

# UML: Diagramas de Interacción

Dpto. Lenguajes y Sistemas Informáticos  
Universidad de Granada

Programación y Diseño Orientado a Objetos

(Curso 2021-2022)

# Créditos

- Las siguientes imágenes e ilustraciones son libres y se han obtenido de:
  - ▶ Emojis, <https://pixabay.com/images/id-2074153/>
- El resto de imágenes e ilustraciones son de creación propia, al igual que los ejemplos de código

# Objetivos

- Saber interpretar los diagramas de secuencia y comunicación
- Saber implementarlos

# Contenidos

- 1 **Introducción**
- 2 **Diagramas de secuencia**
- 3 **Diagramas de comunicación**

# Diagramas de interacción

- Su propósito es **mostrar** el comportamiento del sistema a través de **las interacciones entre los elementos del modelo**
- Hay dos tipos básicos:
  - ▶ **Diagramas de secuencia:** Enfatizan la **secuencia temporal de los mensajes** enviados entre objetos
  - ▶ **Diagramas de colaboración:** Enfatizan la **relación entre los objetos receptores y emisores de los mensajes**
- Elementos:
  - ▶ Participantes: Objetos y clases que forman parte de la interacción
  - ▶ Mensajes: El flujo y su secuencia entre los participantes

# Diagramas de secuencia

- Los **participantes** se muestran en una caja

El nombre del objeto debe ir en minúscula y el de la clase en mayúscula. Presta atención a la localización de los dos puntos entre el nombre del objeto y el de la clase

wally : Robot

Objeto de la clase Robot llamado wally

wally

Objeto llamado wally

: Robot

Objeto anónimo de la clase Robot

Robot

La clase Robot

equipaje : Maleta

Colección de objetos de la clase Maleta llamada equipaje

: Maleta

Colección anónima de objetos de la clase Maleta

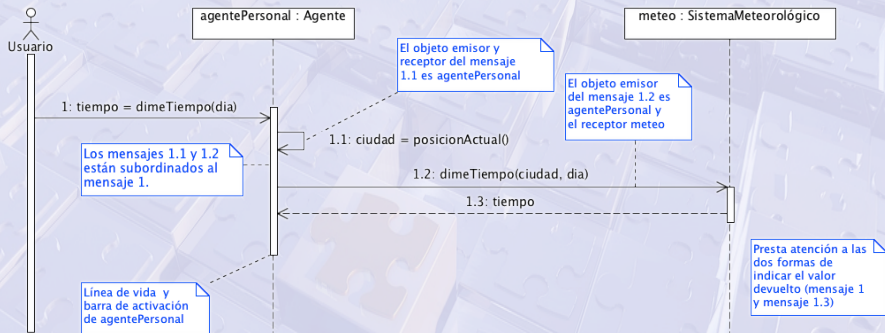
equipaje

Colección de objetos llamada equipaje, sin indicación de a qué clase pertenecen dichos objetos

Los **multiobjetos** o colecciones de objetos se representan con un doble fondo

# Diagramas de secuencia

## ● Mensajes: Emisor y Receptor



# Diagramas de secuencia

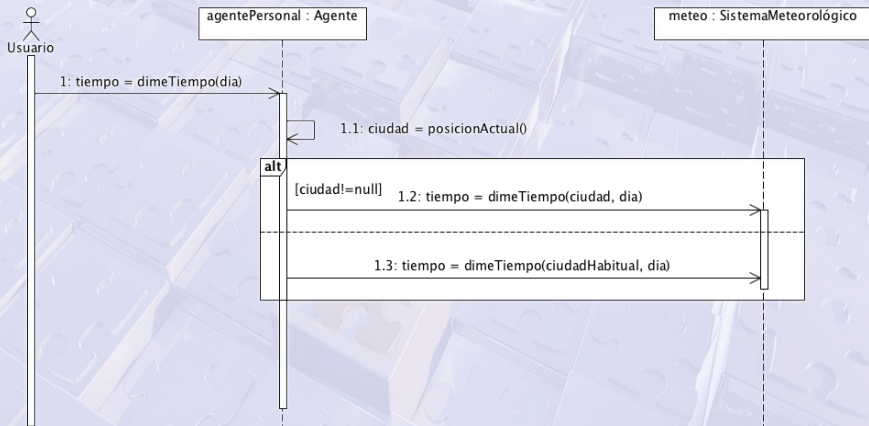
## Ruby: Implementación del diagrama anterior

```
1 class Agente
2
3   . . .
4
5   def dimeTiempo (día)
6     # No se indica receptor, es el propio objeto
7     ciudad = posicionActual
8
9     # ¿Cómo sabemos que meteo es un atributo?
10    @meteo.dimeTiempo (ciudad, día)
11
12    # Devuelve el resultado del último paso de mensaje
13  end
14
15   . . .
16
17 end
```



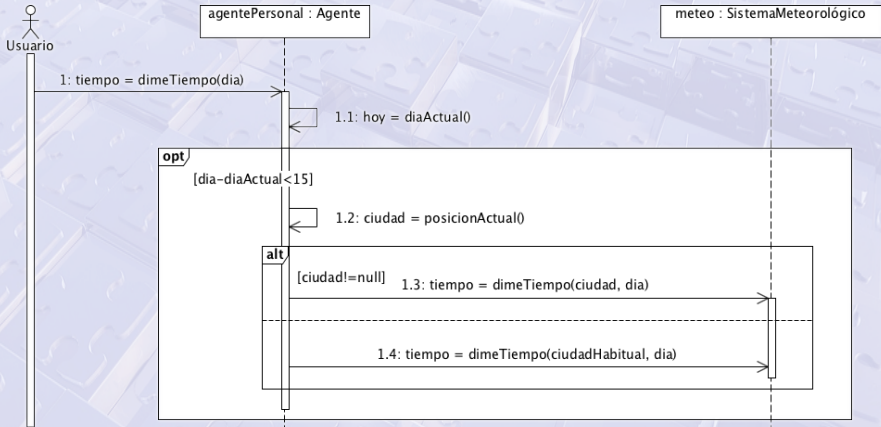
# Diagramas de secuencia

## ● Fragmentos: Condicionales



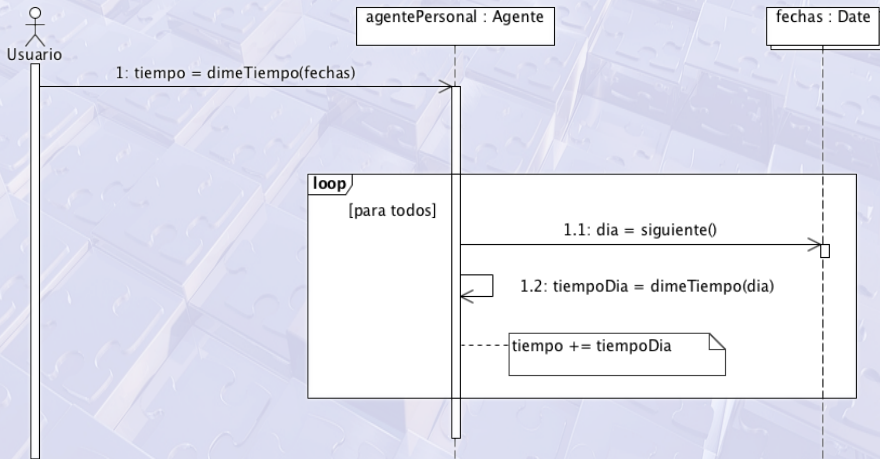
# Diagramas de secuencia

## ● Fragmentos: Condicionales



# Diagramas de secuencia

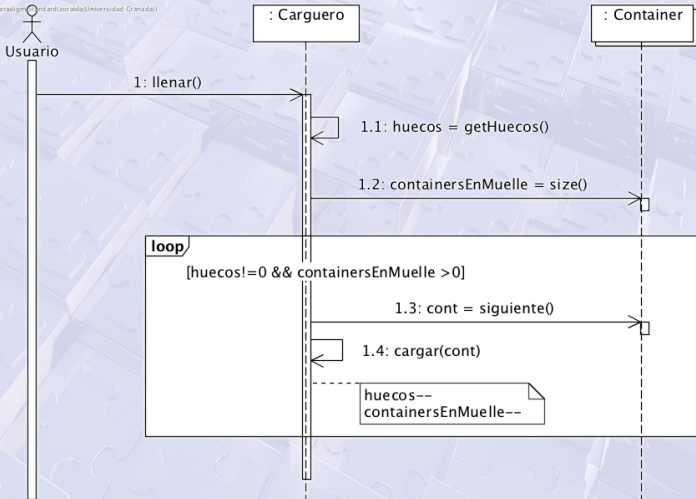
- Fragmentos: **Bucles**



# Diagramas de secuencia

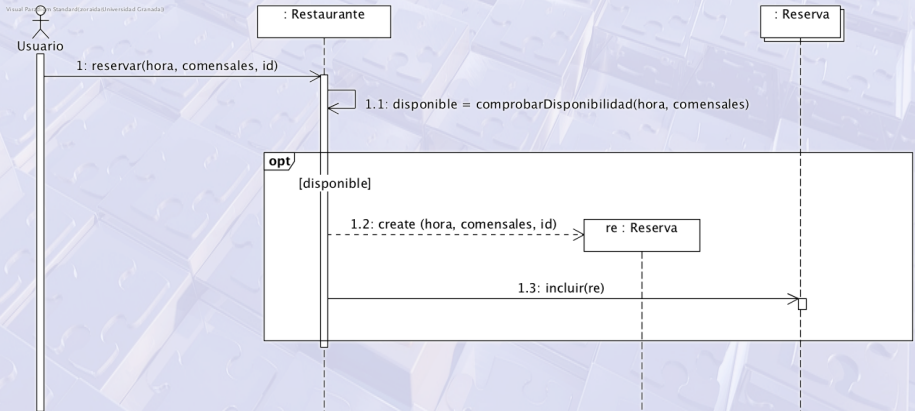
- Fragmentos: **Bucles**

Visual Paradigm (Standard)(Universidad Granada)



# Diagramas de secuencia

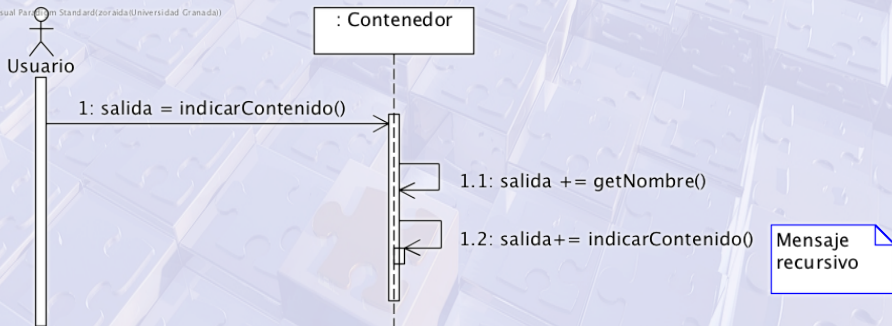
## ● Creación de instancias



# Diagramas de secuencia

## ● Recursividad

Visual Programming Standard (zoraida@Universidad Granada)



# Diagramas de comunicación

- Muestran de forma visual muy clara las vías de comunicación que deben darse entre los participantes para que pueda llevarse a cabo el envío de mensajes entre ellos
- Las vías de comunicación (enlaces) son el elemento principal y el orden temporal de los mensajes un elemento secundario

# Diagramas de comunicación

- Las vías de comunicación se representan mediante líneas que unen a los participantes
- Tipos de enlaces:
  - ▶ **Global (G):** Uno de los participantes pertenece a un ámbito superior. Ej: un atributo de clase
  - ▶ **Asociación (A):** Entre los participantes existe una asociación
  - ▶ **Parámetro (P):** Uno de los objetos es pasado como parámetro a un método del otro participante
  - ▶ **Local (L):** Uno de los participantes es un objeto local a un método del otro participante
  - ▶ **Self (S):** Un objeto también puede enviarse mensajes a sí mismo



# DC para los ejemplos siguientes

## Controlador

+llevarDronA(punto : Lugar)  
 +alturaDron(idDron : int) : float  
 +incluirNuevoDron(dron : Dron, lugar : Lugar)  
 +aterrizarDronesBajoAltura(alt : float)  
 -dronMasCercano(punto : Lugar) : Dron  
 -getDron(idDron : int) : Dron

1..\*

aparatos

## Dron

-id : int  
 -altura : float  
 +getId() : int  
 +getAltura() : float  
 +getPosicion() : Lugar  
 +navegar(destino : Lugar)  
 +aterrizar()  
 +operation2()



posicion

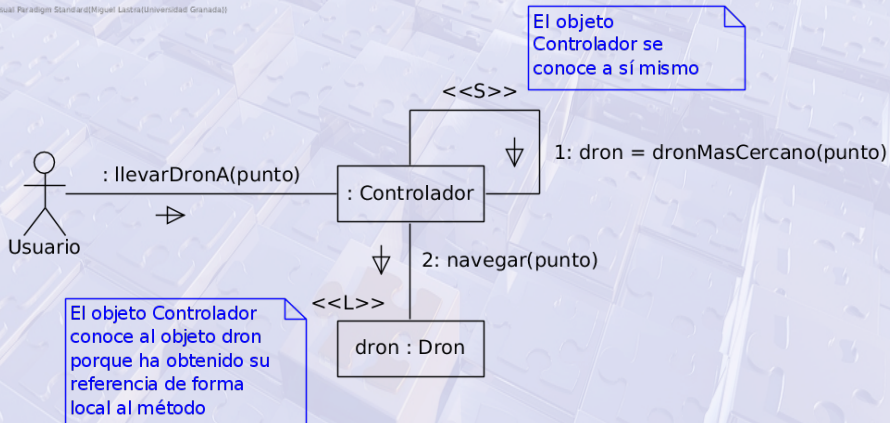


## Lugar

-latitud : int  
 -longitud : int  
 +distancia(punto : Lugar) : float

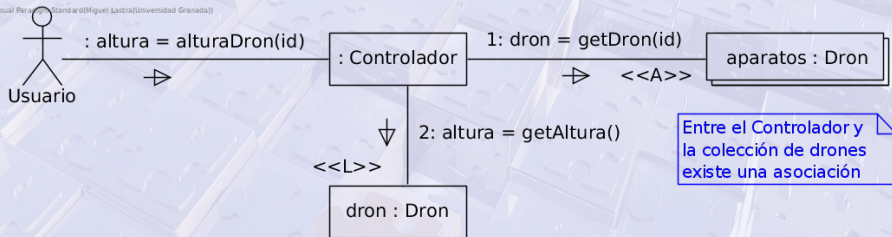
# Ejemplo 1

Visual Paradigm Standard (Miguel Lastra (Universidad Granada))



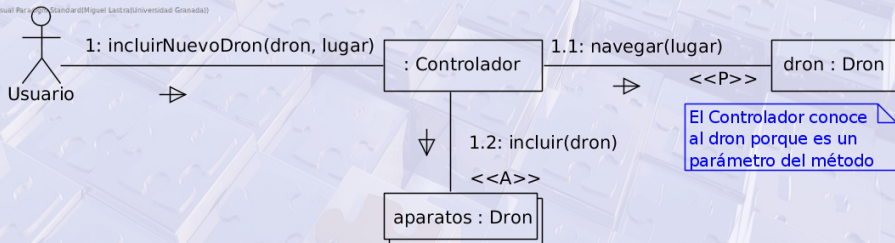
# Ejemplo 2

Visual Paradigm Standard (Miguel Lastra (Universidad Granada))



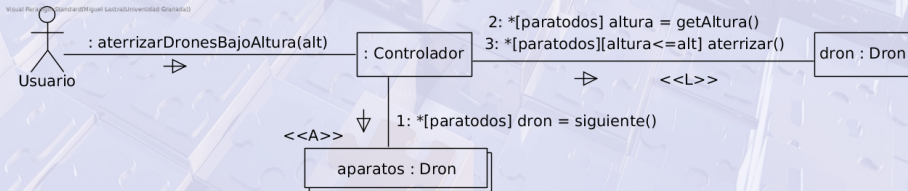
# Ejemplo 3

Visual Paradigm Standard (Miguel Lastra (Universidad Granada))



# Ejemplo 4

## • Condicionales y bucles



# Diagramas de interacción

→ **Diseño** ←

- Recordar que el objetivo de los diagramas UML son:
  - ▶ Especificar las características de un sistema antes de su construcción
  - ▶ Visualizar gráficamente un sistema software de forma que sea entendible
  - ▶ Documentar un sistema para facilitar su mantenimiento, revisión y modificación
- En definitiva, facilitar la tarea del equipo de desarrollo
- Si la especificación de un método (sobre todo los de comunicación) es una maraña de flechas donde es más fácil perderse que aclararse:
  - 1 Tal vez ese tipo de diagrama no sea el más adecuado para esa especificación
  - 2 Tal vez haya que subdividir un diagrama grande en varios pequeños
  - 3 Tal vez el método deba subdividirse en diversas tareas más pequeñas y más fáciles de especificar de una manera clara y fácilmente entendible (supondrá un desarrollo y mantenimiento más fácil)

# UML: Diagramas de Interacción

Dpto. Lenguajes y Sistemas Informáticos  
Universidad de Granada

Programación y Diseño Orientado a Objetos

(Curso 2021-2022)