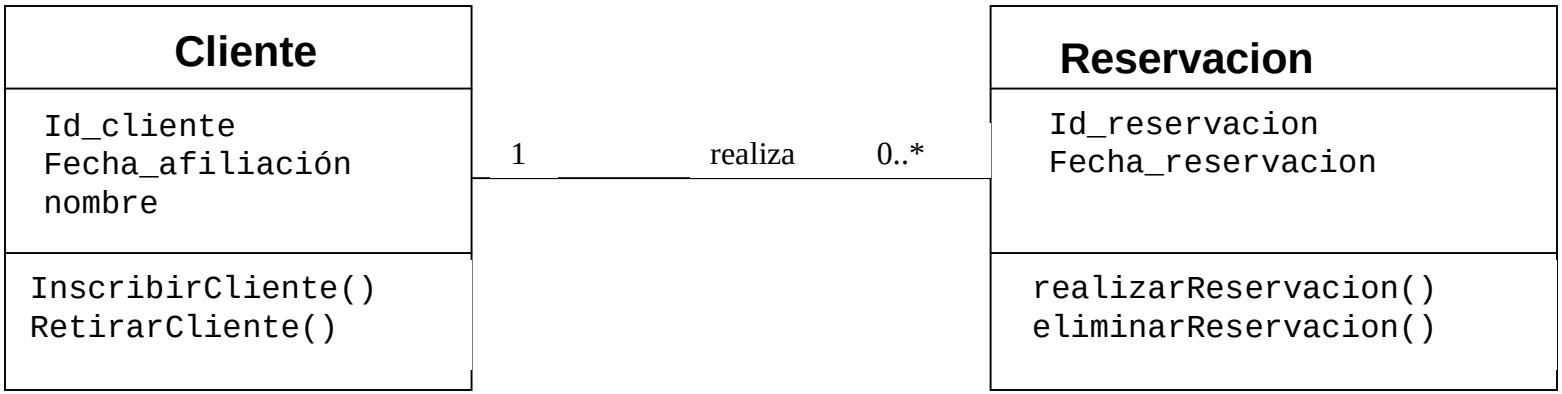
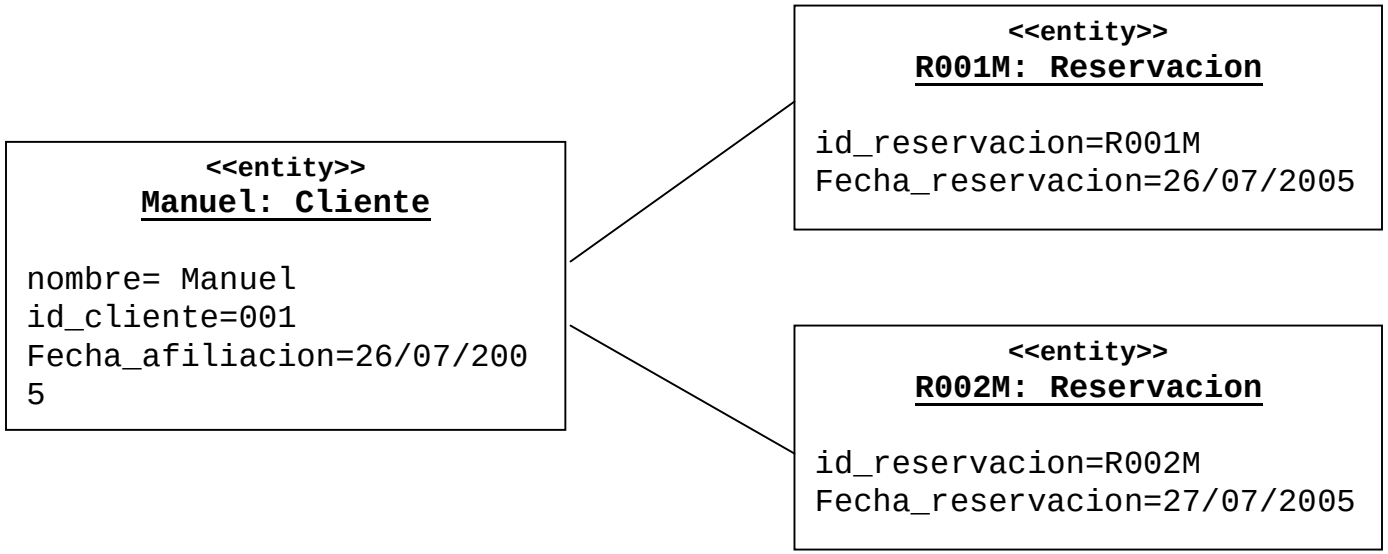


Diagrama de objeto correspondiente al diagrama de clase anterior



Resumen

Ahora que ud. ha completado esta unidad, debe ser capaz de:

- Describir los tipos de clasificadores.
- Explicar acerca de interfaces, roles y paquetes.
- Describir instancias y diagramas de objetos.
- Construir diagramas de objetos.

Modelado de Interacción

Objetivos de Aprendizaje

Al finalizar esta unidad ud. debería ser capaz de:

- Definir colaboraciones.
- Definir interacciones.
- Elaborar diagramas de secuencia.
- Elaborar diagramas de colaboración.
- Describir pasos para elaborar diagramas de colaboración.

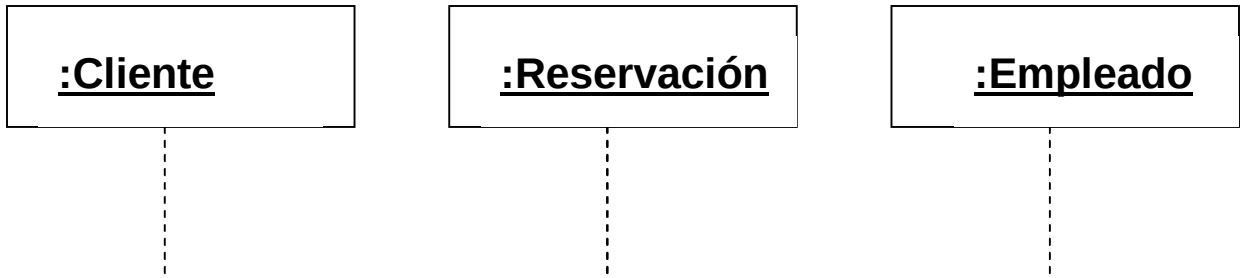
Diagramas de Interacción

- Un diagrama de interacción consiste de:
  - Conjunto de objetos.
  - Relación o enlaces entre objetos.
  - Mensajes entre objetos.

- Tipos de diagramas de interacción:
  - Diagramas de secuencia.
  - Diagramas de colaboración.

Diagramas de Secuencia

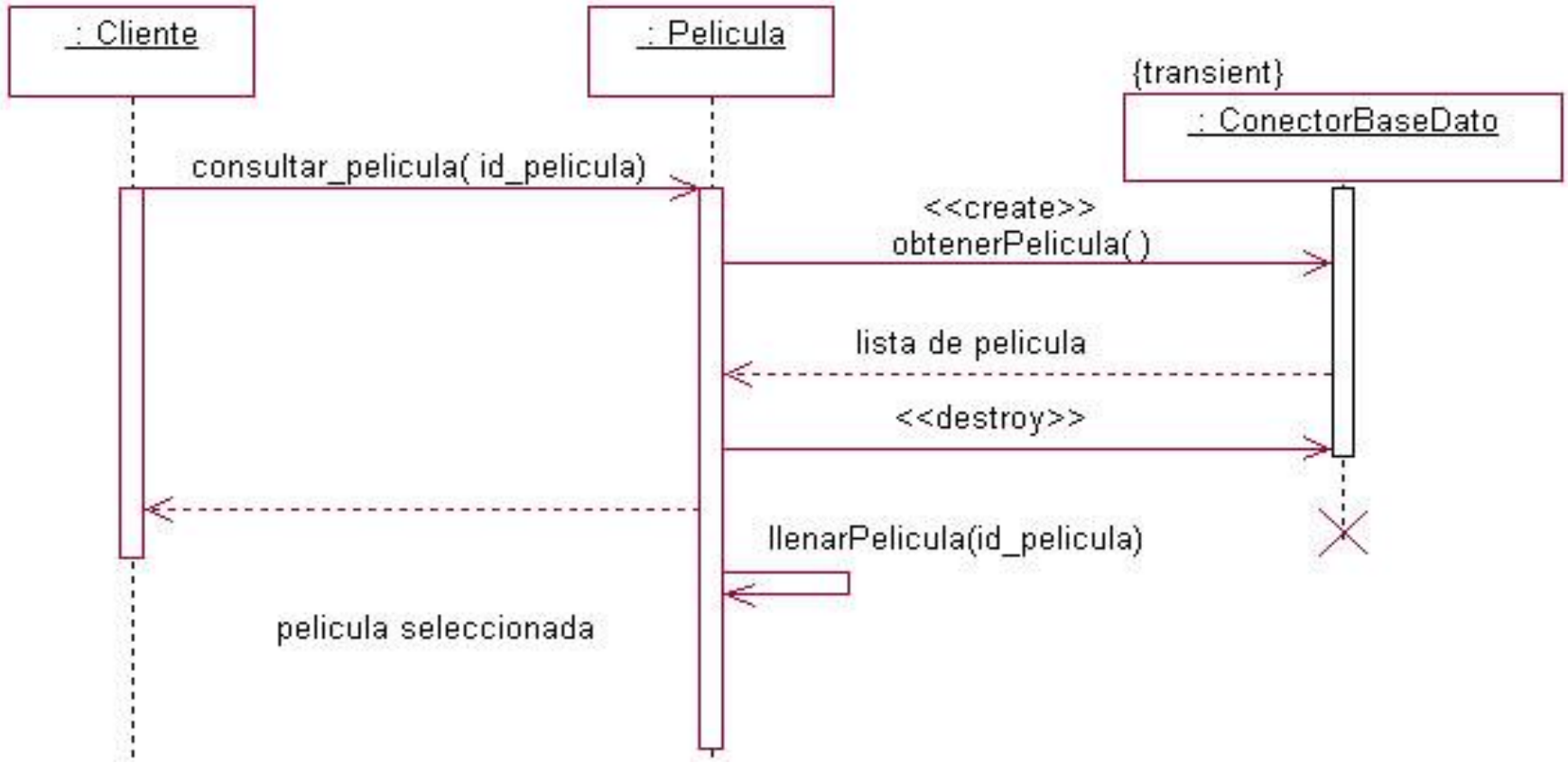
- Línea de vida de un objeto.



- Mensajes: existen diferentes tipos de mensajes.
  - Simples.
  - Síncronos.
  - Asíncronos.

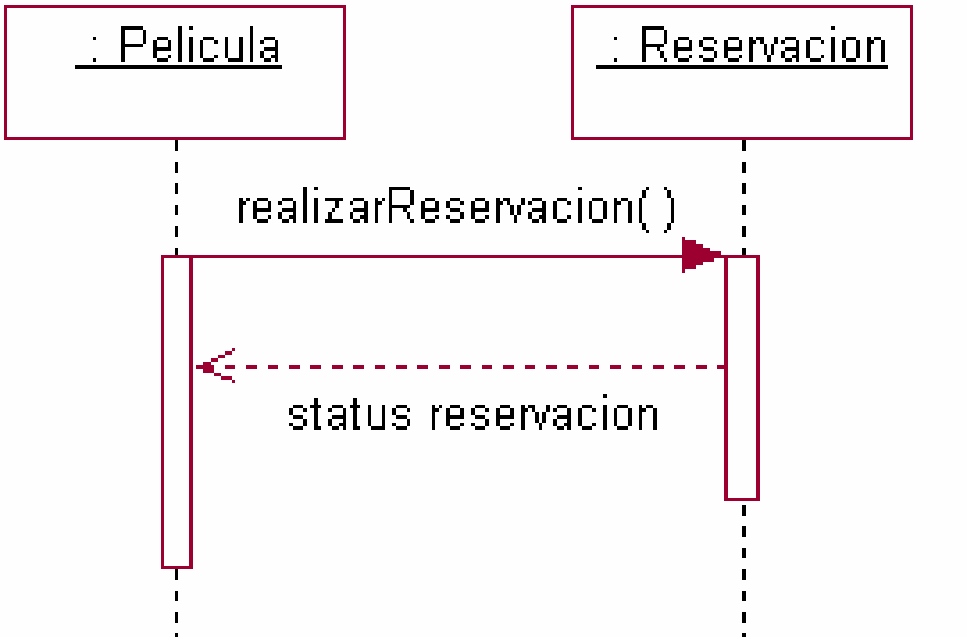
Diagramas de Secuencia (Mensajes)

- Simples**:** Es la transferencia de control de un objeto a otro.



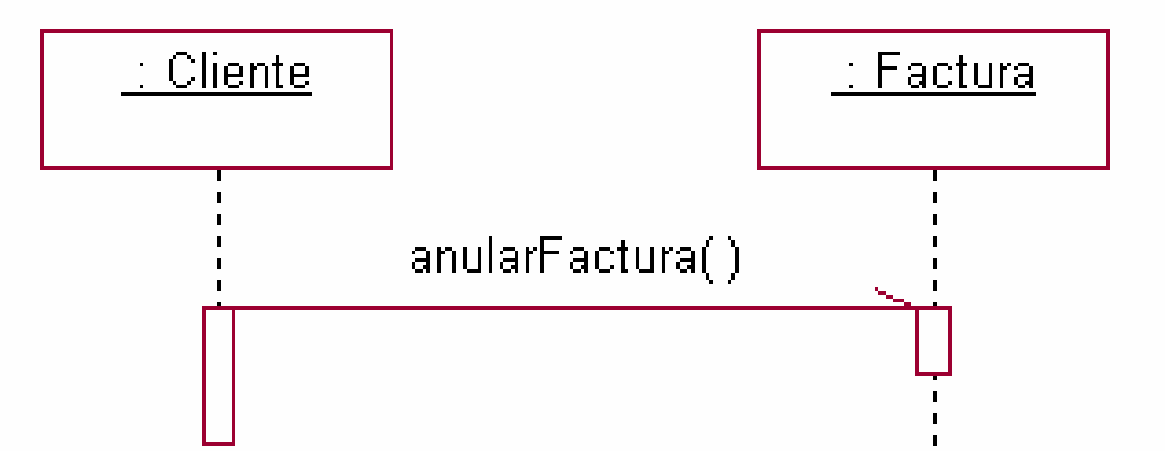
Diagramas de Secuencia (Mensajes)…1

- Síncronos: El objeto remitente pasa un mensaje y espera a que el receptor acepte la llamada.



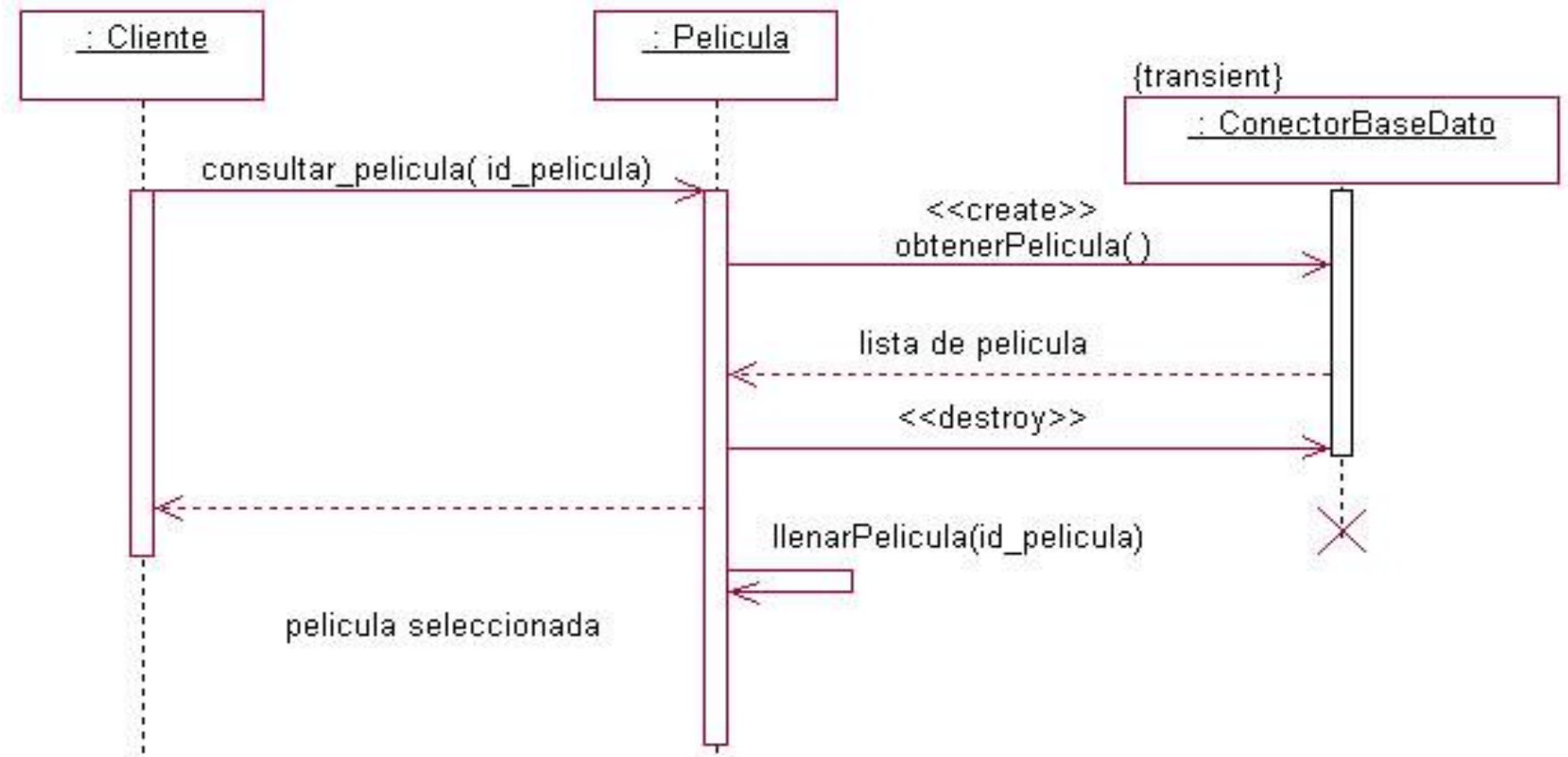
Diagramas de Secuencia (Mensajes)…2

- Asíncronos**:** El objeto remitente envía una señal y continúa con sus tareas independientes.



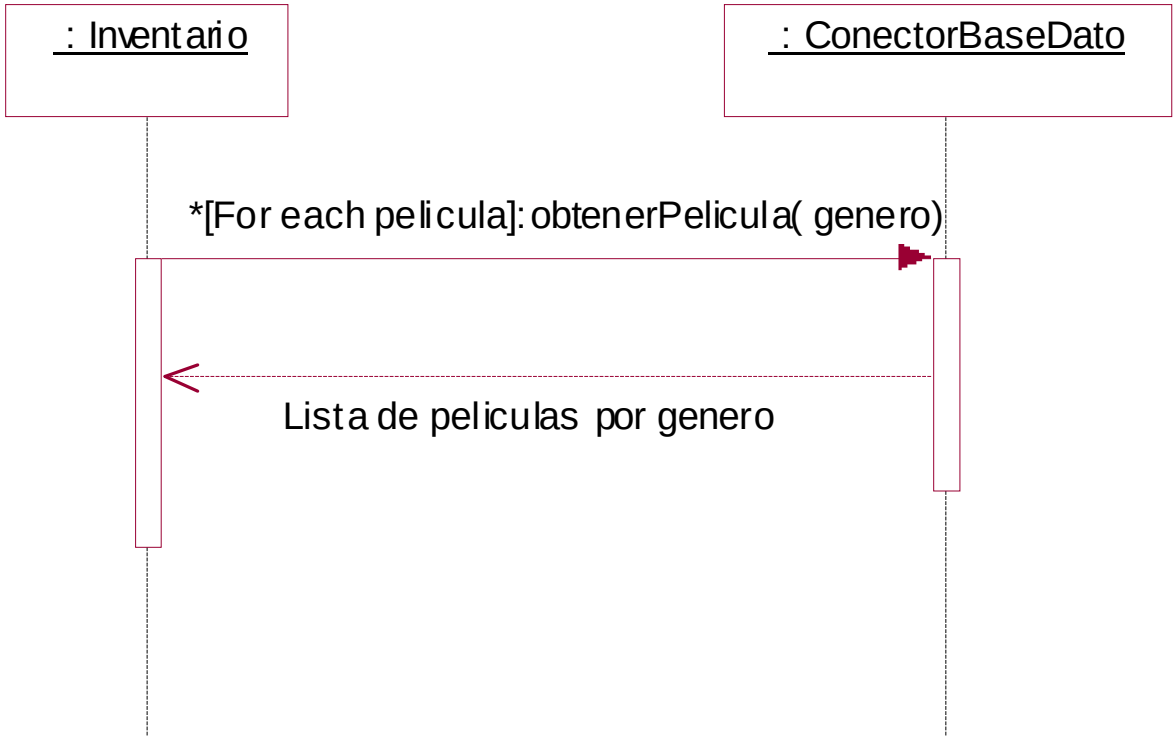
Diagramas de Secuencia (Mensajes)…3

- Reflexivos**:** El remitente y el receptor son el mismo objeto.



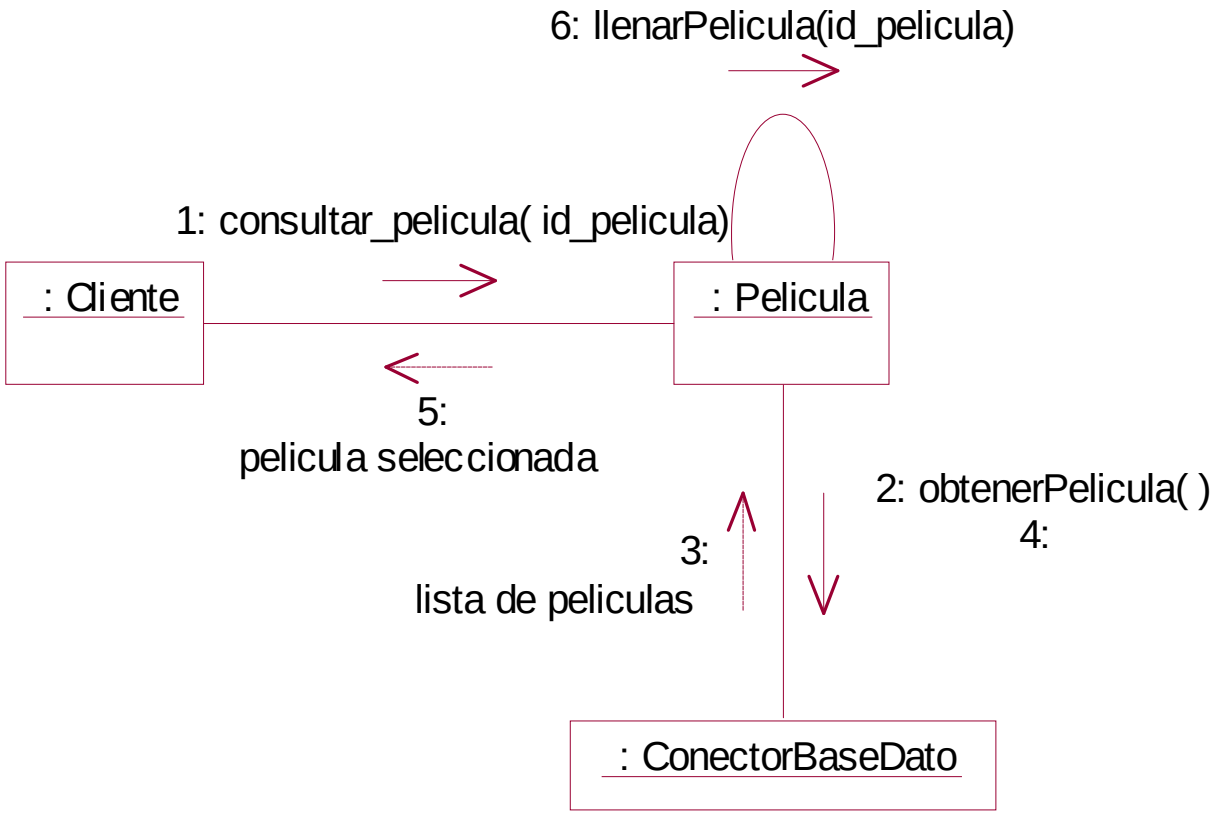
Diagramas de Secuencia (Interacción)

- Uno o más mensajes son ejecutados en secuencia más de una vez.

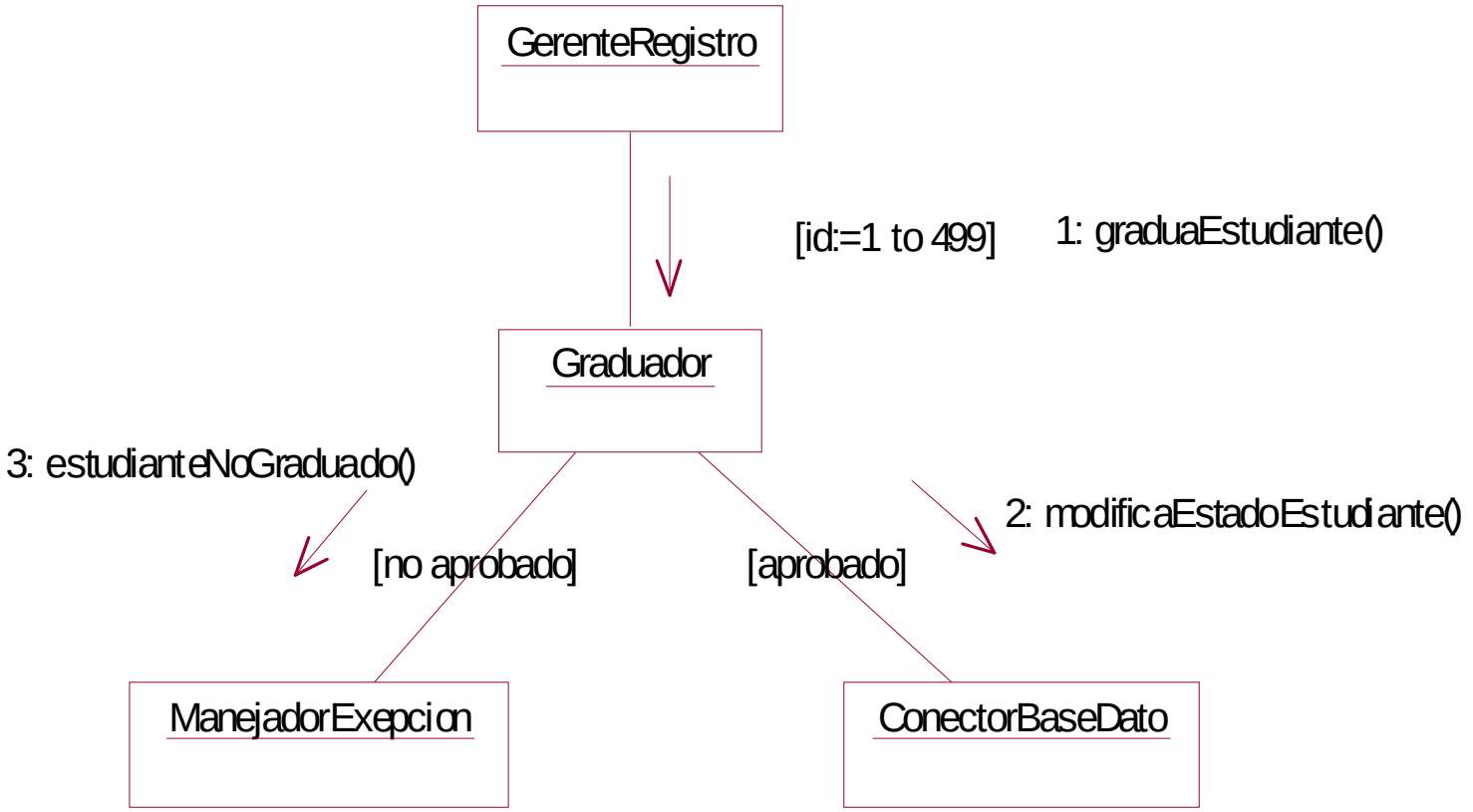


Diagramas de Colaboración

- Brindan una orientación visual acerca del flujo, en el contexto de la organización estructural de objetos que interactúan unos con otros.

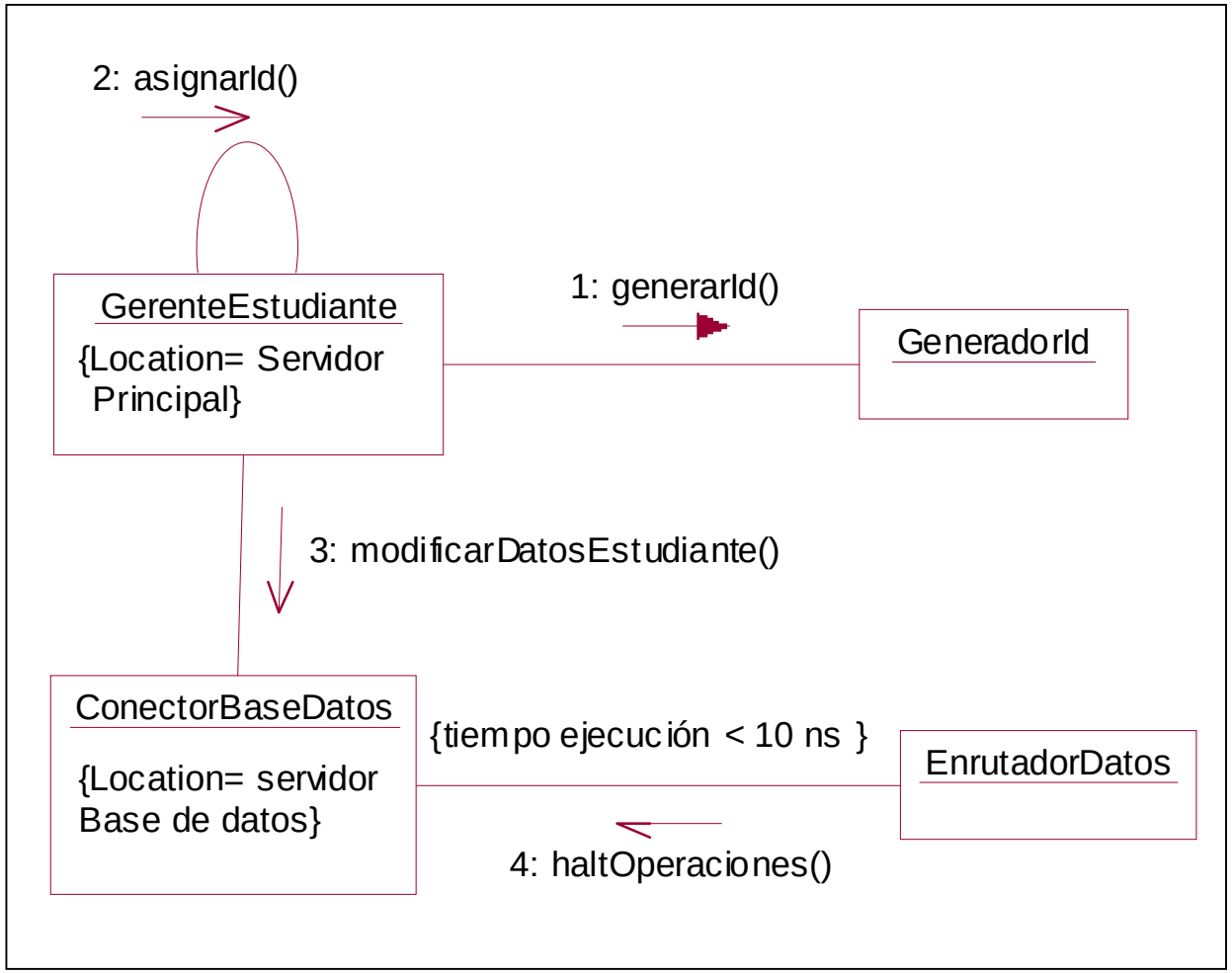


Interacciones en Diagramas de Colaboración



Tiempo y Espacio

La especificación del tiempo y espacio es importante para sistemas en tiempo real y la ubicación para sistemas distribuidos.



Tipos:

- Marca de tiempo
- Expresión de tiempo
- Restricción de tiempo
- Ubicación

Resumen

Ahora que ud. ha completado esta unidad, debe ser capaz de:

- Definir colaboraciones.
- Definir interacciones.
- Elaborar diagramas de secuencia.
- Elaborar diagramas de colaboración.
- Describir pasos para elaborar diagramas de colaboración.

Modelado de Comportamiento Avanzado

Objetivos de Aprendizaje

Al finalizar esta unidad ud. debería ser capaz de:

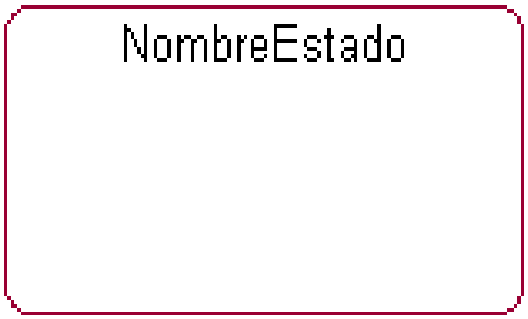
- Discutir que son los eventos y señales.
- Definir máquinas de estado.
- Describir instancias y diagramas de objetos.
- Discutir y construir diagramas de estado.

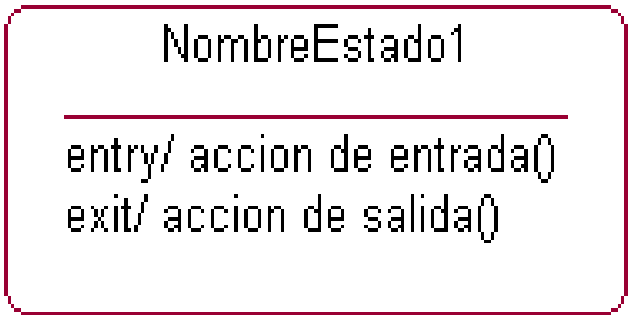
Máquinas de Estado

- Las máquinas de estado permiten modelar el comportamiento de un objeto individual.
- Cada objeto tiene un tiempo de vida.
- Entra en existencia en algún punto en el tiempo y deja de existir en algún otro punto en el tiempo.
- Durante su existencia, puede ir a través de un número de cambios, estos cambios son modelados con máquinas de estado.
- La máquina de estado especifica la secuencia de estados que un objeto pasa a través de su vida como resultado de la ocurrencia de eventos.
- Esto también incluye la reacción del objeto a esos eventos.
- Las máquinas de estado pueden ser usadas para modelar el comportamiento de una instancia de una clase o de un caso de uso.

Estados

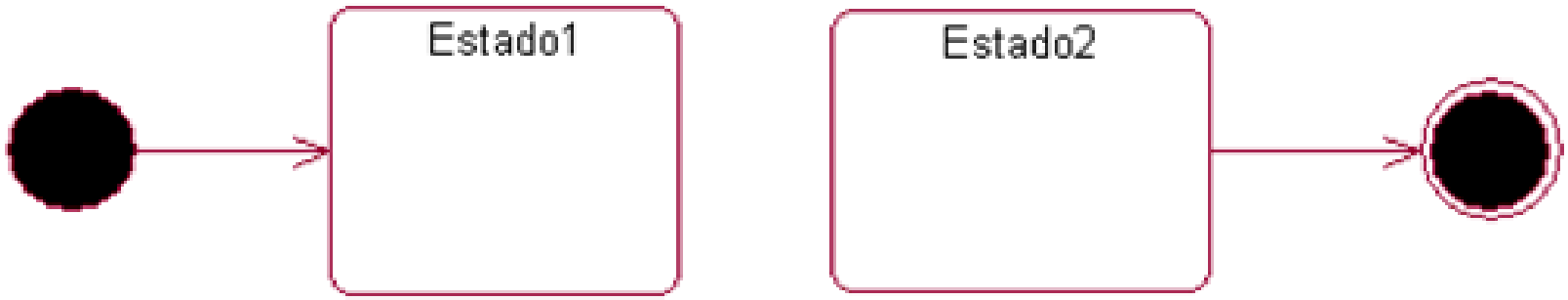
- Un estado es una condición específica durante el tiempo de vida de un objeto.
- Un estado de un objeto tiene varias partes:
  - Nombre.
  - Acciones de Entrada.
  - Acciones de Salida.
  - Transiciones Internas.
  - Transiciones Externas.
  - Subestados.
  - Eventos Diferidos.





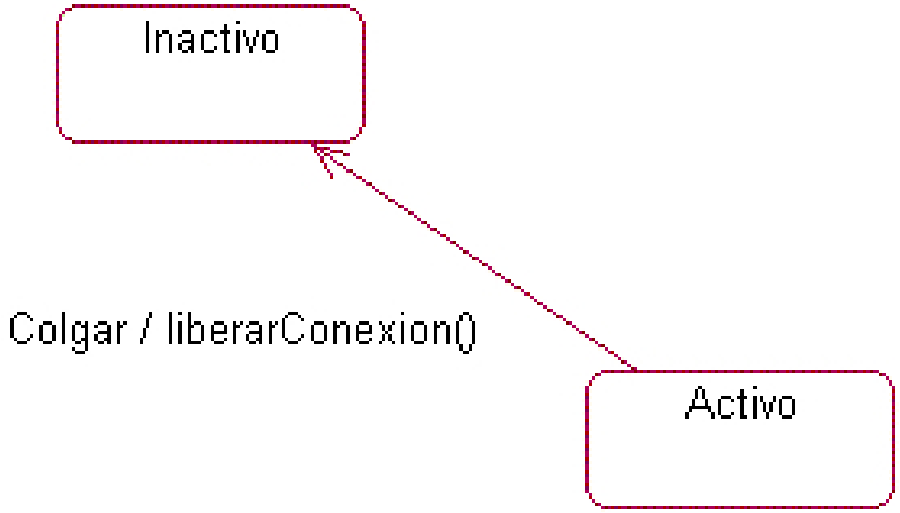
Estados…1

- UML define dos tipos especiales de estados:
  - Estado Inicial.
  - Estado Final.



Eventos

- Un evento es una ocurrencia en el sistema que causa una transición de estado.
- Un evento tiene tiempo y espacio asociados a él.

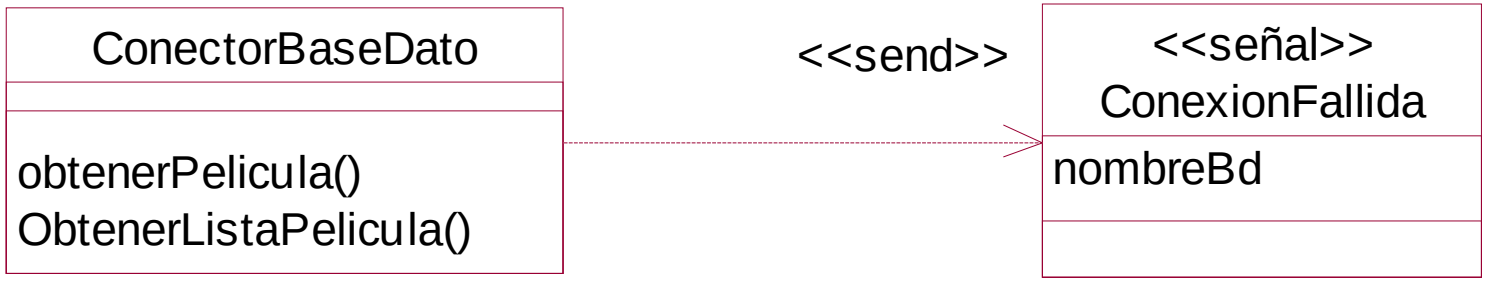


Eventos…1

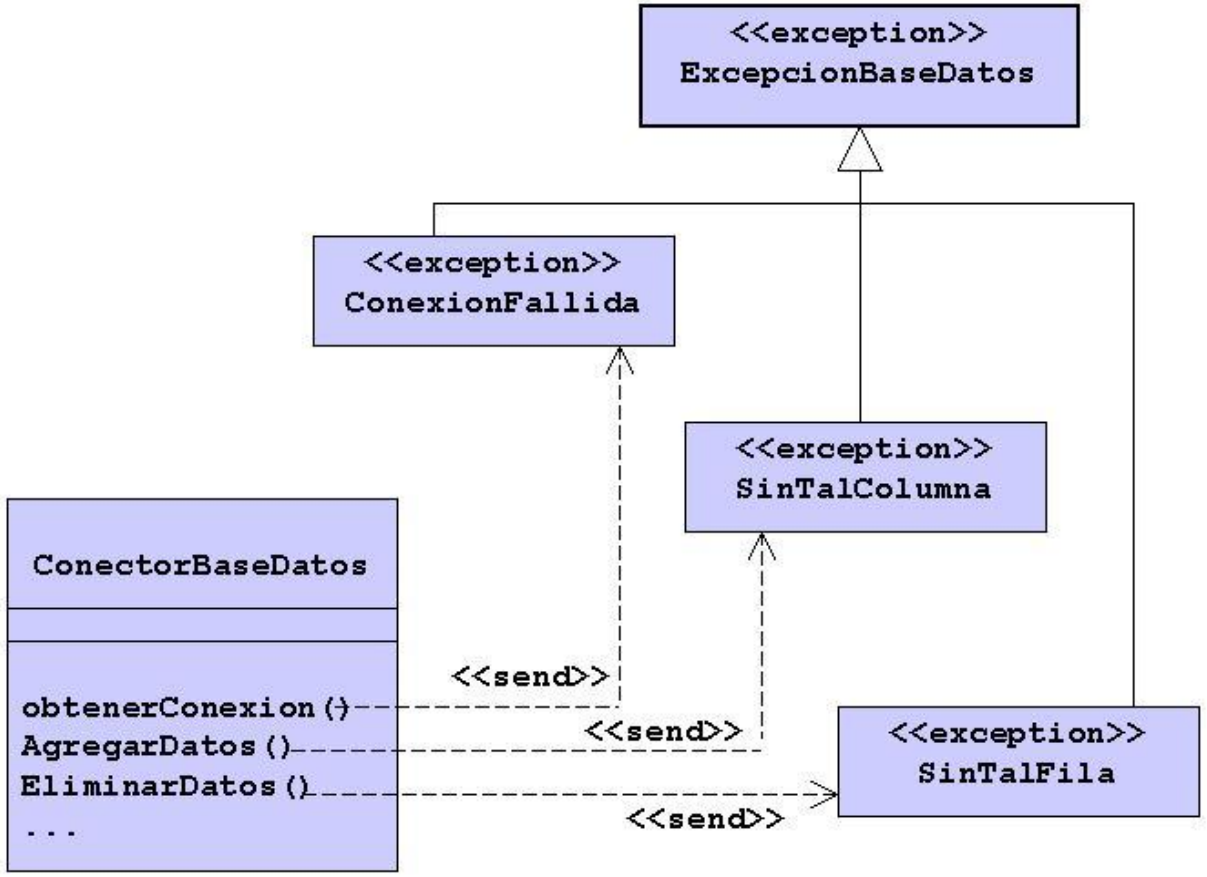
- *Eventos* ***Externos***
  - Los eventos que suceden entre los actores y el sistema son llamados eventos externos.
  - Insertar una tarjeta en un sistema automatizado es un ejemplo de un evento externo.
- *Eventos* ***Internos***
  - Aquellos eventos que pasan entre objetos residiendo en un sistema son referidos como eventos internos.
  - Una excepción por desborde de punto flotante es un ejemplo de un evento interno.
- Los cuatro tipos de eventos que pueden ser modelados en UML son:
  - **Señales.**
  - **Eventos de Llamadas.**
  - **Eventos de Tiempo.**
  - **Eventos de Cambio.**

Señales

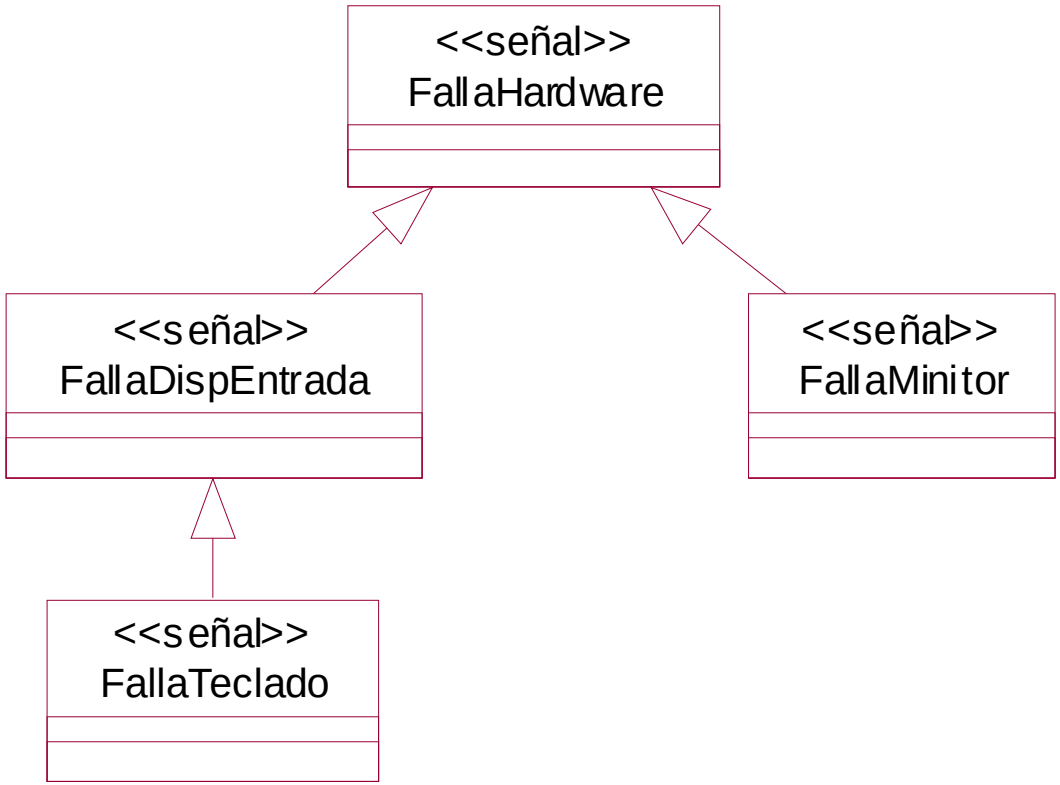
- Se refieren a la comunicación asíncrona entre objetos.
- Una señal es un evento que es lanzado asincrónicamente por un objeto y es capturado por otro objeto en el sistema.
- Las señales se usan para manejar eventos excepcionales, es referida como excepción en los lenguajes de programación modernos.



Excepciones Modeladas con Señales

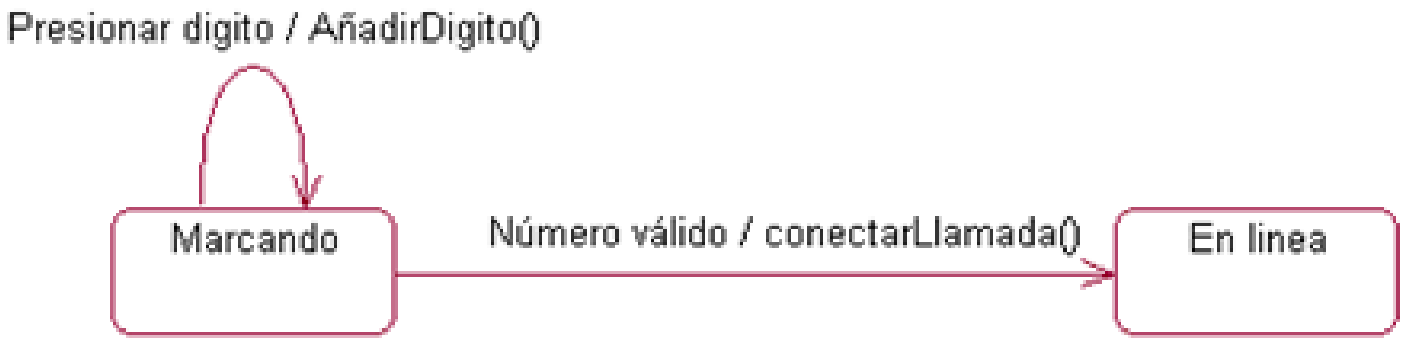


Jerarquía de Señales



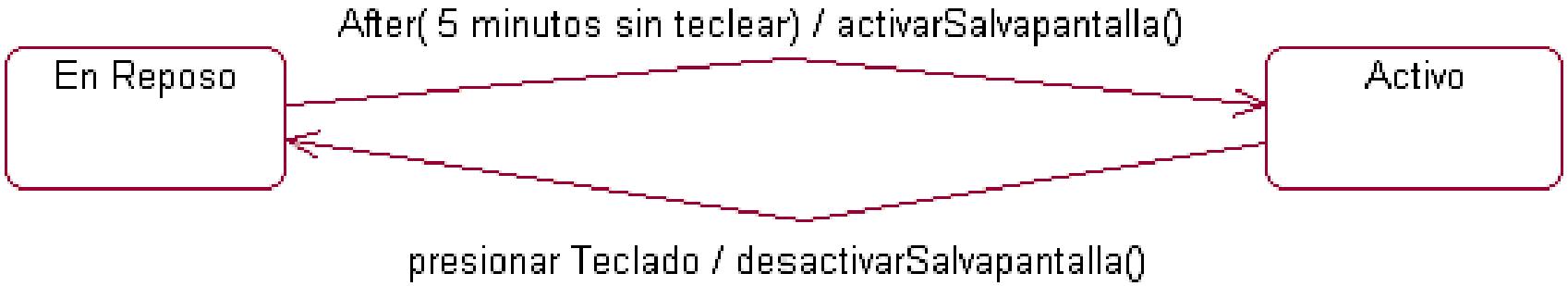
Eventos de Llamadas

- Los eventos de llamada representan el despacho de una operación.
- El despacho resulta en un estado de transición.
- Las señales son asíncronas mientras los eventos de llamada son generalmente síncronos.
- Un objeto invoca una operación en otro objeto que tiene un estado.
- El objeto receptor completa la operación y pasa a un nuevo estado.



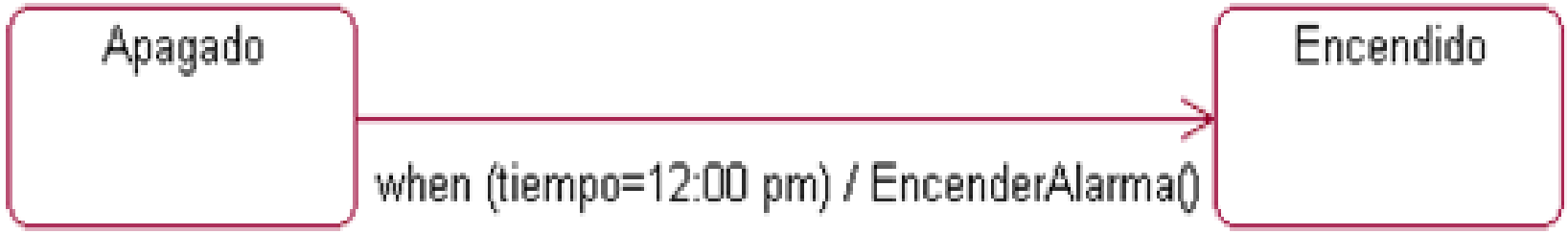
Eventos de Tiempo

- Los eventos de tiempo son usados en instancias donde el paso del tiempo tenga que ser grabado.
- En UML, los eventos de tiempo son modelados usando la palabra reservada **`after`**, seguida por la expresión que caracteriza el tiempo.



Eventos de Cambio

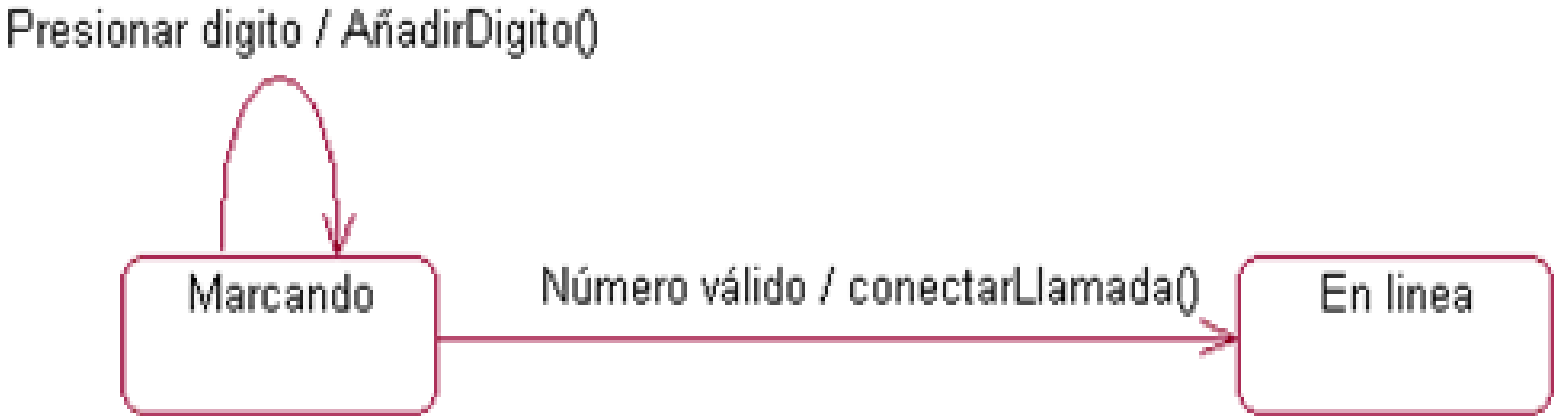
- Es usado para representar un cambio de estado o el cumplimiento de una condición.
- La palabra reservada usada para describir eventos de cambio es **`when`**, seguida de una expresión condicional.
- El evento de cambio ocurre cuando la condición en la expresión condicional cambia de falso a verdadero.



Transiciones

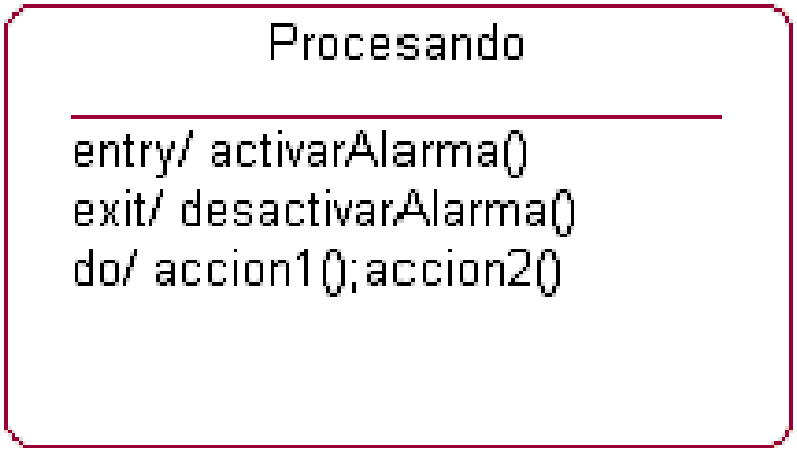
- Se define como la respuesta de un objeto, que se encuentra en un estado a la ocurrencia de un evento.
- Dos estados se relacionan a través de una transición. Cuando un objeto se mueve de un estado a otro, el movimiento se muestra usando una transición.
- Una transición puede tener múltiples fuentes y múltiples destinos.
- Una transición tiene cinco partes:
  - **Estado Fuente.**
  - **Estado Destino.**
  - **Acción.**
  - **Evento Activador.**
  - **Condición de Guarda.**

Estados y Transiciones



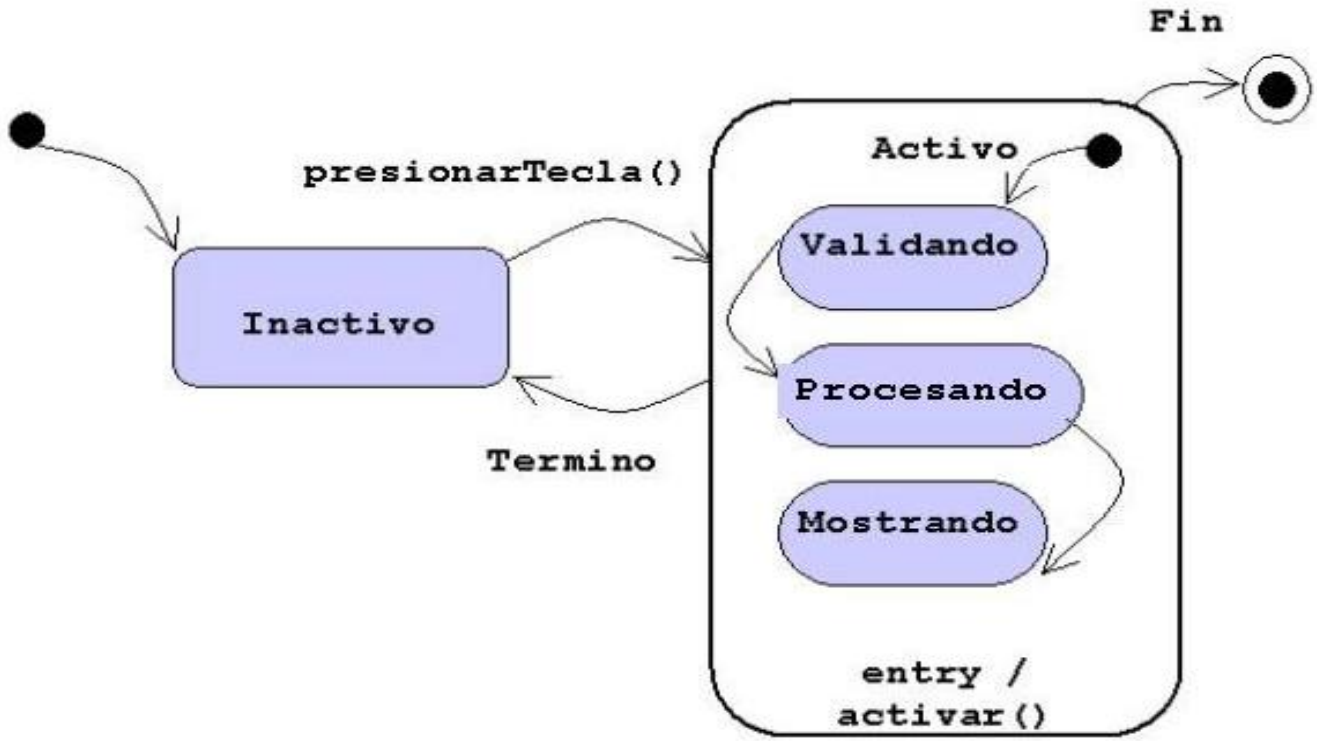
Estados y Transiciones…1

- Se muestran ejemplos de cinco características asociadas a estados y transiciones:
  - **Acción de Entrada.**
  - **Acción de Salida.**
  - **Transiciones Internas.**
  - **Actividades.**
  - **Eventos Diferidos.**



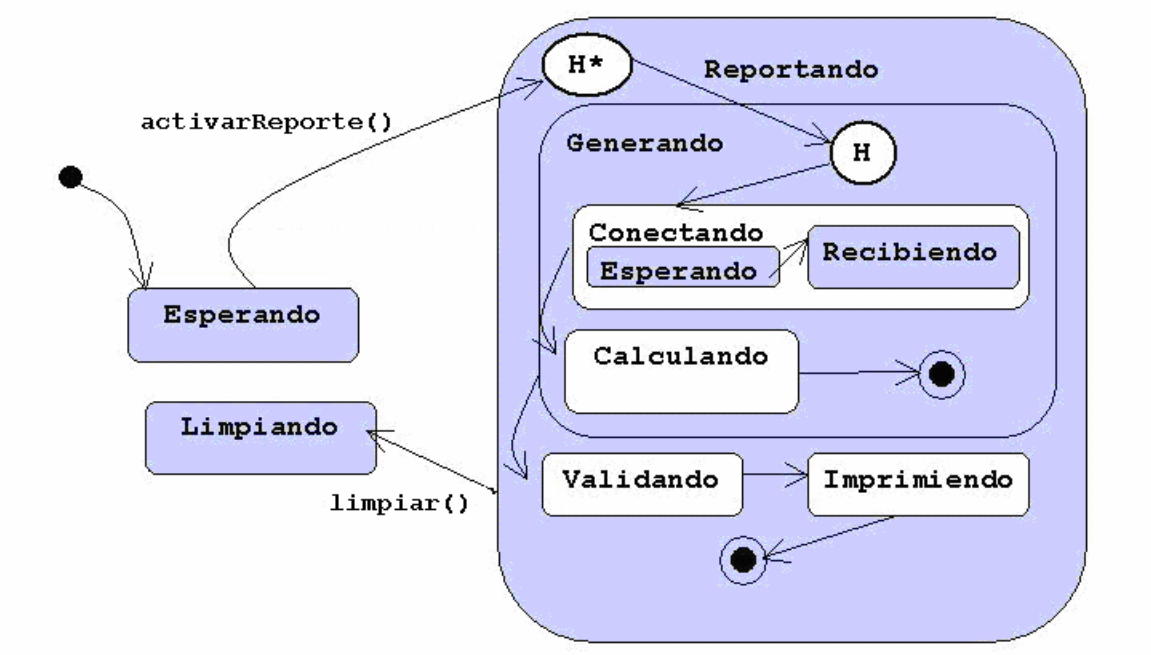
Subestados Secuenciales

- En el estado `Activo`, se encuentran tres subestados: `Validando`, `Procesando` y `Mostrando`. Ellos representan un flujo de control secuencial.



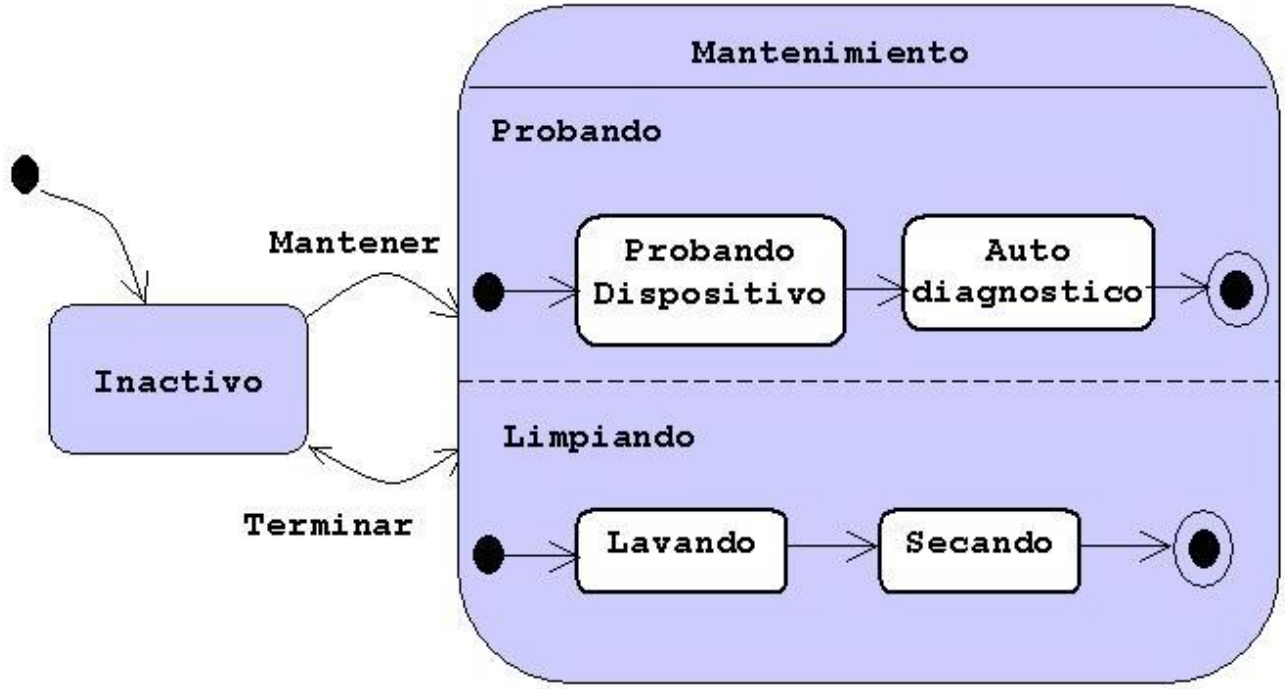
Historia de Estados

- UML proporciona una manera más simple en donde un historial del estado puede ser modelado. Se usa el símbolo H dentro de un círculo para indicar que el estado compuesto mantiene un historial.
- La transición desde fuera del estado compuesto es dirigida hacia la H y luego el control pasa al subestado secuencial (el estado inicial, la primera vez) y en posteriores transiciones al último subestado activo.

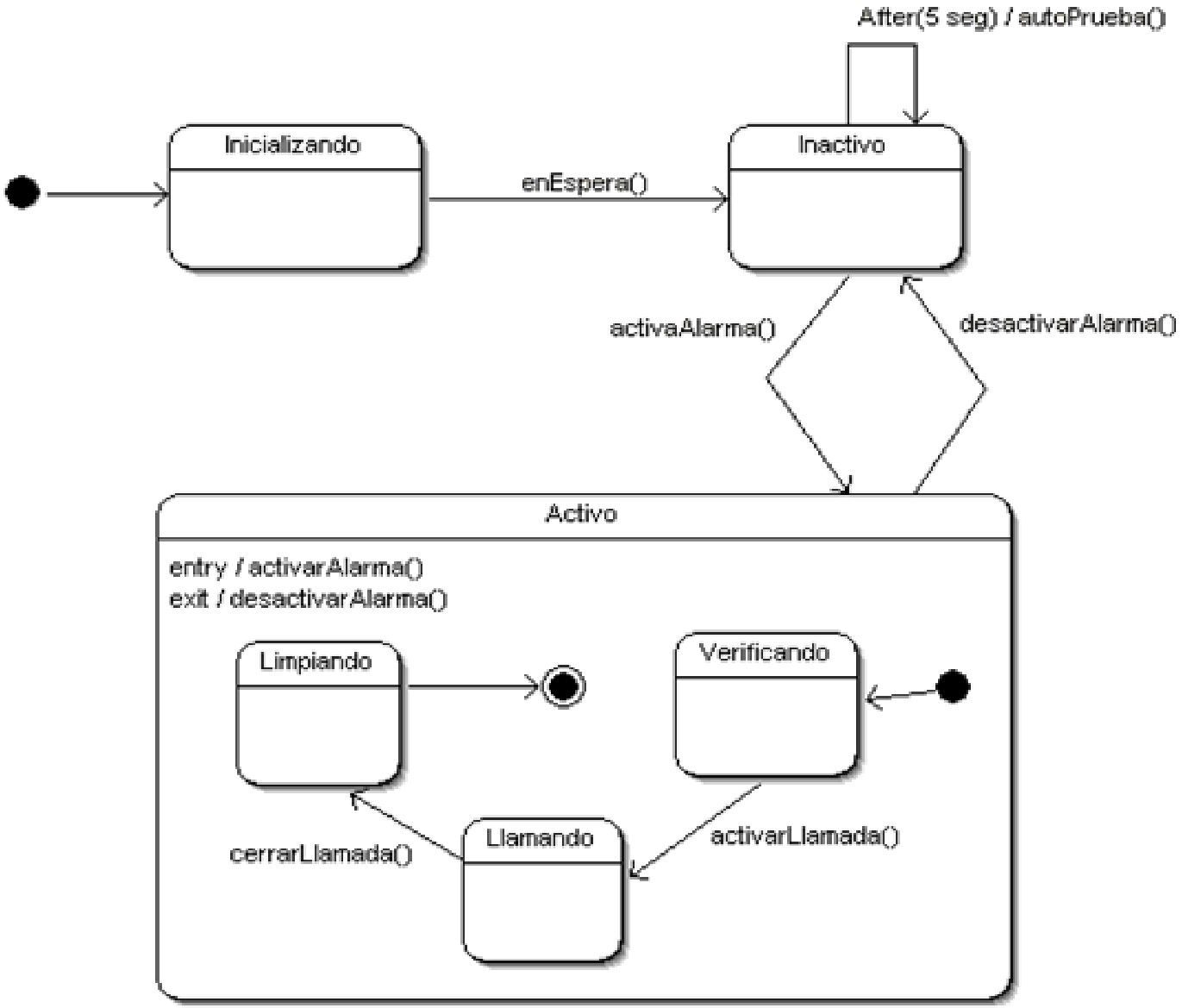


Subestados Concurrentes

- Un estado compuesto puede contener subestados concurrentes.
- El control es pasado al estado inicial de ambos estados concurrentemente. Dentro de los subestados concurrentes, los subestados son secuenciales por naturaleza.

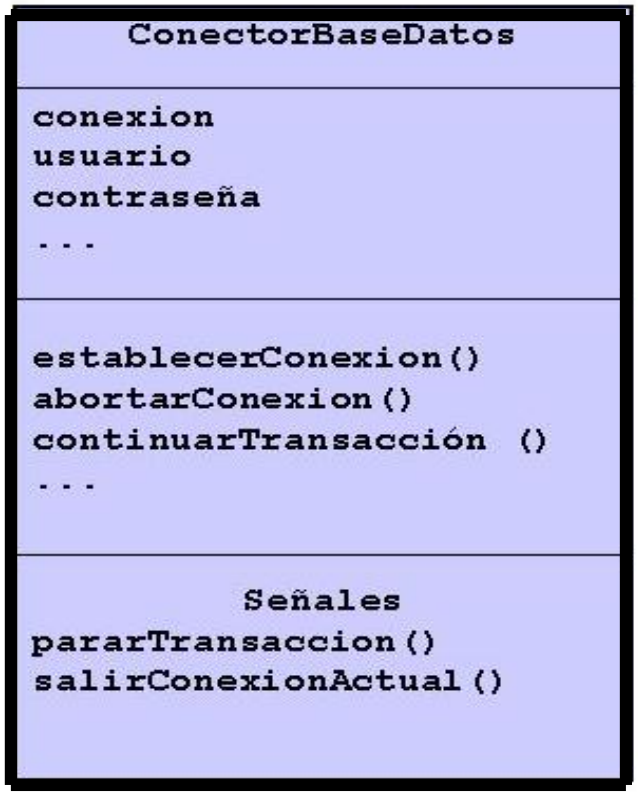


Diagramas de Estados



Clases Activas

- Una clase es referida como una clase activa cuando es capaz de iniciar una actividad de control.
- Una instancia de una clase activa es un objeto activo. Generalmente un objeto activo puede ser un proceso o un hilo, ya que ellos son capaces de iniciar actividades de control.

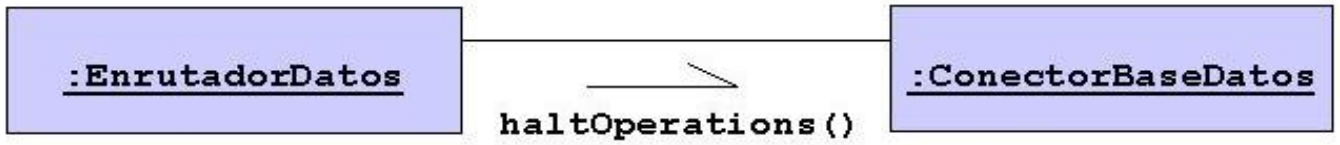


Mensajes-Objetos

- Pasar un mensaje desde un objeto pasivo hacia otro objeto pasivo.

- Un objeto activo invoca una operación sobre otro objeto activo sincrónicamente.

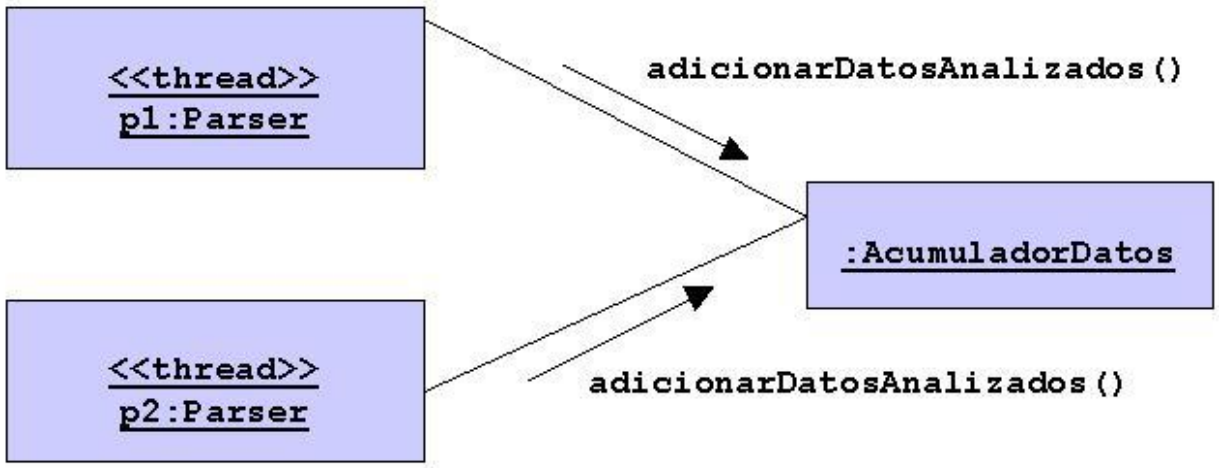
- Un objeto activo invoca una operación sobre otro objeto activo sincrónicamente.



Mensajes-Objetos…1

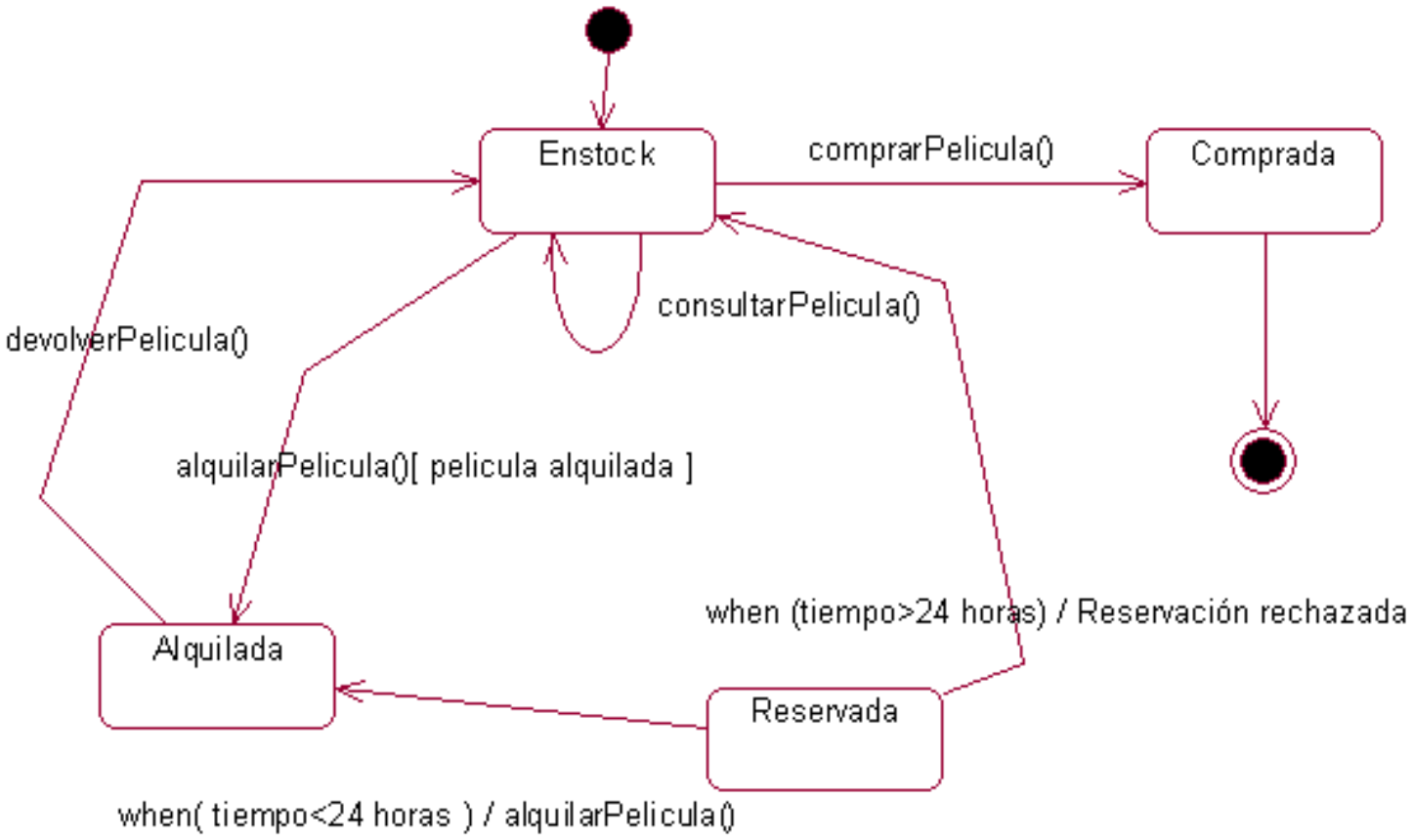
- Pasar un mensaje desde un objeto pasivo hacia un objeto activo.

- Pasar un mensaje desde un objeto activo hacia un objeto pasivo.



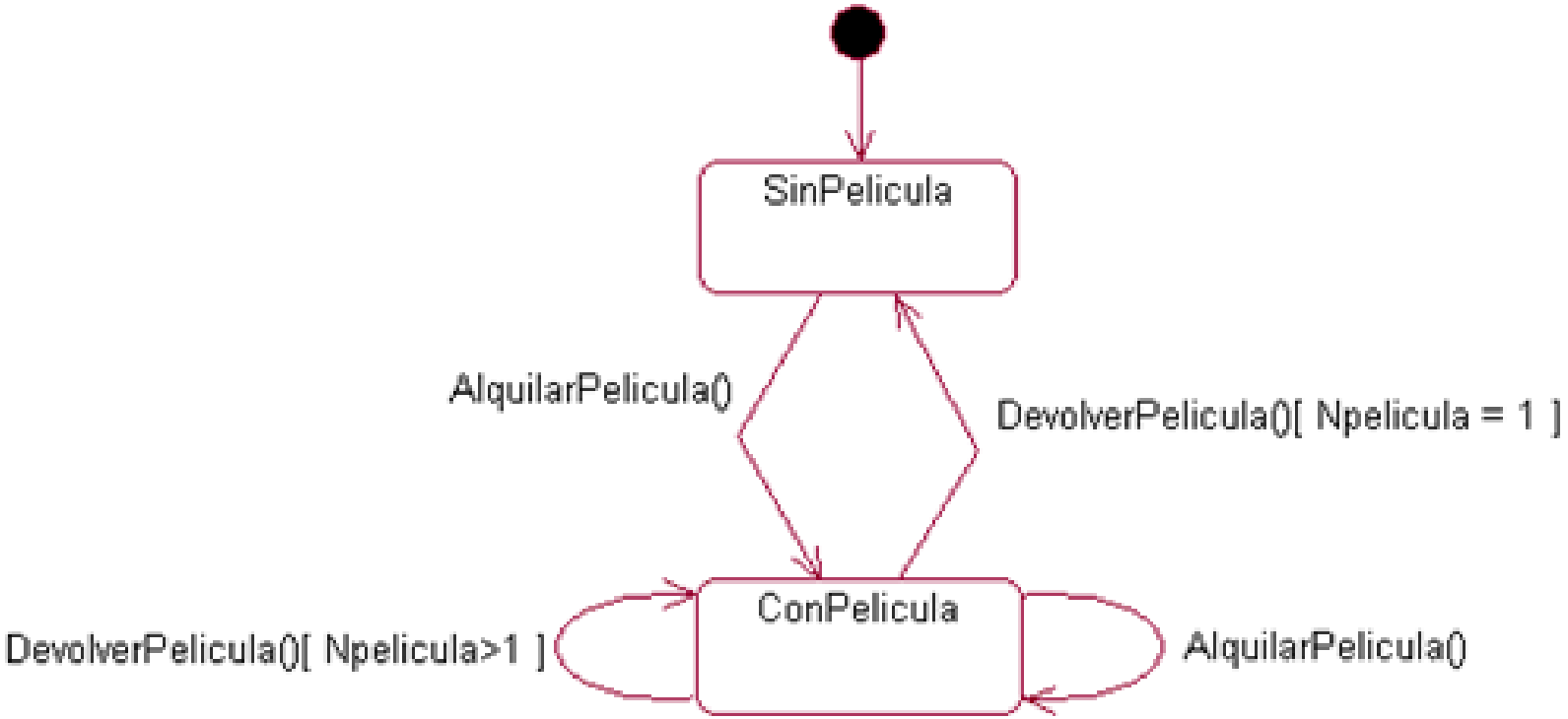
Caso de Estudio: Club de Video

- Diagrama de estado para el objeto película.



Caso de Estudio: Club de Video…1

- Diagrama de estado para el objeto Cliente.



Resumen

Ahora que ud. ha completado esta unidad, debe ser capaz de:

- Discutir eventos y señales.
- Definir máquinas de estado.
- Discutir y construir Diagramas de Estado.

Modelado Arquitectónico

Objetivos de Aprendizaje

Al finalizar esta unidad ud. debería ser capaz de:

- Describir componentes, desplegado y colaboraciones.
- Discutir diagramas de componentes.
- Discutir la necesidad de Diagramas de Despliegue (deployment).

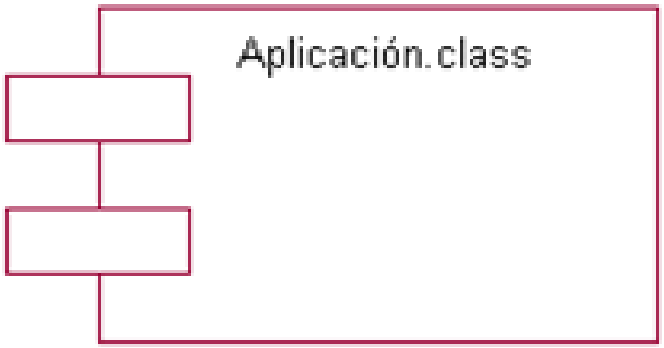
Introducción

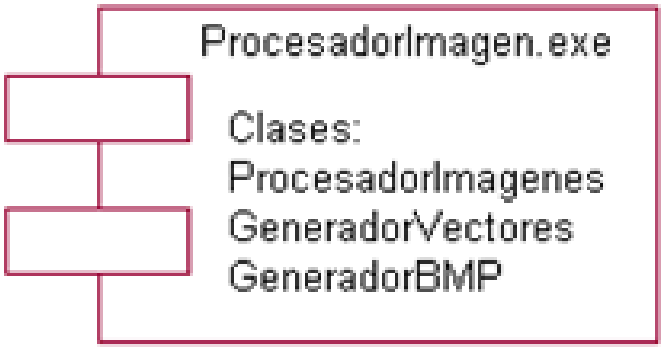
- Los diagramas arquitectónicos permiten modelar ejecutables, librerías, tablas, archivos, código fuente, nodos físicos, etc.
- Los dos diagramas que permitirán proporcionar una perspectiva arquitectónica del sistema son:

  – Diagrama de Componentes.
  – Diagrama de Despliegue.

Componentes

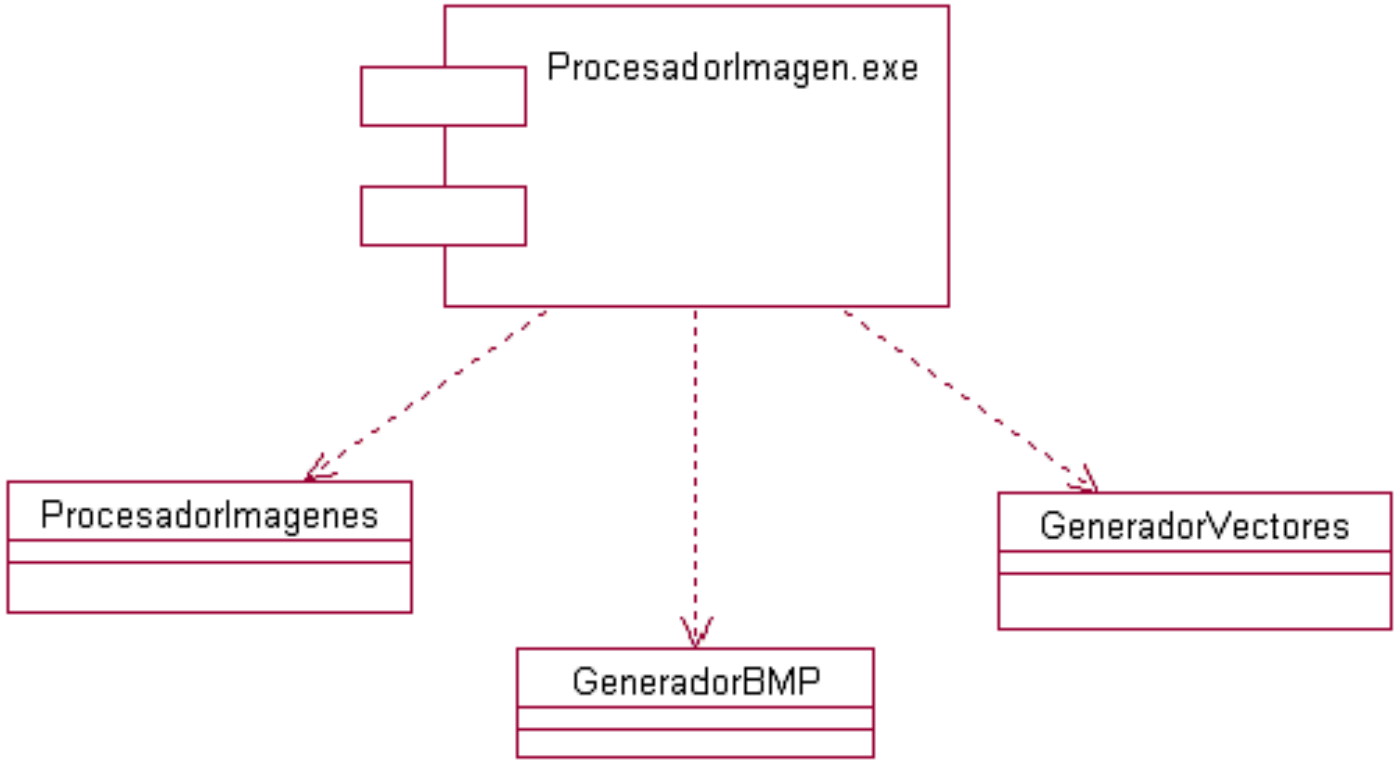
- Un componente es una parte física de un sistema.
- Hay componentes físicos de alto nivel que pueden ser o no equivalentes a otros más pequeños creados para las aplicaciones.





Componentes y Clases

- Las clases representan abstracciones lógicas.
- Los componentes representan partes físicas de un sistema.
- Un componente es la implementación física de un conjunto de elementos lógicos en el sistema.



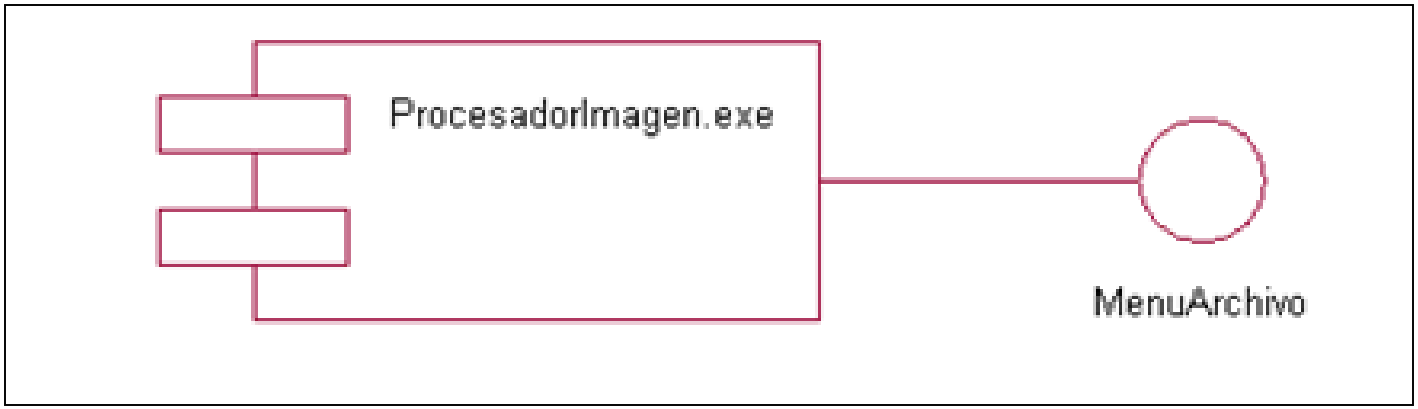
Componentes y Estereotipos

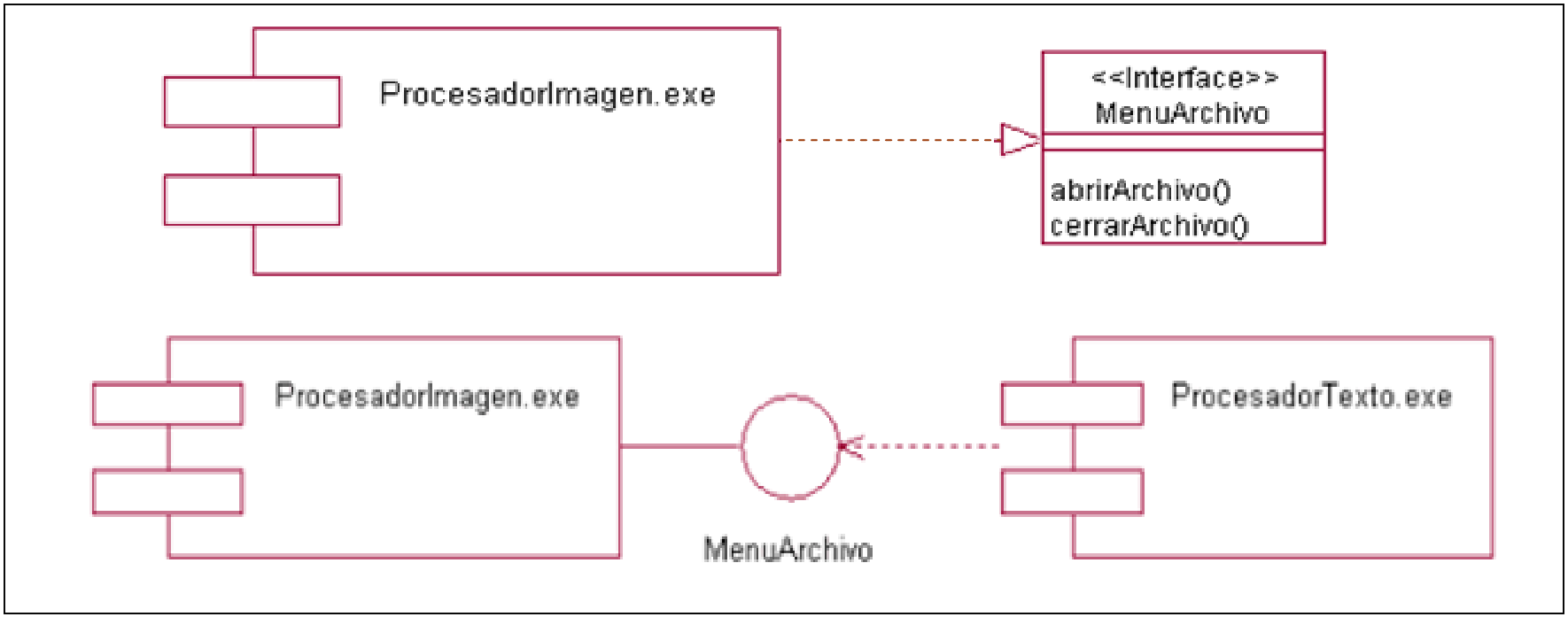
- **<<executable>>**: Para especificar que un componente puede ser ejecutado en un nodo.
- **<<library>>**: Para especificar una librería, ya sea estática o dinámica.
- **<<table>>**: Para especificar que el componente es una tabla de base de datos.
- **<<file>>**: Para especificar que es un código fuente o un archivo de datos.
- **<<document>>**: Para especificar que el componente representa un documento.

Componentes e Interfaces

- Una interfaz es un conjunto de operaciones que especifica los servicios ofrecidos por una clase o un componente.
- Las interfaces establecen un puente entre componentes y clases.
- Una interfaz puede ser realizada tanto por un componente como por una clase.
- La relación entre un componente y una interfaz puede ser mostrada de dos maneras:
  - Forma abreviada (icónica).
  - Forma expandida
- Un componente puede tanto importar y como exportar interfaces.

Componentes e Interfaces…1





Tipos de Componentes

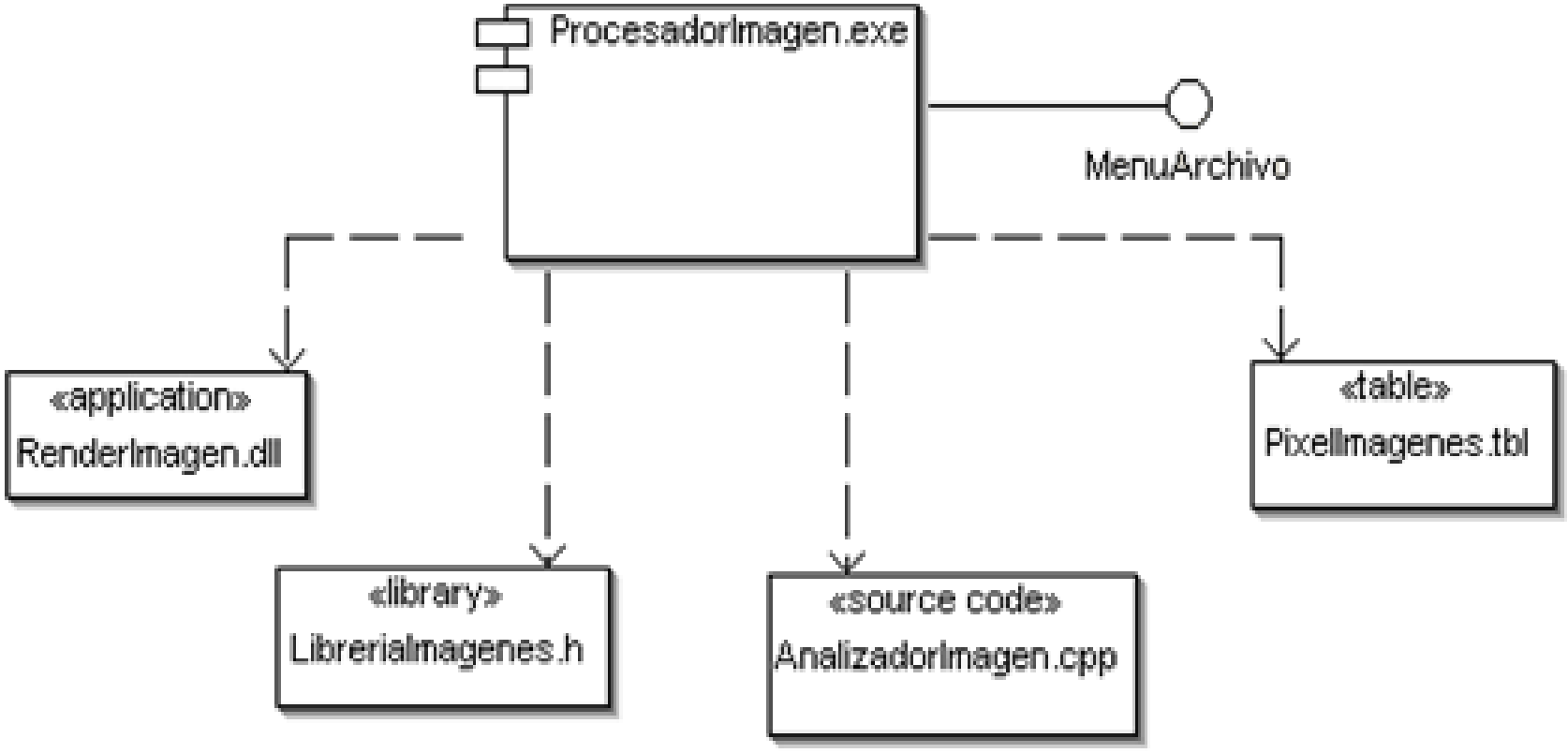
- Hay tres tipos de componentes:
  - **Despliegue** (Deployment): Estos componentes se usan para formar un sistema ejecutable.
  - **Producto de Trabajo** (Work product): Estos componentes son parte del proceso de desarrollo que es esencial para el sistema.
  - **Ejecución** (Execution): Estos componentes son el resultado de un sistema que se está ejecutando.

Diagramas de Componentes

- Los diagramas de componentes son usados para modelar la implementación estática de los elementos físicos residiendo en un nodo, tales como: ejecutables, librerías, tablas, archivos, etc.
- Los diagramas de componentes también son útiles para construir sistemas ejecutables ambos a través de ingeniería hacia delante como en reversa.
- Es posible agregar notas y restricciones a un diagrama de componentes.

Diagramas de Componentes…1

Elementos del componente ProcesadorImagen.exe.

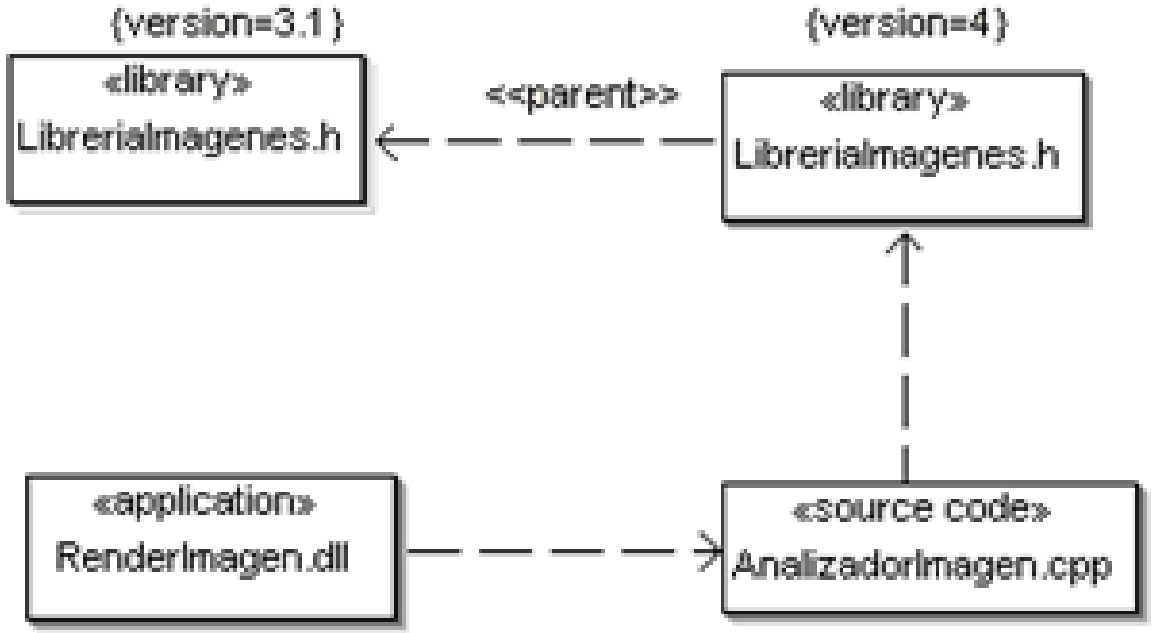


Diagramas de Componentes…2

- Los diagramas de componentes son usados generalmente para modelar los siguientes elementos:
  - Código fuente.
  - Lanzamiento de ejecutables (releases).
  - Bases de Datos físicas.
  - Sistemas adaptables.
  - Ingeniería hacia Adelante y en Reversa.

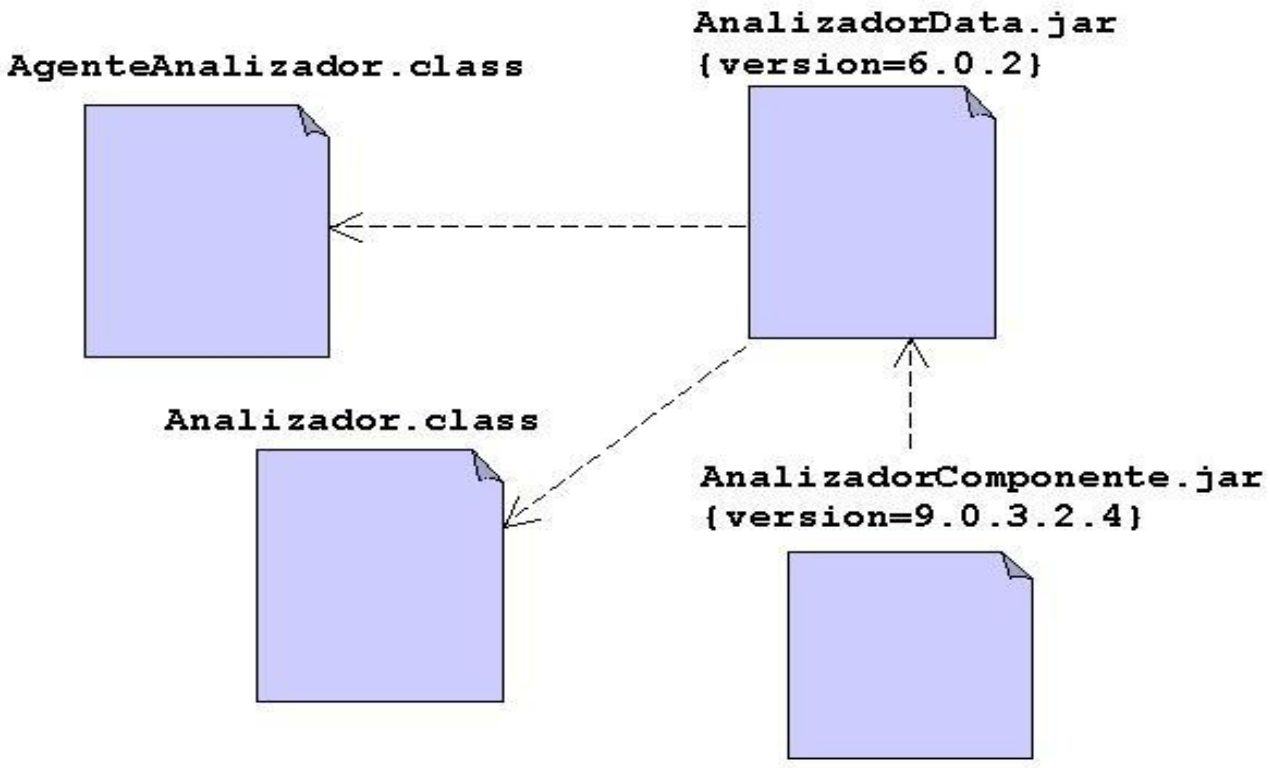
Modelar el Código Fuente

- Pueden usarse paquetes para agrupar archivos fuente. El estereotipo <<parent>> puede mostrar versiones anteriores. Los valores etiquetados pueden usarse para indicar la versión o cualquier otra información acerca de un archivo.



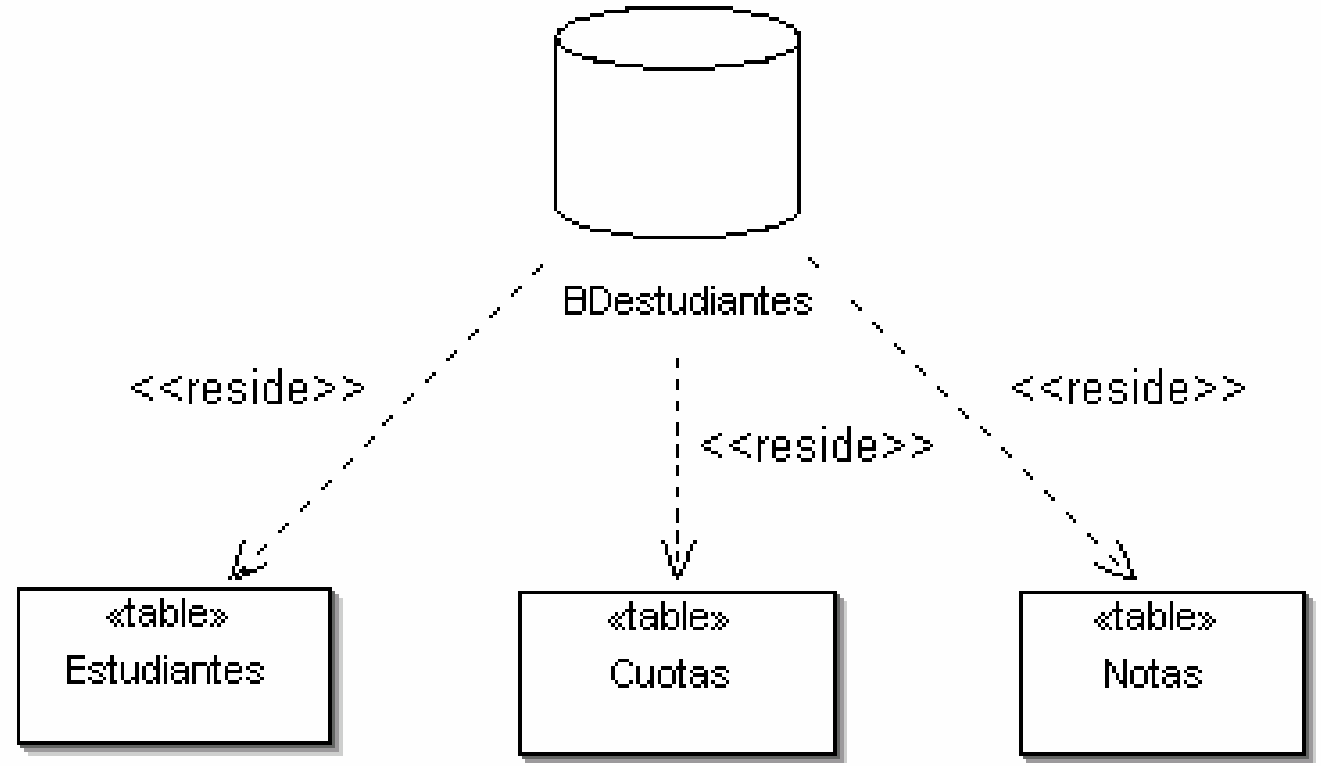
Modelar el Lanzamiento de Ejecutables

- Para sistemas grandes, puede ser necesario un archivo ejecutable, una librería de enlace dinámico y un archivo de base de datos. En Java se necesitan archivos `.class` y `.jar`.



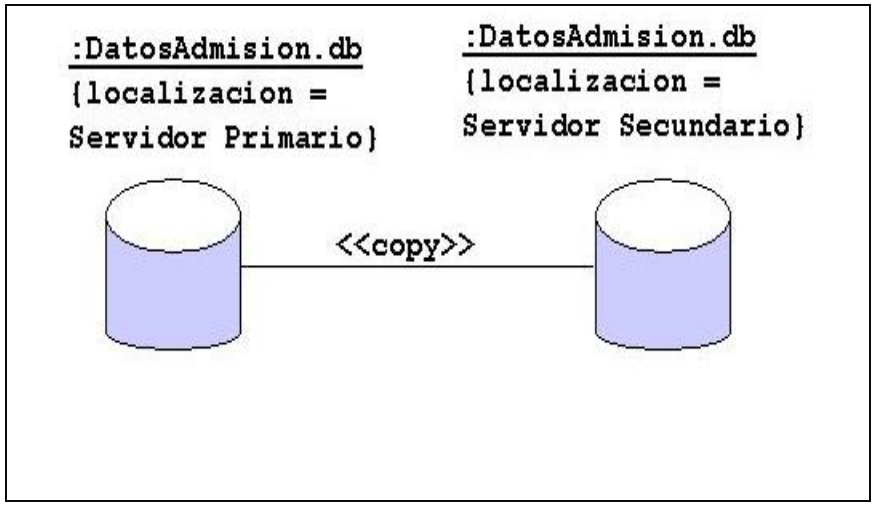
Modelar Bases de Datos Físicas

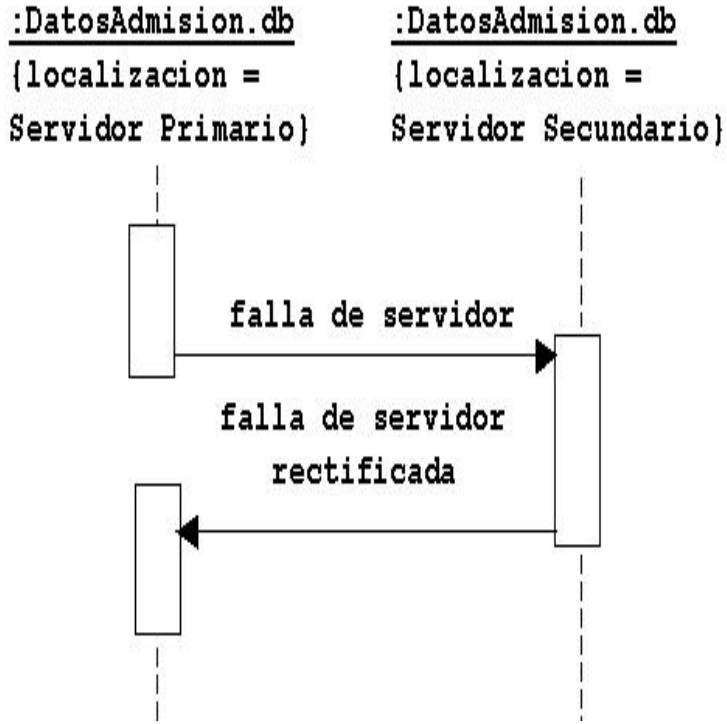
La figura muestra el modo en UML en que se modela bases de datos físicas.



Modelar Sistemas Adaptables

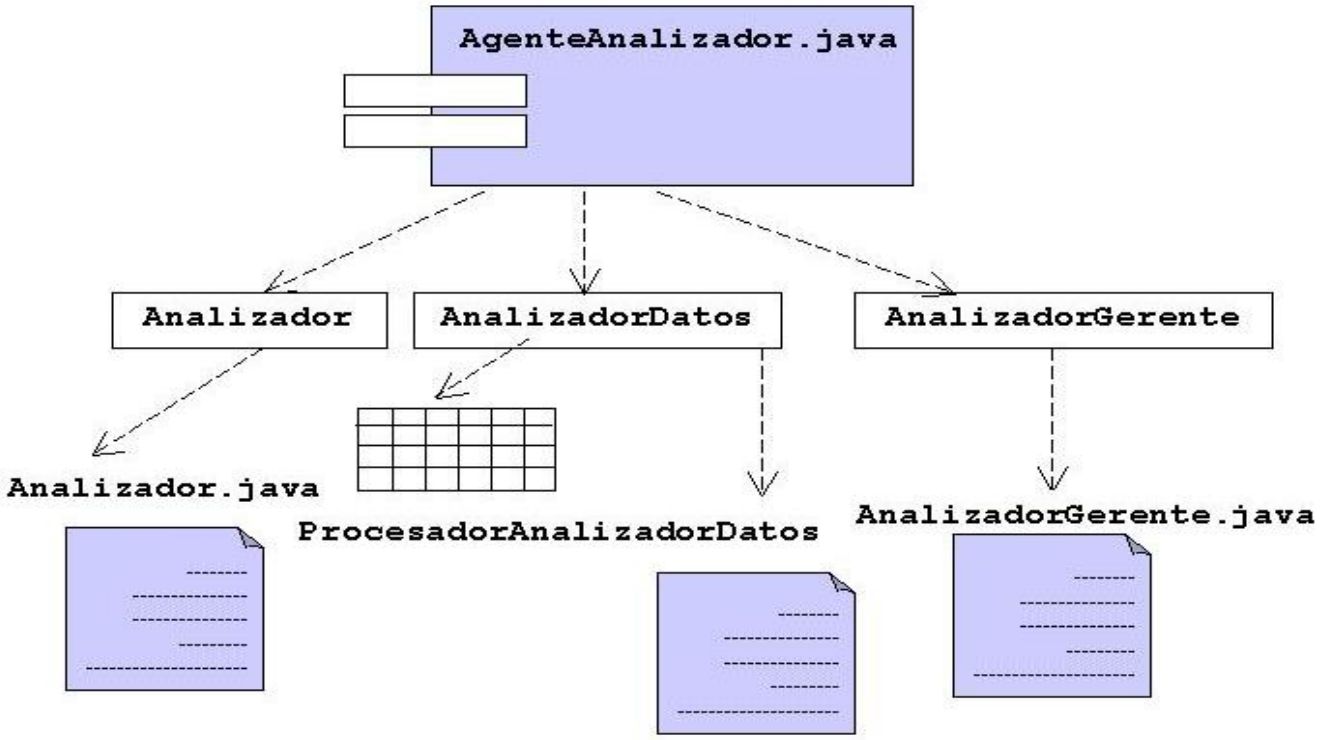
- Los sistemas dinámicos contienen agentes que son móviles. Estos agentes móviles se desplazan de un procesador a otro para manejar el balance de carga y la recuperación ante fallas.





Ingeniería hacia Delante y Reversa

- La ingeniería Hacia Adelante es la creación del código a partir de un modelo existente.
- La ingeniería en Reversa es la creación del modelo a partir del código existente.



Colaboraciones

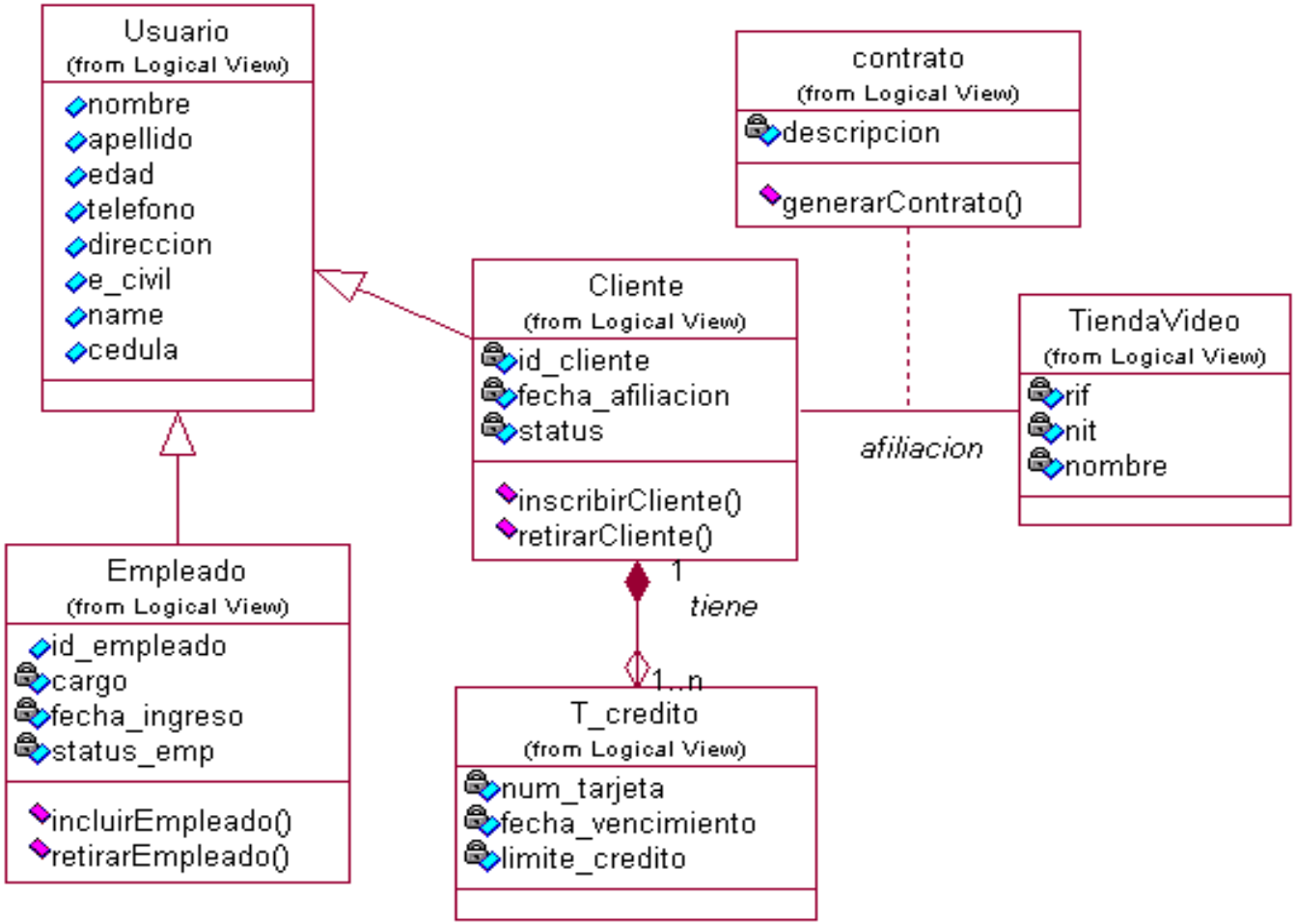
- Las colaboraciones son un conjunto de clases, interfaces, nodos, componentes y casos de uso.
- Estos elementos juntos proporcionan un comportamiento que es mayor que la suma de todas sus partes.
- Una colaboración también especifica cómo los clasificadores y asociaciones realizan un elemento.

Colaboraciones…1

- Las colaboraciones tratan con dos aspectos:
  - ***Aspectos Estructurales:*** Estos incluyen:
    - Clases.
    - Interfaces.
    - Nodos.
    - Componentes.
    - Casos de uso.
  - ***Aspectos de comportamiento***: Estos incluyen:
    - Los aspectos dinámicos de cómo los elementos estructurales interactúan unos con otros.
    - Los aspectos de comportamiento son representados como un diagrama de interacción.

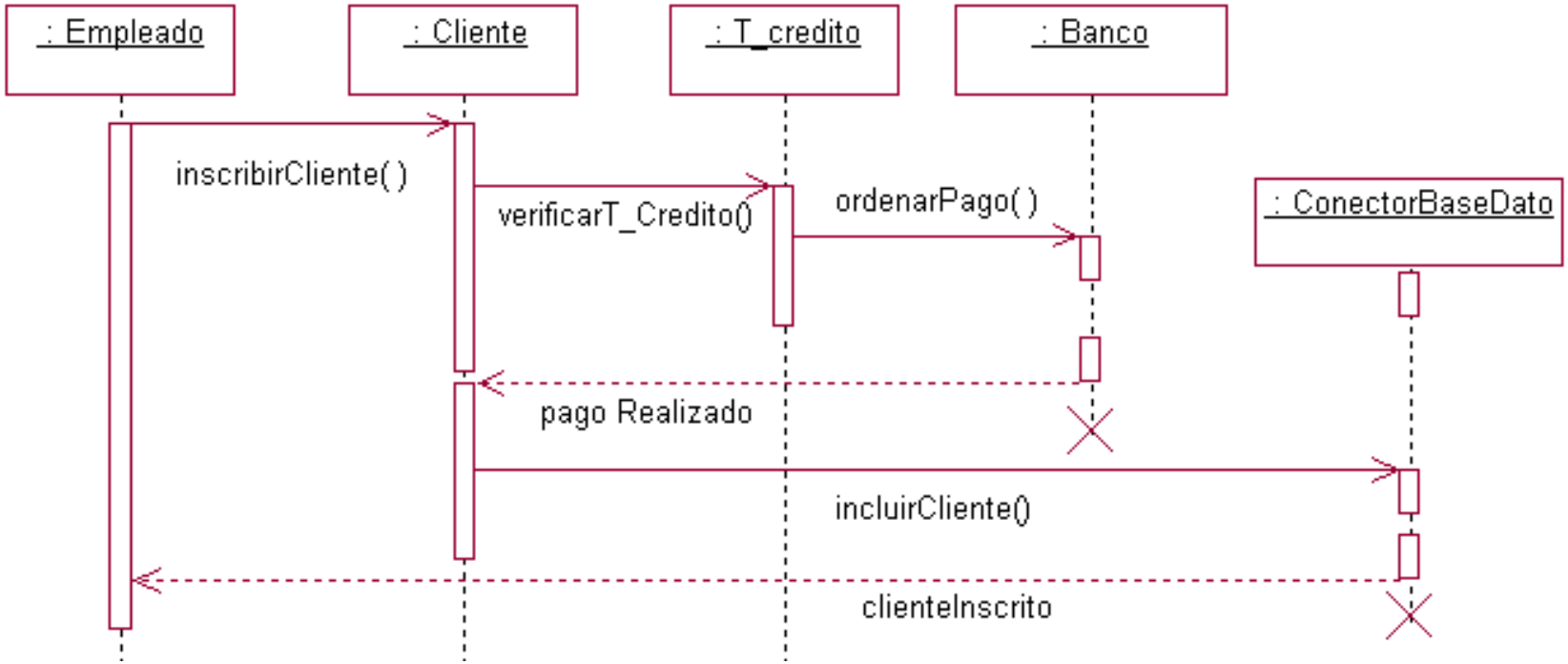
Colaboraciones Aspecto Estructural

El diagrama de clases mostrado es la realización de aspectos estructurales de la colaboración `AfiliaciónCliente`



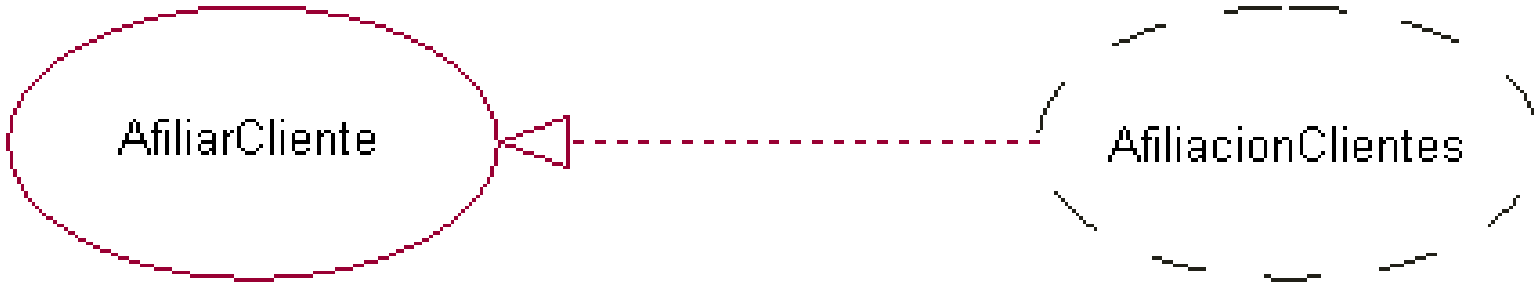
Colaboraciones Aspecto Comportamiento

El diagrama de interacción mostrado es la realización de aspectos del comportamiento de la colaboración `AfilicionCliente`.



Colaboraciones y Casos de Uso

- Los casos de uso son derivados y luego diseñados durante el proceso de comprensión del sistema.
- Durante la implementación final del sistema, los casos de uso son realizados como colaboraciones.
- La colaboración realizada a partir de casos de uso se puede describir más usando la modelación estructural y de comportamiento.

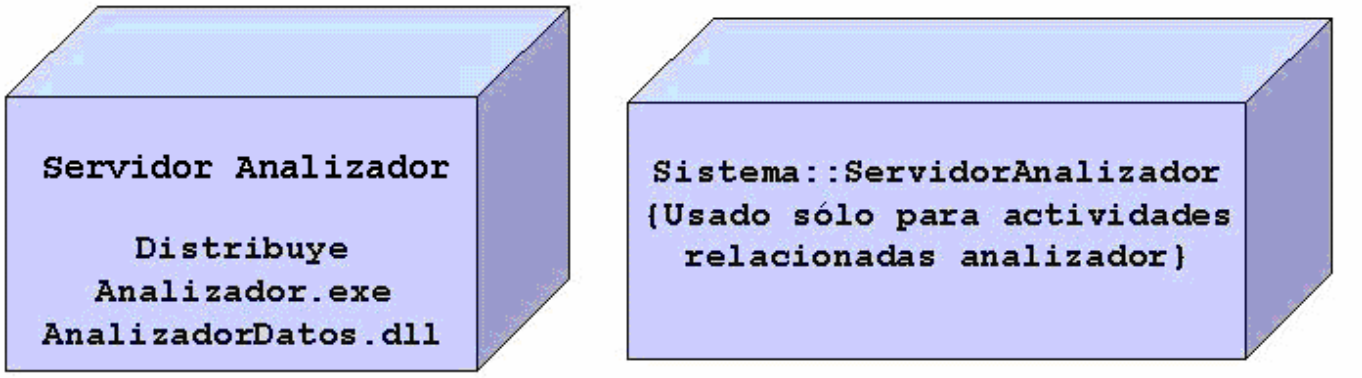


Diagramas de Despliegue

- Estos elementos contienen:
  - Componentes.
  - Procesos.
  - Objetos.
- Estos elementos de procesamiento en tiempo de ejecución son referidos como ***nodos*** representando un recurso computacional con capacidad de memoria y procesamiento.
- Un diagrama de despliegue es un grafo de nodos. Cada nodo es conectado para describir la asociación de comunicación.
- Los diagramas de componentes y de despliegue son llamados colectivamente ***diagramas de implementación***.

Nodos

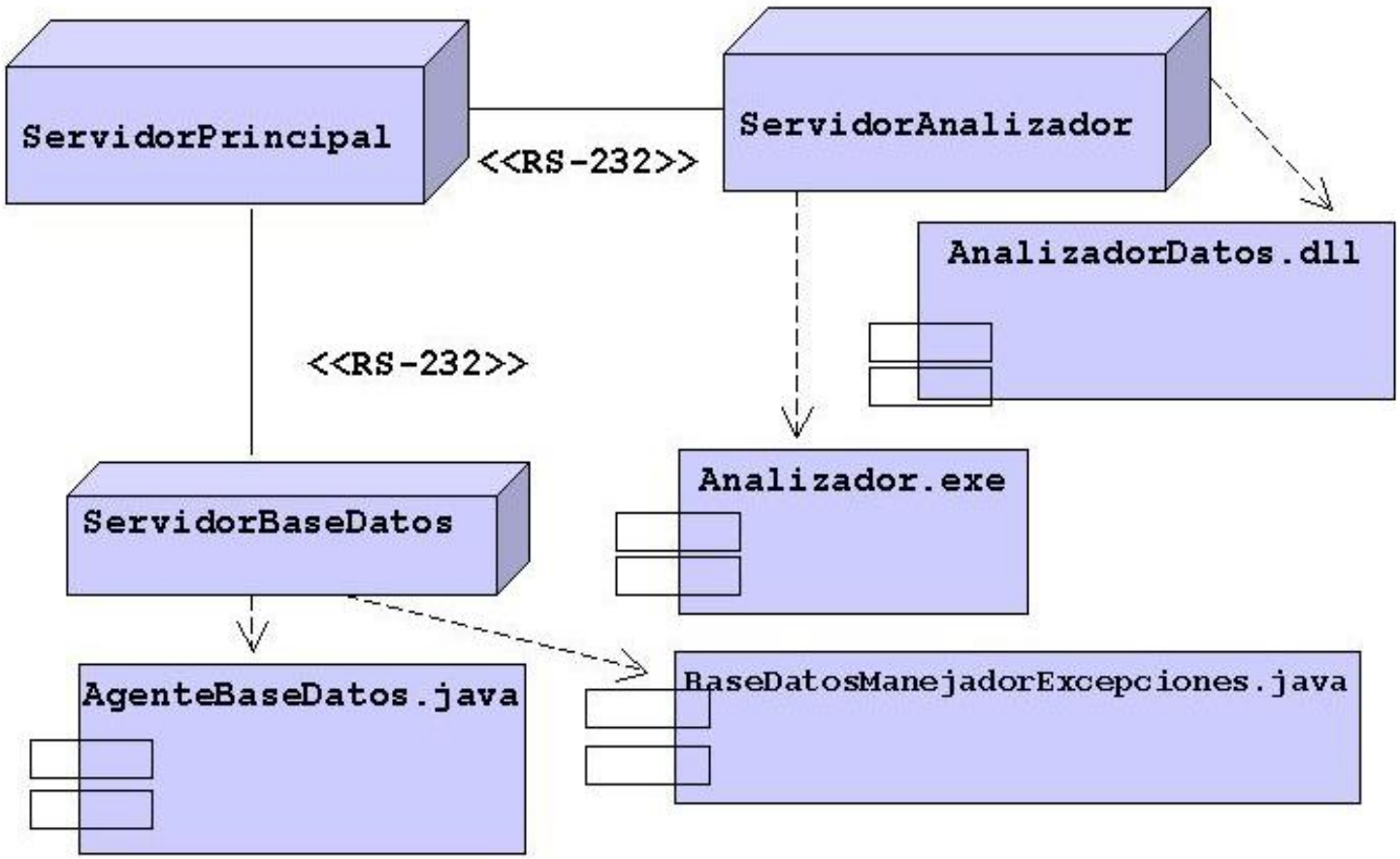
- Representan un recurso computacional que tiene memoria y capacidad de procesamiento.
- Cada nodo tiene un nombre, ya sea un nombre simple o nombre de ruta.



Nodos y Componentes

- Los Nodos y los Componentes participan en relaciones de dependencia, generalización y asociación.
- Ellos pueden participar en diagramas de interacción y es posible crear instancias de nodos y componentes.
- La diferencia entre ellos son:
  - Los componentes forman parte en la ejecución de un sistema, mientras que los nodos ejecutan componentes.
  - Los componentes caracterizan el empaquetamiento de elementos lógicos, tales como colaboraciones y clases, mientras que los nodos representan físicamente los componentes desplegados.
- Cuando un conjunto de componentes es desplegado en un nodo, el grupo es referido como una *unidad de distribución.*

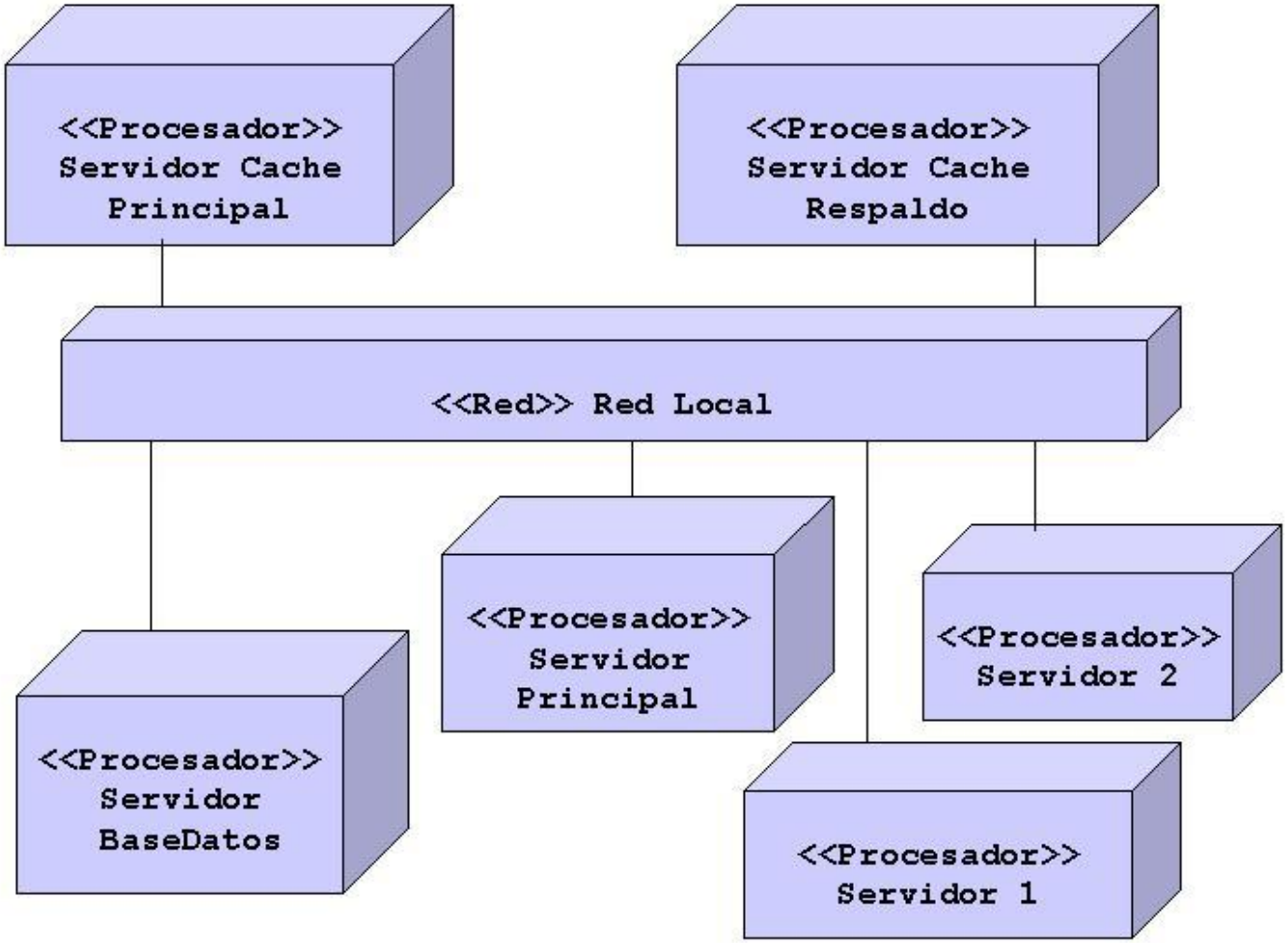
Nodos y Componentes: Un Ejemplo



Uso de los Diagramas de Despliegue

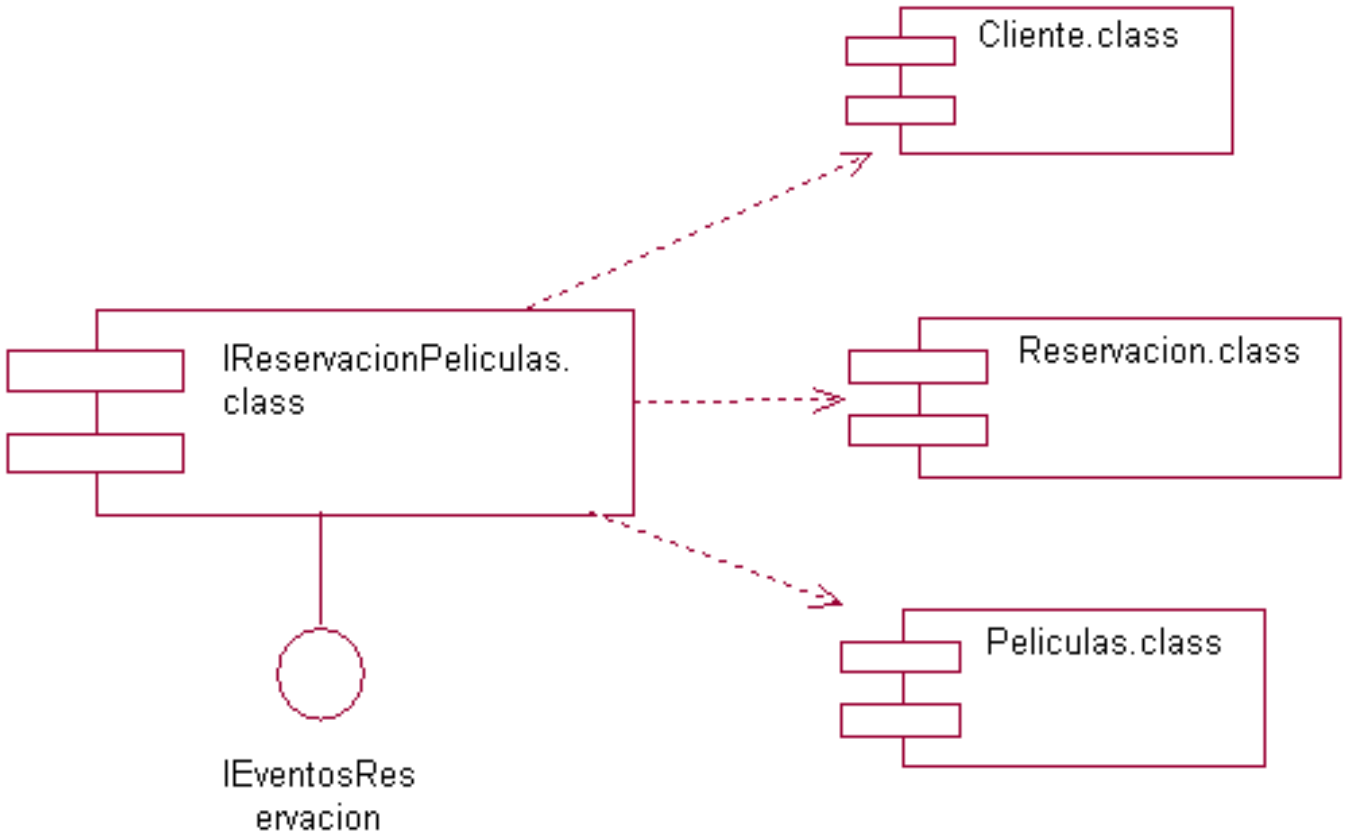
- Los diagramas de despliegue se usan para modelar lo siguiente:
  - Sistemas incorporados (embebidos).
  - Sistemas cliente-servidor.
  - Sistemas distribuidos.
- Para el modelamiento de un sistema cliente-servidor, el cliente y el servidor pueden ser mostrados como íconos de paquetes estereotipados, con los nodos embebidos dentro de ellos.
- Para los otros dos, las diferentes partes del sistema pueden ser mostradas como un diagrama de despliegue.

Diagramas de Despliegue



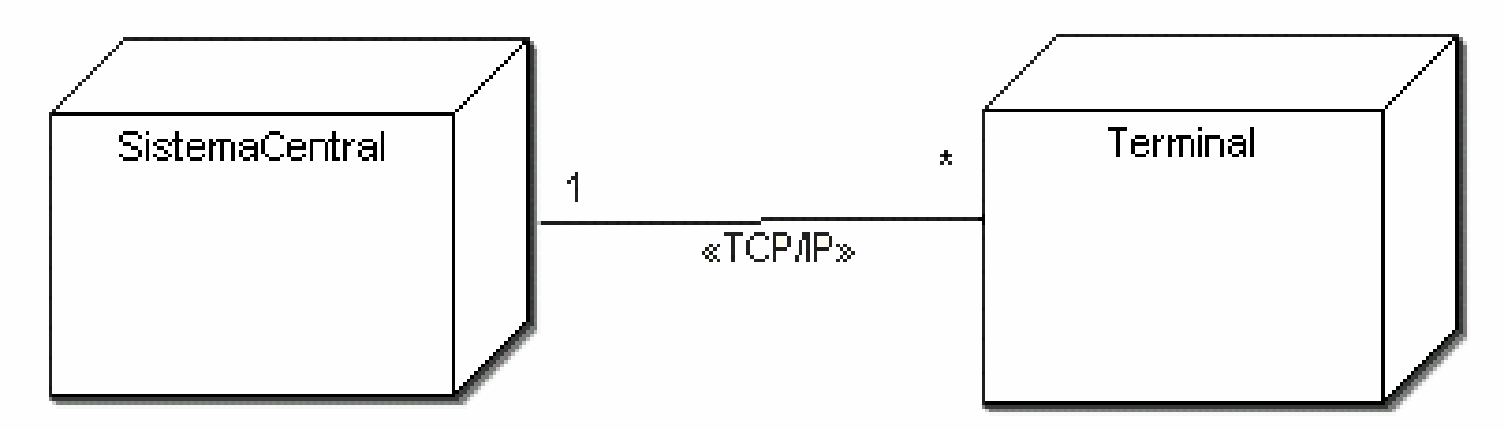
Caso de Estudio: Club de Video

Diagrama de componentes para reservar una película



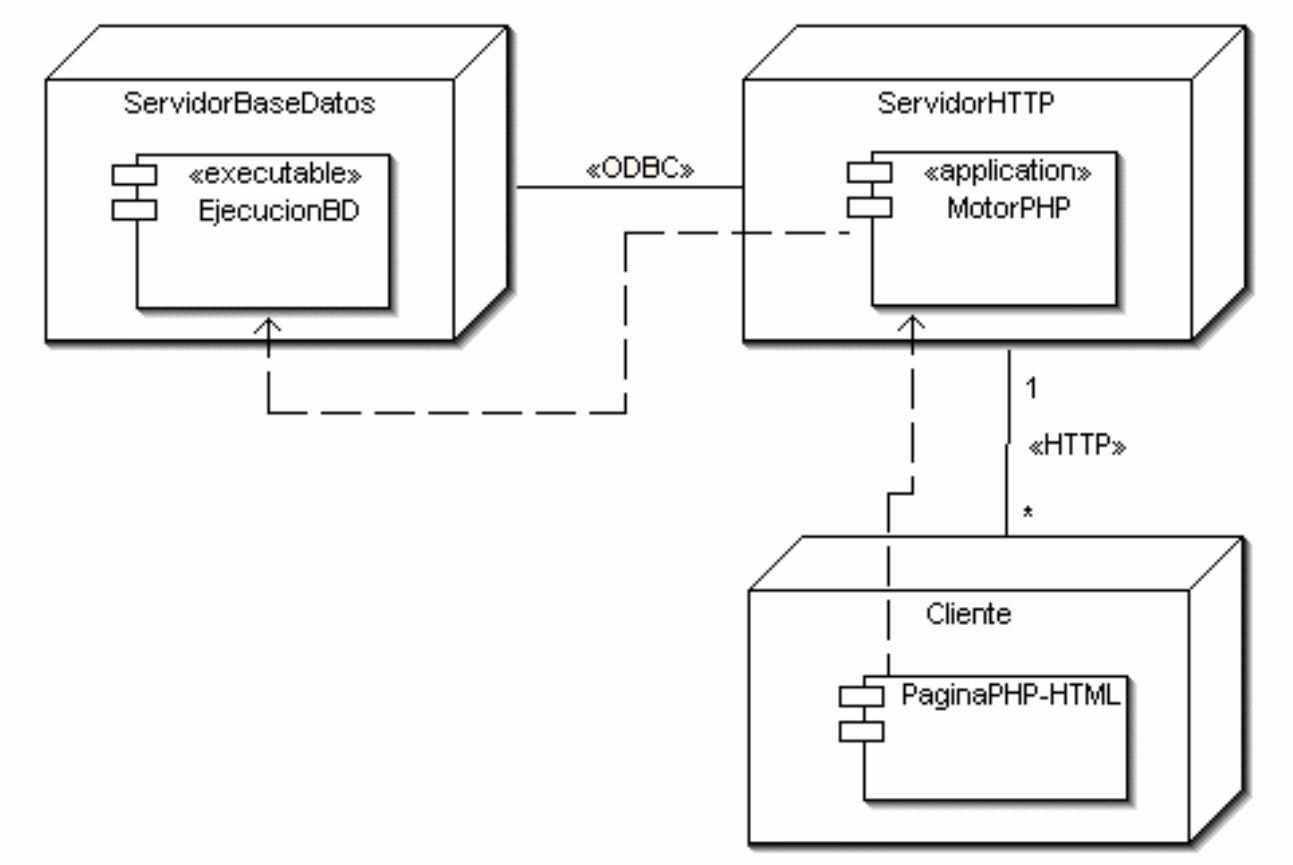
Caso de Estudio: Club de Video…1

Diagrama de despliegue para consultar películas desde los terminales remotos



Caso de Estudio: Club de Video…2

Diagrama de despliegue del modelo cliente servidor



Resumen

Ahora que ud. ha completado esta unidad, debe ser capaz de:

- Describir componentes, desplegado y colaboraciones.
- Discutir diagramas de componentes.
- Discutir la necesidad de Diagramas de Despliegue (deployment).