Tema 2
Diseño

Proyecto de Análisis y Diseño de Software

2º Ingeniería Informática

Universidad Autónoma de Madrid

Indice

- **Introducción**.
  - Conceptos de Orientación a Objetos.
- **Modelado de la Estructura.**
  - Diagramas de Clases.
- **Modelado del Comportamiento.**
  - Diagramas de Transición de Estados.
  - Diagramas de Secuencia.
- Trazabilidad de Requisitos

Orientación a Objetos

- En el paradigma estructurado de construcción de software, el bloque fundamental es la operación.
  - Funciones y datos van por separado.
  - Se ha mostrado útil para sistemas de pequeña escala (≈5000 LOC)

- El paradigma orientado a objetos presta igual atención a datos y operaciones.
  - Objetos: información de estado y operaciones asociadas en una sola unidad.
  - “Registros+funciones”.

Orientación a Objetos

Clases y Objetos

nombre: Jorge III
pais: Gran Bretaña
inicioReinado: 1760
finReinado: 1820

nombre: Luis XIV
pais: Francia
inicioReinado: 1774
finReinado: 1792

- Todos los reyes tienen aspectos en común.
- Podemos representar dichos aspectos en una clase.
- Reyes concretos serían instancias (objetos) de dicha clase.

Rey
- nombre: String - pais: String - inicioReinado: Date - finReinado: Date
+ reinar() + abdicar()

clase

jorge:Rey
nombre="Jorge III" pais="Gran Bretaña" inicioReinado=1760 finReinado=1820

luis:Rey
nombre="Luis XIV" pais="Francia" inicioReinado=1774 finReinado=1792

objetos

Orientación a Objetos

Clases y Objetos

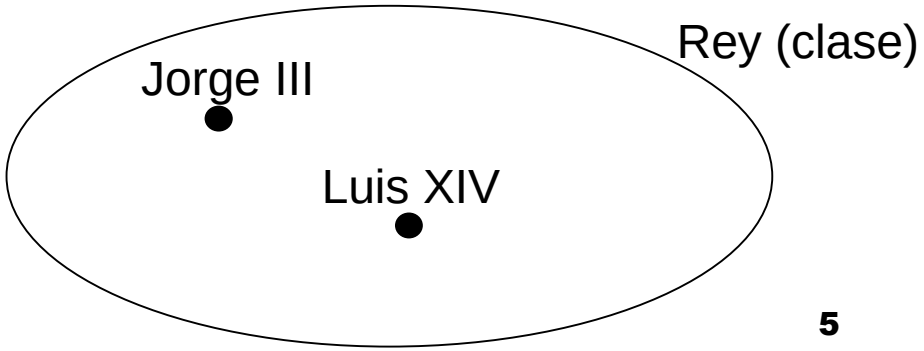
Clase

- Declara Propiedades (atributos) de todas las instancias.
- Declara Operaciones (métodos) aplicables a todas las instancias.

Objeto

- Da valores a los atributos declarados en la clase.
- Reacciona a invocaciones de los métodos declarados en la clase, usando como estado los valores de sus atributos.

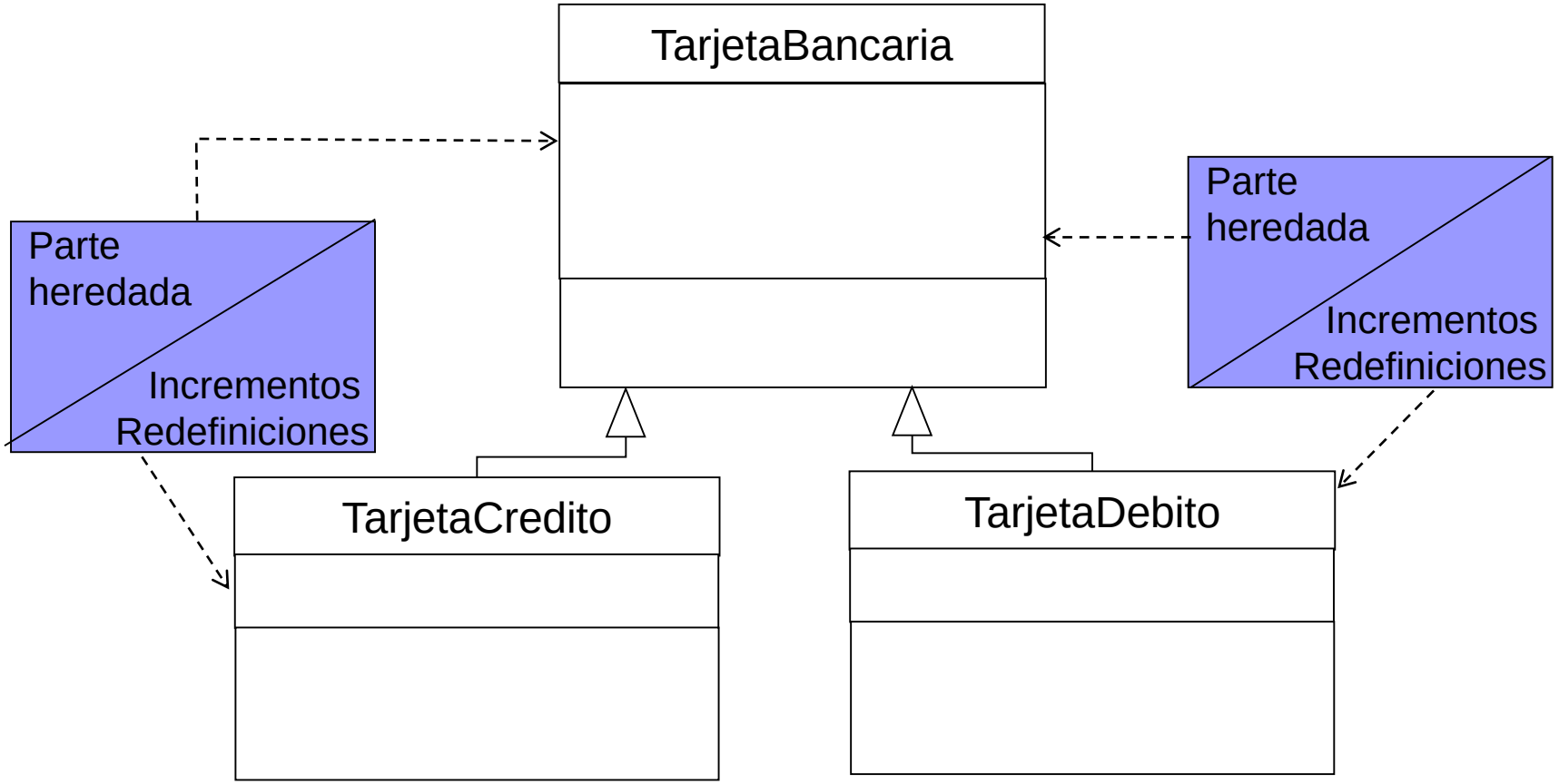
Usando conjuntos, podemos entender una clase como la definición del conjunto de todas sus instancias



Orientación a Objetos

Herencia

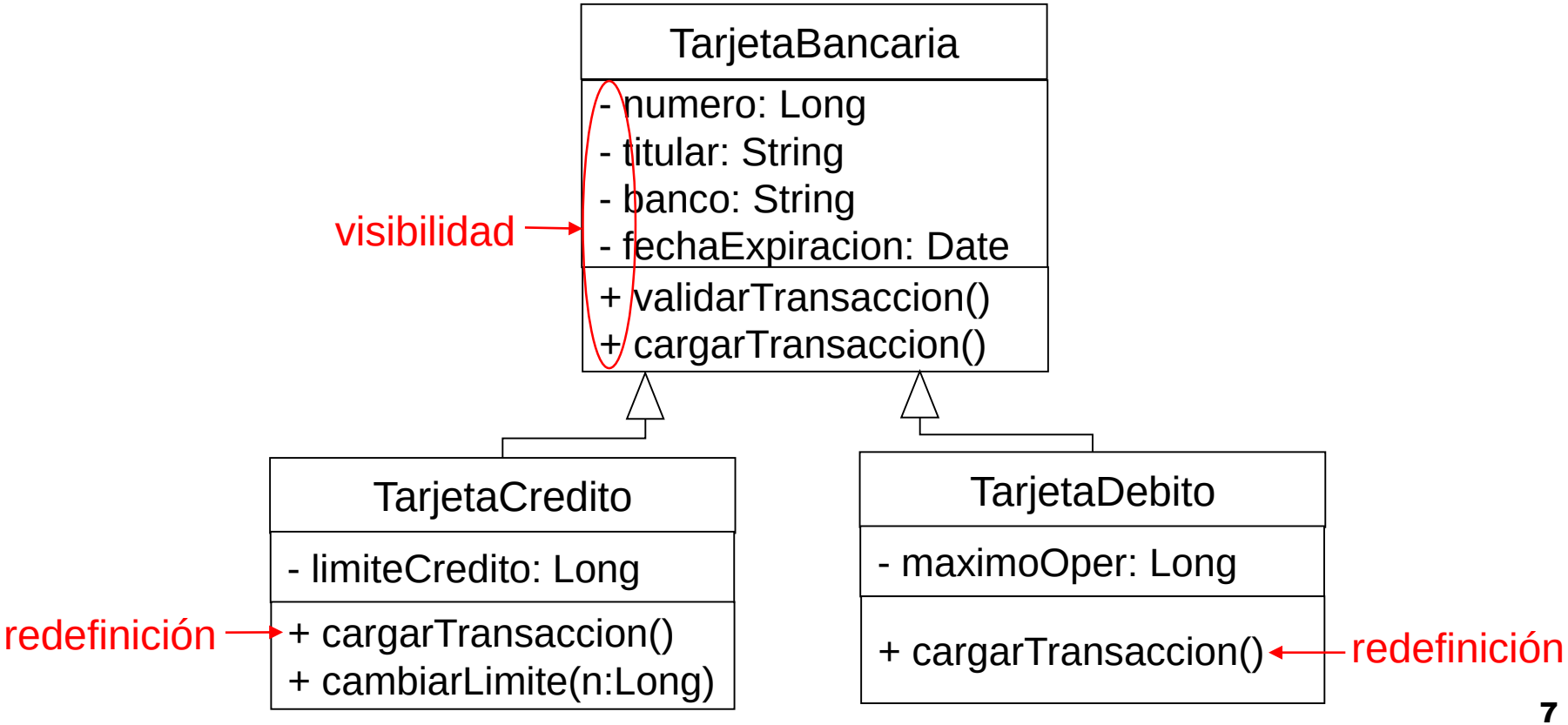
- Jerarquías de especialización de clases.
- Herencia de propiedades y operaciones.



Orientación a Objetos

Herencia

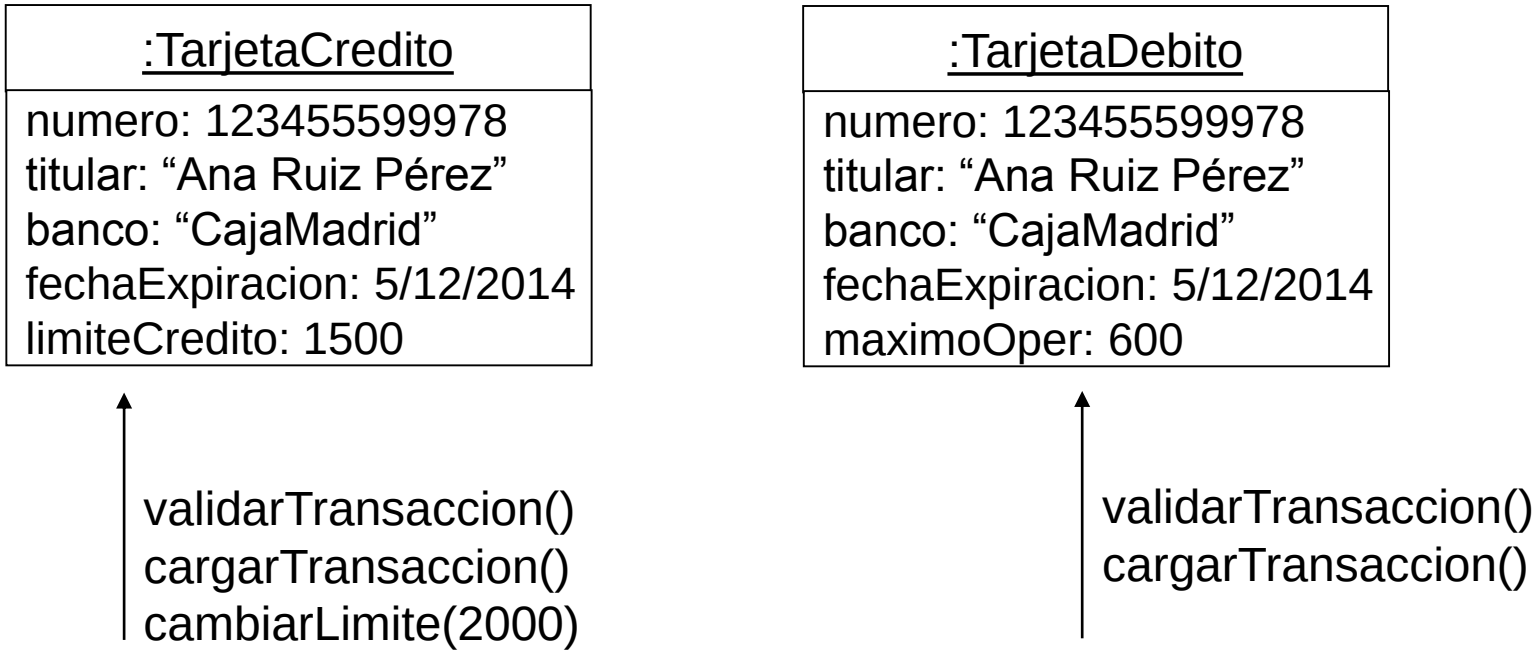
- Jerarquías de especialización de clases.
- Herencia de propiedades y operaciones.



Orientación a Objetos

Herencia

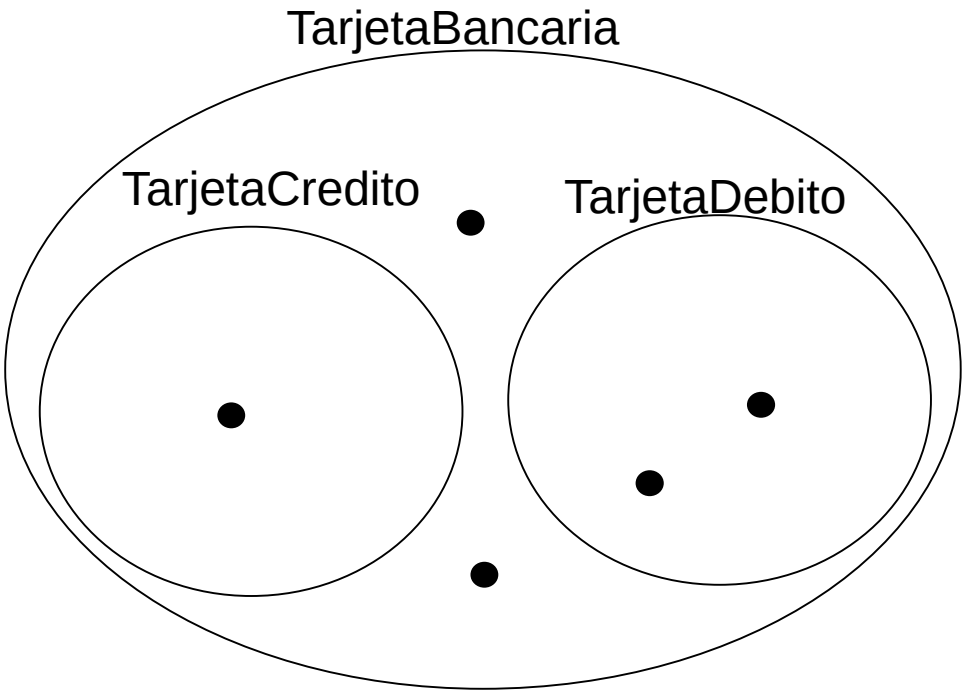
- Un objeto de una clase hija hereda las propiedades de la clase padre.
- Podemos invocarle las operaciones definidas en la clase padre.



Orientación a Objetos

Herencia

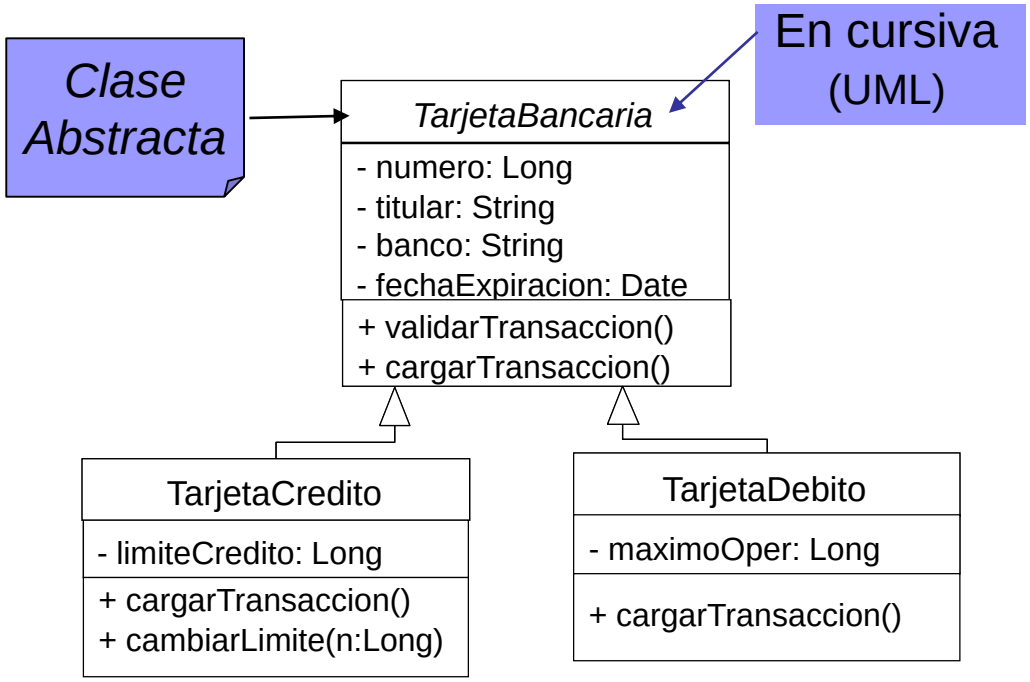
- Substitución segura de supertipos por subtipos.
  - Todas las tarjetas de crédito y de débito son tarjetas bancarias.
  - Existen tarjetas bancarias que no son de crédito ni de débito (clase base no abstracta).
  - No hay tarjetas que sean de crédito y de débito a la vez (herencia simple, no múltiple en este caso).

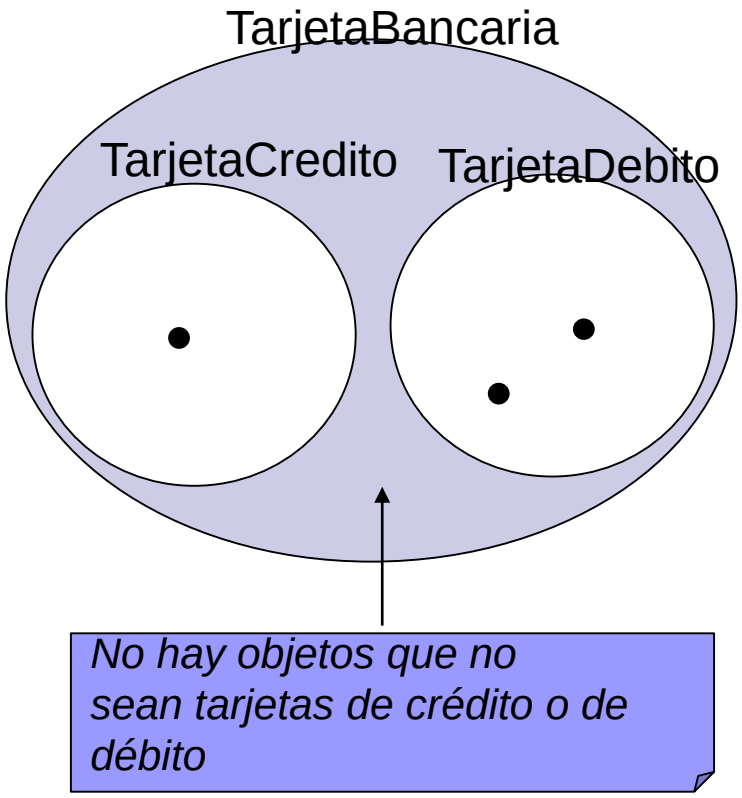


Orientación a Objetos

Clase Abstracta

- Una clase abstracta no puede instanciarse

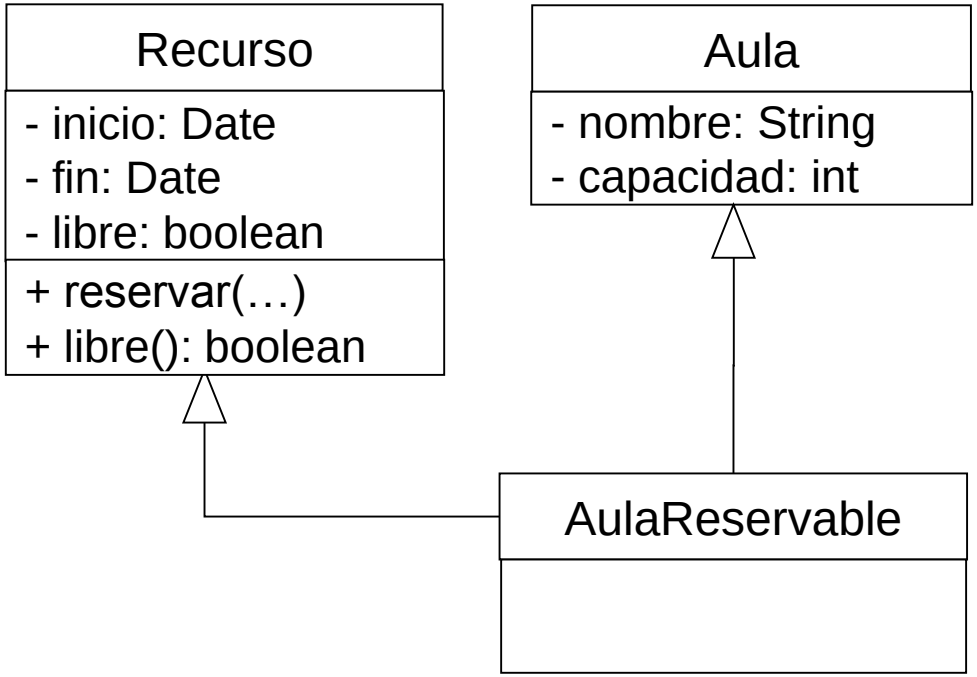


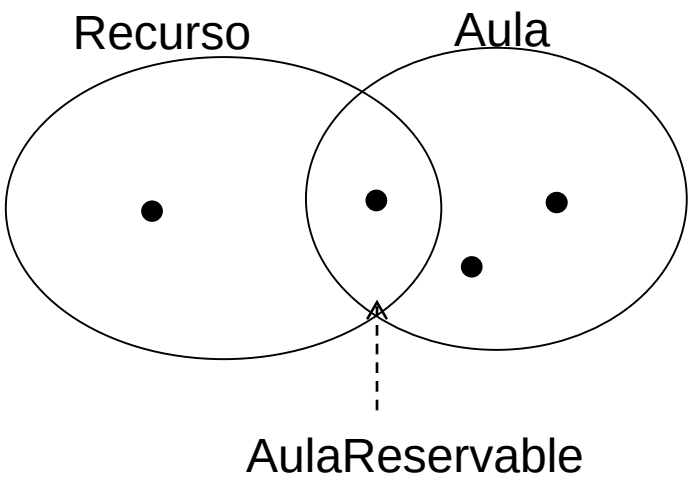


Orientación a Objetos

Herencia Múltiple

- Una clase puede heredar de varias.
- No es posible en Java, pero sí en otros lenguajes de programación como C++.





Orientación a Objetos

Asociaciones

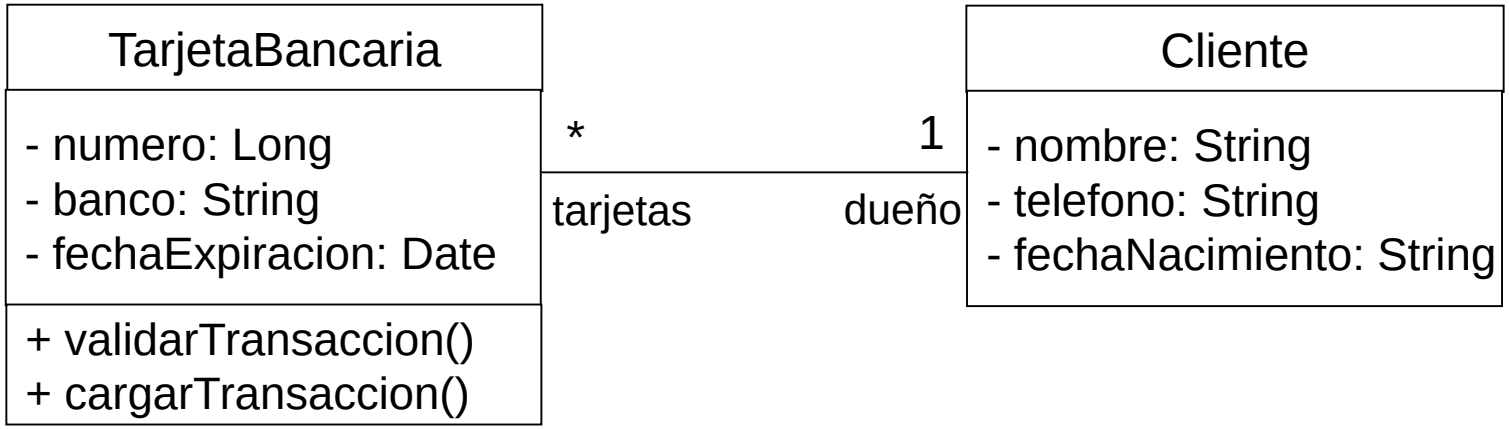
- Los objetos no están aislados, si no que necesitan "conocerse" para poder invocar métodos de otros.
  - La funcionalidad se implementa mediante colaboraciones de varios objetos.

- Conceptualmente, se representa mediante un enlace (una línea) entre dos clases.
  - En programación, esto significa que un objeto tienen una referencia a otro objeto (un atributo).

Orientación a Objetos

Asociaciones

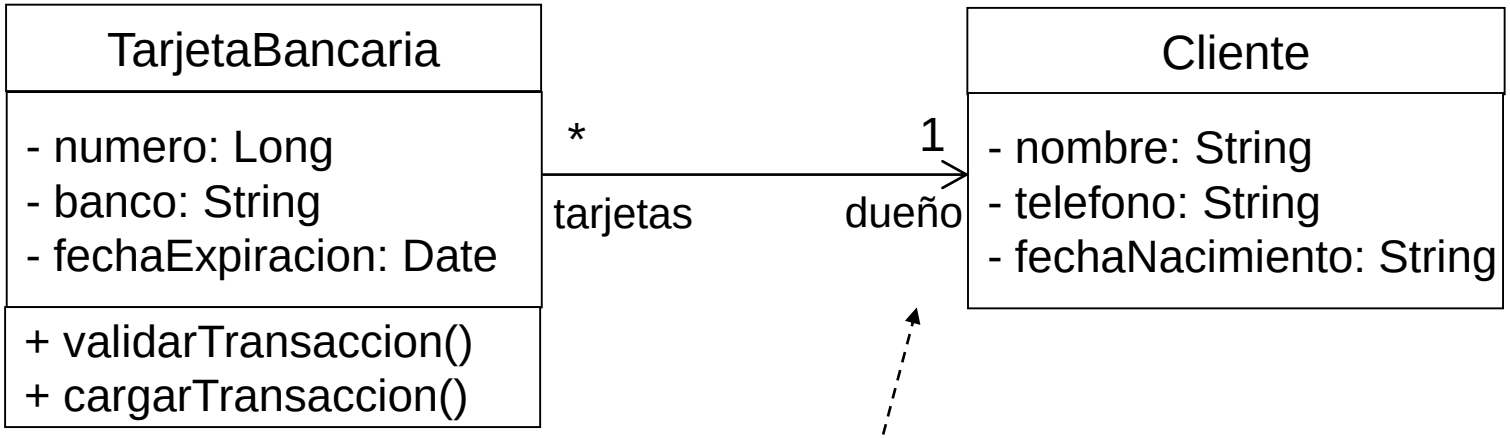
- Multiplicidad:
  - Con cuántos objetos destino se puede relacionar un objeto fuente, y viceversa.
- Roles:
  - Nombres de los extremos de las asociaciones.



Orientación a Objetos

Asociaciones

- Navegación: Indica si un objeto puede acceder al otro.



Desde un objeto de tipo tarjetaBancaria podemos acceder a su dueño, pero no al revés

Orientación a Objetos

Herencia de asociaciones

- Una asociación declarada en una clase base, se hereda en las clases hijas.

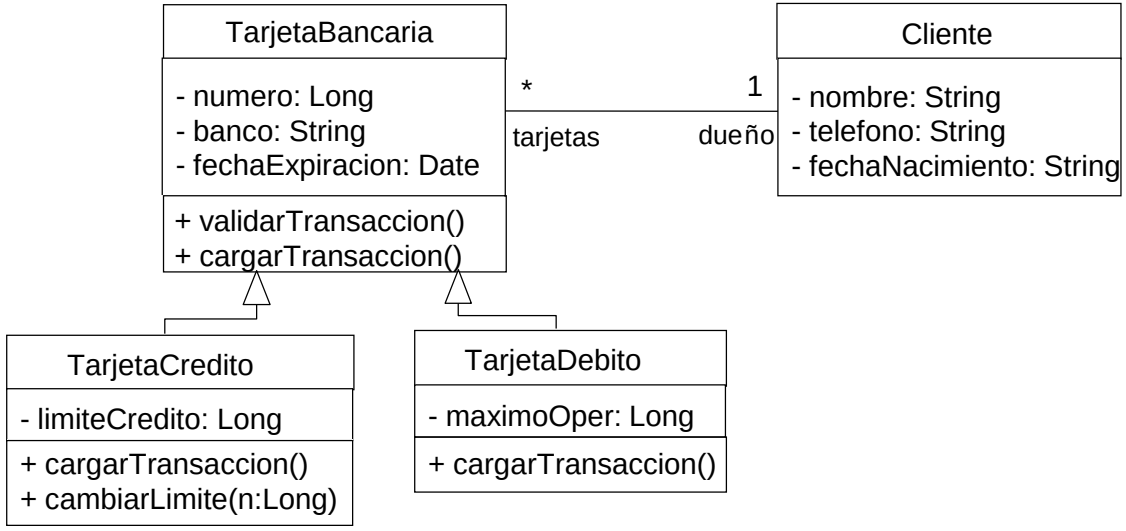


Diagrama de Clases

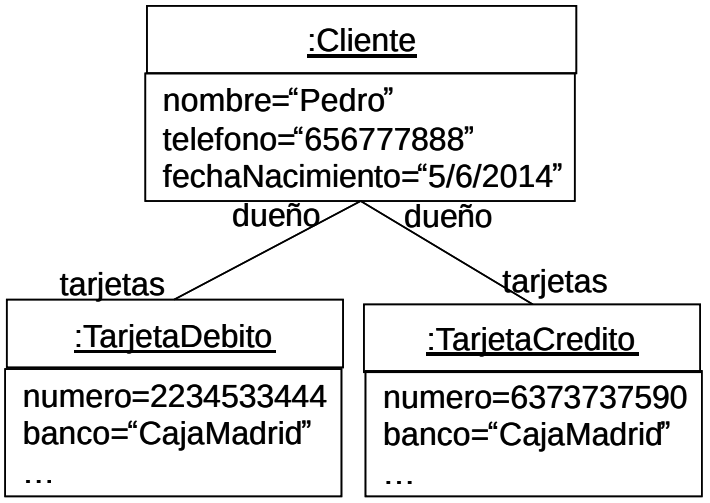
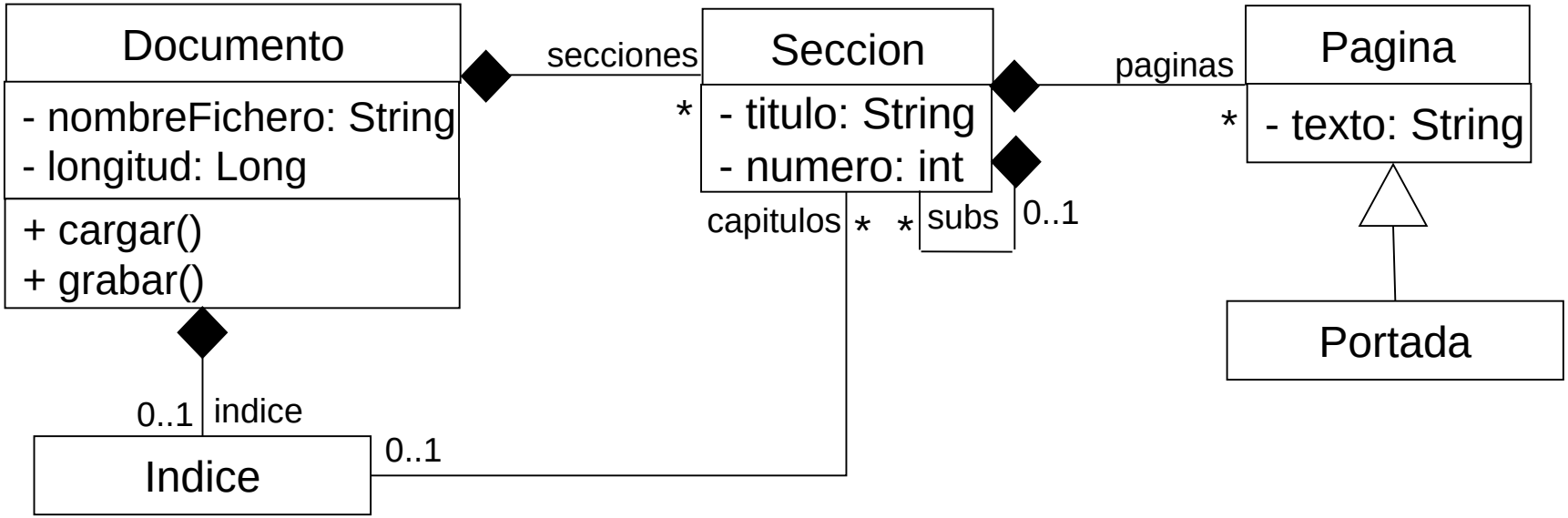


Diagrama de Objetos

Orientación a Objetos

Composición y Agregación

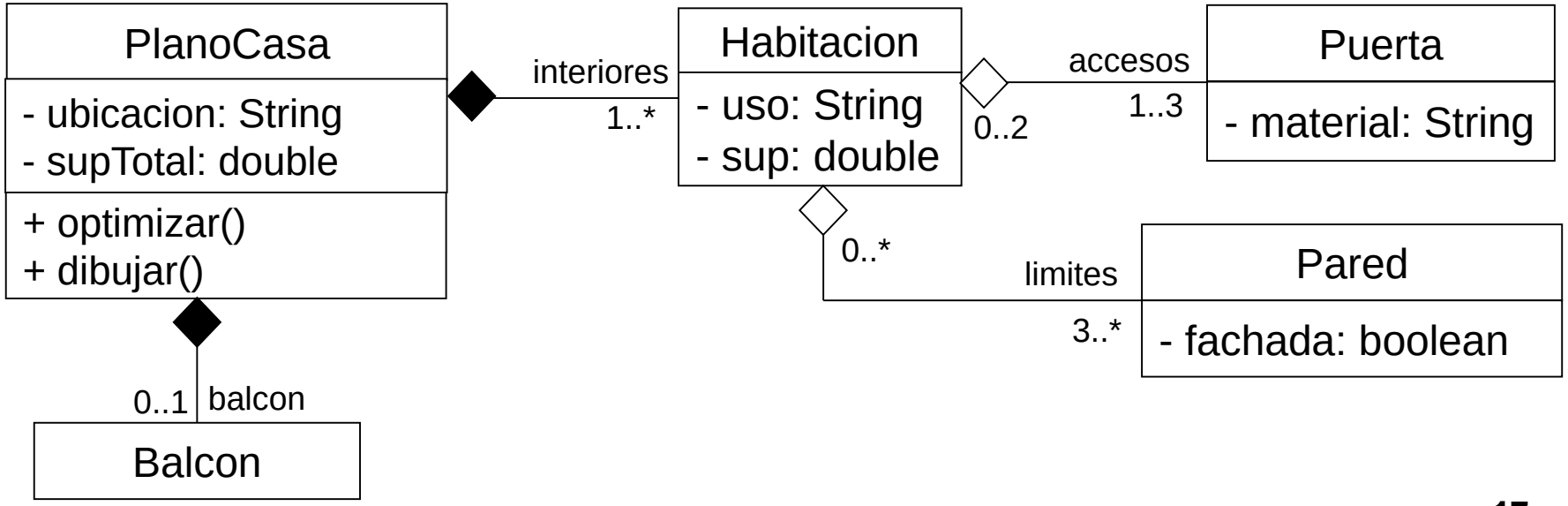
- Hay relaciones con semánticas especiales:
  - ◆ Composición (una clase hecha de partes)
  - ◇ Agregación (composición más débil)



Orientación a Objetos

Composición y Agregación

- Hay relaciones con semánticas especiales:
  - ◆ Composición (partes en no más de un compuesto)
  - ◇ Agregación (partes pueden estar en varios compuestos)



Proyecto

- Realiza el diagrama de clases de la aplicación del proyecto.

Indice

- **Introducción**.
  - Conceptos de Orientación a Objetos.
- **Modelado de la Estructura.**
  - Diagramas de Clases.
- **Modelado del Comportamiento.**
  - Diagramas de Transición de Estados.
  - Diagramas de Secuencia.
- Trazabilidad de Requisitos

Modelado del Comportamiento

- Un diagrama de clases nos dice la estructura de la aplicación, pero no el comportamiento:
  - ¿qué hace cada método?
  - ¿cómo cambia el estado de un objeto ante invocaciones de métodos?
  - ¿cómo colabora un conjunto de objetos para realizar una tarea?

Modelado del Comportamiento

Diagramas de Comportamiento

- Un diagrama de clases nos dice la estructura de la aplicación, pero no el comportamiento:
  - ¿qué hace cada método?
    - Pseudocódigo (lenguaje de “acción semántica”).
    - Diagrama de actividad
  - ¿cómo cambia el **estado de un objeto** ante invocaciones de métodos?
    - **Diagrama de Transición de Estados** (“statecharts”).
  - ¿cómo **colabora un conjunto de objetos** para realizar una tarea?
    - **Diagrama de Secuencia**.
    - Diagrama de Colaboración/comunicación.

Diagrama de Transición de Estados

- Van asociados a una clase.
- Describen cómo evoluciona cuando se le invocan métodos.
- Similar a un autómata finito.

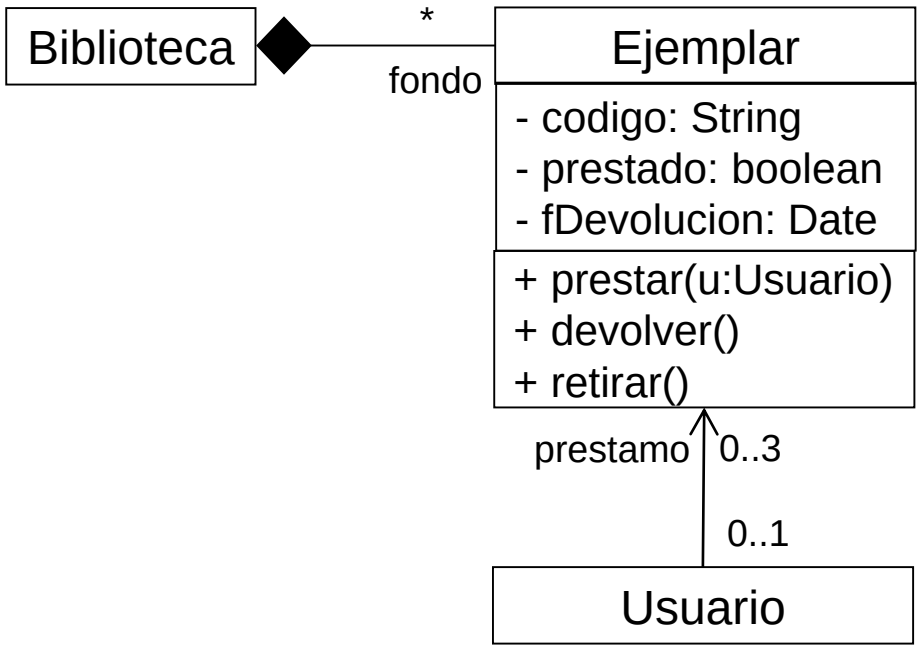


Diagrama de clases

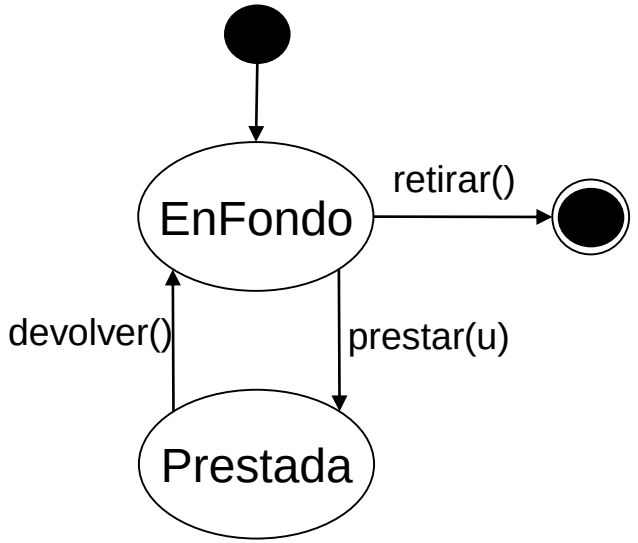


Diagrama de transición de estados (clase Ejemplar)

Diagrama de Transición de Estados

- Estados jerárquicos.
- Guardas y Acciones en las transiciones.

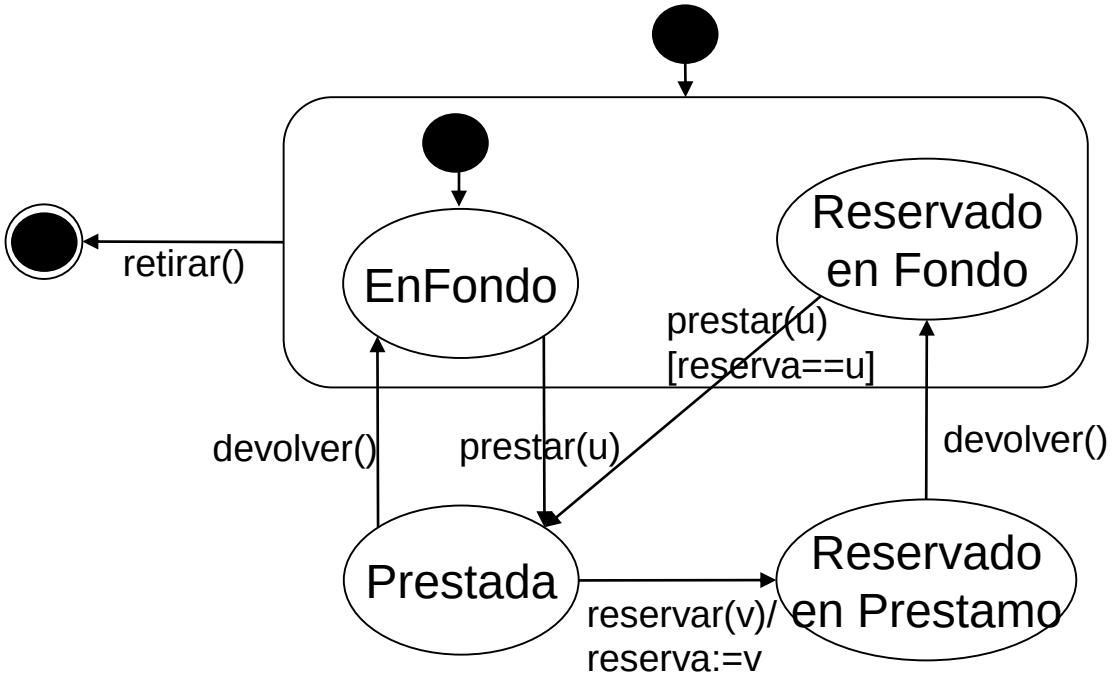


Diagrama de Transición de Estados

Ejercicio

- Modelar el comportamiento de una cadena de música.
- Una cadena de música puede estar encendida (ON) o apagada (Standby).
- La cadena tiene reproductor de CD, Radio y MP3. Se cambia de uno a otro con el botón “mode”.
- Cuando se enciende la cadena se recuerda el último estado en el que estuvo.

Diagrama de Transición de Estados

Solución

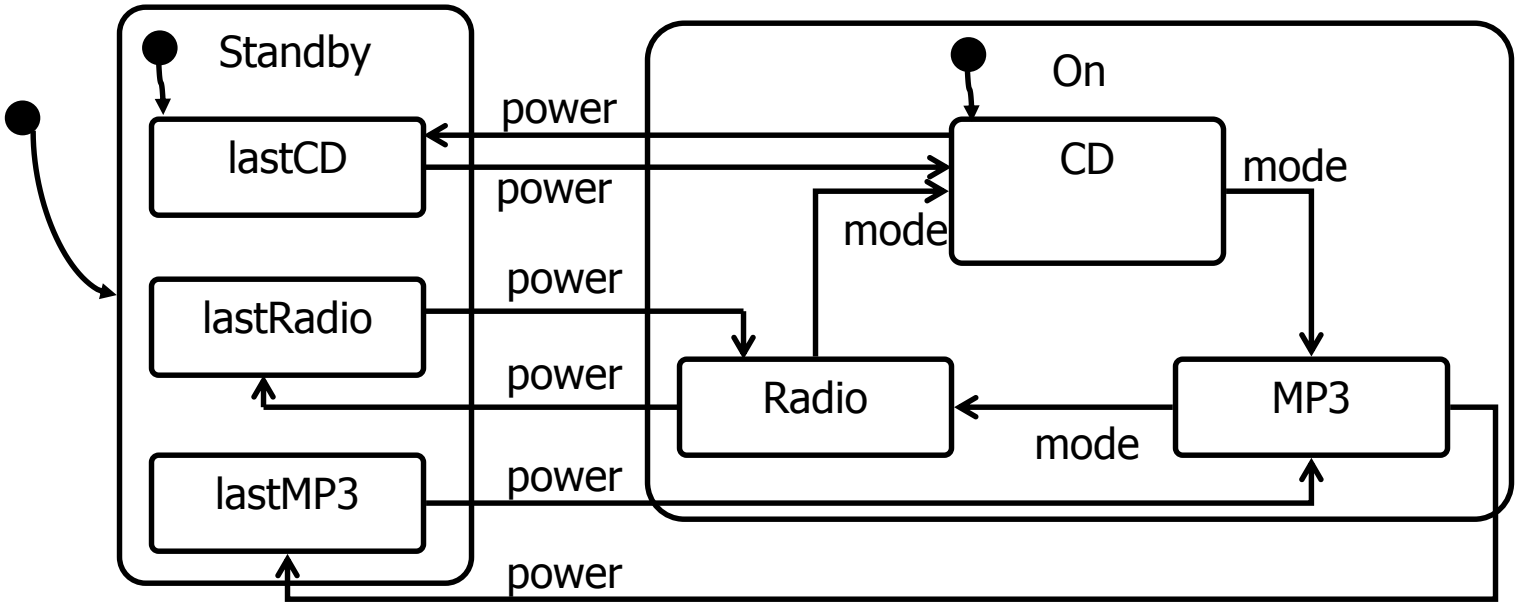
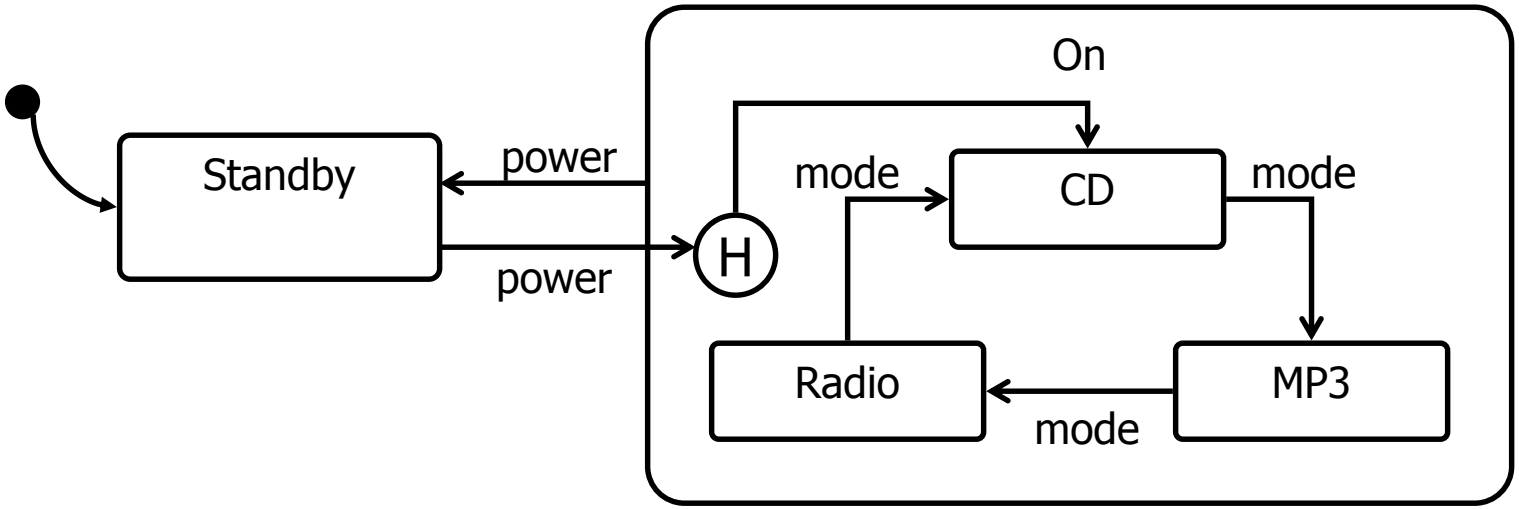


Diagrama de Transición de Estados

Estado Histórico

- Recuerda el último estado en el que se estuvo en un estado jerárquico.
- Si es la primera vez que se accede, se sigue la transición.



Ejercicio

- Modela el comportamiento de un "*recurso compartido*" en un sistema P2P. Su ciclo de vida es el siguiente:
  - El usuario activa la descarga de un recurso compartido de un tamaño *T*.
  - Cuando existe alguna fuente para dicho recurso, éste se pone en cola. Pueden llegar más fuentes en cualquier momento.
  - Una vez que está en cola, en algún momento puede iniciarse la descarga.
  - Cuando el recurso se está descargando, llegan paquetes de datos de tamaño *P*.
  - La descarga puede bien finalizar antes de que se hayan obtenido todos los datos del recurso, o bien cuando se ha descargado completamente.
  - En cualquier momento antes de que el recurso se descargue completamente, el usuario puede cancelar la compartición del recurso.

Solución

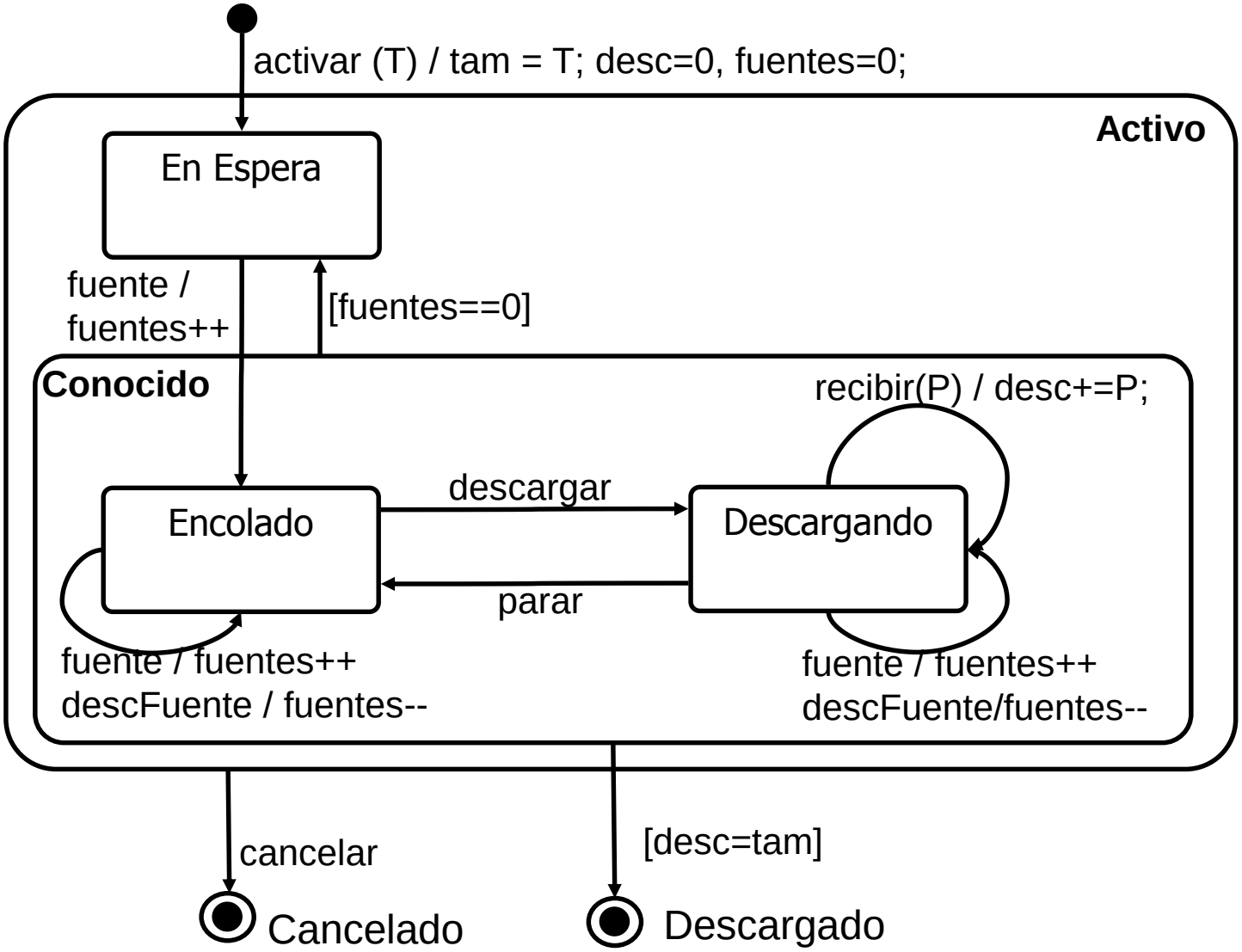

activar (T) / tam = T; desc=0, fuentes=0;
Activo
En Espera
fuente /
fuentes++
[fuentes==0]
Conocido
recibir(P) / desc+=P;
descargar
Encolado
Descargando
parar
fuente / fuentes++
descFuente / fuentes--
fuente / fuentes++
descFuente/fuentes--
cancelar
[desc=tam]
Cancelado
Descargado

Diagrama de Transición de Estados

- Selecciona dos clases de tu proyecto y construye su diagrama de transición de estados.

Diagrama de Secuencia

- Representa una secuencia de mensajes que intercambia **un conjunto de objetos**.
- Cada objeto tiene una línea de vida (representada verticalmente).
  - El paso del tiempo se representa de arriba abajo.
  - Invocación de mensajes: flechas entre líneas de vida.
  - Cajas de activación.

Diagrama de Secuencia

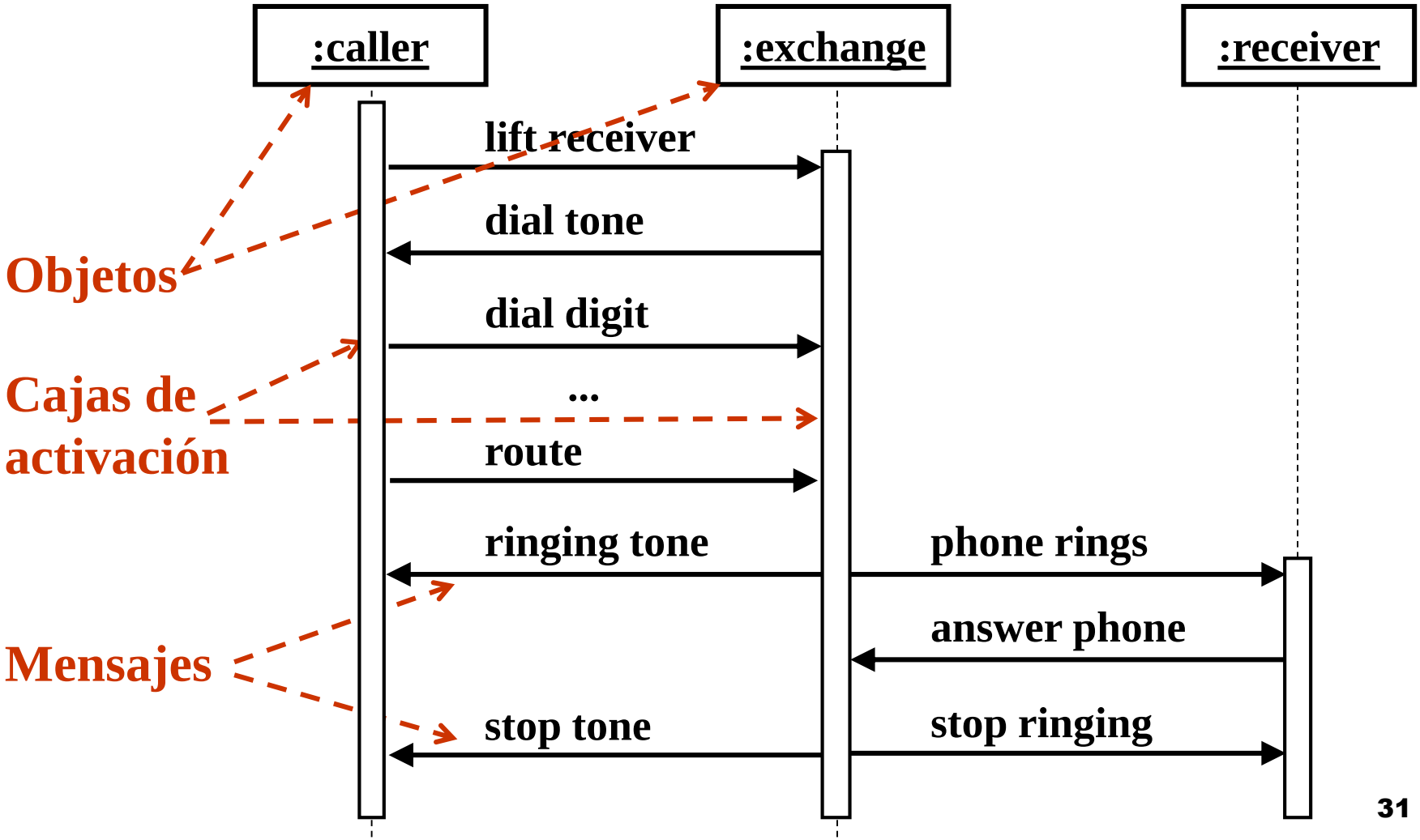


Diagrama de Secuencia

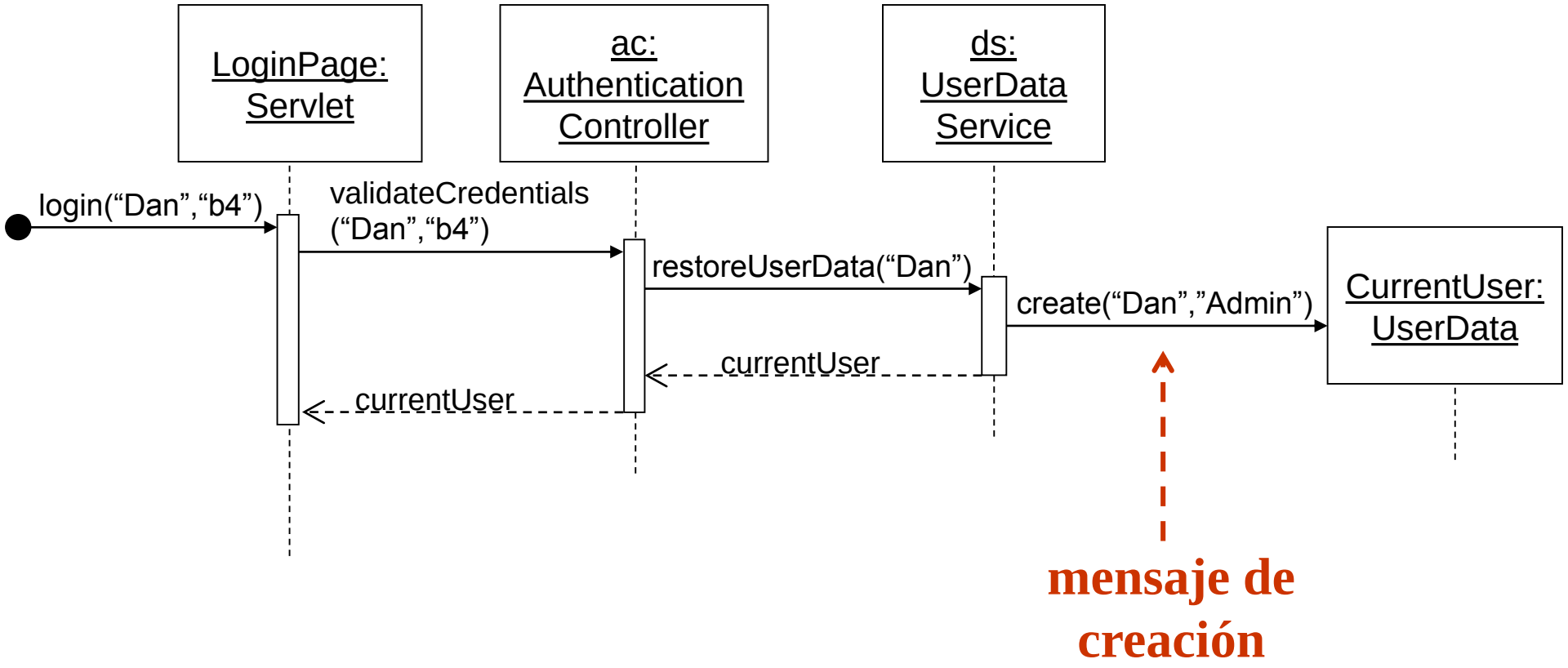
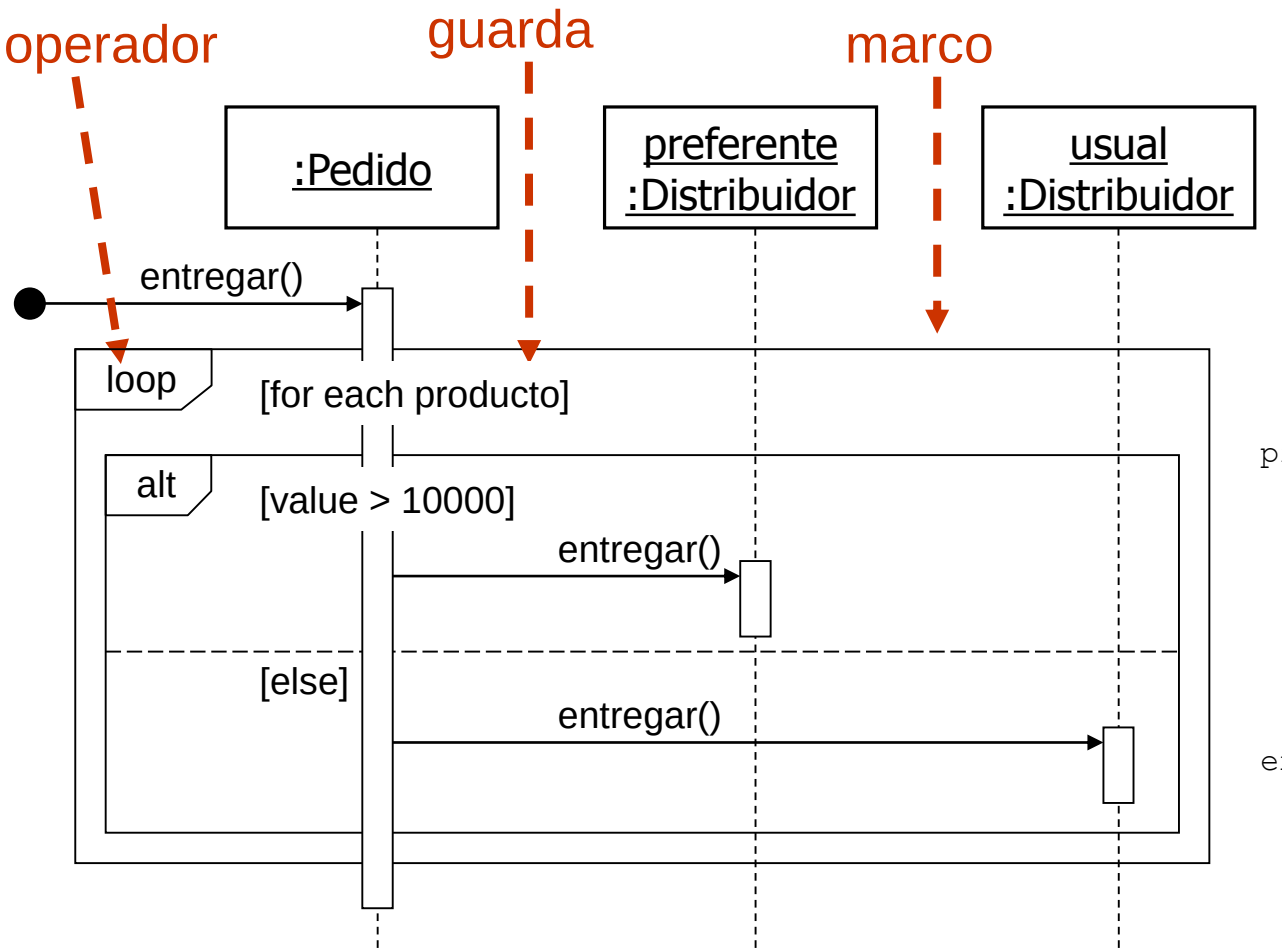


Diagrama de Secuencia

Operadores




```
procedure entregar()
   foreach producto:
      if producto.value>10000
         preferente.entregar()
      else
         usual.entregar()
      end if
   end for
end procedure
```

Diagrama de Secuencia

Operadores

- Fragmentos combinados, operadores:
  - **Alternativa (*alt*)**: elección (mediante una guarda) de una interacción. Múltiples fragmentos, sólo se ejecuta el que cumpla la guarda.
  - **Opción (*opt*)**: equivale a un operador *alt* con un solo fragmento. Se ejecuta si la guarda se cumple.
  - **Bucle (*loop*)**: el fragmento se ejecuta múltiples veces. La guarda indica cómo realizar la iteración.
  - **Negativa (*neg*)**: define una interacción inválida.
  - **Paralelo (*par*)**: cada fragmento se ejecuta en paralelo.
  - **Región crítica (*critical*)**: sólo puede haber un proceso ejecutando simultáneamente el fragmento
  - **Diagrama de secuencia (*sd*)**: rodea un diagrama de secuencia
  - **Referencia (*ref*)**: el marco hace referencia a una interacción definida en otro diagrama. El marco dibujado cubre las líneas involucradas en la interacción. Puede incluir parámetros y un valor de retorno.

Diagrama de Secuencia

Operadores

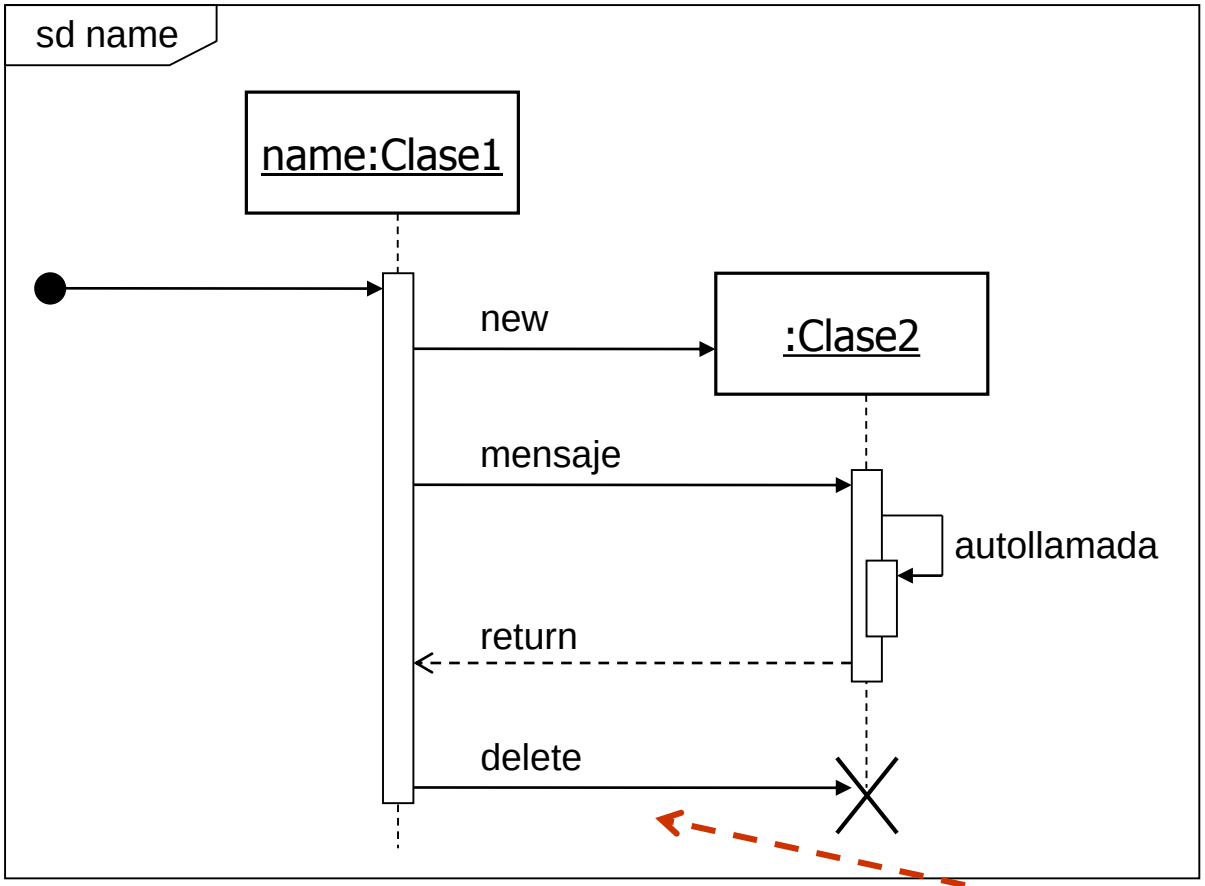


Diagrama de Secuencia

Ejercicio

- Especificar el diagrama de secuencia de la operación “realizarJugada” definida en la clase Jugador, para el juego del parchís

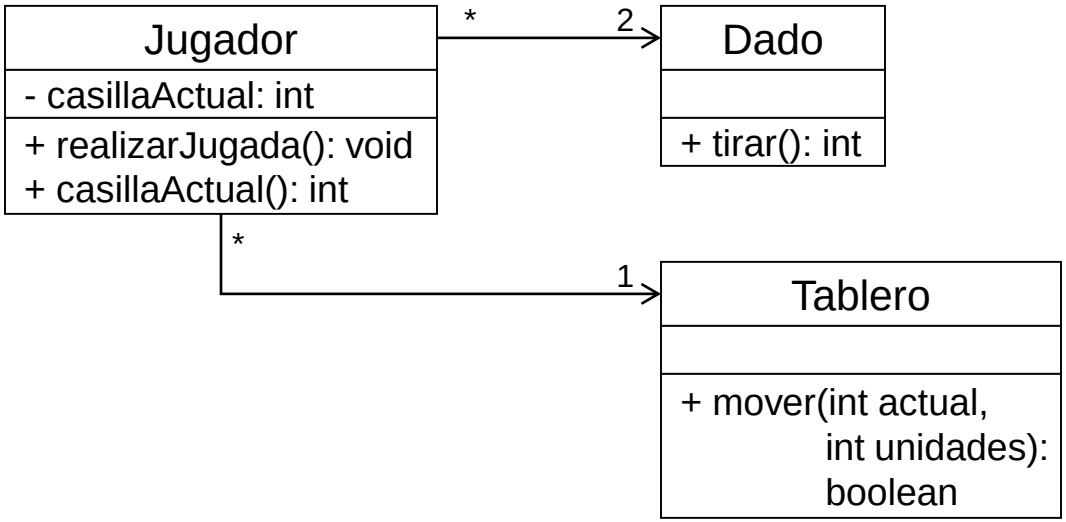


Diagrama de Secuencia

Ejercicio

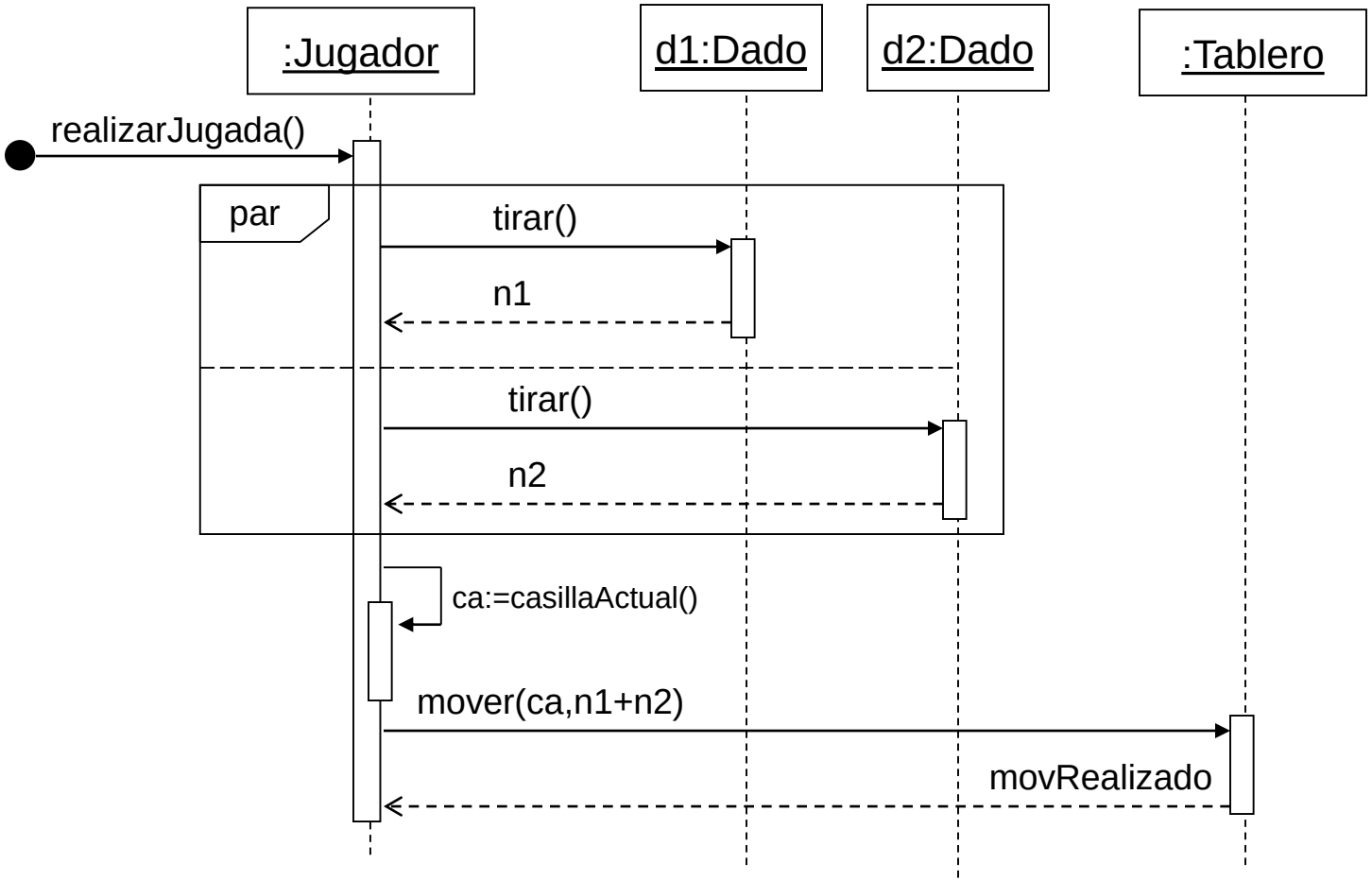


Diagrama de Secuencia

Ejercicio

Especificar el diagrama de secuencia del método “crearLaberinto”

```
public class JuegoLaberinto {
   public Laberinto crearLaberinto () {
      Laberinto lab = new Laberinto();
      Habitacion h1 = new Habitacion();
      Habitacion h2 = new Habitacion();
      Puerta puerta = new Puerta(h1, h2);
      lab.añadeHabitacion(h1);
      lab.añadeHabitacion(h2);
      h1.añadePuerta(puerta);
      return lab;
   }
}
```

Diagrama de Secuencia

Ejercicio

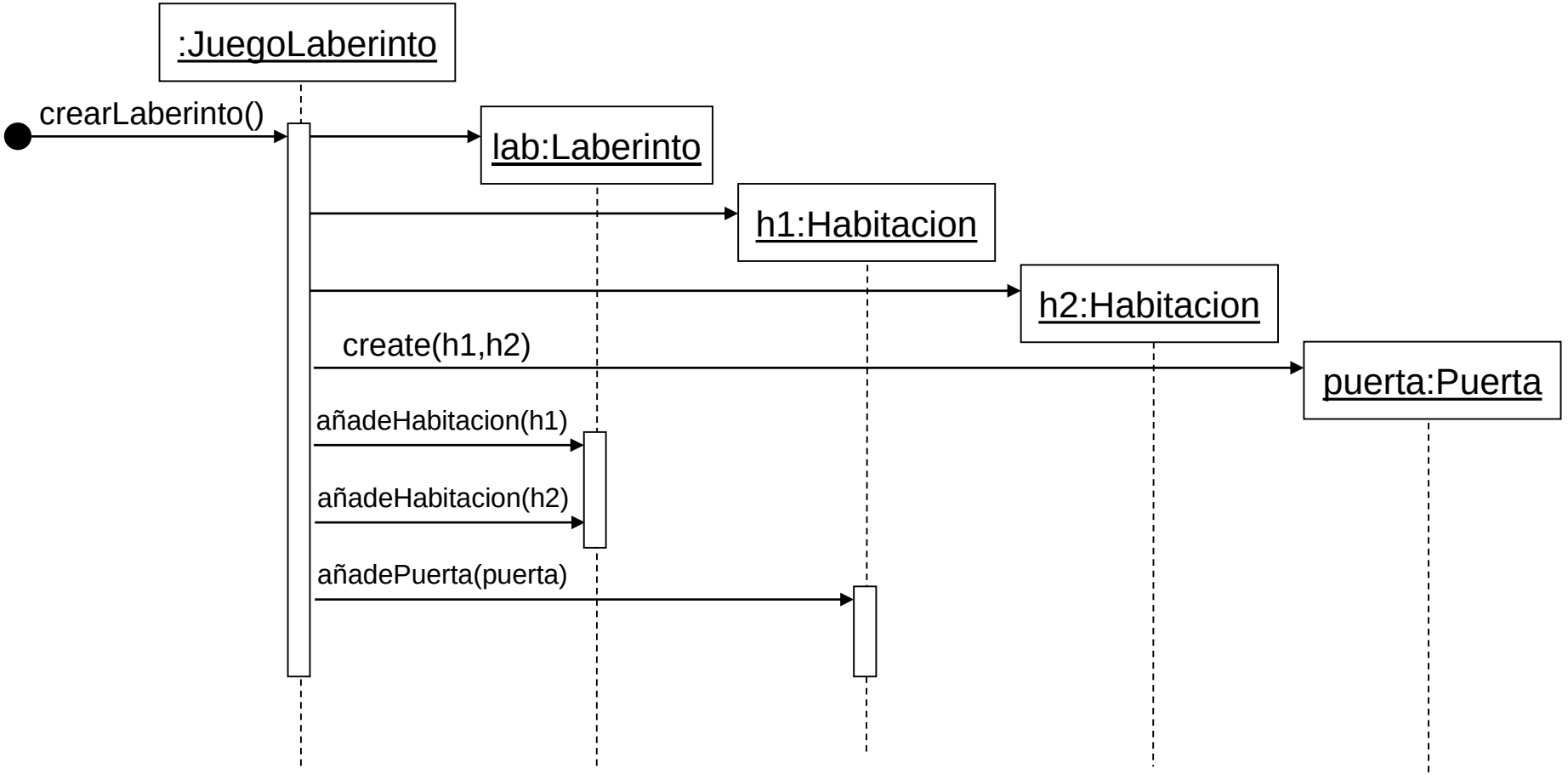


Diagrama de Secuencia

- Selecciona dos escenarios relevantes de tu proyecto y construye los diagramas de secuencia.
  - Puedes seleccionar algún escenario de los casos de uso.

Indice

- **Introducción**.
  - Conceptos de Orientación a Objetos.
- **Modelado de la Estructura.**
  - Diagramas de Clases.
- **Modelado del Comportamiento.**
  - Diagramas de Transición de Estados.
  - Diagramas de Secuencia.
- Trazabilidad de Requisitos

Trazabilidad de Requisitos

- Al finalizar el diseño (detallado) de la aplicación, es necesario comprobar que todos los requisitos educidos en la fase de análisis del problema tienen su correspondiente representación en el dominio de la solución a implementar.
- Matriz de Trazabilidad de Requisitos

	Elementos Funcionales			
Requisitos	ClaseN.Método1	ClaseN.Método2	…	ClaseN.MétodoM
Requisito 1	X	X		
Requisito 1.1		X		
…				
Requisito N	X			X
Requisito N.M				X

Trazabilidad de Requisitos

- Elabora la Matriz de Trazabilidad de Requisitos para el Proyecto diseñado
  - Se entregará también actualizada más adelante, en la práctica de programación

Bibliografía

Existen numerosos libros y manuales sobre UML2.0. A continuación se listan unos cuantos:

- Diagramas de Clases:
  - Utilización de UML en Ingeniería de Software con Objetos y Componentes. Stevens&Poley. Addison Wesley. 2002. Capítulos 5 y 6.
  - UML Distilled, 3rd Edition. Fowler. Addison Wesley. Capítulo 3.
- Diagramas de Objetos:
  - UML Distilled, 3rd Edition. Fowler. Addison Wesley. Capítulo 6.
- Diagramas de Transición de Estados:
  - Utilización de UML en Ingeniería de Software con Objetos y Componentes. Stevens&Poley. Addison Wesley. 2002. Capítulo 11.
  - UML Distilled, 3rd Edition. Fowler. Addison Wesley. Capítulo 10.
- Diagramas de Secuencia:
  - UML Distilled, 3rd Edition. Fowler. Addison Wesley. Capítulo 4.