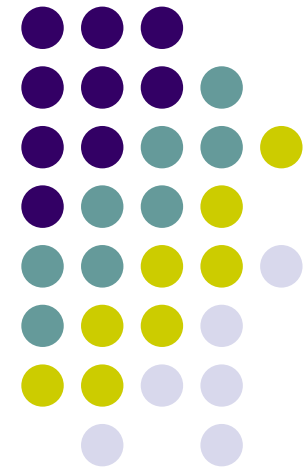


# Tema 1

## UML: Lenguaje Unificado de Modelado



# Introducción



- Dominio: contexto de un problema
- Modelo: abstracción de un problema
- UML:
  - Unified Modeling Language. Versión 2.0 (finales de 2004)
  - Lenguaje de modelado de sistemas
  - Diagramas de estructura, comportamiento, ...
    - Especificar sistemas
    - Visualizar sistemas
    - Construir sistemas
    - Documentar sistemas
  - Ingeniería directa: hacer diagramas y luego implementar
  - Ingeniería inversa: a partir de código extraer los diagramas

# Contenido



- Diagramas de estructura
  - Clases y objetos
  - Relaciones
  - Interfaces
- Diagramas de comportamiento
  - Colaboraciones
  - Diagramas de secuencia
  - Diagramas de comunicación

# Diagramas de estructura

## Clases y objetos



- **Diagramas de clases**: estructura del sistema.
  - Clases: conceptos dentro del sistema que comparten los mismos atributos, operaciones, relaciones y semántica
    - Atributos: tipo, visibilidad, posible valor inicial
    - Operaciones: signatura, visibilidad
  - Relaciones: asociaciones entre clases
- **Diagramas de objetos**: estructura del sistema en tiempo de ejecución.
  - Objetos: instancias de una clase
    - Atributos (valores actuales)
  - Links: relaciones entre objetos, instancias de asociaciones

# Diagramas de estructura

## Clases y objetos. Ejemplo



Diagrama de Clases:

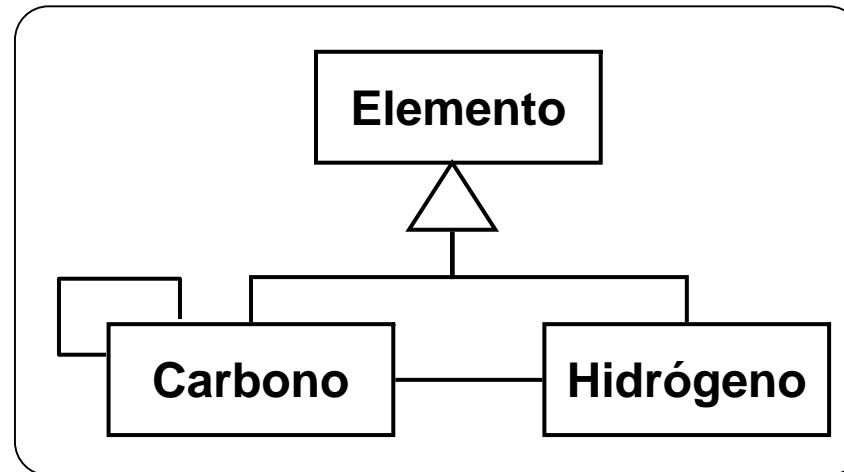
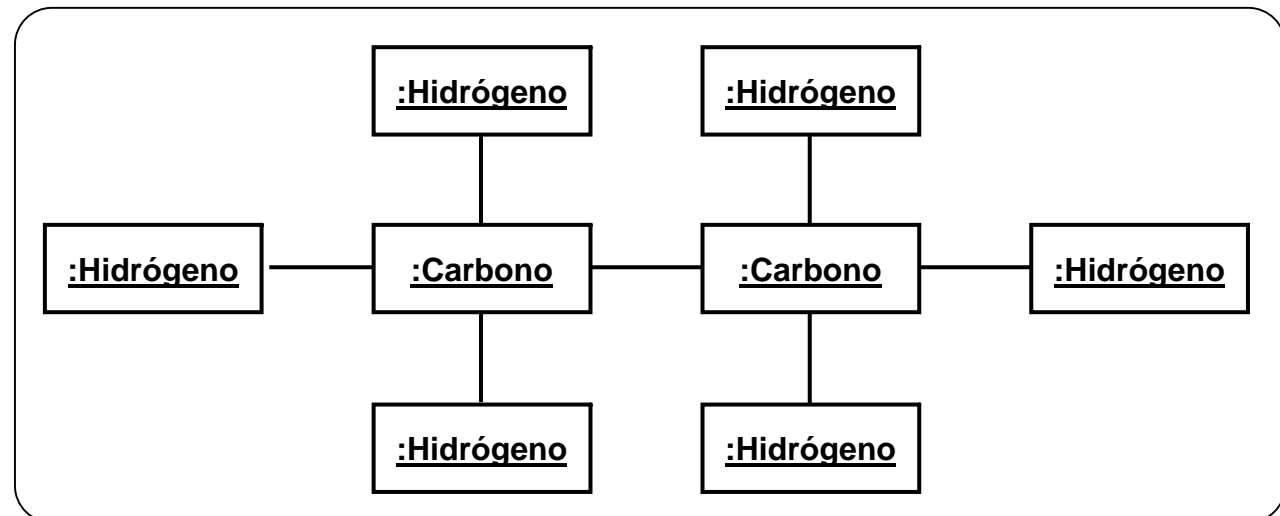
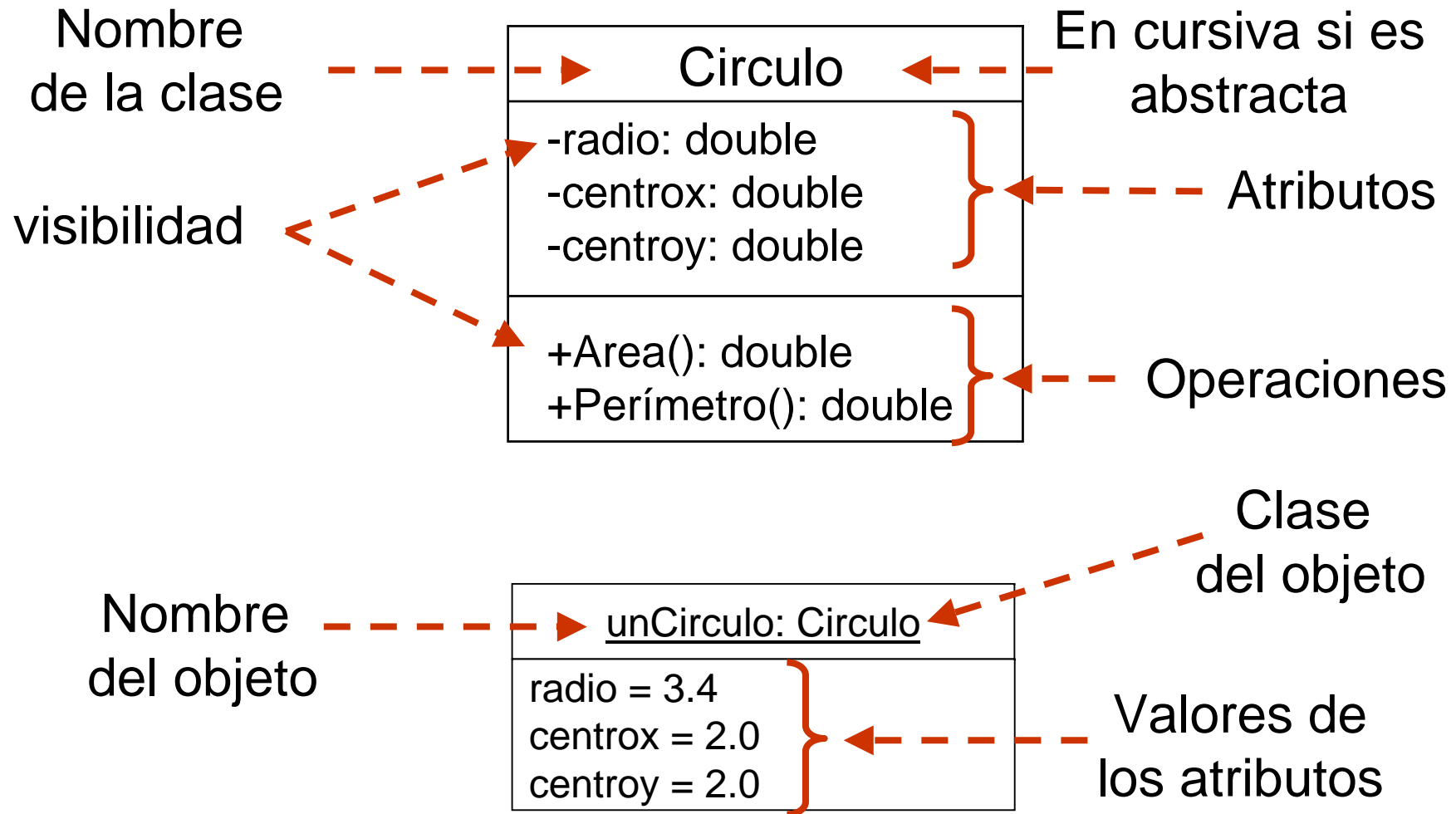


Diagrama de Objetos:



# Diagramas de estructura

## Clases y objetos



# Diagramas de estructura

## Atributos



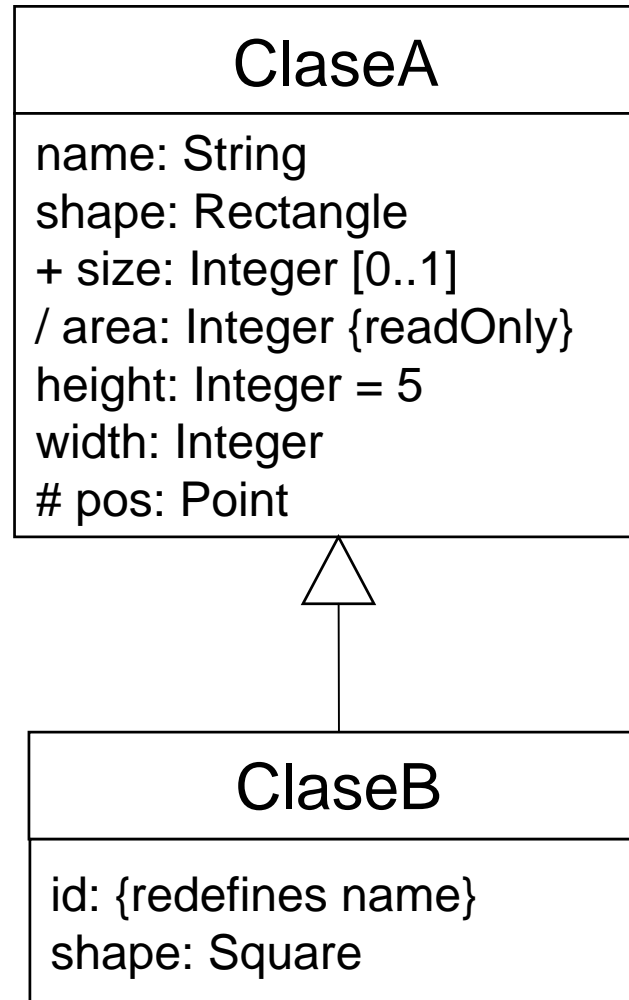
- Notación para atributos de clases:

*[visibilidad] [/] nombre [:tipo] [multiplicidad] [= valor] [{propiedad}]*

- Visibilidad: + (público) – (privado) # protegido
- /: indica que el atributo es derivado
- Multiplicidad: va entre [ ] y vale 1 por defecto
- Propiedades: {readOnly}, {union}, {subsets <property-name>}, {redefines <property-name>}, {ordered}, {bag}, {seq}, {sequence}, {composite}
- Un atributo subrayado es estático

# Diagramas de estructura

## Atributos. Ejemplo





# Diagramas de estructura

## Métodos



- Notación para métodos de clases:

*[visibilidad] nombre ( [parámetros] ) [: tipo-retorno] [{propiedad}]*

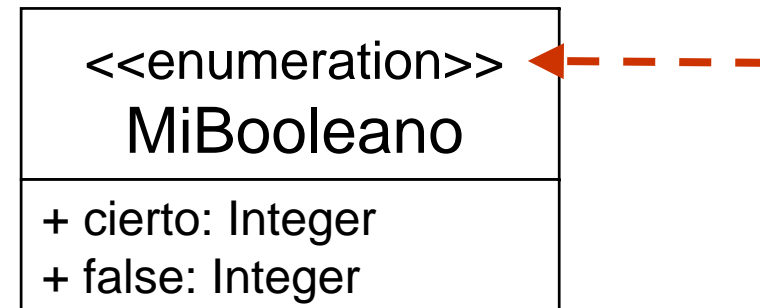
- Visibilidad: + (público) – (privado) # protegido
- Parámetros: separados por comas
- Propiedades: {query}, {update}, {concurrent}, {abstract}, {constructor}
- Un método subrayado es estático
- Ejemplos:
  - display()
  - + toString(): String

# Diagramas de estructura

## Estereotipos



- Estereotipo: extensión del vocabulario de UML que permite crear nuevos bloques derivados de los existentes, pero específicos a un problema concreto
- Ejemplos:
  - <<type>>
  - <<enumeration>>
  - <<interface>>
  - ...

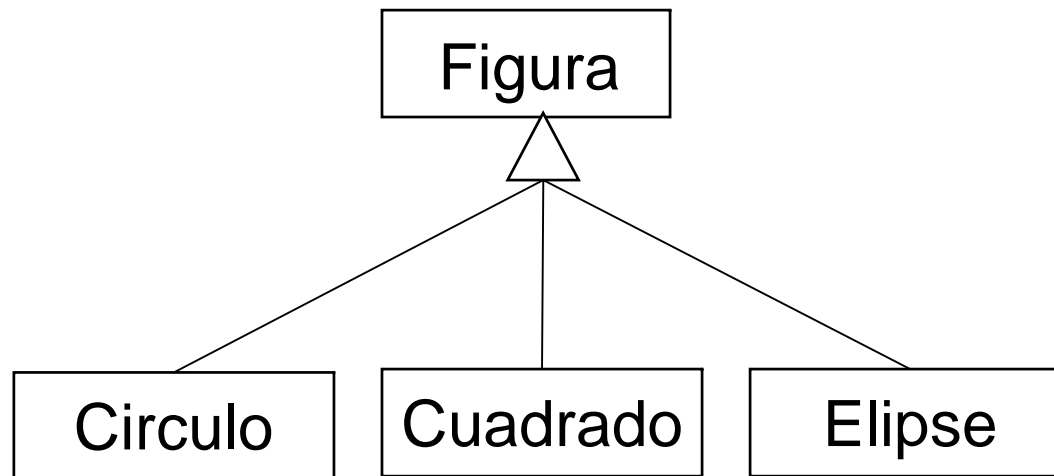


# Diagramas de estructura

## Relación de generalización



- Especialización



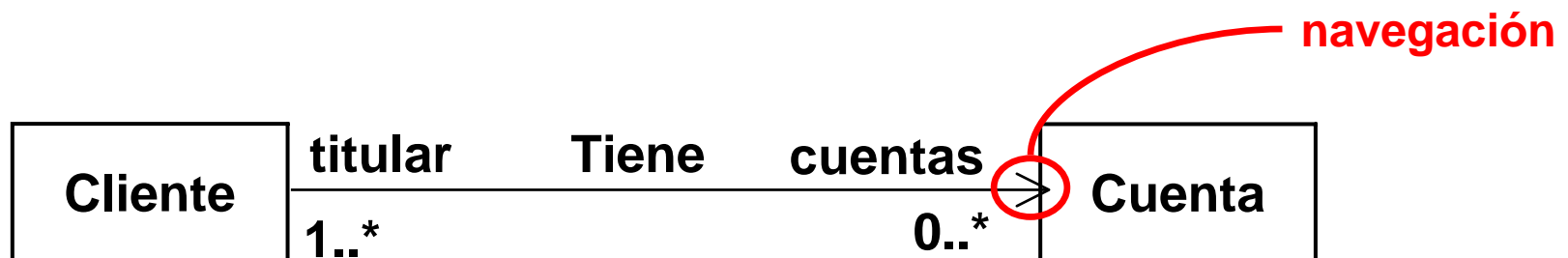
- Clasificación múltiple
- Clasificación dinámica

# Diagramas de estructura

## Relación de asociación



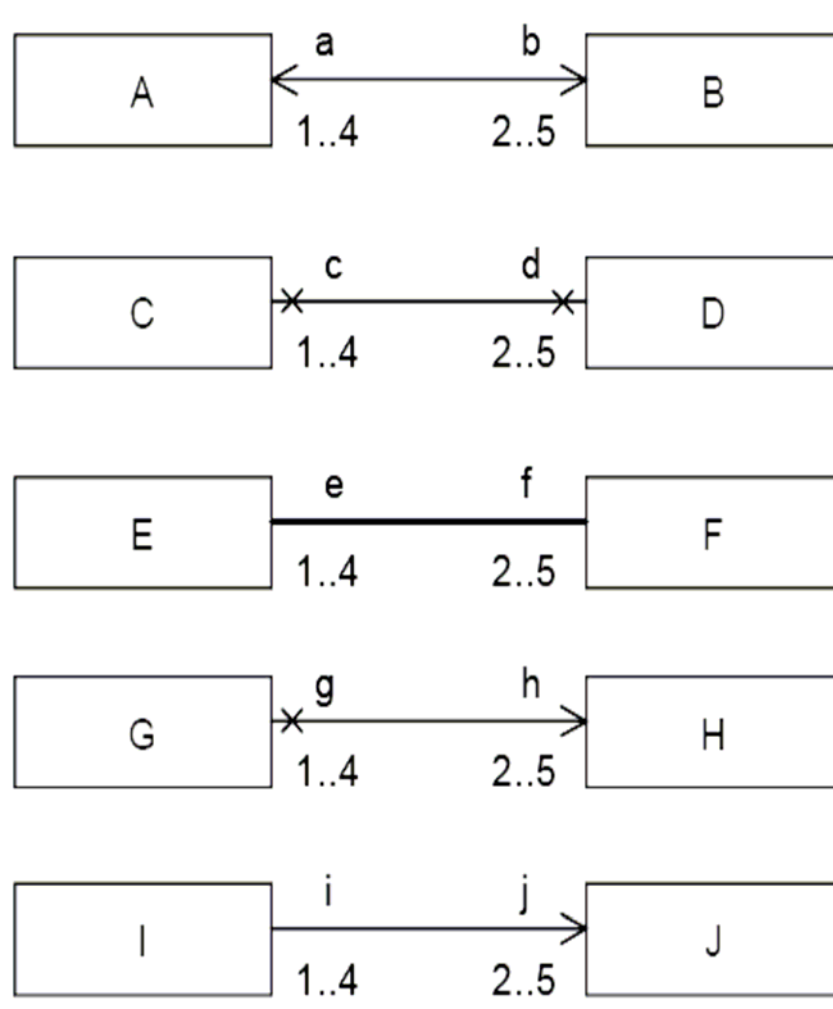
- Especifica que una clase utiliza otra clase
- Pueden tener etiquetas: nombre, roles, multiplicidad



- Ejemplos de cardinalidad:
  - 1..\* mínimo 1, no hay máximo
  - 0..1 mínimo 0, máximo 1
  - 1,2 uno o dos
  - 3 exactamente 3
- Navegación:
  - unidireccional
  - bidireccional
  - no especificado
  - no navegable (x)

# Diagramas de estructura

## Relación de asociación

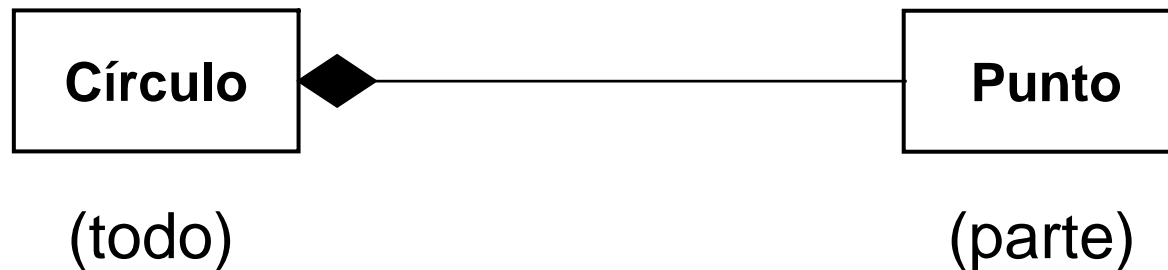


# Diagramas de estructura

## Relación de composición



- Relación del tipo todo/parte



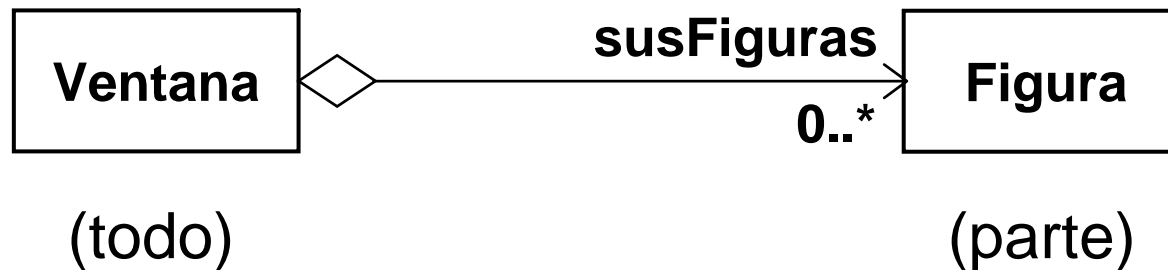
- Es una relación fuerte:
  - Si el *Círculo* se copia o elimina, también lo hace el *Punto*
  - La cardinalidad en la parte del todo es 0..1 ó 1

# Diagramas de estructura

## Relación de agregación



- Relación del tipo todo/parte



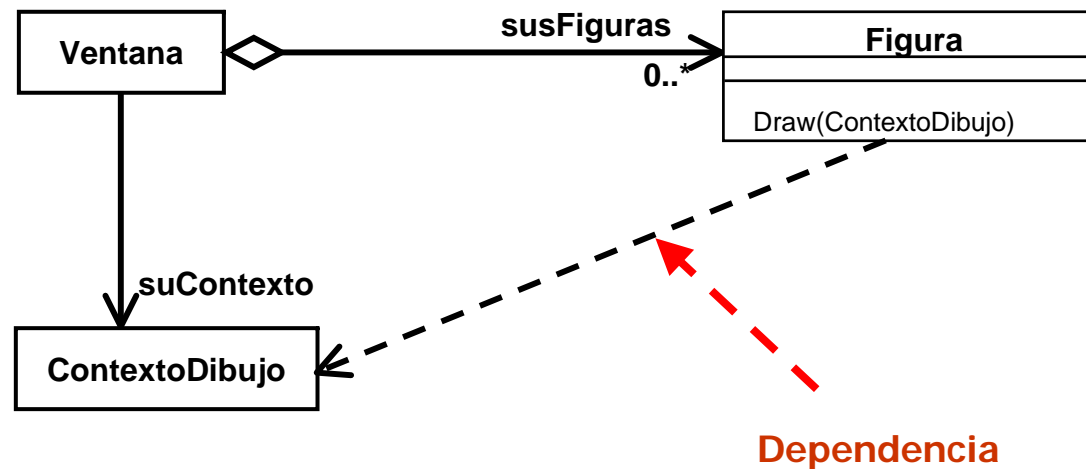
- No es una relación fuerte:
  - La *Ventana* contiene *Figuras*, pero cada una puede existir sin la otra

# Diagramas de estructura

## Relación de dependencia



- Relación débil de uso que declara que un cambio en la especificación de un elemento puede afectar a otro que lo utiliza





# Diagramas de estructura

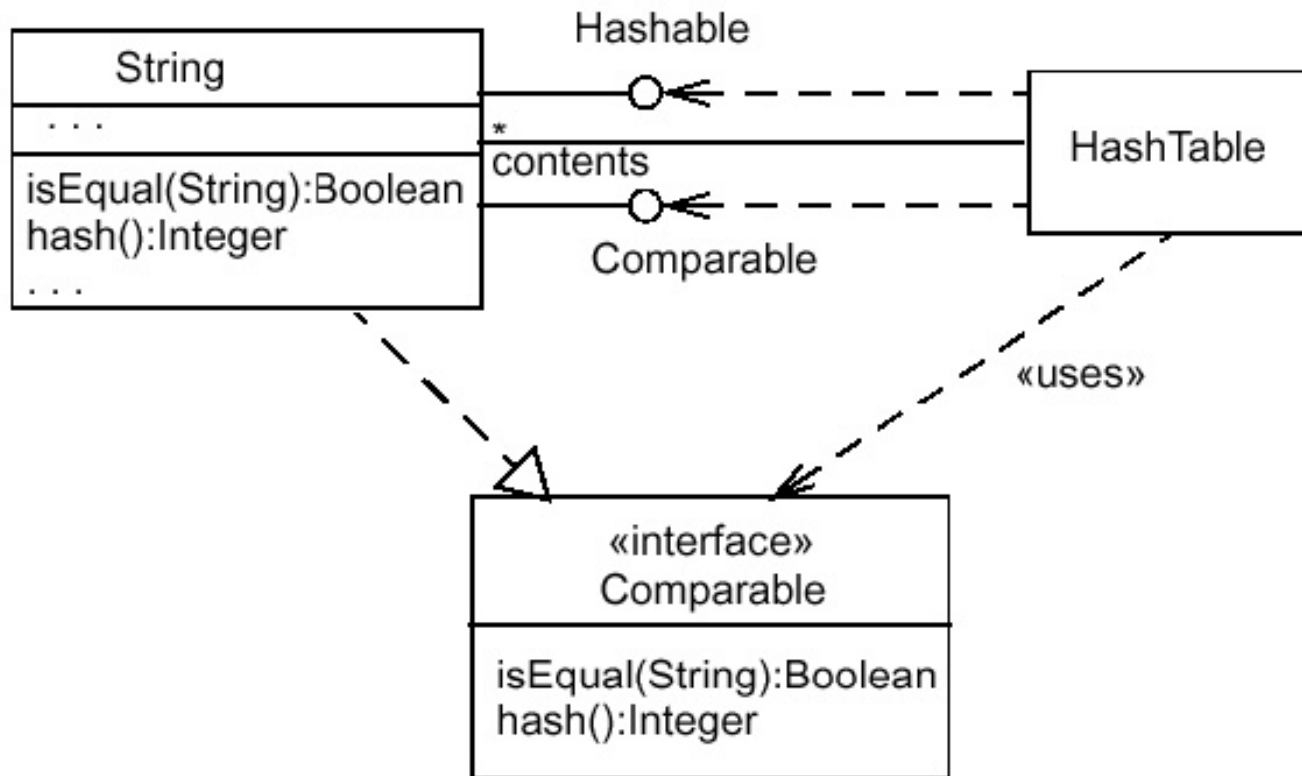
## Interfaces



- Colección de operaciones que especifican un servicio de una clase o componente
- Separa especificación e implementación de una abstracción
- Incluyen:
  - Nombre
  - Operaciones sin implementación
  - Relaciones de realización
  - Pueden tener relaciones de generalización
- No incluyen:
  - Atributos
  - Asociaciones

# Diagramas de estructura

## Interfaces. Ejemplo



# Diagramas de estructura

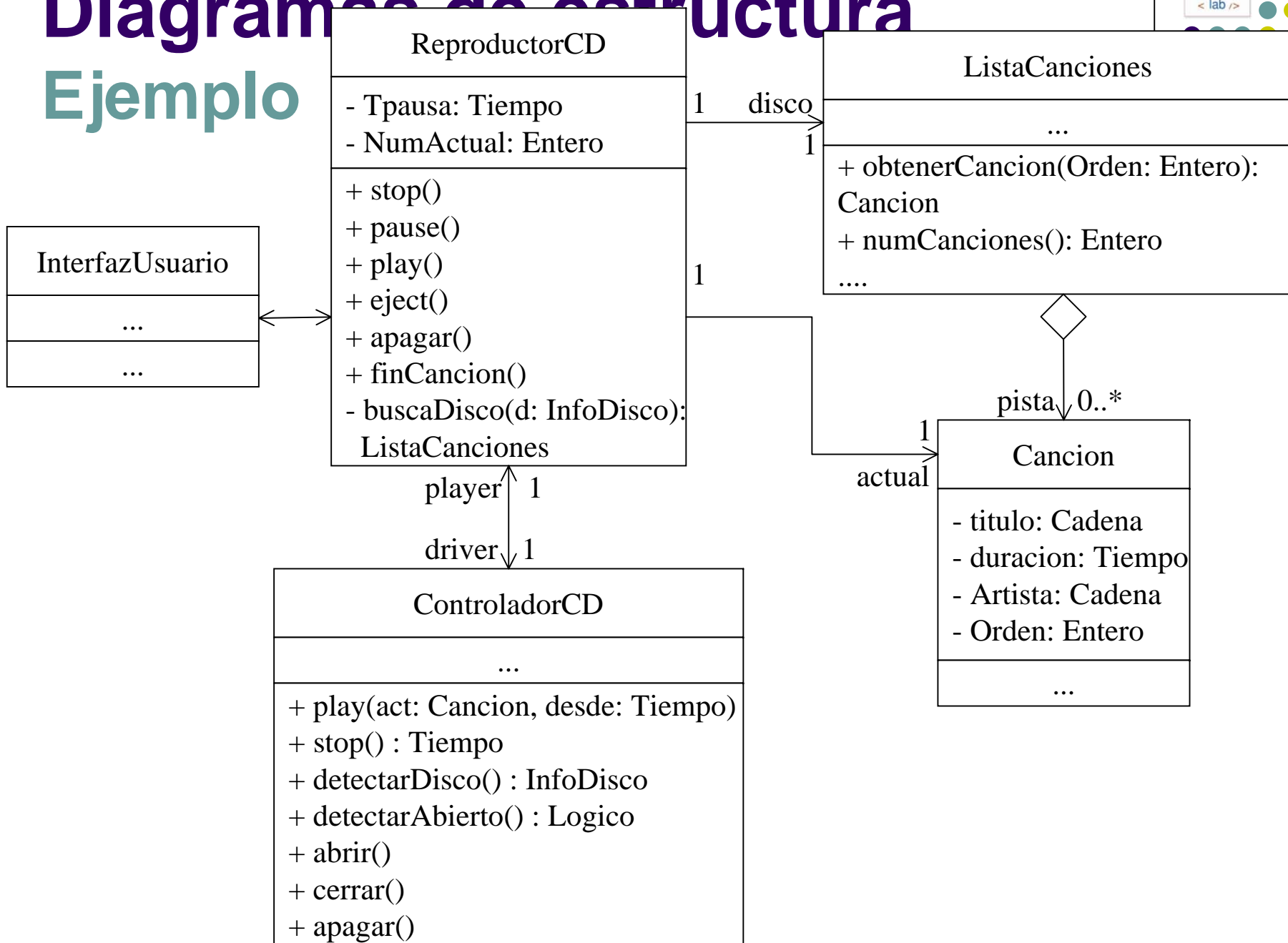
## Estilo



- Los atributos no deben ser objetos (usar relaciones en tal caso)
- Los diagramas de clases no suelen incluir (son detalles de implementación):
  - Constructores
  - Métodos de acceso (“*get*”/“*set*”)
  - Métodos de gestión de elementos de una asociación o agregación (por ej. “*add*”/“*remove*”)

# Diagramas de estructura

## Ejemplo



# Diagramas de estructura

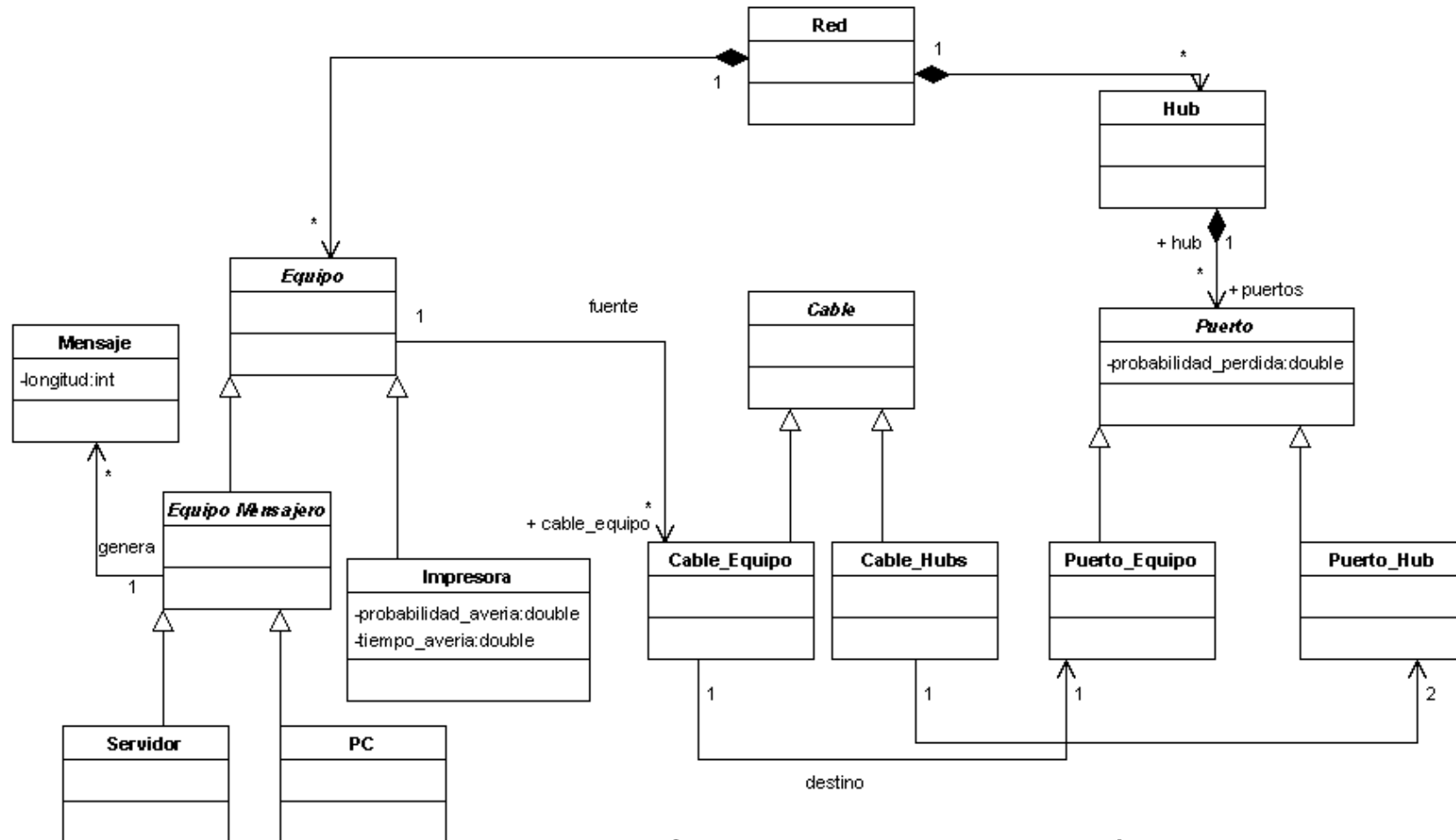
## Ejercicio



- Definir un diagrama de clases que describa una red de ordenadores.
- Los elementos que puede incluir la red son:
  - Servidores, PCs, Impresoras
  - Hubs, Cables de red
- Los PCs pueden conectarse con un único Hub, mientras que los Servidores pueden conectarse con uno o varios.
- Servidores y PCs pueden generar mensajes de cierta longitud.
- Los Hubs tienen un número de puertos, algunos de los cuales puede usarse para conectar con otros Hubs. Además tienen cierta probabilidad de “perder” mensajes.
- Existe cierta probabilidad de que las Impresoras se averíen durante cierto tiempo.

# Diagramas de estructura

## Ejercicio: posible solución



El que los PCs puedan conectarse con un único Hub y los servidores con uno o varios puede modelarse mediante una restricción OCL, o añadiendo asociaciones desde Servidor y PC

# Diagramas de comportamiento

## Colaboraciones

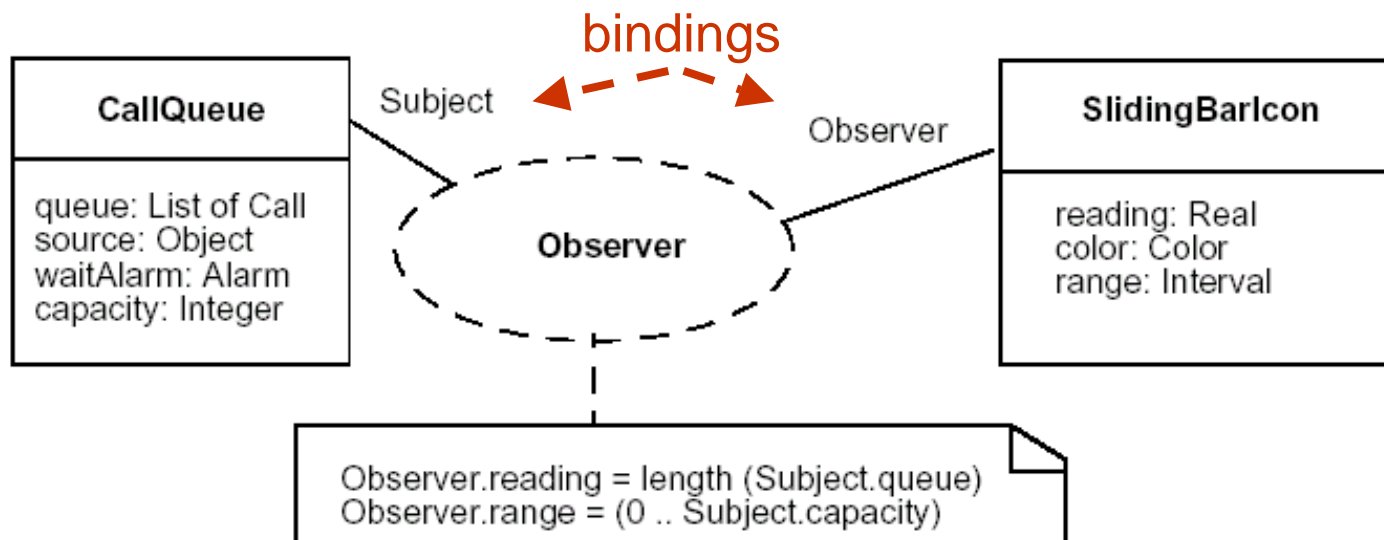
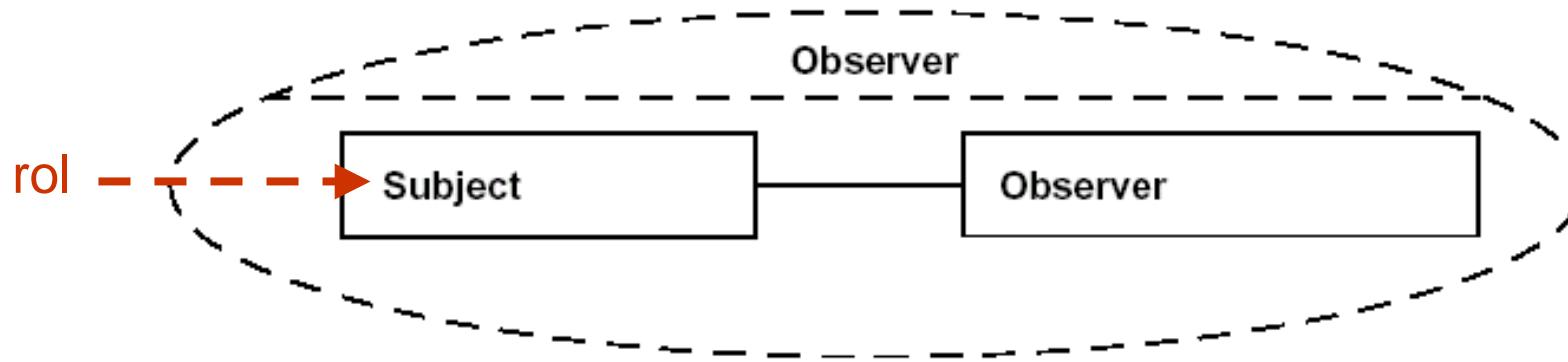


- Una colaboración es una interacción entre dos o más clases
- Muestra la implementación de una operación o la realización de un caso de uso
- Pueden describirse con más de un diagrama de interacción
- Patrones = colaboraciones parametrizadas

# Diagramas de comportamiento Colaboraciones. Ejemplo



Estructura interna de una colaboración





# Diagramas de comportamiento

## Diagramas de interacción



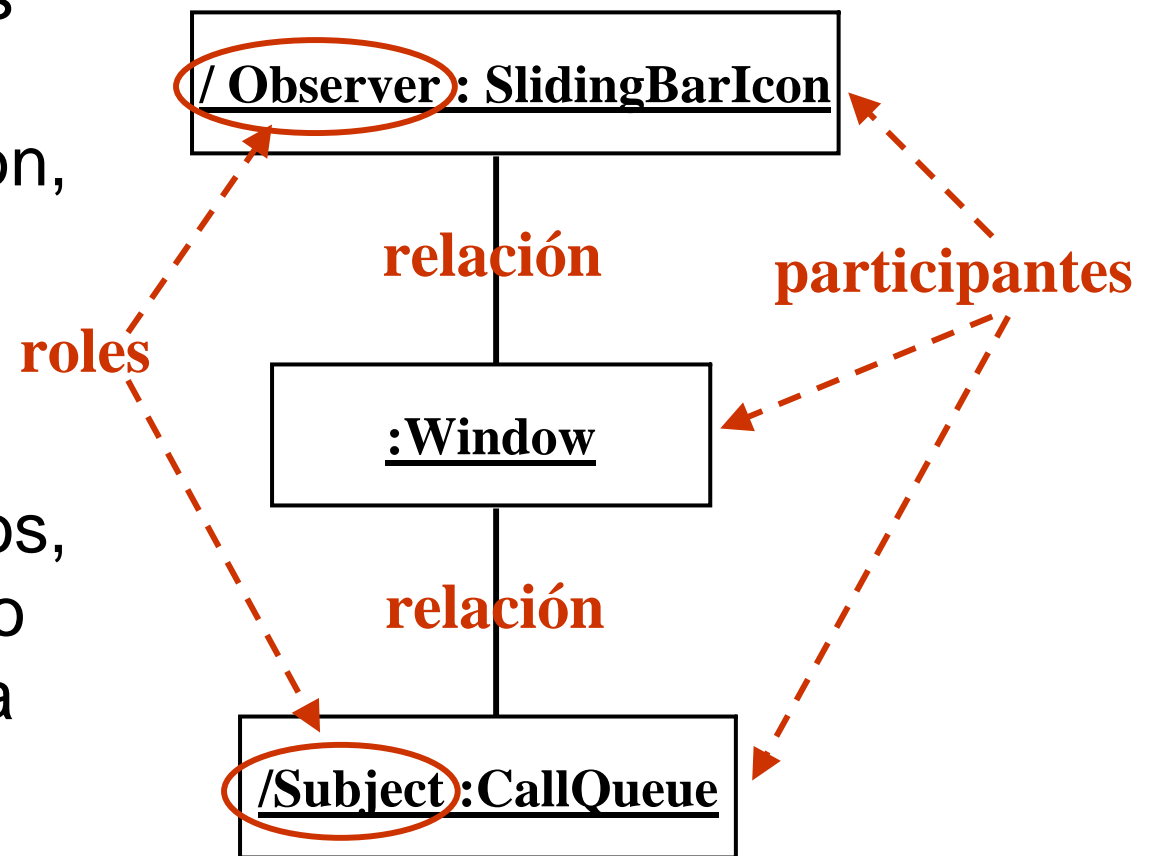
- Interacción: comportamiento que comprende una secuencia de mensajes intercambiados entre un conjunto de objetos de una colaboración, dentro de un contexto determinado, para alcanzar cierto fin
- Especifican la secuencia de mensajes para cumplir el objetivo de la colaboración
- Tipos de diagrama de interacción:
  - de secuencia: destaca la ordenación temporal
  - de comunicación (o colaboración): destaca la organización estructural de los objetos involucrados

# Diagramas de comportamiento

## Diagramas de comunicación



- Objetos necesarios (roles o instancias) para una interacción, y sus relaciones
- Similar a un diagrama de objetos, muestra el contexto necesario para una colaboración



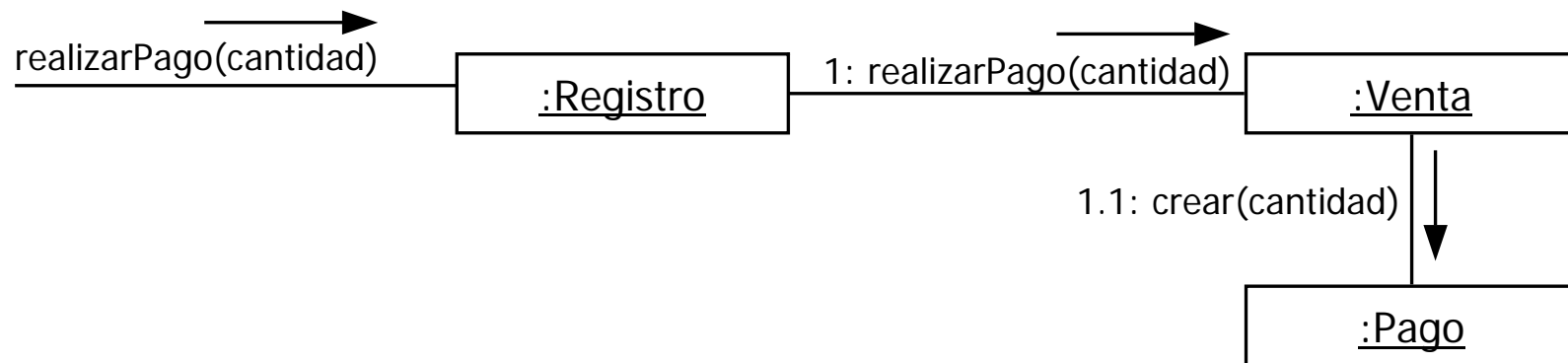
# Diagramas de comportamiento

## Diagramas de comunicación

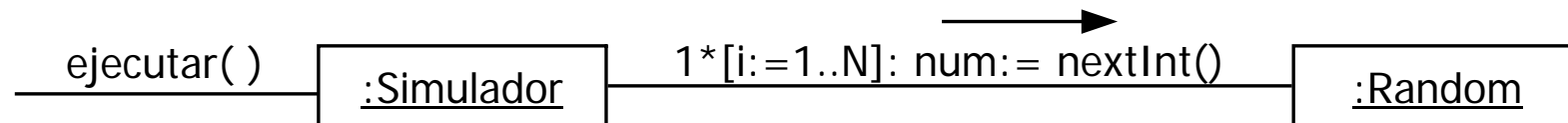


- Pueden mostrar la secuencia de mensajes

### *Llamadas anidadas:*

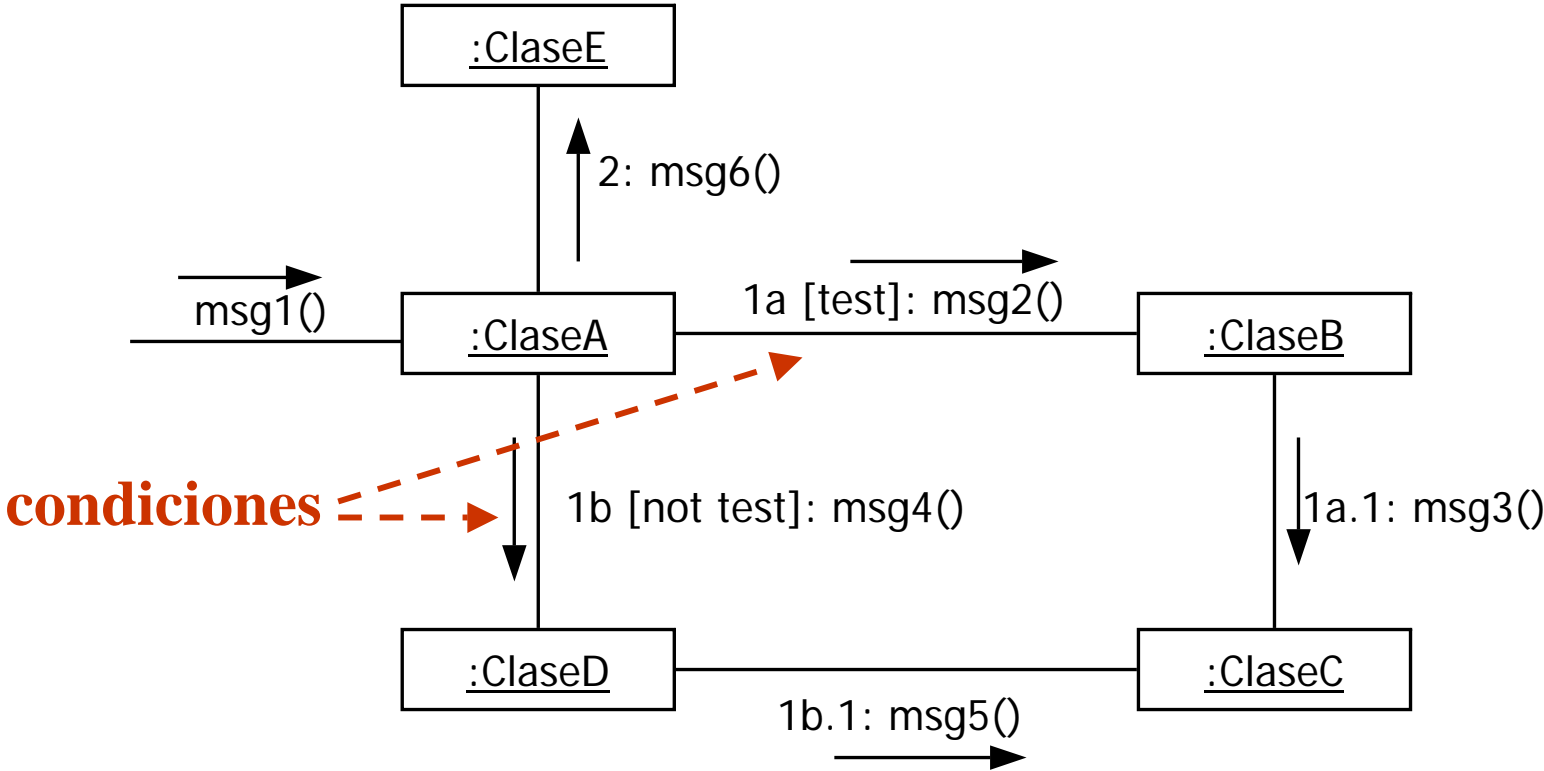


### *Iteraciones:*



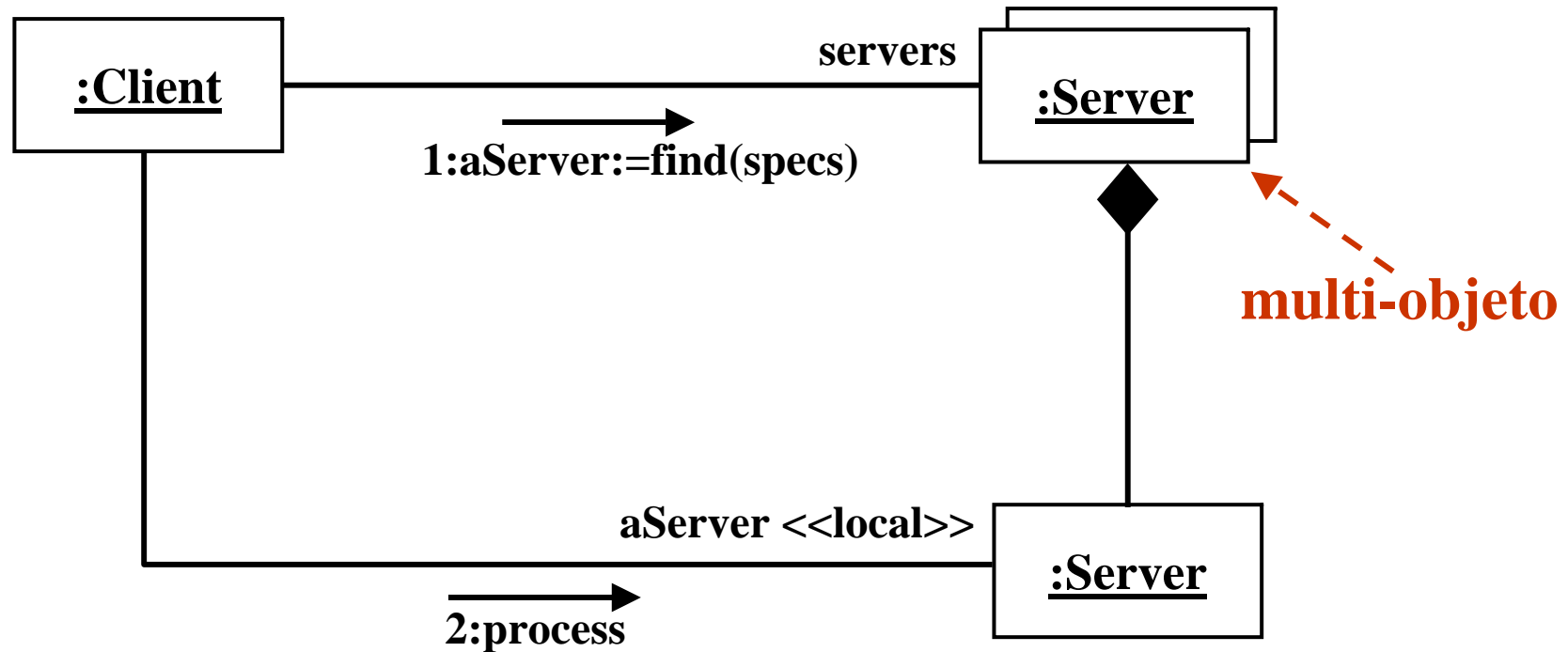
# Diagramas de comportamiento

## Diagramas de comunicación



# Diagramas de comportamiento

## Diagramas de comunicación



# Diagramas de comportamiento

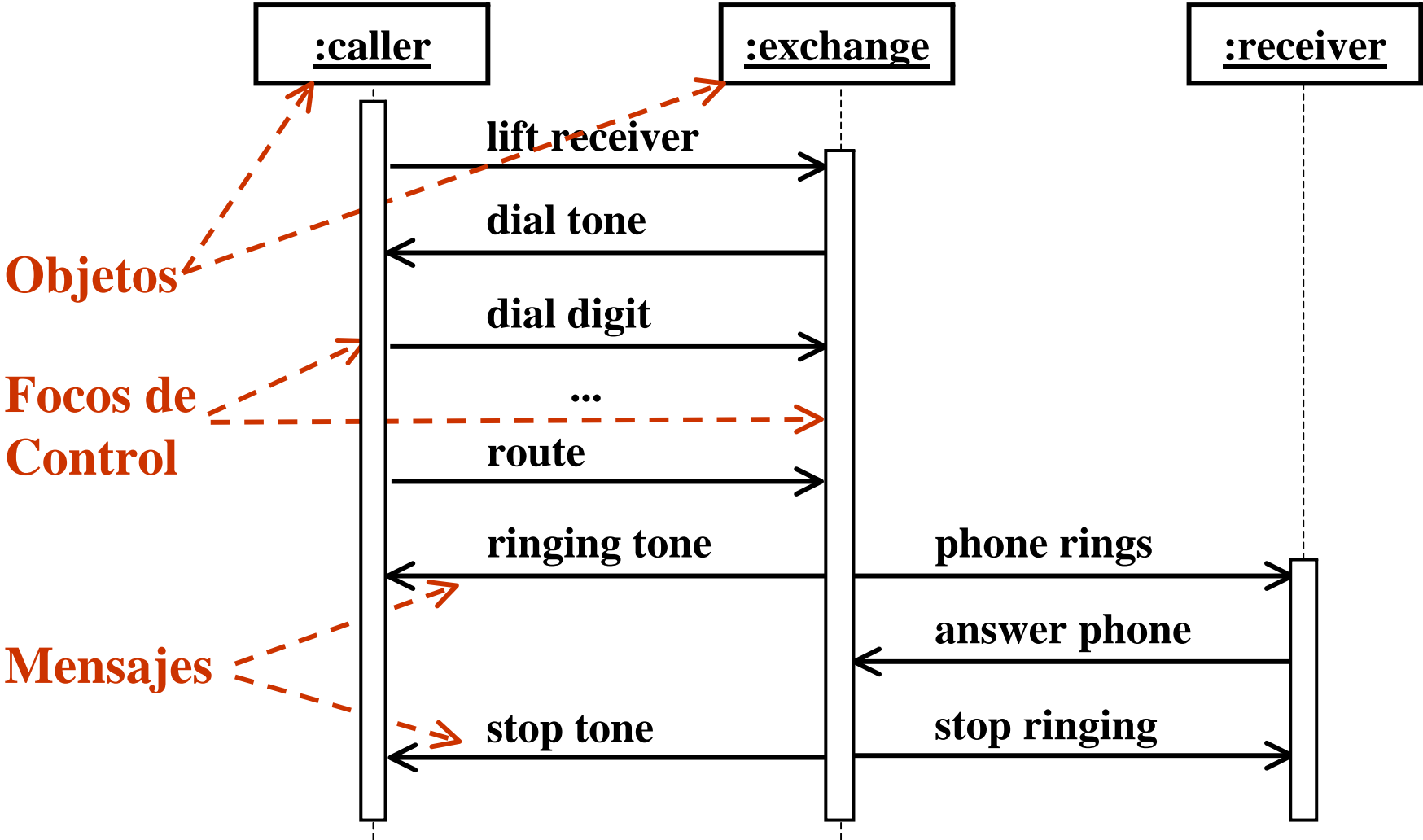
## Diagramas de secuencia



- Mensajes entre objetos (roles o instancias) para una interacción
- Dos dimensiones:
  - Temporal: generalmente vertical
  - Instancias: generalmente horizontal. El orden relativo de las instancias no tiene importancia
- Muestra la existencia y duración de las instancias, pero no sus relaciones

# Diagramas de comportamiento

## Diagramas de secuencia






# Diagramas de comportamiento

## Diagramas de secuencia



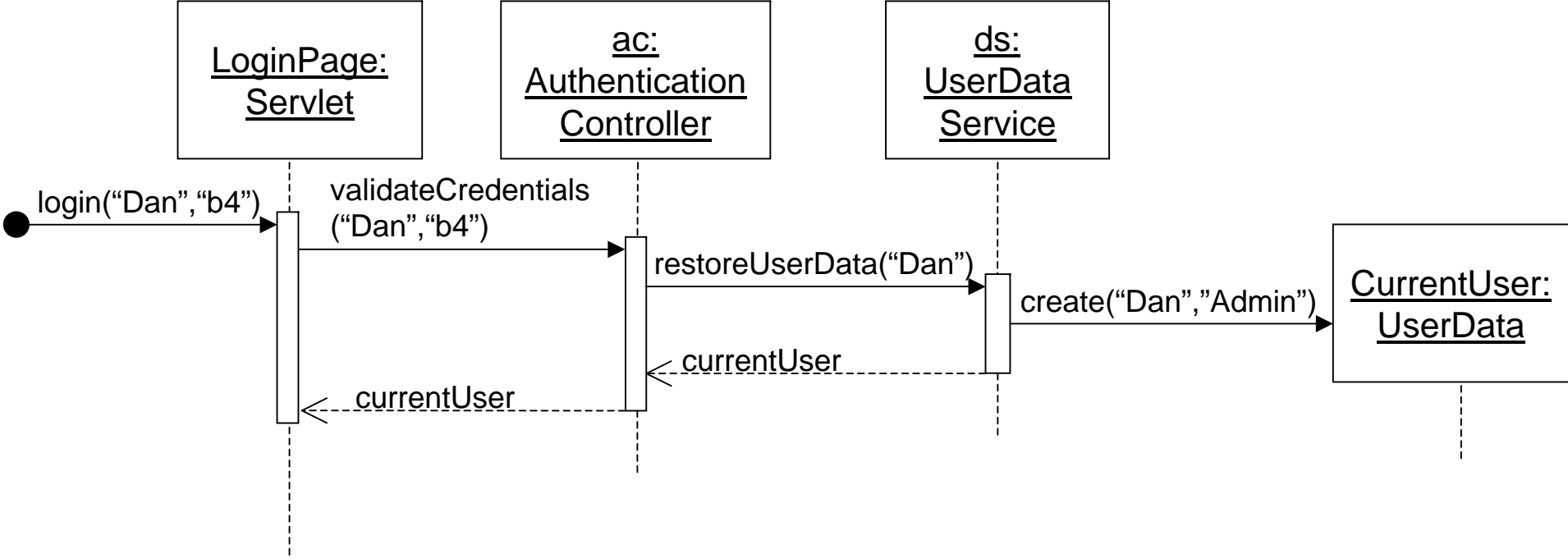
### Tipos de flujo de control

	Llamada a procedimiento u otra forma de llamada con anidamiento de control. La secuencia anidada termina antes de que siga la operación que invocó. Puede usarse para procesos concurrentes cuando el mensaje es síncrono.
	Comunicación asíncrona, sin anidamiento de control. El objeto que envía no se detiene a esperar respuesta.
	Retorno de una llamada a procedimiento. Puede omitirse si queda claro por el fin de la activación.



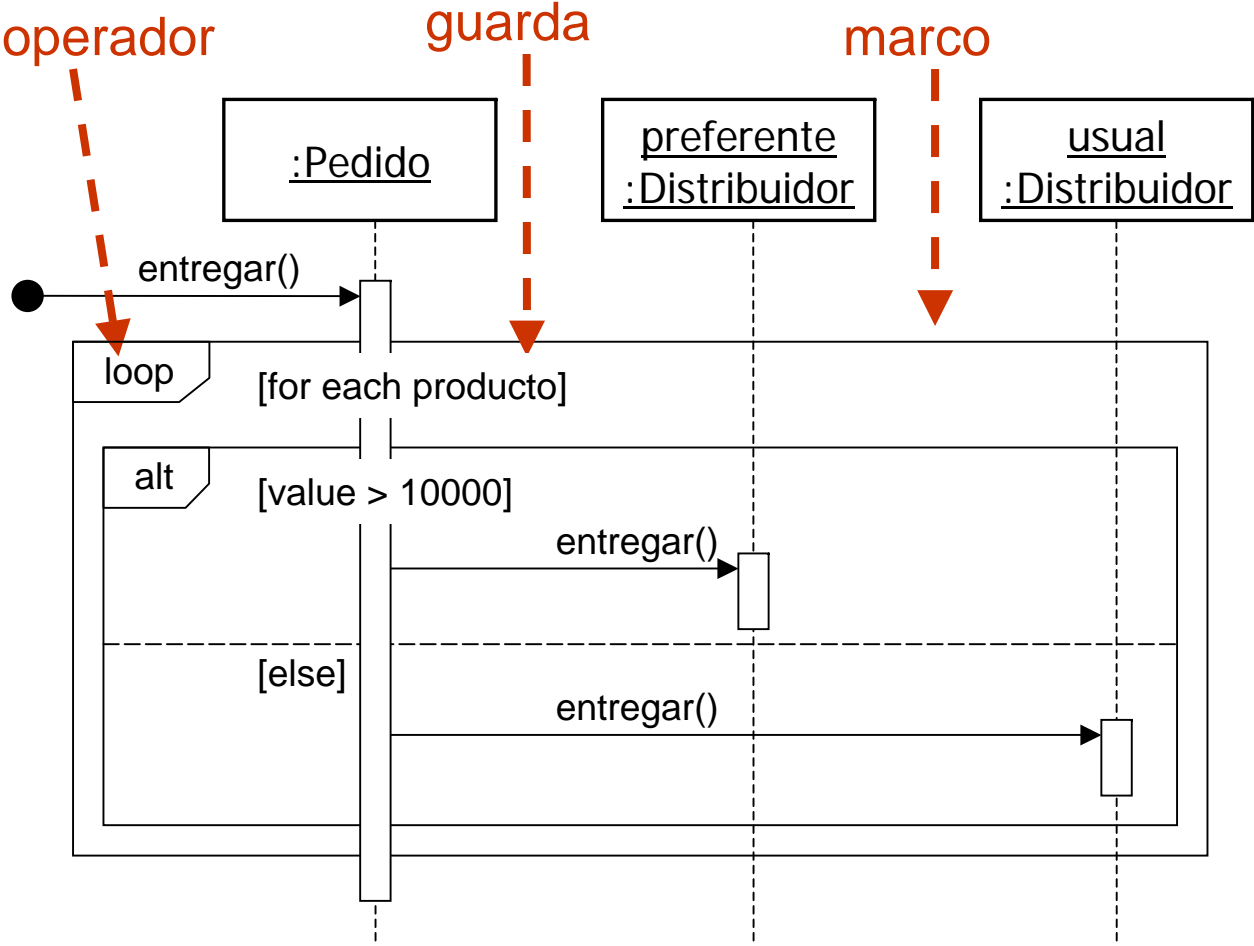
# Diagramas de comportamiento

## Diagramas de secuencia



# Diagramas de comportamiento

## Diagramas de secuencia



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

# Diagramas de comportamiento

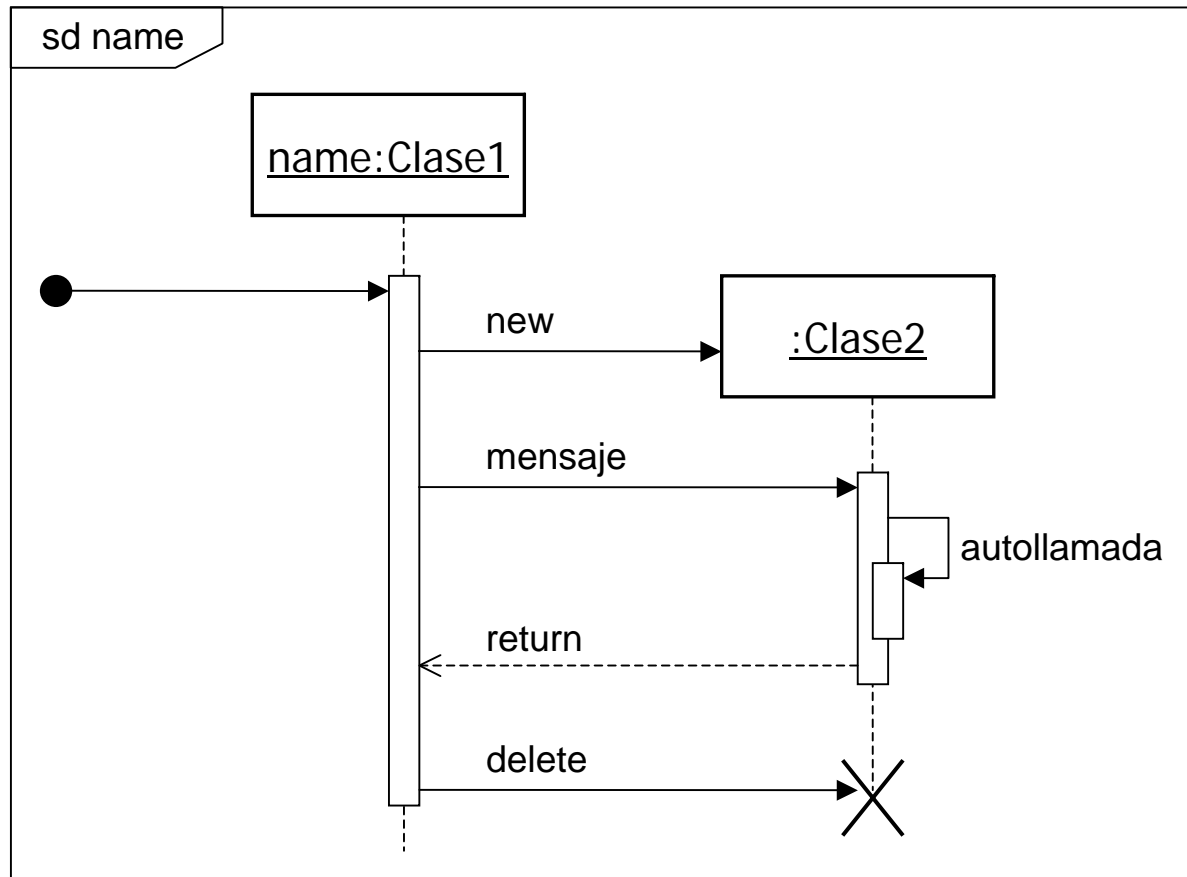
## Diagramas de secuencia



- 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.

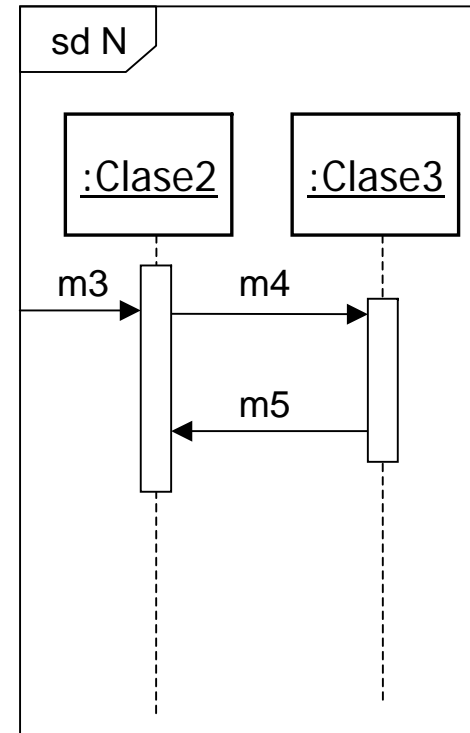
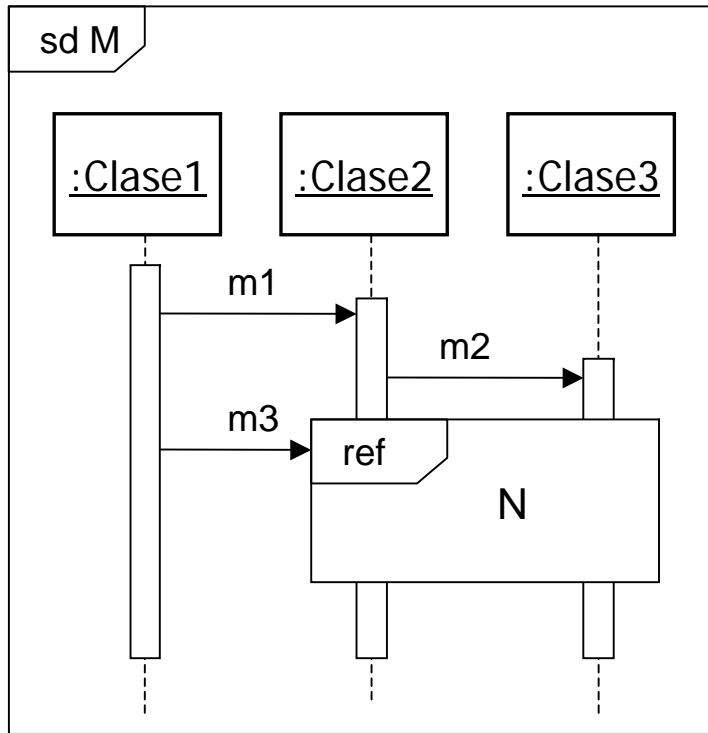
# Diagramas de comportamiento

## Diagramas de secuencia



# Diagramas de comportamiento

## Diagramas de secuencia



# Diagramas de comportamiento

## Ejercicio

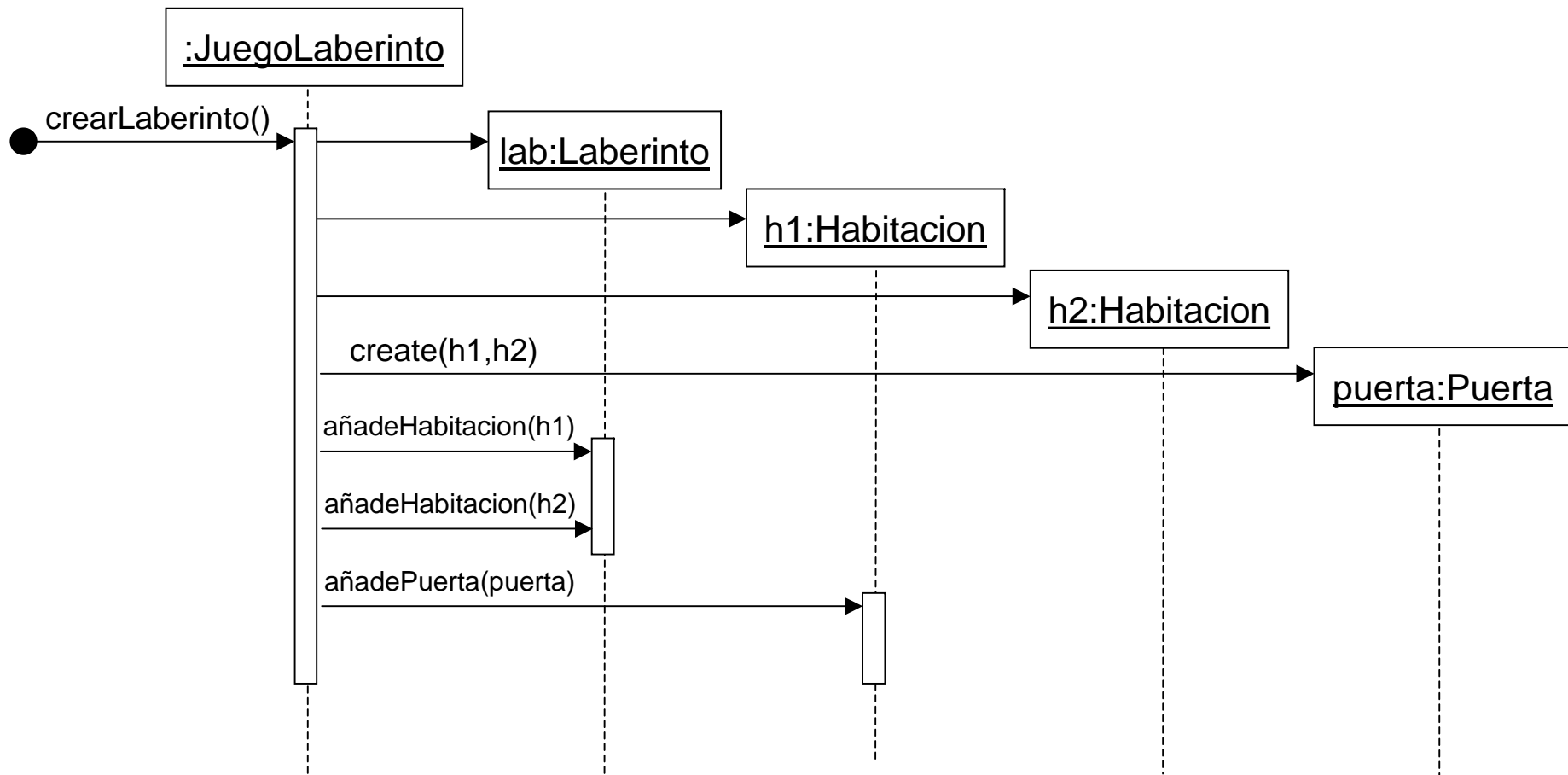


Especificar el diagrama de secuencia de la operación “crearLaberinto”

```
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;
    }
}
```

# Diagramas de comportamiento

## Solución



# Diagramas de comportamiento

## Ejercicio



Especificar el diagrama de secuencia de la operación “crearLaberinto”

```
public class JuegoLaberinto {
    private Laberinto lab;
    private boolean conVentana;

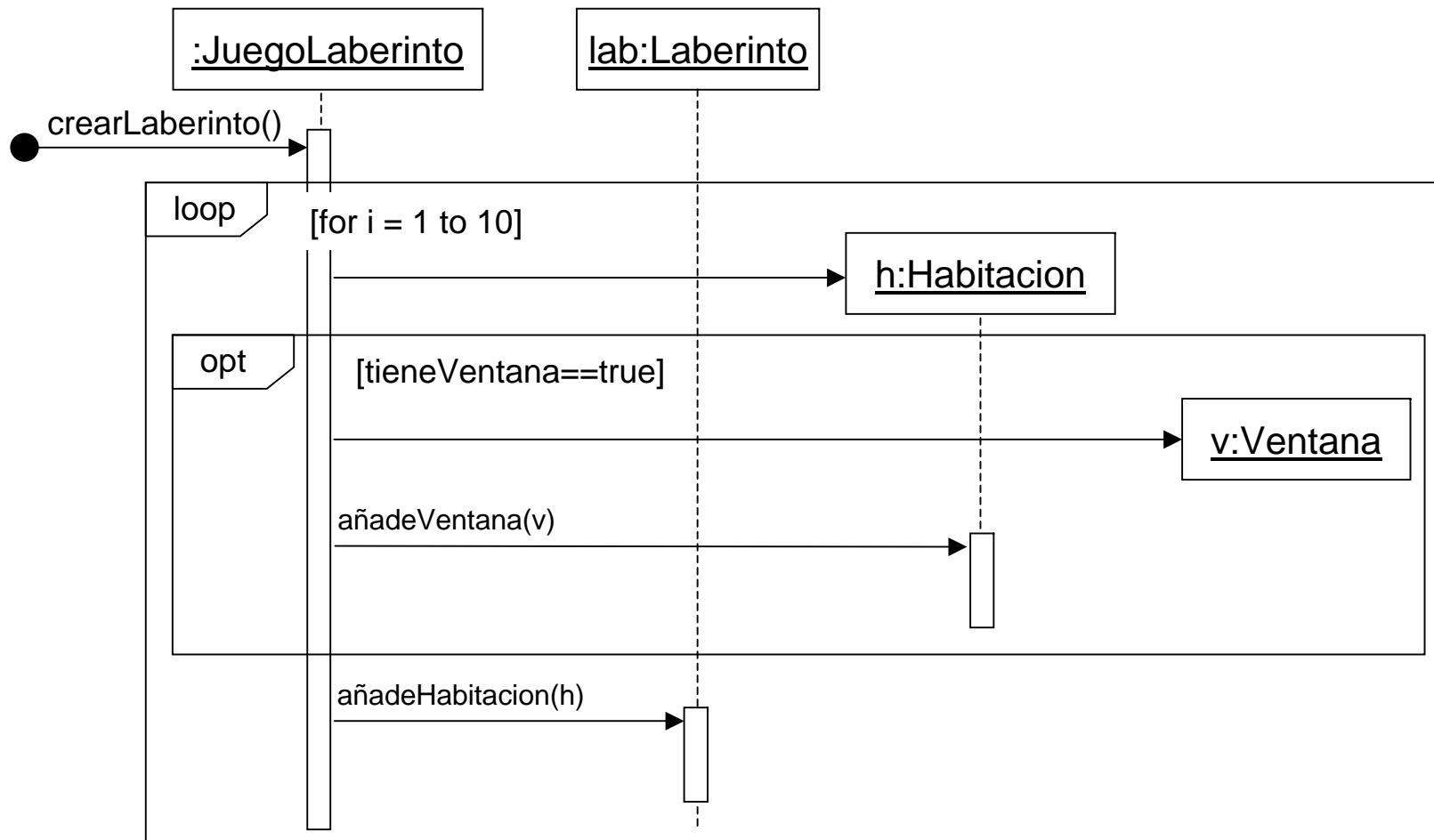
    public JuegoLaberinto() {
        lab = new Laberinto();
        conVentana = true;
    }

    public void crearLaberinto () {
        Habitacion h;
        for (int i=0; i<10; i++) {
            h = new Habitacion();
            if (conVentana == true)
                h.añadeVentana(new Ventana());
            lab.añadeHabitacion(h);
        }
    }
}
```



# Diagramas de comportamiento

## Solución

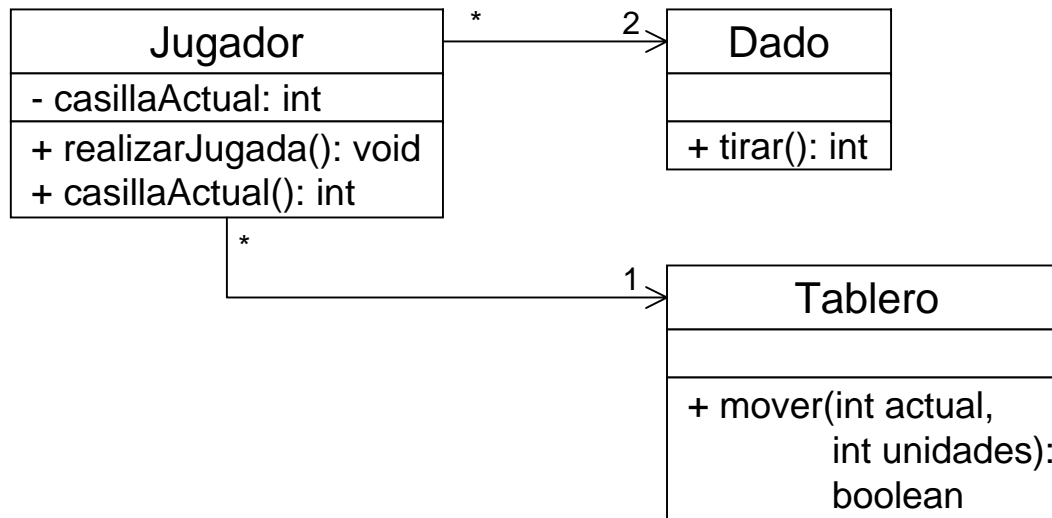


# Diagramas de comportamiento

## Ejercicio

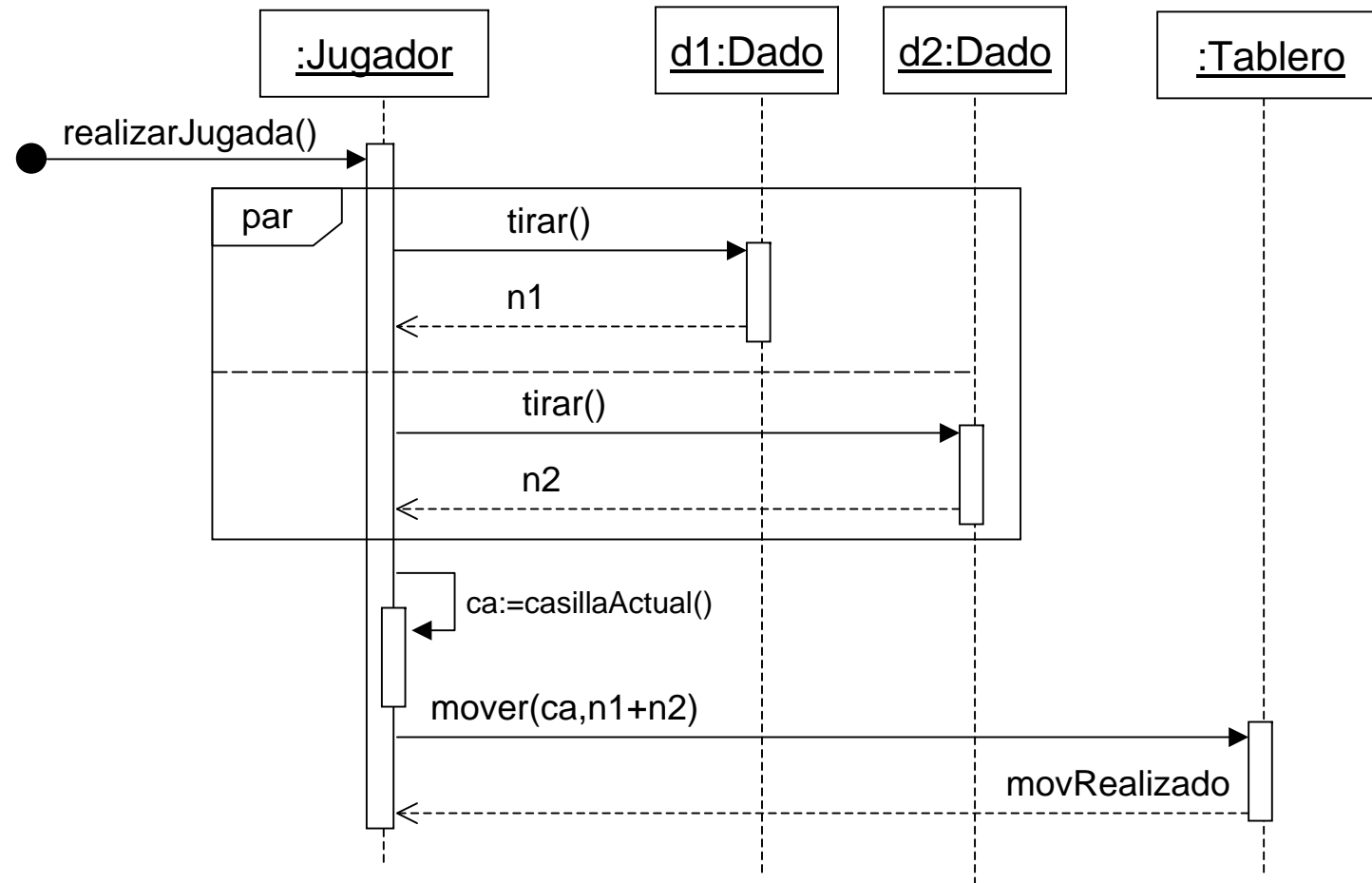


Especificar el diagrama de secuencia de la operación “realizarJugada” definida en la clase Jugador, para el juego del parchís



# Diagramas de comportamiento

## Solución



# Diagramas de comportamiento

## Ejercicio

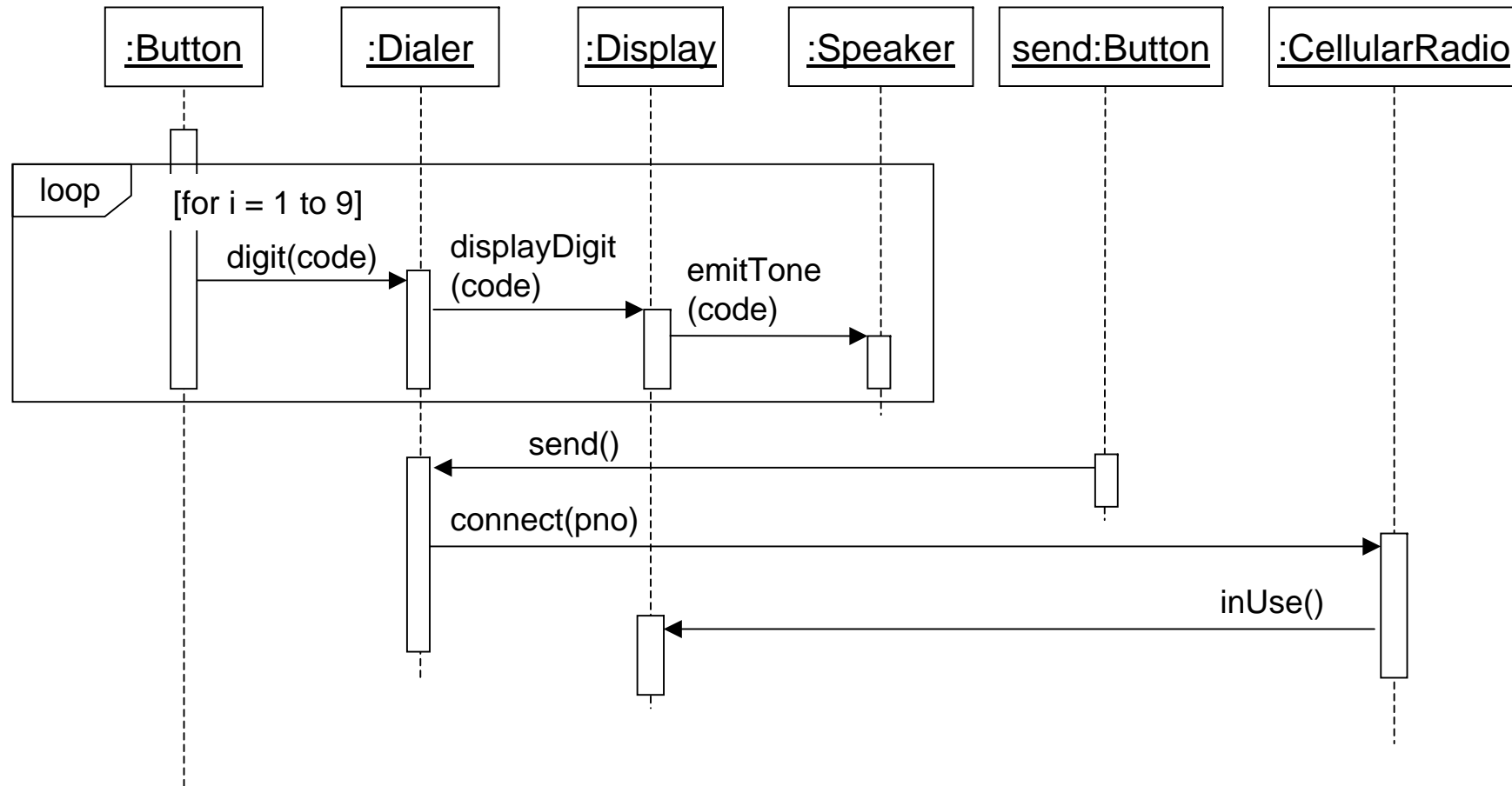


Identificar las clases relevantes y realizar el diagrama de secuencia para el siguiente caso de uso, que corresponde a la realización de una llamada desde un teléfono móvil.

- El usuario pulsa los dígitos del número de teléfono
- Para cada dígito
  - la pantalla se actualiza para añadir el dígito marcado
  - se emite un tono por el receptor
- El usuario pulsa el botón “Enviar”
- El indicador “en uso” se ilumina en pantalla
- El móvil establece conexión con la red
- Los dígitos acumulados se mandan a la red
- Se establece la conexión con el número marcado

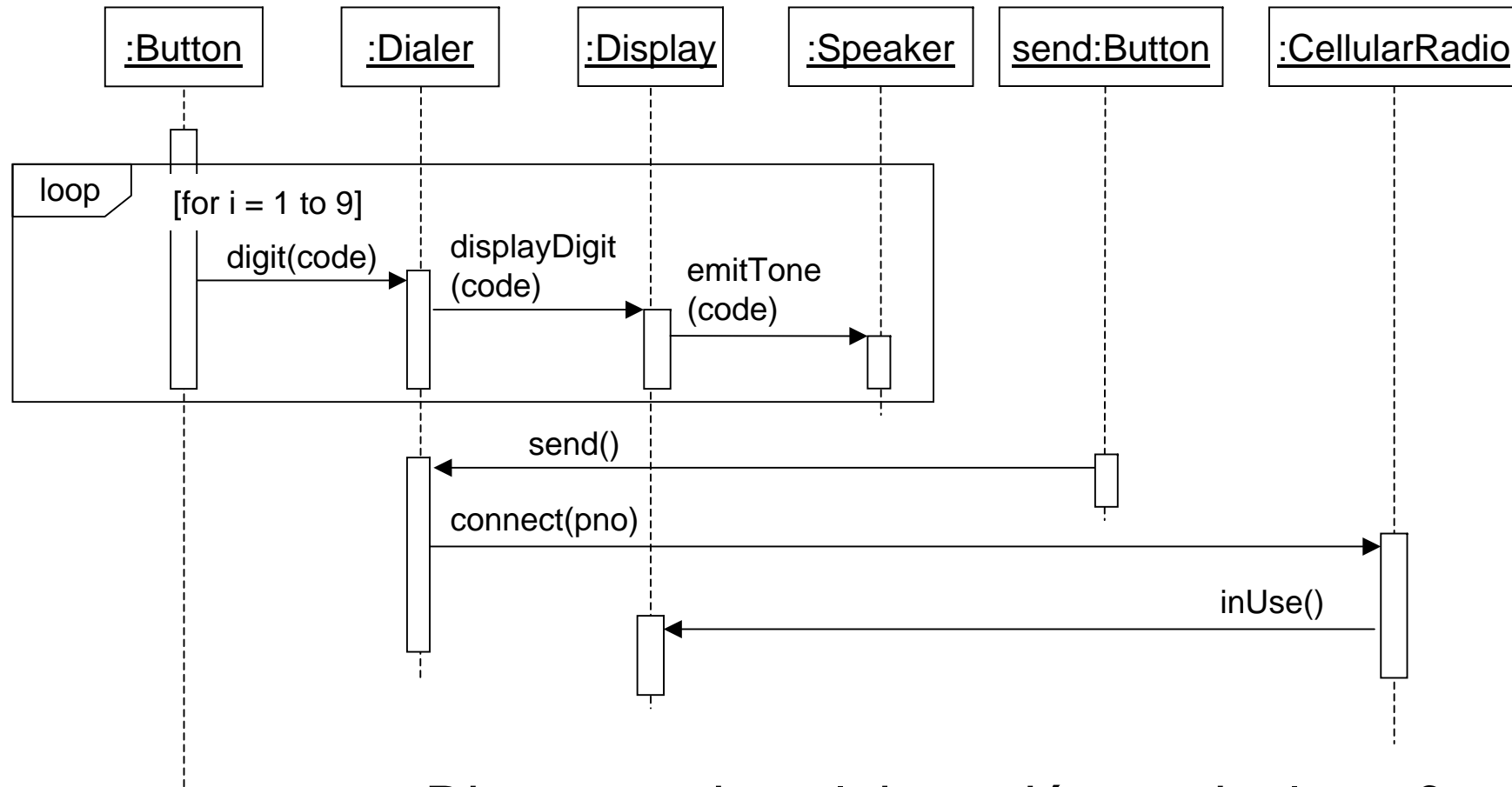
# Diagramas de comportamiento

## Ejercicio: posible solución



# Diagramas de comportamiento

## Ejercicio: posible solución



¿Diagrama de colaboración equivalente?

# Diagramas de comportamiento

## Solución

