

- Planificación
 - ✓ Identificación. Título, descripción, versión, fecha,...
 - ✓ Documento de análisis
 - ✓ Especificación de requisitos o requerimientos
 - ✓ Diagramas de casos de uso
 - ✓ Escenarios y sub-escenarios
- Análisis y Diseño OO
 - ❖ **Clases, Objetos y mecanismos de colaboración**
 - ✓ **Diagramas de interacción (secuencia, colaboración)**
 - ✓ **Diagramas de Clases y patrones de diseño.**
 - ✓ **Diagramas de objetos**
 - ❖ **Modelado del comportamiento de clases y objetos**
 - ✓ Diagramas de actividades
 - ✓ Diagramas de estados
 - ❖ **Construcción del modelo físico**
 - ✓ Diagramas de componentes
 - ✓ Diagramas de despliegue
- Implementación
- Prueba y Mantenimiento: Informes de errores y Nueva especificación

4. AyDOO con UML

➤ Diagrama de Clases

- ❖ El más utilizado dentro de la fase de Análisis y Diseño. Uso:

- ✓ Para modelar el vocabulario de un sistema.
- ✓ Para modelar colaboraciones simples.

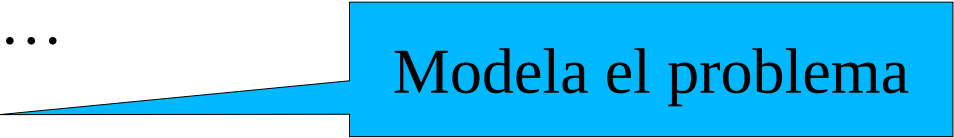

➤ Para realizar un Diagrama de clases, debe comprenderse claramente:

- ❖ Diferencia entre objeto vs clases

- ❖ Relaciones entre clases

- ✓ Asociación
- ✓ Composición
- ✓ Agregación
- ✓ Herencia
- ✓ Interfaces
- ✓ Clases Abstractas
- ✓ Paquetes

4. AyDOO con UML. Diagrama de clase

- Un diagrama de clases puede ilustrar varios niveles de detalles y diferentes objetivos, por ejemplo...
- Diagrama de Clases Conceptual 
 - ❖ Desarrollado por los analistas de sistema (negocio) para modelar los recursos del sistema (negocio)
 - ❖ Contiene tipos de datos y operaciones del negocio
 - ❖ Contiene solo clases del “negocio” – no clases de la implementación
- Diagrama de Clases de Implementación 
 - ❖ Desarrollado por los diseñadores para modelar los requisitos del código
 - ❖ Contiene detalles de programación tales como tipos de datos del lenguaje o de la base datos y modificadores de acceso
 - ❖ Se puede usar para generar el código

4. AyDOO con UML. Diagrama de clase

➤ Clase en UML

❖ Formado por tres partes

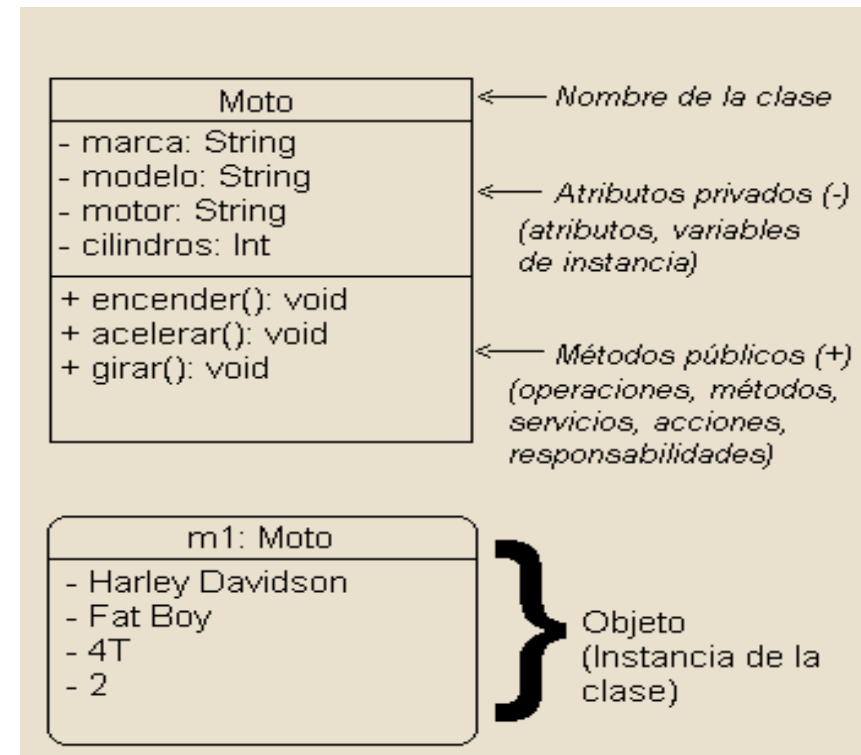
- ✓ Nombre
- ✓ Atributos
- ✓ Métodos

❖ Formato para los atributos y métodos

Visibilidad nombre_atributo: tipo_atributo [=valor_inicial {propiedades}]

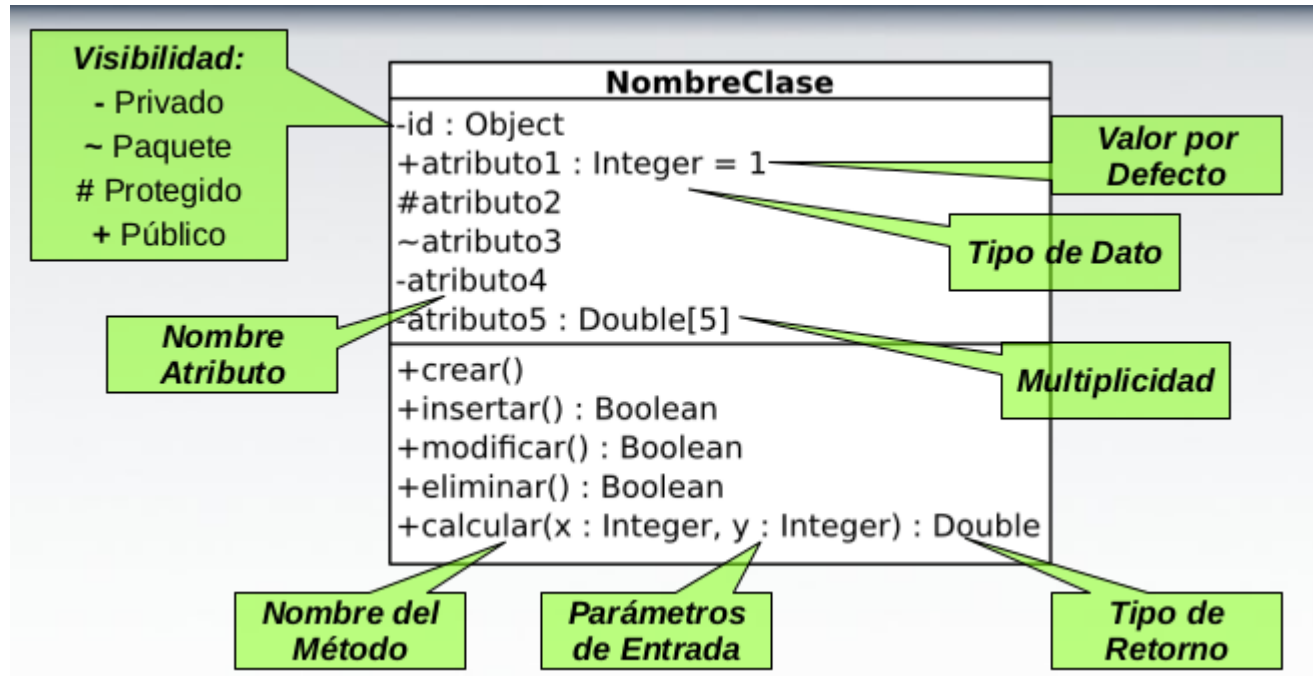
❖ Donde visibilidad

- ✓ (-) private
- ✓ (+) public
- ✓ (#) protected



4. AyDOO con UML. Diagrama de clase

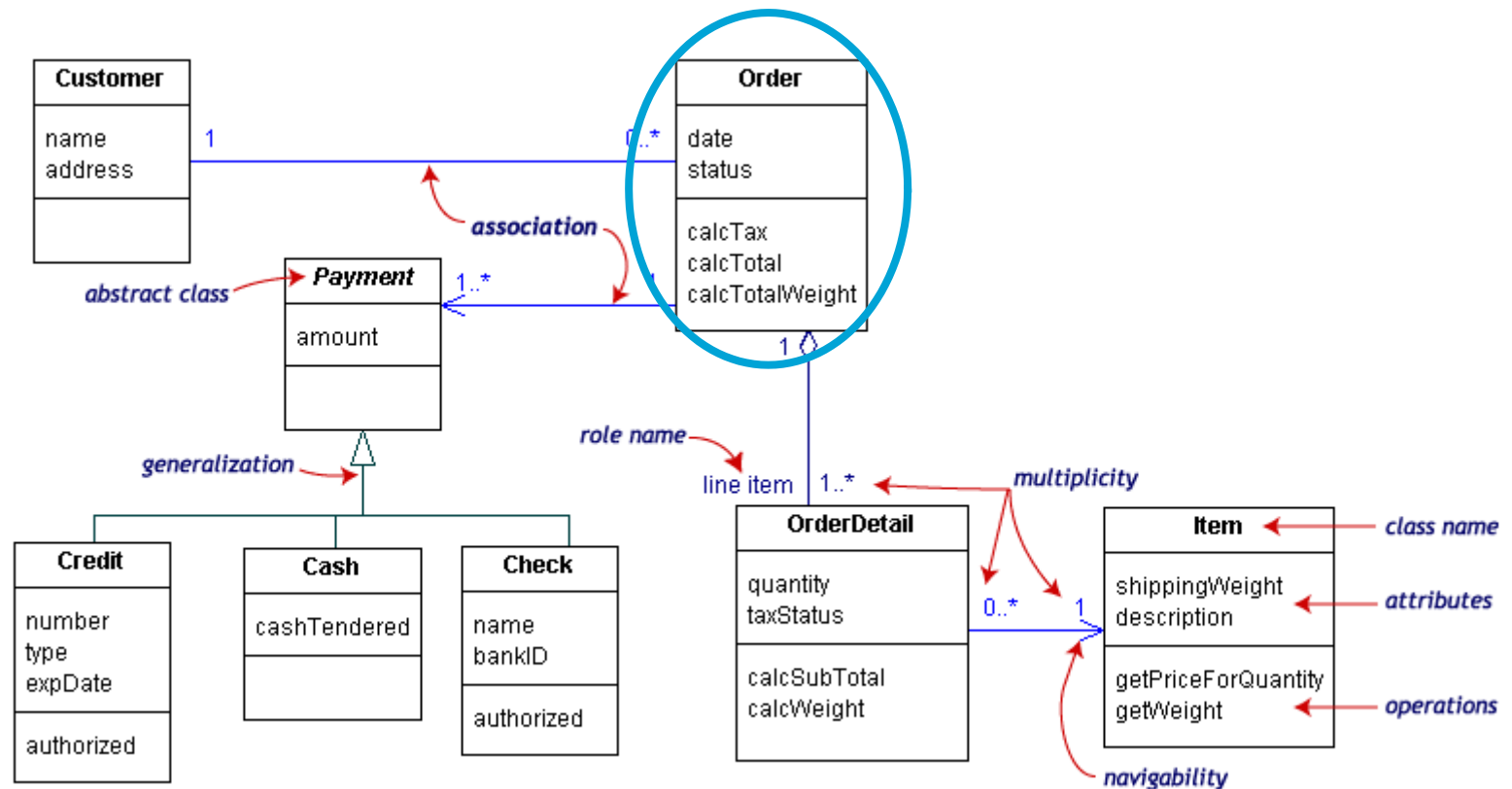
➤ Clase en UML



Obtenido de: UML Diagramas de Clases. Universidad de Los Andes

4. AyDOO con UML. Diagrama de clase

- Relaciones en el diagrama de clase:
 - ❖ Tiene: Agregación, Composición y Asociación
 - ❖ Es: Herencia, Herencia abstracta e Implementación



4. AyDOO con UML

➤ Diagrama de Clases

❖ Para realizar el Diagrama de clases, varias tareas:

1. Identificar objetos y clases
2. Identificar y depurar relaciones
3. Identificar atributos de objetos y relaciones
4. Ver si existen jerarquías de generalización
5. Comprobar los casos de uso (iterar)
6. Añadir a los diagramas cardinalidad.
7. Añadir y simplificar métodos

4. AyDOO con UML

➤ Diagrama de Clases

- ❖ ¿Cómo obtener las clases y las relaciones?
- ❖ Mediante un **análisis gramatical** para búsqueda de clases, identificando clases potenciales. Dado un enunciado del problema, buscar verbos y sustantivos.
 - ✓ **verbos**: pueden ser **operaciones o relaciones** entre clases.
 - ✓ **sustantivos**: pueden ser **clases o atributos**.
 - ✓ Hacer lista de sustantivos.
 - ✓ Seleccionar o rechazar cada sustantivo.
- ❖ De este modo, seremos capaces de
 - ✓ **detectar atributos**
 - ✓ **detectar operaciones**: análisis gramatical de verbos
 - ✓ **obtener clases y relaciones** entre clases

4. AyDOO con UML

➤ Diagrama de Clases

❖ Pasos para seleccionar las clases:

- ✓ Seleccionar nombres en los requisitos.
- ✓ Añadir clases adicionales procedentes de nuestro conocimiento del dominio.
- ✓ Eliminar redundancias.
- ✓ Eliminar clases irrelevantes.
- ✓ Eliminar clases vagas.
- ✓ Separar atributos.
- ✓ Separar métodos.
- ✓ Eliminar objetos de diseño.
- ✓ Resultado: Preparar diccionario de clases.

•Diagrama de Clase. Ejemplo Biblioteca

- Libros y revistas :
 - La biblioteca contiene libros y revistas, es posible que de cada libro haya varias copias.
 - Algunos de los libros se prestan por pequeños periodos de tiempo, el resto pueden ser prestados por periodos de hasta tres semanas
 - Solo el personal pueden tomar prestadas las revistas.
 - Es posible tomar prestados hasta seis elementos a la vez, el personal hasta doce.
 - Regularmente llegan nuevos ejemplares de libros y revistas, algunas veces se eliminan los antiguos.
 - Al final del año se envían las revistas para ser encuadernadas.
- Prestamos :
 - El sistema debe mantener información sobre los libros prestados y devueltos. El sistema actual ya lo hace.
 - Deben emitirse recordatorios cuando un libro se excede del tiempo de préstamo.
 - Es posible que en un futuro se permita a los usuarios extender el periodo de préstamo si un libro no esta reservado
- Búsquedas :
 - El sistema debe permitir a los usuarios la búsqueda de libros sobre un tópico, un autor, etc...
 - El sistema debe permitir comprobar si un libro está disponible.
 - El sistema debe permitir la reserva de libros.
 - Cualquiera puede realizar búsquedas en la biblioteca.

4. AyDOO con UML

•Diagrama de Clase. Ejemplo Biblioteca

- Libros y revistas :
 - La biblioteca contiene libros y revistas, es posible que de cada libro haya varias copias.
 - Algunos de los libros se prestan por pequeños periodos de tiempo, el resto pueden ser prestados por periodos de hasta tres semanas
 - Solo el personal pueden tomar prestadas las revistas.
 - Es posible tomar prestados hasta seis elementos a la vez, el personal hasta doce.
 - Regularmente llegan nuevos ejemplares de libros y revistas, algunas veces se eliminan los antiguos.
 - Al final del año se envían las revistas para ser encuadernadas.
- Prestamos :
 - El sistema debe mantener información sobre los libros prestados y devueltos. El sistema actual ya lo hace.
 - Deben emitirse recordatorios cuando un libro se excede del tiempo de préstamo.
 - Es posible que en un futuro se permita a los usuarios extender el periodo de préstamo si un libro no esta reservado
- Búsquedas :
 - El sistema debe permitir a los usuarios la búsqueda de libros sobre un tópico, un autor, etc...
 - El sistema debe permitir comprobar si un libro está disponible.
 - El sistema debe permitir la reserva de libros.
 - Cualquiera puede realizar búsquedas en la biblioteca.

•Diagrama de Clase. Ejemplo Biblioteca

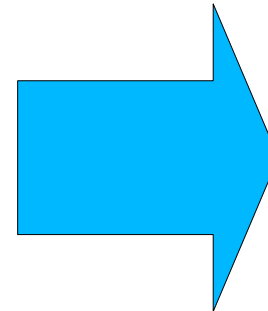
- Analizando la lista de clases
 - Biblioteca
 - Libros
 - Revistas
 - Pequeño periodo de tiempo
 - Semanas
 - Persona
 - Elementos
 - Ejemplares
 - Sistema
 - Recordatorio
 - Tópico
 - Autor
 - Reserva

4. AyDOO con UML

•Diagrama de Clase. Ejemplo Biblioteca

– Refinamiento

- ~~XXXXXXXX~~ Sistema
- Libros Elementos
- Revistas Elementos
- Copias Ejemplares
- ~~Pequeño periodo de tiempo~~
- ~~Semanas~~
- Persona **Miembro Personal**
- ~~Recordatorio~~
- ~~Tópico~~
- ~~Autor~~
- ~~Reserva~~
- ~~Búsqueda~~



- Libros
- Revistas
- Copias
- Miembro
- Personal

(Redundancia) (Vagas) (XXXXXXXXXX) Atributos Conocimiento

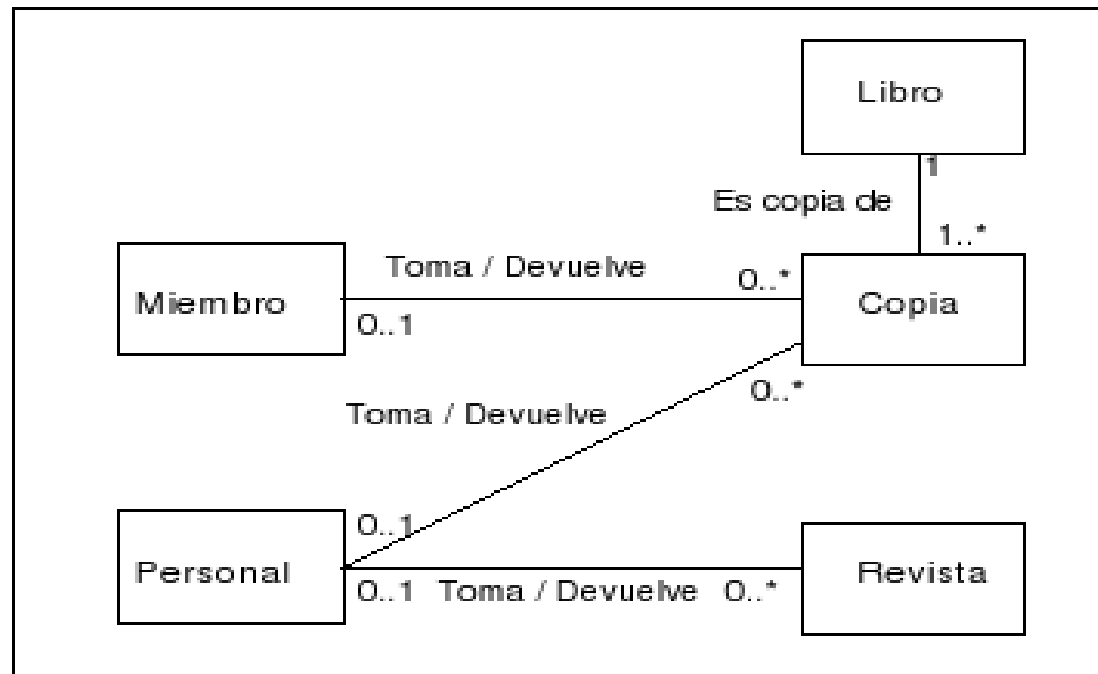
Diseñar de Clases

- **Relaciones. Fases:**
 1. Seleccionar verbos relacionales en los requisitos.
 2. Añadir relaciones adicionales procedentes de nuestro conocimiento del dominio.
 3. Eliminar relaciones de diseño o entre clases eliminadas.
 4. Eliminar eventos transitorios.
 5. Reducir relaciones ternarias.
 6. Eliminar relaciones redundantes o derivadas.
 7. Añadir relaciones olvidadas.
 8. Definir la multiplicidad de cada relación.

4. AyDOO con UML

Diseñar de Clases

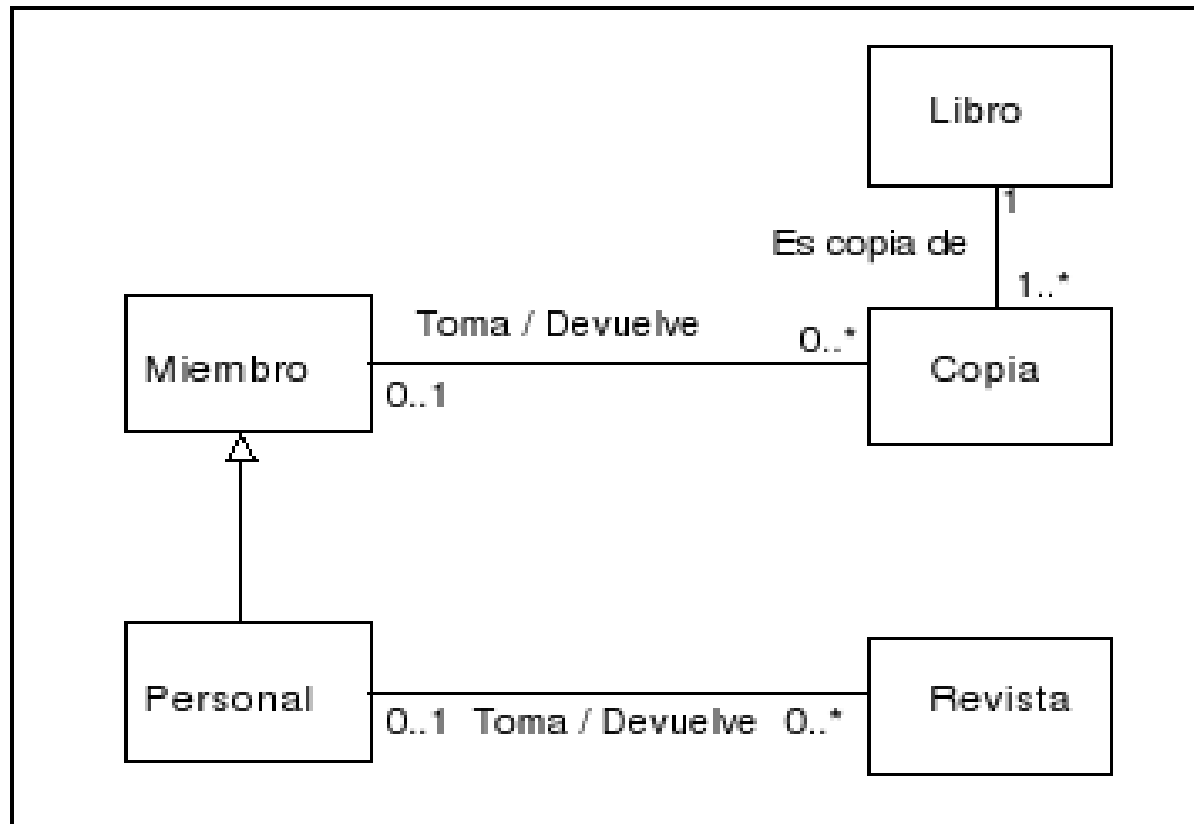
- **Ejemplo de la biblioteca. Relaciones.**
 1. Una copia lo es de un libro
 2. Un miembro puede devolver o tomar un libro
 3. Un miembro del personal puede devolver o tomar un libro
 4. Un miembro del personal puede devolver o tomar una revista



4. AyDOO con UML

Diseñar de Clases

- **Ejemplo de la biblioteca. Refinamiento**



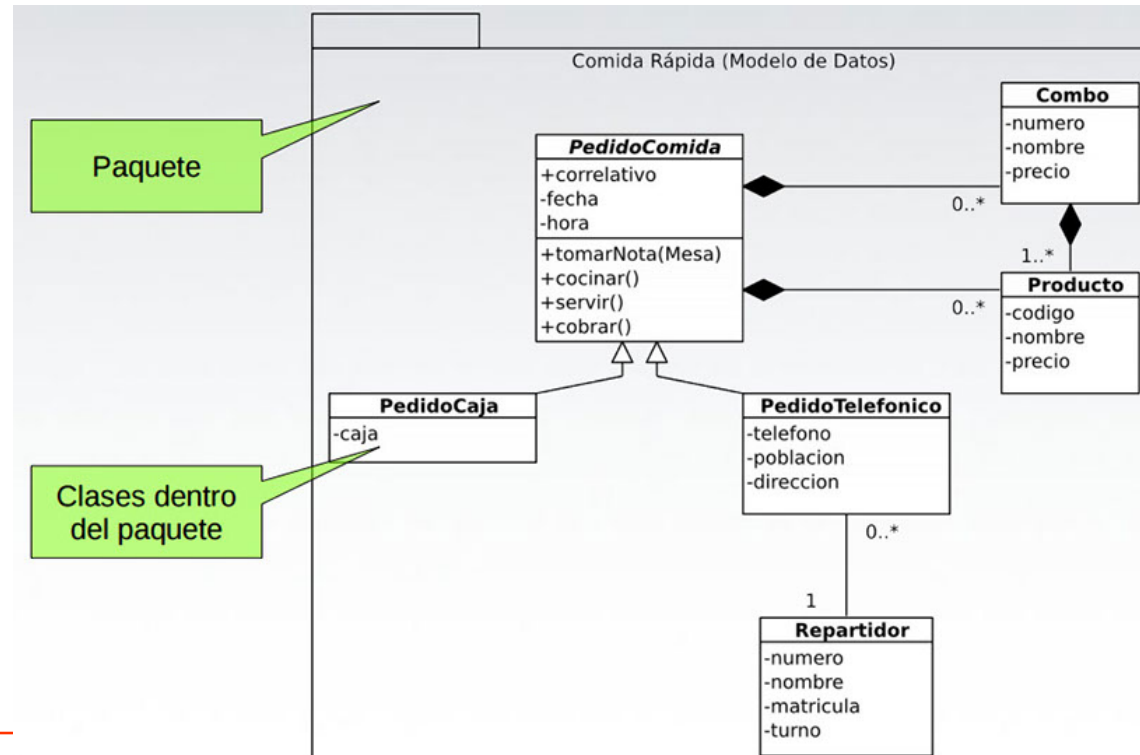
Más ejemplos en documentos adicionales

4. AyDOO con UML

- Análisis y Diseño Orientado a Objetos mediante UML
 - Modelo Estructural
 - Diagrama de clases
 - Diagrama de objetos
 - Modelo de comportamiento
 - Diagrama de Interacción
 - Diagrama de Colaboración
 - Diagrama de Secuencia
 - Diagrama de Actividades
 - Diagrama de Componentes
 - Diagrama de Despliegue
 - Diagrama de Estados

4. AyDOO con UML. Diagrama de paquetes

- Un paquete es un mecanismo utilizado para agrupar elementos de UML que permite organizar los elementos modelados con UML, facilitando de ésta forma el manejo de los modelos de un sistema complejo
- Define un espacio de nombres: Dos elementos de UML pueden tener el mismo nombre pero encontrarse en paquetes distintos

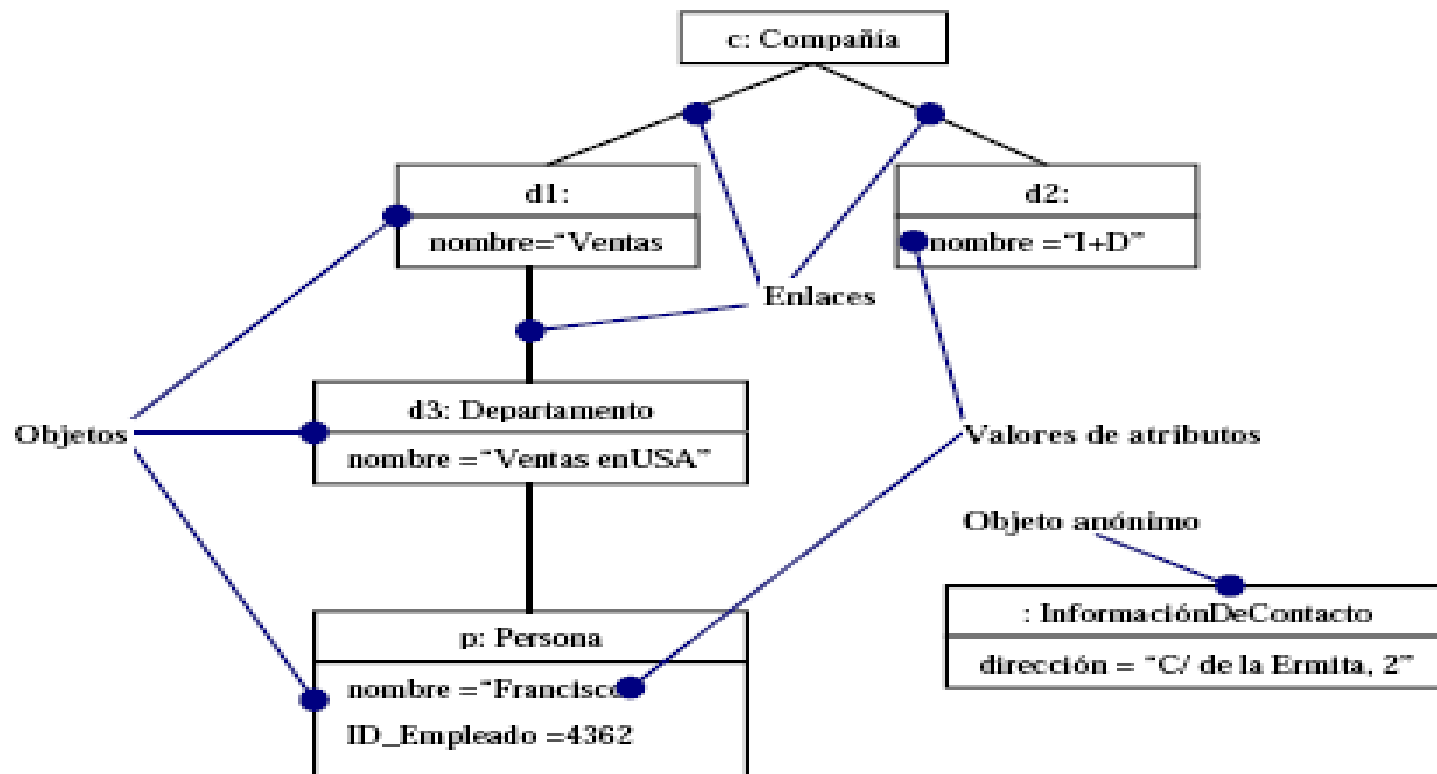


4. AyDOO con UML. Diagrama de paquetes

- Entre paquetes pueden existir relaciones de dependencia y generalización.
- Las dependencias entre paquetes denotan que algún elemento de un paquete depende de los elementos en otro paquete. Existen diferentes tipos de relaciones de dependencia entre paquetes:
 - ❖ Importación: Modelado como una dependencia estereotipada con `<<import>>`.
 - ❖ Acceso: Modelado como una dependencia estereotipada con `<<access>>`.
 - ❖ Combinación: Modelado como una dependencia estereotipada con `<<merge>>`.
 - ❖ Exportación: Modelado implícitamente a través de la visibilidad pública en los elementos del paquete. No se exporta explícitamente a algún paquete.

4. AyDOO con UML

- Modelo Estructural. Diagrama de objetos
 - ❖ Un diagrama de objetos contiene un conjunto de instancias de los elementos encontrados en un diagrama de clases.
 - ❖ Es decir, un diagrama de objetos expresa la parte estática de una interacción, consistiendo en los objetos que colaboran pero sin ninguno de los mensajes enviados entre ellos.



4. AyDOO con UML

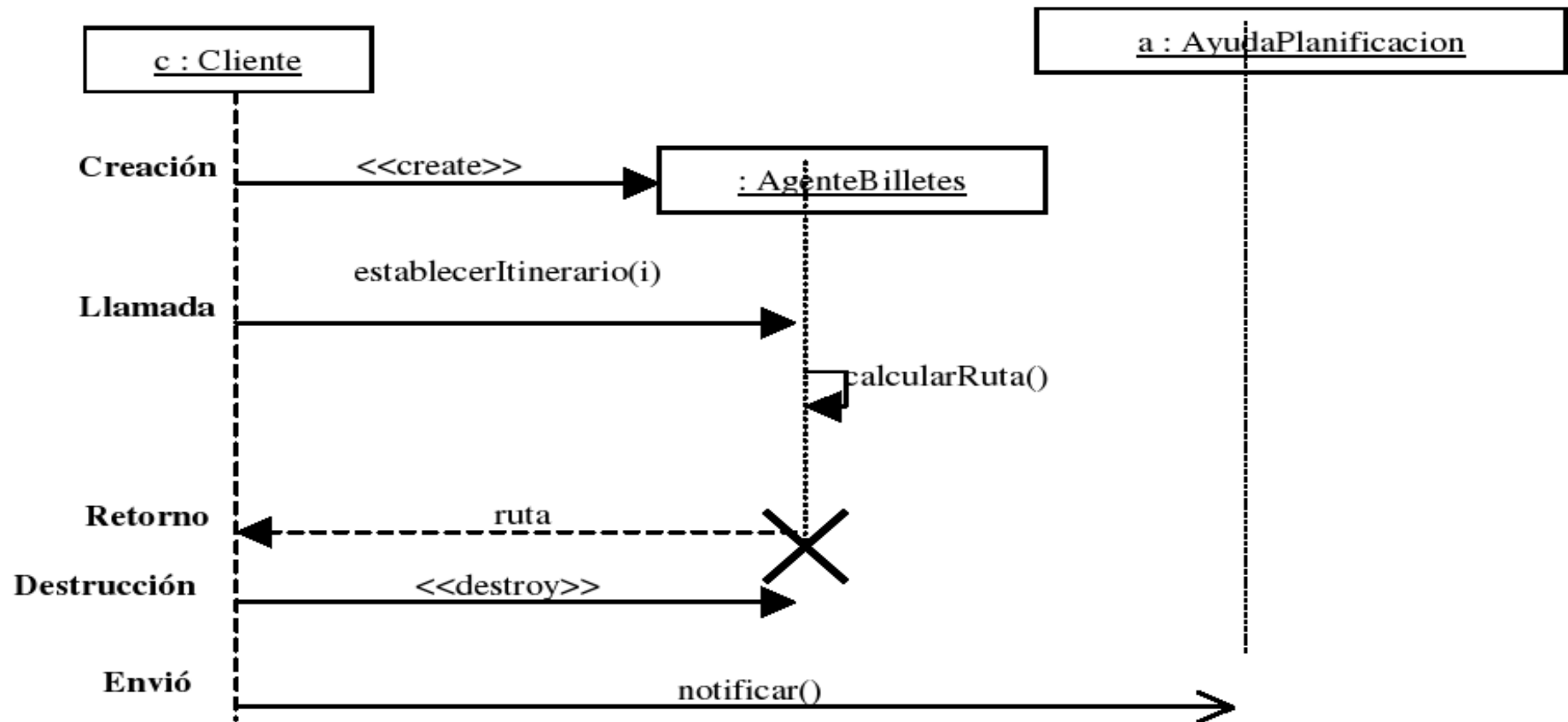
- Análisis y Diseño Orientado a Objetos mediante UML
 - Modelo Estructural
 - Diagrama de clases
 - Diagrama de objetos
 - Modelo de comportamiento
 - **Diagrama de Interacción**
 - **Diagrama de Colaboración**
 - **Diagrama de Secuencia**
 - Diagrama de Actividades
 - Diagrama de Componentes
 - Diagrama de Despliegue
 - Diagrama de Estados

4. AyDOO con UML

- Interacción: Es un comportamiento que implica un intercambio de mensajes entre varios objetos en un contexto determinado con un objetivo determinado
 - ❖ Se utilizan para expresar los aspectos dinámicos de las colaboraciones
 - ❖ Las interacciones se llevan a cabo entre objetos no entre clases
 - ❖ En UML se modelan varios tipos de acciones:
 - ✓ Llamada. Invoca una operación sobre un objeto; un objeto puede enviarse un mensaje a sí mismo, lo que resulta en la invocación local de una operación.
 - ✓ Retorno. Devuelve un valor al invocador.
 - ✓ Envío. Envía una señal a un objeto.
 - ✓ Creación. Crea un objeto.
 - ✓ Destrucción. Destruye un objeto; un objeto puede “suicidarse” al destruirse a sí mismo.

4. AyDOO con UML

- Interacción:
 - Ejemplo de Mensaje



4. AyDOO con UML

➤ Diagrama de interacción

❖ Los mensajes se extraen de los casos de uso (escenarios). Pueden ser de los siguientes tipos

- ✓ Señales
- ✓ Entradas
- ✓ Decisiones
- ✓ Interrupciones
- ✓ Transiciones
- ✓ Acciones externas
- ✓ Condiciones de error

❖ Resultados:

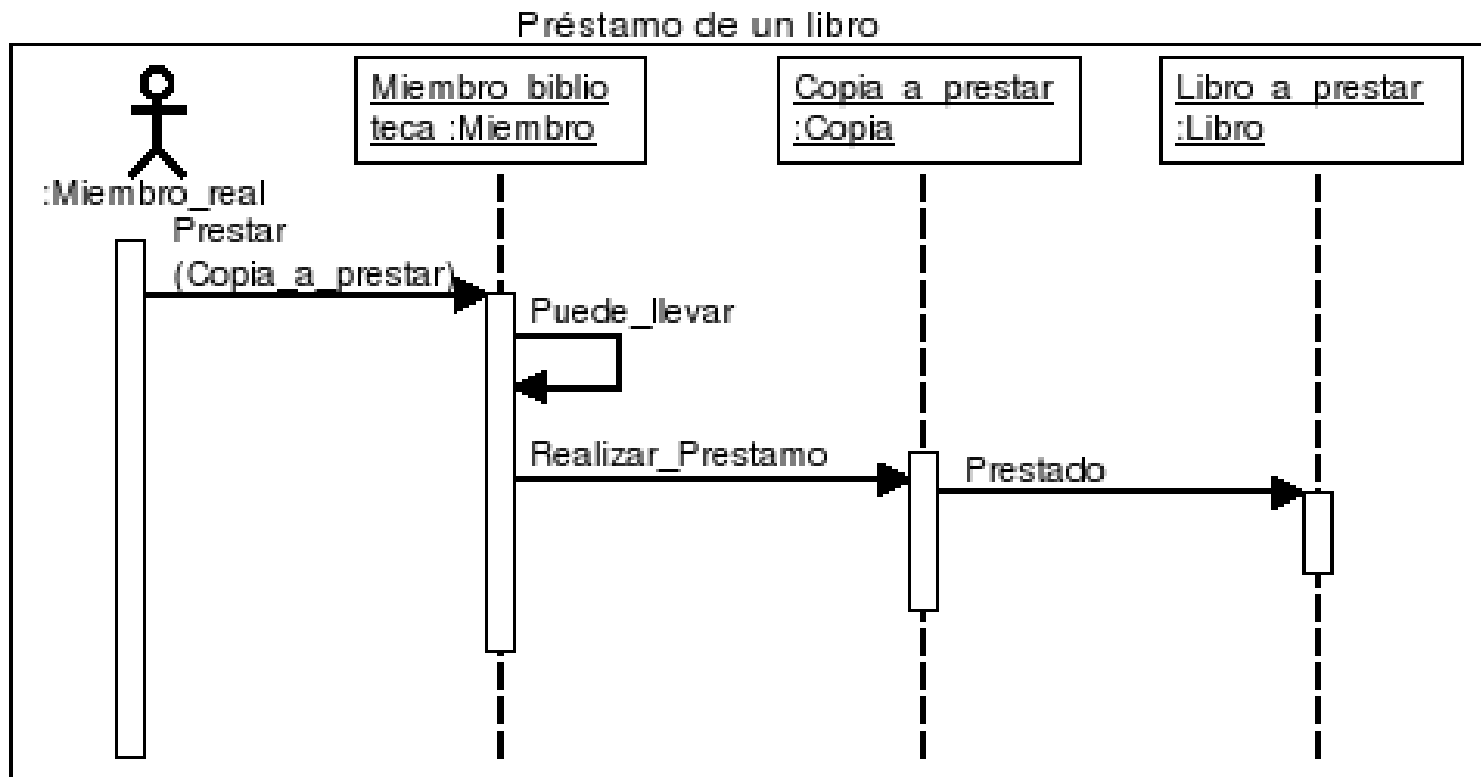
- ✓ Diagramas de secuencia
- ✓ Diagrama de colaboración.

➤ Diagrama de secuencia

- ❖ Un diagrama de secuencia destaca la ordenación temporal de los mensajes, es decir, coloca los objetos que participan en la interacción y los mensajes que estos objetos envían y reciben.
- ❖ Características:
 - ✓ En el eje X aparecen los objetos
 - ✓ En el eje Y el orden de sucesión en el tiempo, desde arriba hasta abajo.
- ❖ Esto ofrece al lector una señal visual clara del flujo de control a lo largo del tiempo.
- ❖ Se extraen de los casos de uso, escenarios.
- ❖ Se pueden modelar
 - ✓ En la fase de análisis de forma generalizada
 - ✓ En la fase de diseño, especificando los objetos exactos que forman parte de la interacción.

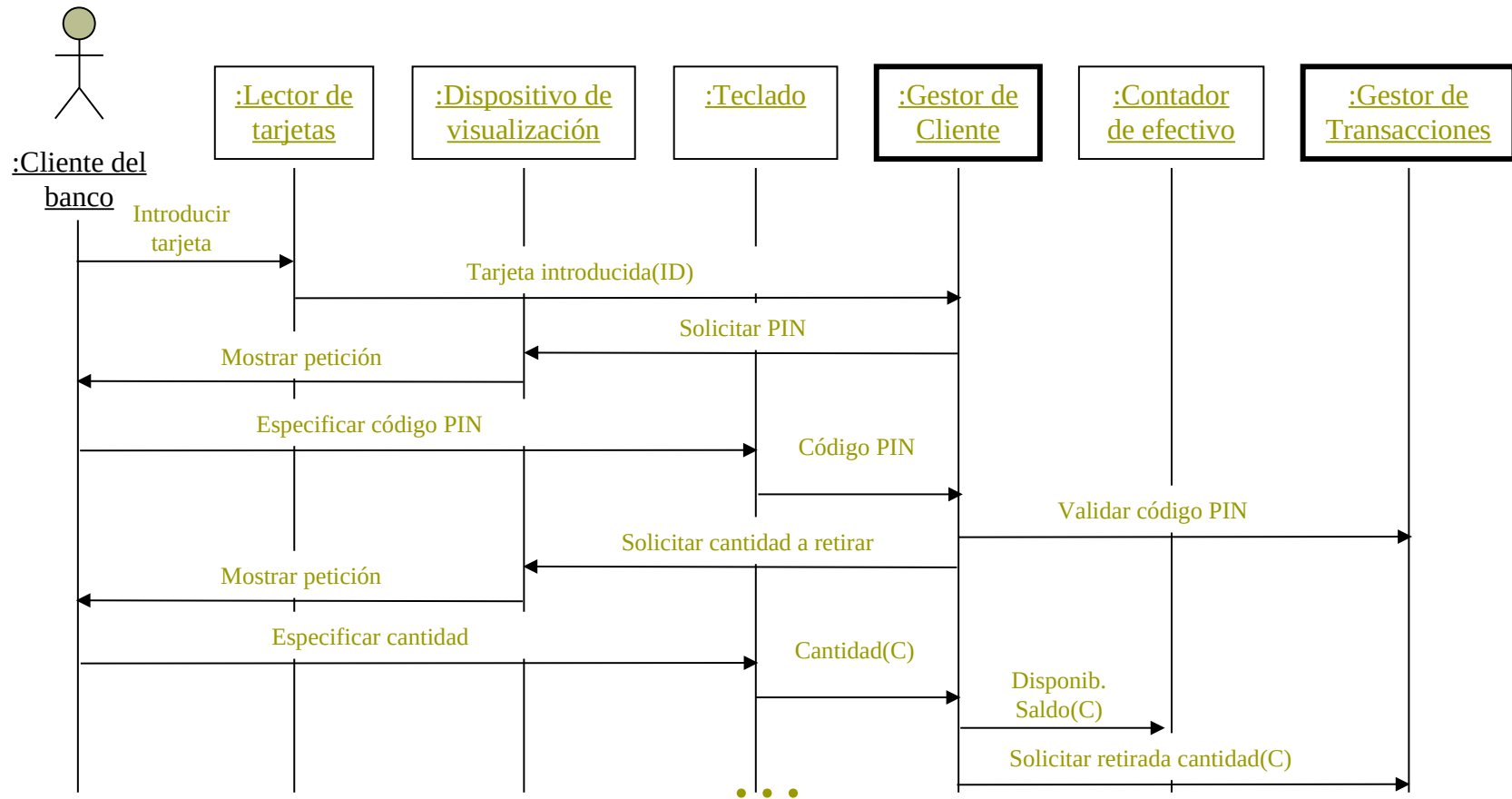
4. AyDOO con UML

- Diagrama de secuencia. Ejemplo



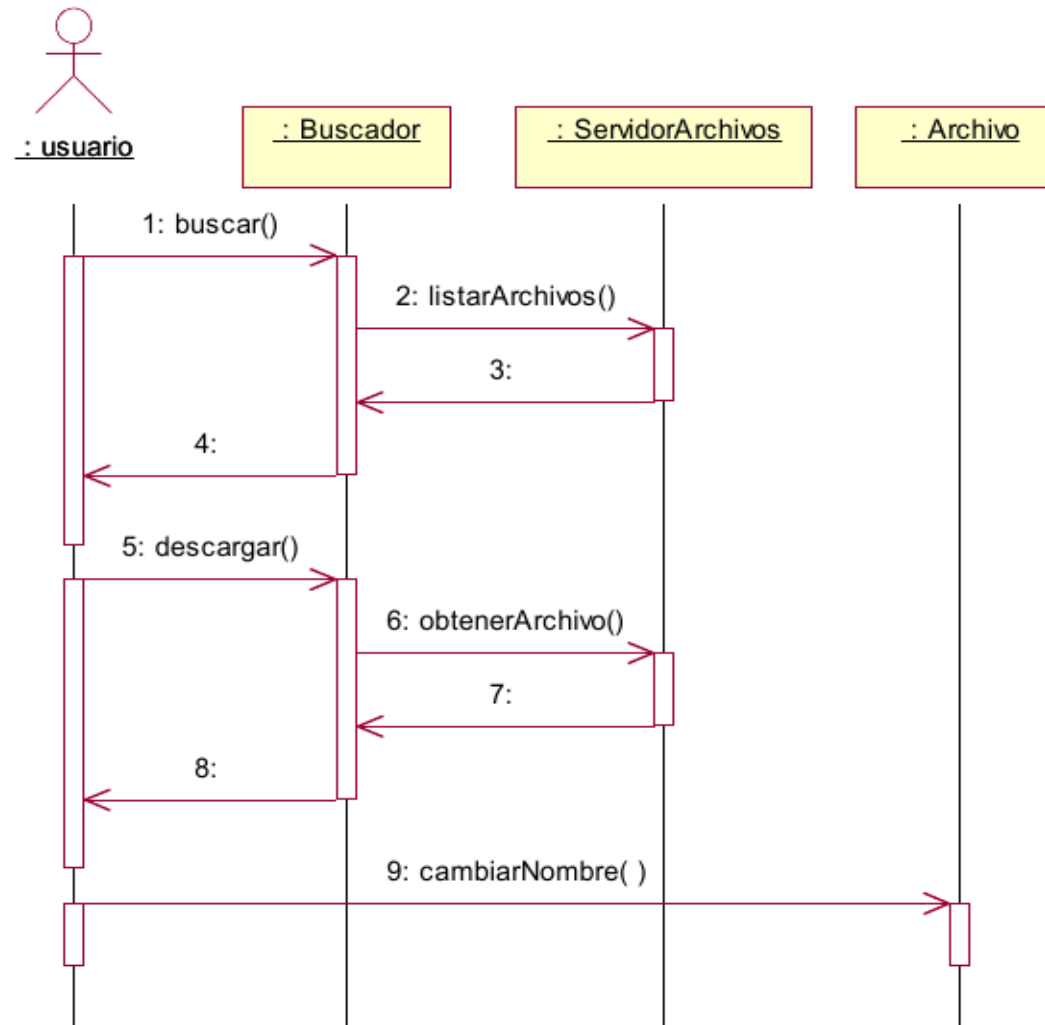
4. AyDOO con UML

Diagrama de secuencia. Ejemplo



4. AyDOO con UML

Diagrama de secuencia. Ejemplo



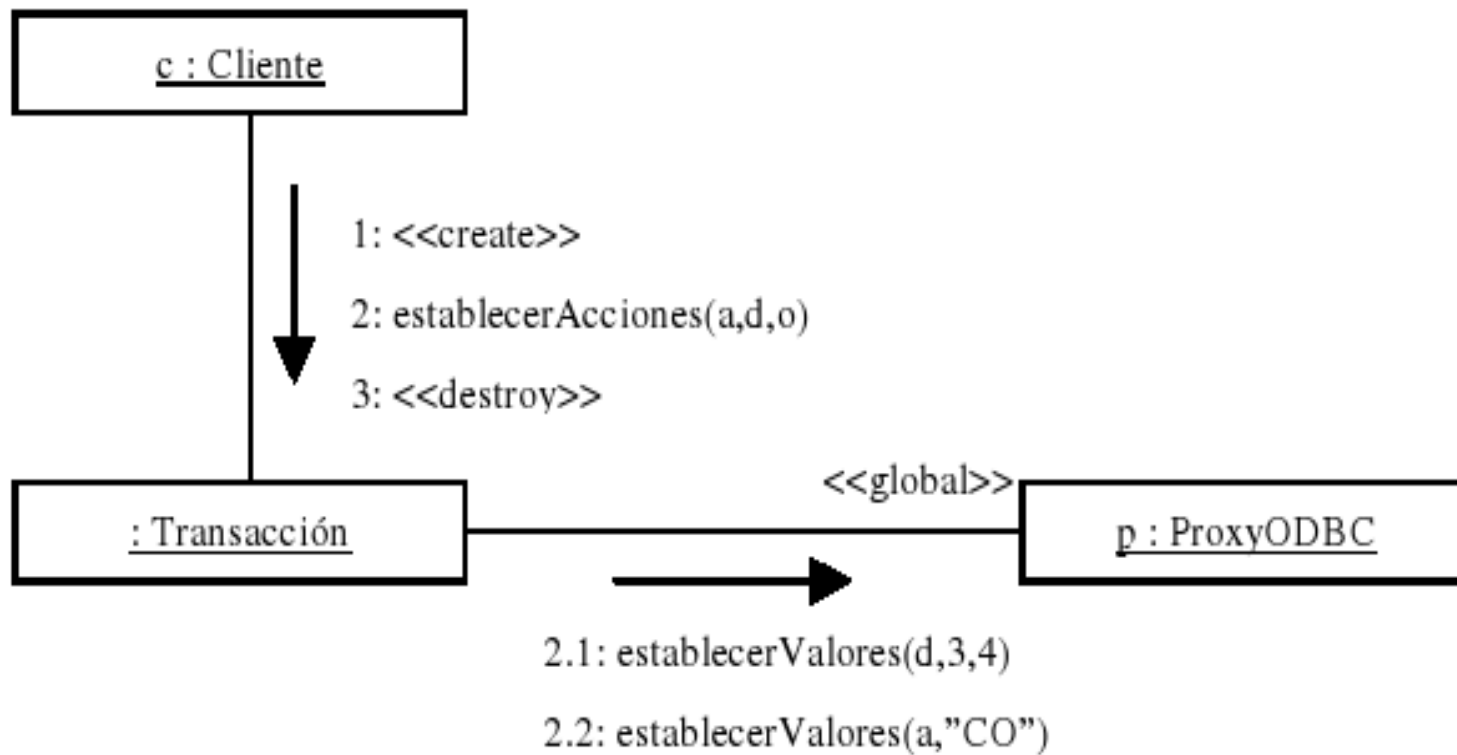
4. AyDOO con UML

➤ Diagrama de colaboración

- ❖ Un diagrama de colaboración destaca la organización de los objetos que participan en una interacción, es decir, coloca los objetos que participan en la colaboración como nodos del grafo y se muestran los enlaces entre ellos.
- ❖ Por último, estos enlaces se adornan con los mensajes que envían y reciben los objetos. Esto da al lector una señal visual clara del flujo de control en el contexto de la organización estructural de los objetos que colaboran.

4. AyDOO con UML

- Diagrama de colaboración



4. AyDOO con UML

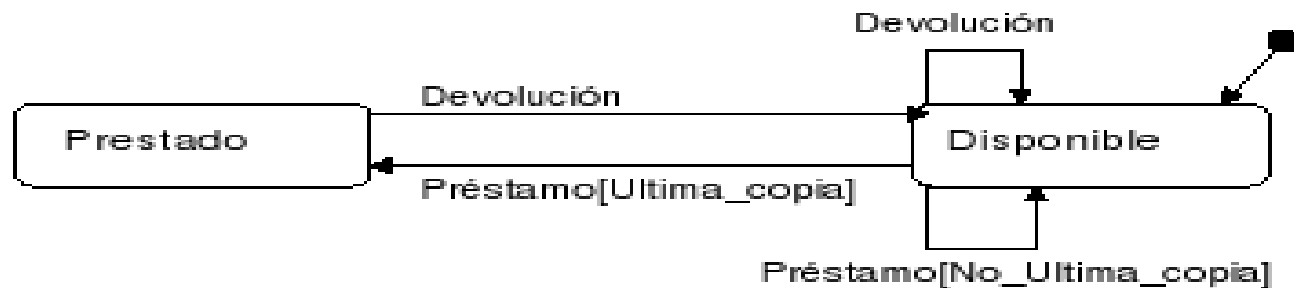
- Diagrama de secuencia y colaboración
 - Los diagramas de secuencia tienen dos características que los distinguen de los diagramas de colaboración:
 - En primer lugar, está la línea de vida de un objeto, es la línea vertical discontinua que representa la existencia de un objeto a lo largo de un periodo de tiempo.
 - En segundo lugar está el foco de control que es un rectángulo estrecho situado sobre la línea de vida que representa el período de tiempo durante el cual un objeto ejecuta una acción, bien sea directamente o a través de un procedimiento subordinado.

4. AyDOO con UML

- Análisis y Diseño Orientado a Objetos mediante UML
 - Modelo Estructural
 - Diagrama de clases
 - Diagrama de objetos
 - Modelo de comportamiento
 - Diagrama de Caso de Uso
 - Diagrama de Interacción
 - Diagrama de Colaboración
 - Diagrama de Secuencia
 - Otros diagramas
 - **Diagrama de Actividades**
 - **Diagrama de Estados**
 - **Diagrama de Componentes**
 - **Diagrama de Despliegue**

4. AyDOO con UML

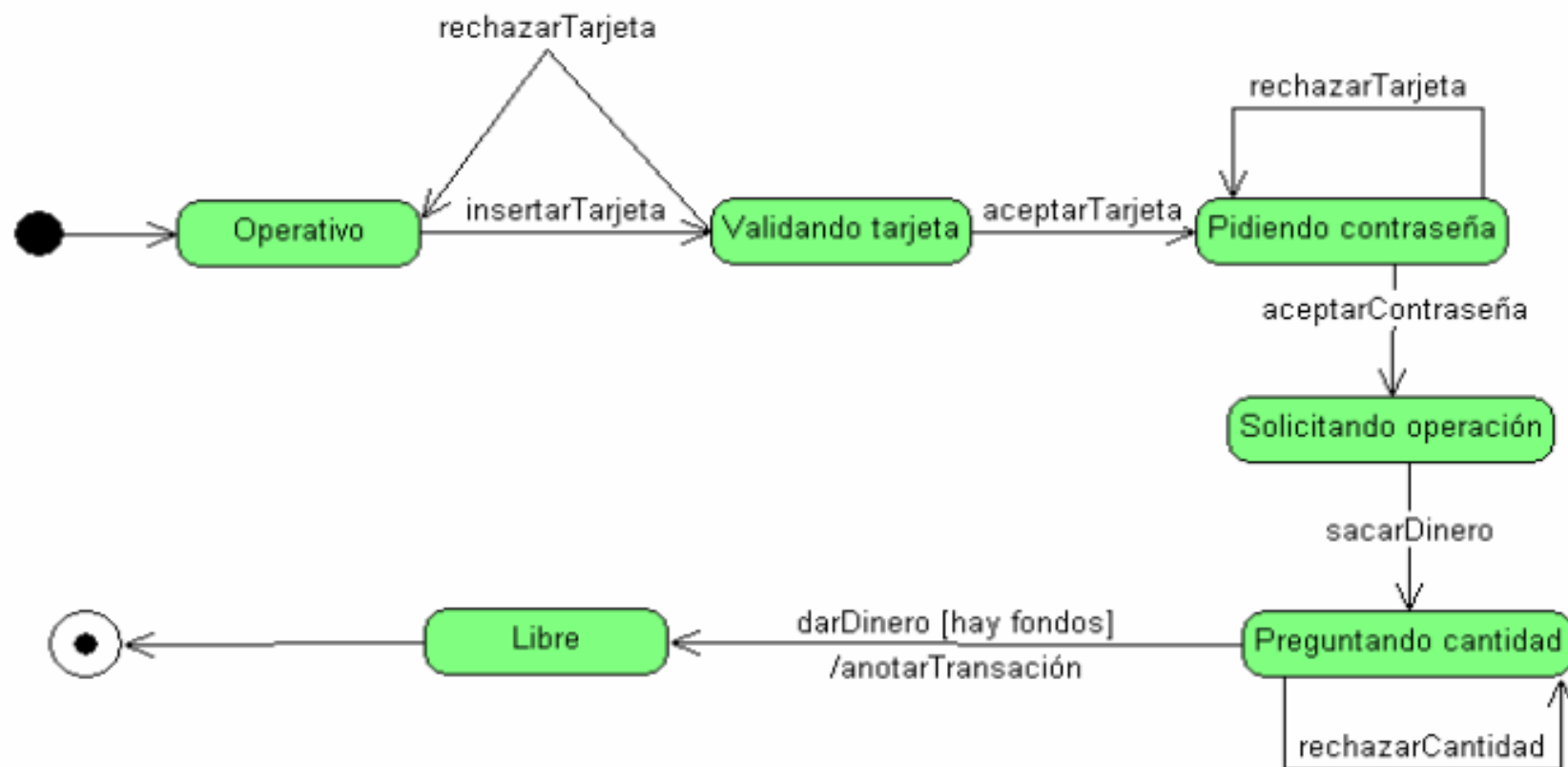
- **Diagramas de Estados**
 - Muestran una maquina de estados compuesta por estados, transiciones, eventos y actividades. Estos diagramas cubren la vista dinámica de un sistema y son muy importantes a la hora de modelar el comportamiento de una interfaz, clase o colaboración.



4. AyDOO con UML

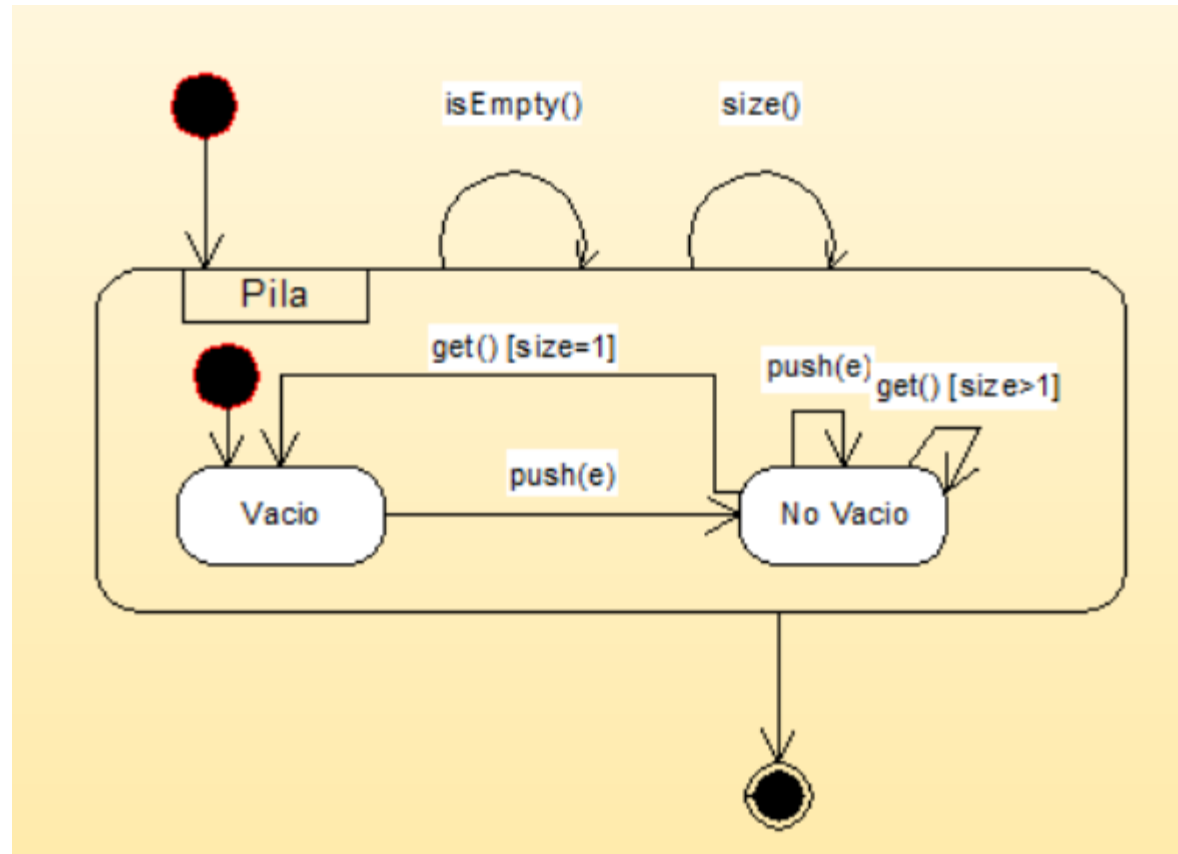
- Diagramas de Estados

—



4. AyDOO con UML

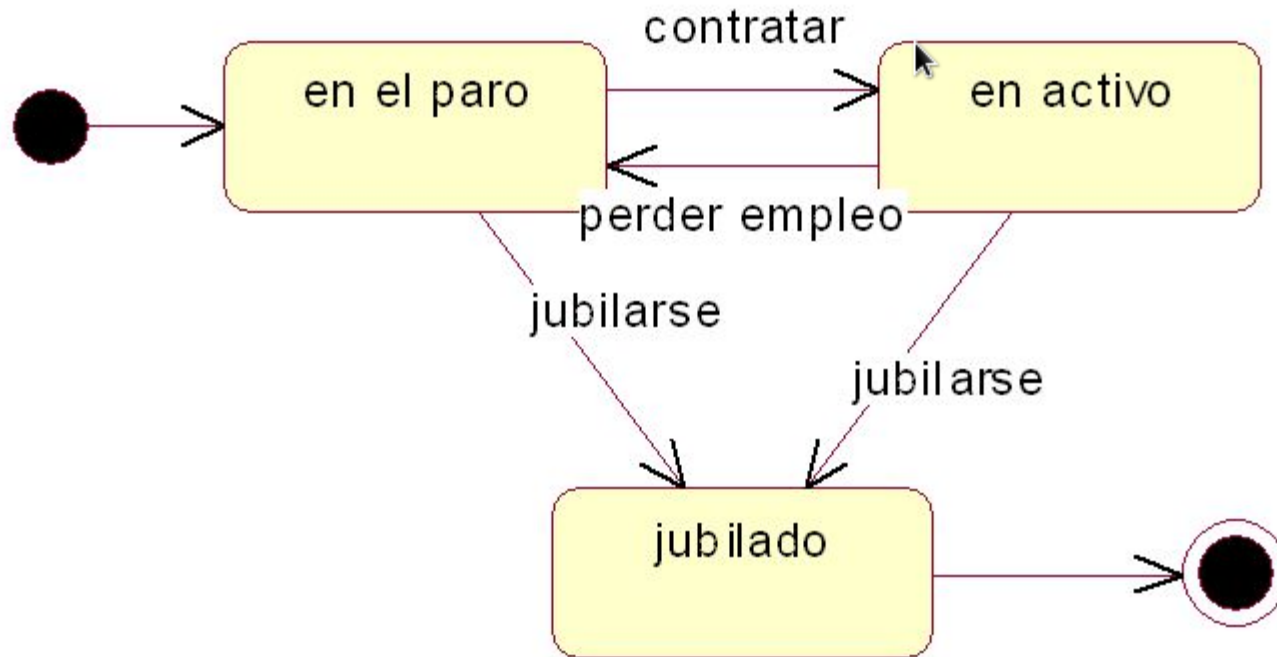
- **Diagramas de Estados**
 - **De una pila**



4. AyDOO con UML

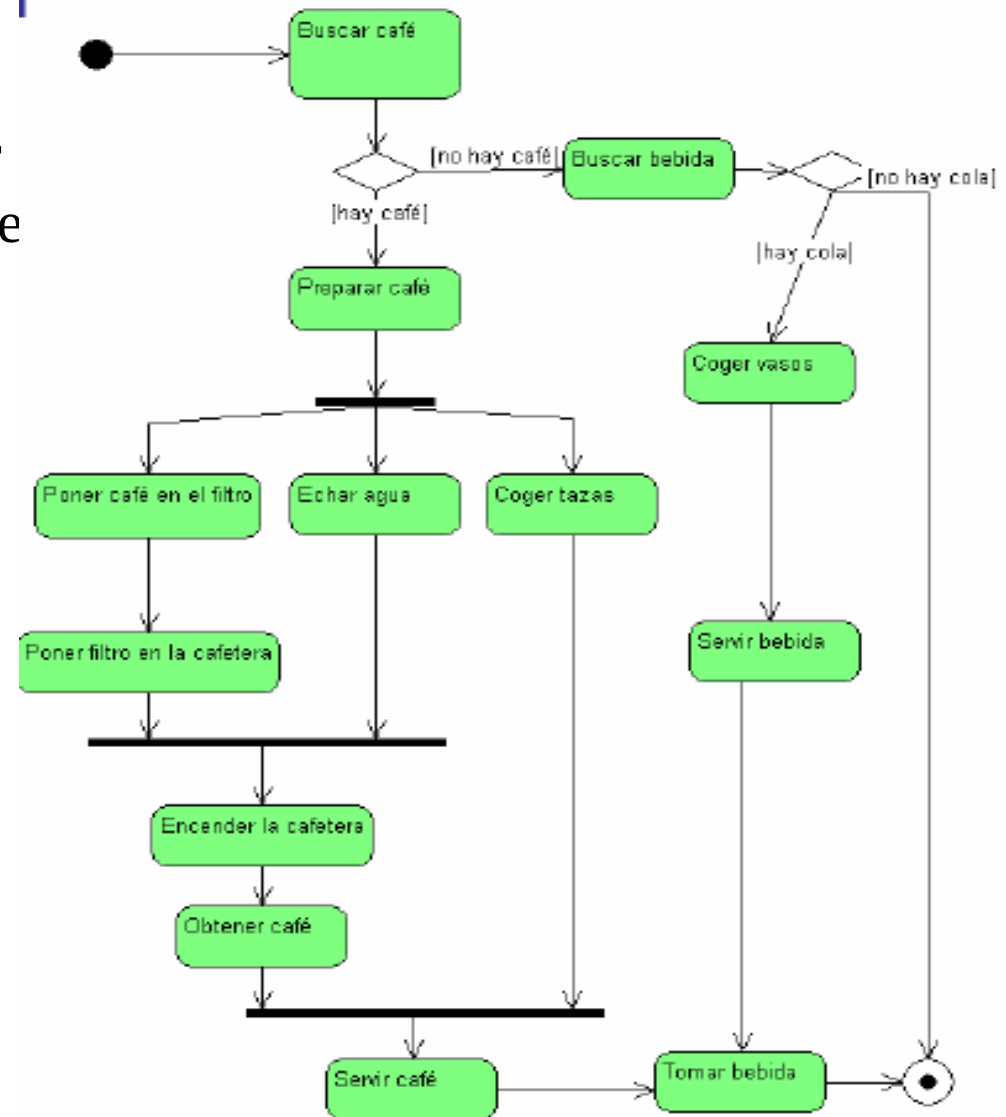
- **Diagramas de Estados**

—



4. AyDOO con UML

- **Diagramas de Actividades**
 - Tipo especial de diagramas de estados que se centra en mostrar el **flujo de actividades** dentro de un sistema. Los diagramas de actividades cubren la parte dinámica de un sistema y se utilizan para modelar el funcionamiento de un sistema resaltando el flujo de control entre objetos.

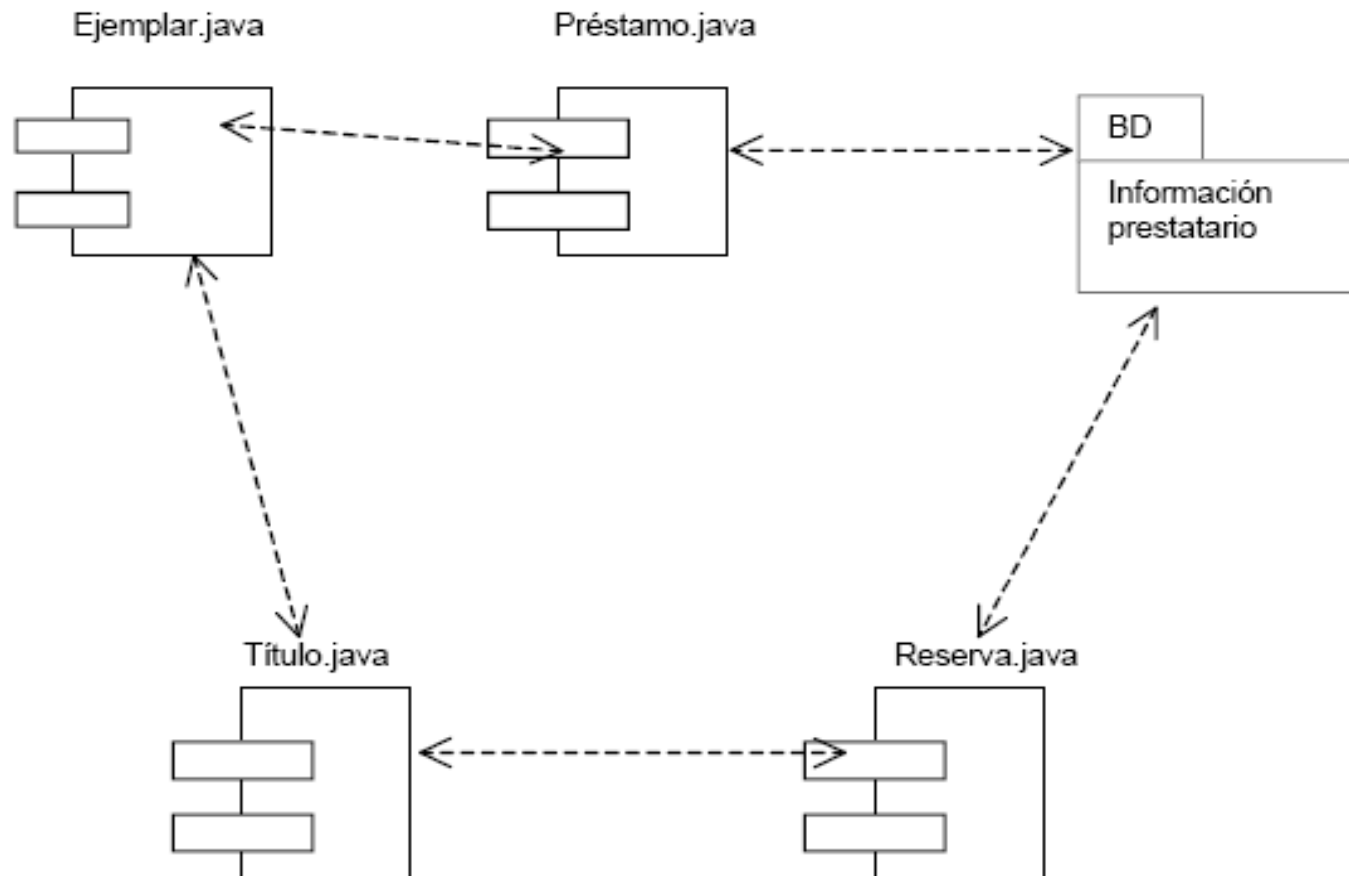


4. AyDOO con UML

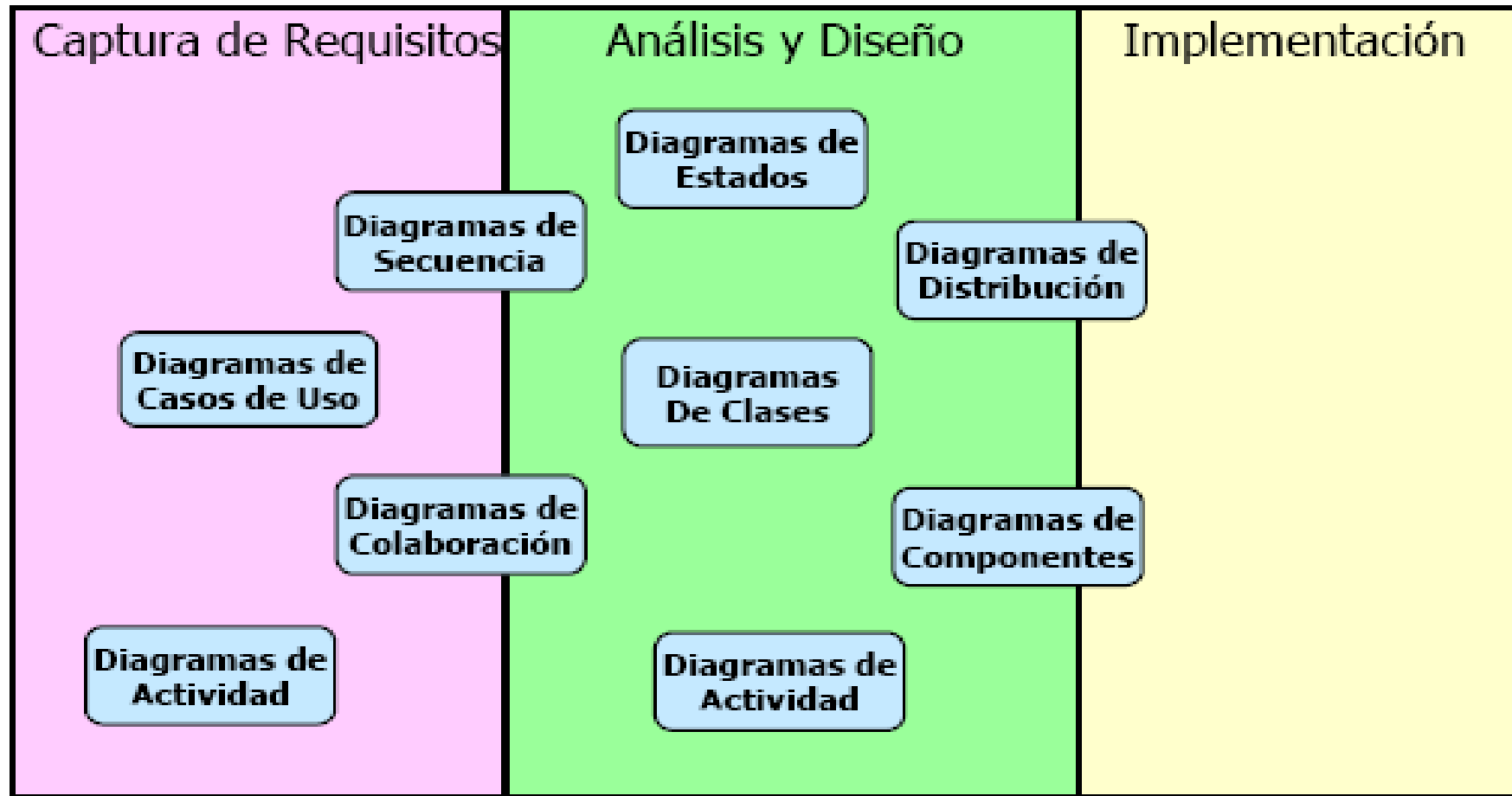
- **Diagramas de Componentes**
 - Muestra la organización y las dependencias entre un conjunto de componentes. Cubren la vista de la implementación estática y se relacionan con los diagramas de clases ya que en un componente suele tener una o más clases, interfaces o colaboraciones
- **Diagramas de Despliegue**
 - Representan la configuración de los nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos. Muestran la vista de despliegue estática de una arquitectura y se relacionan con los componentes ya que, por lo común, los nodos contienen uno o más componentes.

4. AyDOO con UML

- Diagramas de Componentes



4. AyDOO con UML



"You can model 80 percent of most problems by using about 20 percent of the UML."-- Grady Booch

5. Introducción. Proceso Unificado

➤ Proceso unificado de Rational (RUP)

❖ Definición: Es el conjunto de pasos ordenados que se realizan para alcanzar un objetivo

✓ Transformar los requisitos de usuario en un producto software

❖ El Proceso Unificado (PU)

✓ Es una metodología adaptable. Se puede modificar para adaptarlo al sistema a desarrollar en cada momento

✓ Es una técnica para elaborar modelos que se adapta a UML

✓ Su objetivo es producir un software de calidad

✓ PU es independiente del lenguaje de modelado y de la plataforma

❖ Pilares del PU

✓ Dirigido por casos de uso

✓ Centrado en la arquitectura

✓ Iterativo e incremental

1. Introducción. Proceso Unificado

Dirigido por casos de uso

- **Capturar los requisitos funcionales** del sistema y expresarlos desde el **punto de vista del usuario**.
- **Guiar** todo el proceso de desarrollo del sistema de información.
- En definitiva, resuelve las siguientes preguntas:
 - ❖ ¿Qué debe hacer el sistema ... para cada usuario?
 - ❖ ¿Qué tipos de usuarios hay?
 - ❖ ¿Qué funcionalidades hace cada uno?
 - ❖ ¿Existe una relación entre ellos y entre las funcionalidades que pueden hacer?

Centrado en la arquitectura

- Describe diferentes **vistas del sistema**
- Incluye los aspectos estáticos y dinámicos más significativos
- La **arquitectura y los casos de uso** evolucionan en paralelo
- Responsable: el arquitecto
 - ❖ Empieza por la parte que no es específica de los casos de uso
 - ❖ Trabaja con casos de uso claves
 - ❖ Progresa con la especificación de más casos de uso

Iterativo e incremental

- Se divide el trabajo en mini-proyectos
- Cada mini-proyecto es una iteración que resulta en un incremento
- La iteración
 - ❖ Trata un conjunto de casos de uso
 - ❖ Trata los riesgos más importantes
- En cada iteración se persiguen unos objetivos concretos

5. Introducción. Proceso Unificado

➤ Proceso unificado

- ❖ La filosofía del Proceso Unificado es empezar a trabajar en el desarrollo con un conocimiento relativo del problema,
 - ✓ A partir de él se hacen los primeros diagramas
- ❖ A medida que se va conociendo más el problema,
 - ✓ los diagramas se hacen más precisos (en cada iteración) para ampliarlos después (incremento).
- ❖ El proceso se repite hasta asegurarse que los diagramas realizados son una expresión exacta del sistema de información a desarrollar.

5. Introducción. Proceso Unificado

- En cada fase o incremento hay varias iteraciones.
 - ❖ Cada iteración es como un subproyecto: productos software + docum.
 - ✓ Puntos de verificación y control, (proceso continuo de pruebas)
 - ✓ Se pasa por una secuencia de actividades (flujos de trabajo), que realiza una parte de la funcionalidad del proyecto
 - ✓ Representa un ciclo de desarrollo completo, desde la captura de requisitos hasta implementación y prueba
 - ✓ Produce la entrega de una versión del proyecto ejecutable
 - Esta versión es un subconjunto del producto final que se va detallando en cada iteración (versión mejorada)
 - ✓ Continúa hasta que los flujos de trabajo sean satisfactorios

5. Introducción. Proceso Unificado

➤ Las cuatro “P” dentro del RUP: Personas, Proyecto, Producto y Proceso

- ❖ Personas: arquitectos, desarrolladores, ingenieros de prueba, etc.
- ❖ Proyecto: elementos organizativo a través del cual se gestiona el desarrollo software
- ❖ Producto: artefactos que se crean durante la vida del proyecto (modelos, código fuente, ejecutables, documentación, etc.)
- ❖ Proceso: conjunto completo de actividades necesarias para transformar los requisitos de un usuario en un producto
 - ✓ Plantilla para crear proyectos

Bibliografía

- Ivar Jacobson, Grady Booch, James Rumbaugh, “El Proceso Unificado de Desarrollo Software”, Addison Wesley, 1999
- UML. Diseño Orientado a Objetos con UML. Raúl Alarcón. Grupo Eidos.
- UML y Patrones. C. Larman. Prentice Hall, 1999.
- El Lenguaje Unificado de Modelado. G. Booch, J. Rumbaugh, I. Jacobson. Addison Wesley Iberoamericana, 1999.
- Object-Oriented Analysis and Design. G. Booch. Benjamin/Cummings, 1994.
- The UML Specification Document. G. Booch, I. Jacobson and J. Rumbaugh. Rational Software Corp., 1997.
- Object-Oriented Software Engineering: A Use Case Driven Approach. I. Jacobson. Addison-Wesley, 1992.
- Object-Oriented Modeling and Design. J. Rumbaugh et al. Prentice-Hall, 1991.