Programación 4

Desarrollo Orientado a Objetos basado en UML

Proceso de Desarrollo

- ¿Qué es?
  - Un proceso de desarrollo de software describe un enfoque para construir, instalar y mantener sistemas de software
- ¿Por qué necesitamos uno?
  - Es necesario conocer de antemano qué actividades debemos realizar

Algunas Actividades

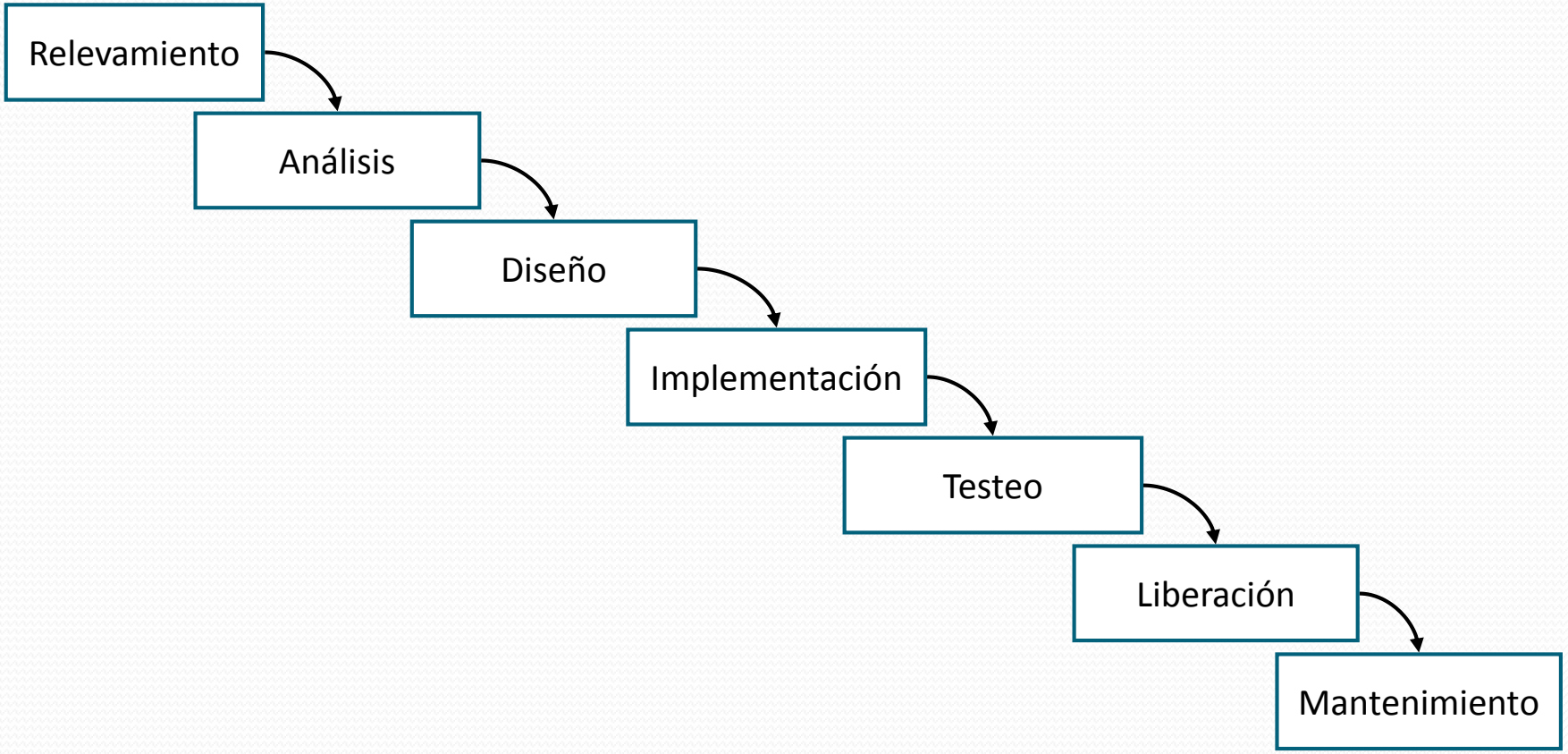
1. Entrar en contexto con la realidad del problema
2. Obtener una descripción de lo que se espera del producto
3. Comprender qué se debe hacer
4. Determinar cómo se debe hacer
5. Hacerlo
6. Probar que esté bien hecho
7. Entregar el producto
8. Hacerle retoques varios
9. Mantenerlo

Pero Hay Más…

- Realizar estimaciones de tiempo, de costos, de recursos
- Planificar
- Asegurarse que las cosas se hagan:
  - En el tiempo previsto
  - De la forma establecida
- Administrar las diferentes versiones de lo que se va produciendo
- Montar y mantener los ambientes de desarrollo y prueba

Un Modelo de Proceso

- Cascada:

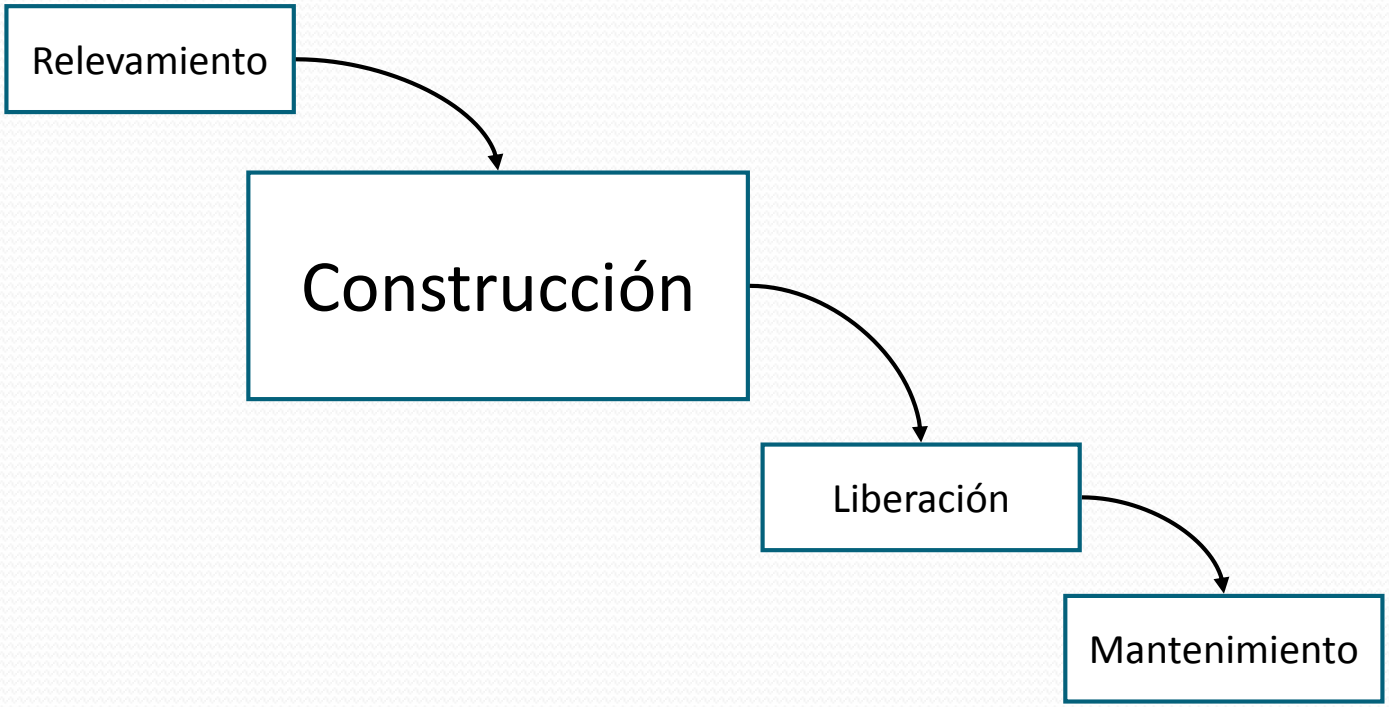


Problemas…

- Cada actividad se realiza en secuencia y luego de finalizar la anterior
- Para problemas grandes y complejos no resulta una estrategia adecuada:
  - Errores en etapas tempranas se descubren tardíamente
  - No hay visibilidad hasta muy avanzado el proyecto

Otro Modelo

- Iterativo e Incremental (I&I):



Características

- Se divide el problema en varios subproblemas
- Las iteraciones se producen en “Construcción”
- Se itera sobre una “mini cascada” donde se resuelve cada subproblema:

```
for each (sp:Subproblema) {
  analisis(sp);
  diseño(sp);
  implementacion(sp);
  testeo(sp);
}
```

- En la iteración *i* se resuelve sp_i llevándose resueltos los subproblemas: sp_1, sp_2, … , sp_{i-1}

Nuestro Proceso

- Para poder realizar un proceso I&I es necesario conocer un proceso en cascada
- Nos concentraremos en algunas actividades dentro de la “cascada” de Construcción:
  - Análisis
  - Diseño
  - Implementación
- Los pasos concretos a realizar en estas actividades depende del paradigma de desarrollo a seguir

Nuestro Proceso (2)

- Los requerimientos vendrán dados por Casos de Uso y descripciones generales del sistema
- Un Caso de Uso narra la historia completa (junto a todas sus variantes) de un conjunto de actores mientras usan el sistema

Caso de Estudio

- Biblioteca de música
  - El usuario necesita organizar las canciones en su reproductor de música (artista, canción, género)
  - Similar al modelo que se puede encontrar en cualquier reproductor de música online u offline (Winamp, Grooveshark, etc.)

Caso de Uso

Nombre	Agregar Canción	Actor	Usuario
Sinopsis	El caso de uso comienza cuando el usuario necesita agregar una nueva canción al sistema. Para ello especifica su título y el año en que se compuso. Luego El sistema muestra una lista con los Géneros y otra con los Artistas y el usuario selecciona cuáles agrega a la canción. Finalmente si la información es correcta, se da de alta la canción en el sistema		

Orientación a Objetos

- Enfoque diferente al tradicional
- Puede ser entendida como:
  - Una forma de pensar basada en abstracciones de conceptos existentes en el mundo real
  - Organizar el software como una colaboración de objetos que interactúan entre sí por medio de mensajes

Enfoque Tradicional

- Una aplicación implementada con un enfoque tradicional presenta la siguiente estructura general:

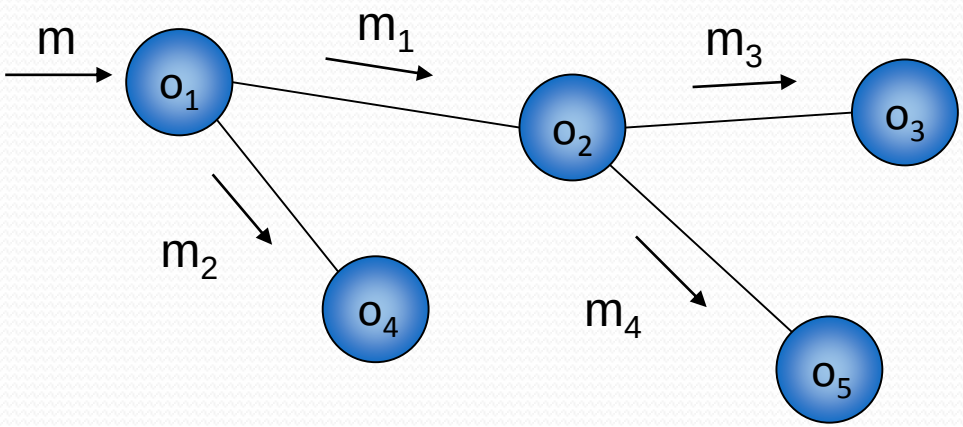
```
type T = …

f1(T t) {…}
…
fn() {…}

main() {
      //invocaciones a fi
}
```

Enfoque Orientado a Objetos

- Una aplicación orientada a objetos es el resultado de la codificación en un lenguaje de programación orientado a objetos del siguiente esquema:



Desarrollo OO

- Los pasos generales de desarrollo se mantienen en el enfoque orientado a objetos
- Pero las actividades que constituyen algunos de ellos son particulares:
  - Análisis ⇨ Análisis Orientado a Objetos
  - Diseño ⇨ Diseño Orientado a Objetos
  - Implementación ⇨ Implemenentación Orientada a Objetos

Desarrollo OO (2)

- Ciertas actividades son demasiado complejas para realizarlas mentalmente en el desarrollo de una aplicación de mediano porte en adelante

Desarrollo OO (3)

- Es necesaria una herramienta (conceptual) que permita a la vez:
  - Servir de ayuda para el desarrollo de la tarea (uno mismo)
  - Visualizar lo hecho hasta el momento (uno mismo)
  - Comunicar el avance obtenido (el cliente y el equipo de desarrollo)
  - Documentar el desarrollo de la aplicación (el equipo de desarrollo)

Desarrollo OO (4)

- UML es el estándar para modelado de software
- Es un lenguaje que puede ser aplicado cualquiera sea el método particular de desarrollar software orientado a objetos
- Utilizaremos algunos de sus diagramas para asistir nuestro desarrollo

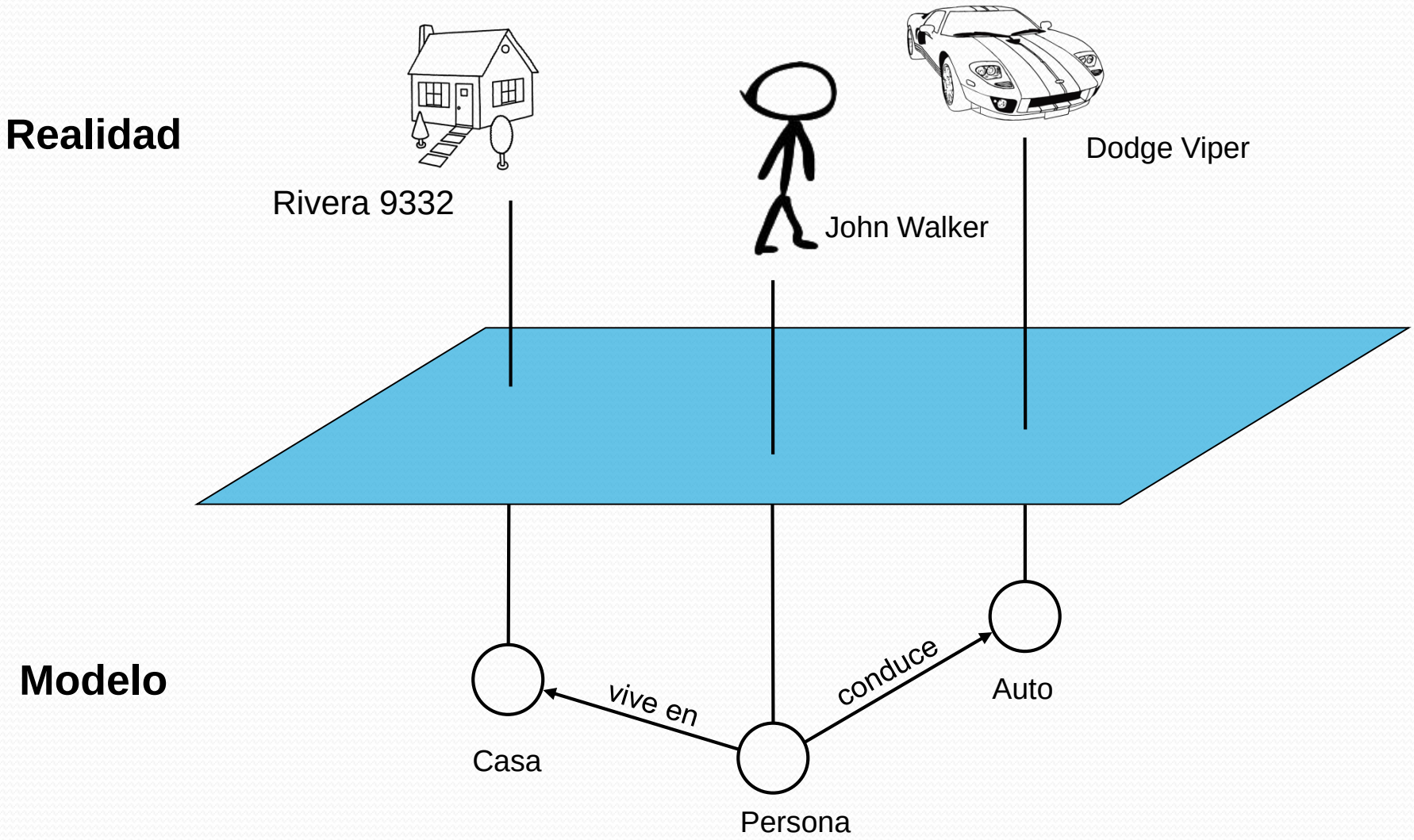
Análisis Orientado a Objetos

- Considerar el dominio de la aplicación y su solución lógica en términos de conceptos (cosas, entidades)
- Concepto clave: *abstracción*
- Objetivo: encontrar y describir los conceptos en el dominio de la aplicación:
  - Esto permite comprender mejor la realidad y el problema

Análisis Orientado a Objetos (2)

- Estos conceptos pueden entenderse como una primera aproximación a la solución al problema
- En un sistema de software orientado a objetos (bien modelado) existe un *isomorfismo* entre estos conceptos y los elementos que participan en el problema en la vida real

Análisis Orientado a Objetos (3)



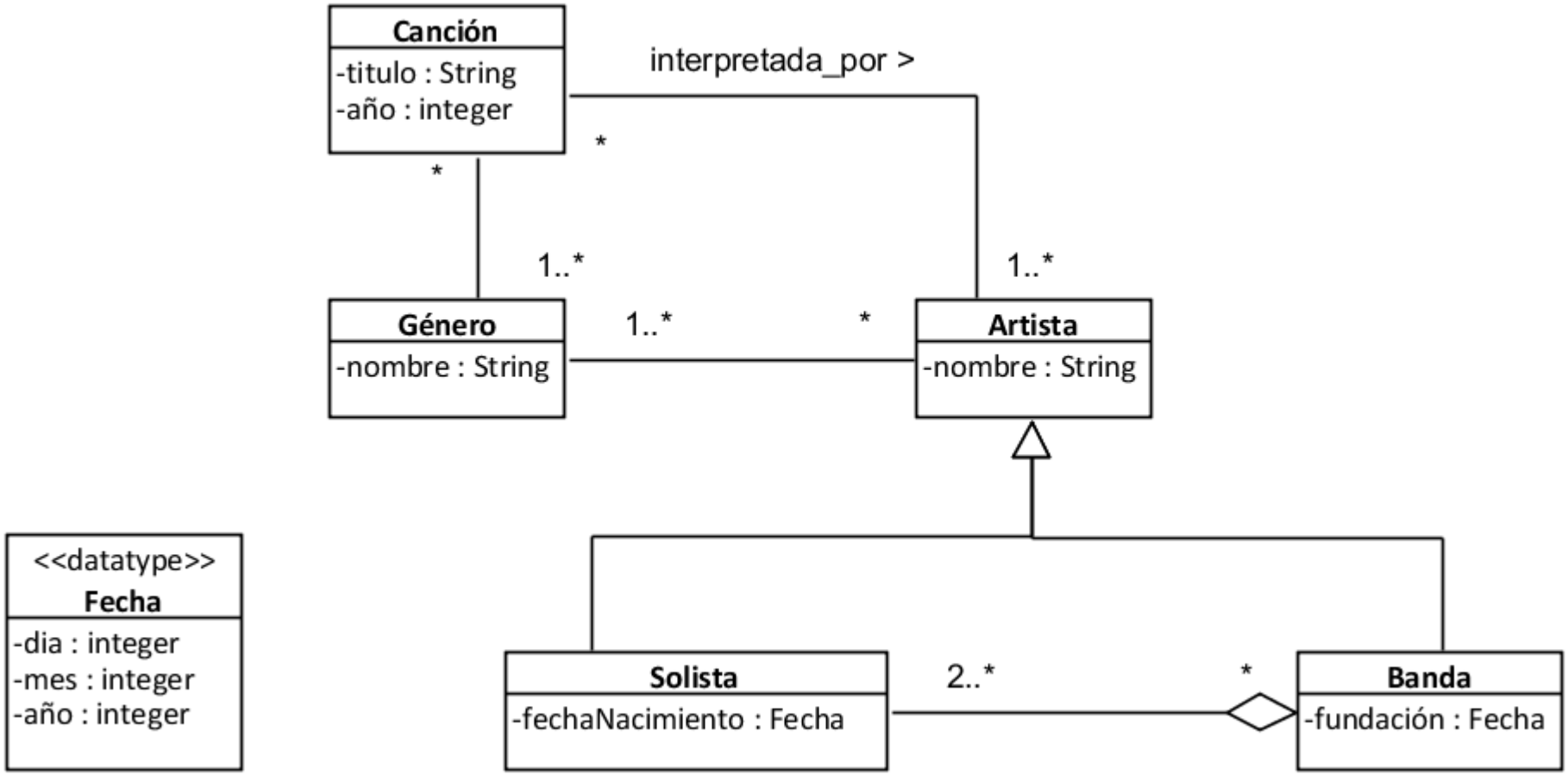
Análisis OO :: Actividades

- Modelado de Dominio
  - Modelar el dominio para comprender mejor el contexto del problema
  - Herramienta: Diagrama de Clases
- Especificación del Comportamiento
  - contar con una descripción más precisa de qué es lo que se espera del sistema
  - Herramientas: Diagramas de Secuencia del Sistema y Contratos

Análisis OO :: Dominio

- Un Modelo de Dominio contiene los conceptos y sus relaciones que sean significativos en el dominio del problema
- La información es provista principalmente por los Casos de Uso
- Se incluyen además las restricciones a las cuales está sujeto el dominio

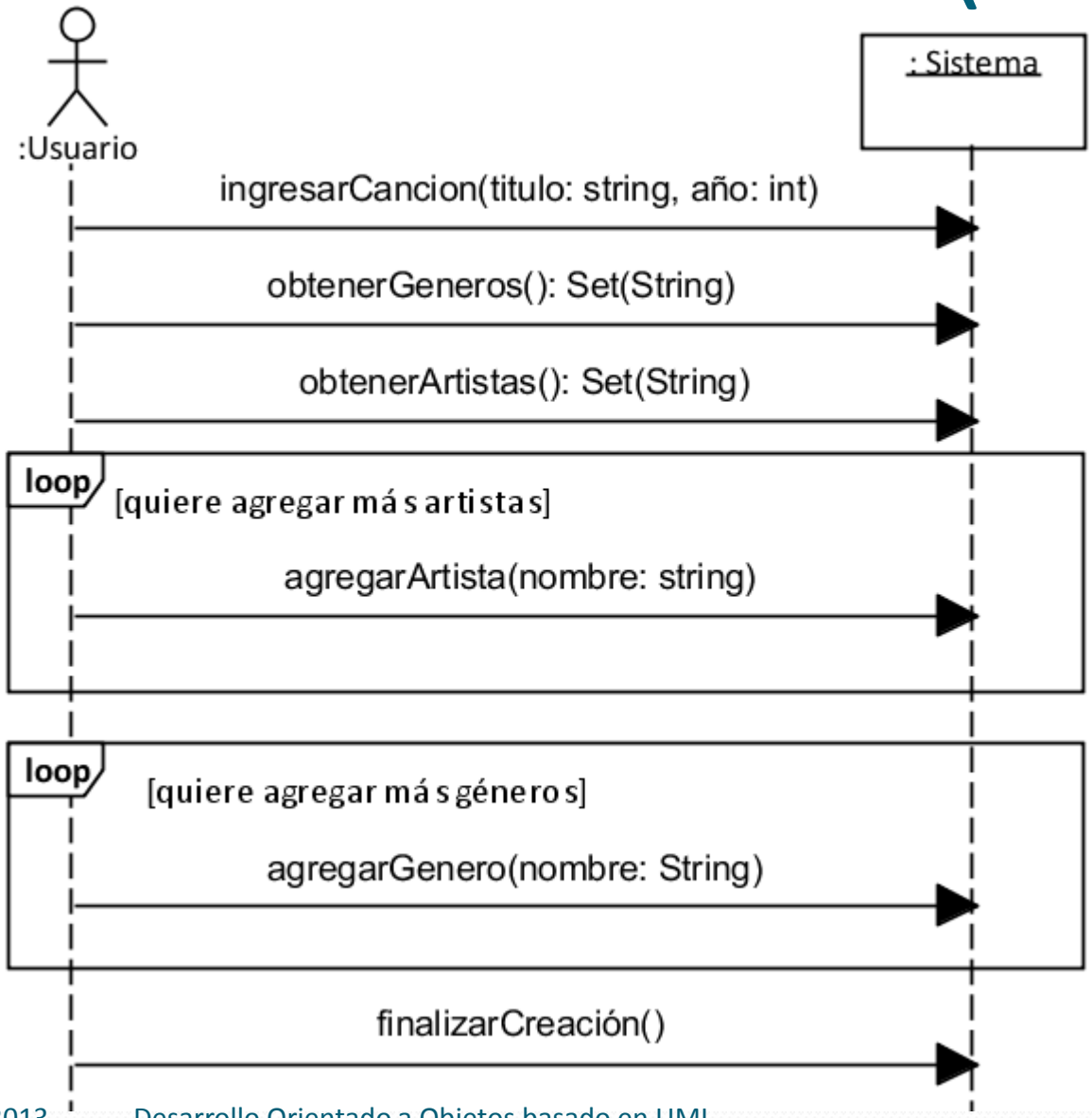
Modelo de Dominio



Análisis OO :: Comportamiento

- Los Diagramas de Secuencia del Sistema ilustran la forma en que los actores realizan “invocaciones” sobre el sistema
- Los diferentes escenarios de uso son los definidos en los Casos de Uso
- El efecto de cada mensaje es especificado en forma precisa por medio de un Contrato

Diagrama de secuencia (DSS)



Caso de Estudio

Contrato

Operación	**`ingresarCancion(titulo:string, año: int)`**
Descripción	Comienza el ingreso de una nueva canción al sistema especificando la información básica
Pre-condiciones	1. No existe una canción de nombre igual a **titulo** 2. El parámetro **año** no es mayor al año actual
Post-condiciones	El sistema recuerda el titulo y el año especificados, para usarse posteriormente.

Diseño Orientado a Objetos

- Objetivo: definir objetos lógicos (de software) y la forma de comunicación entre ellos para una posterior programación
- En base a los “conceptos candidatos” encontrados durante el análisis y por medio de ciertos principios y técnicas, se debe decidir:
  - Cuáles de éstos serán los objetos que participarán en la solución
  - Cómo se comunican entre ellos para obtener el resultado deseado

Diseño Orientado a Objetos (2)

- Concepto clave: *responsabilidades*
- En esta transición:
  - No todos los conceptos necesariamente participarán de la solución
  - Puede ser necesario “reflotar” conceptos inicialmente dejados de lado
  - Será necesario fabricar “ayudantes” (también objetos) para que los objetos puedan llevar a cabo su tarea

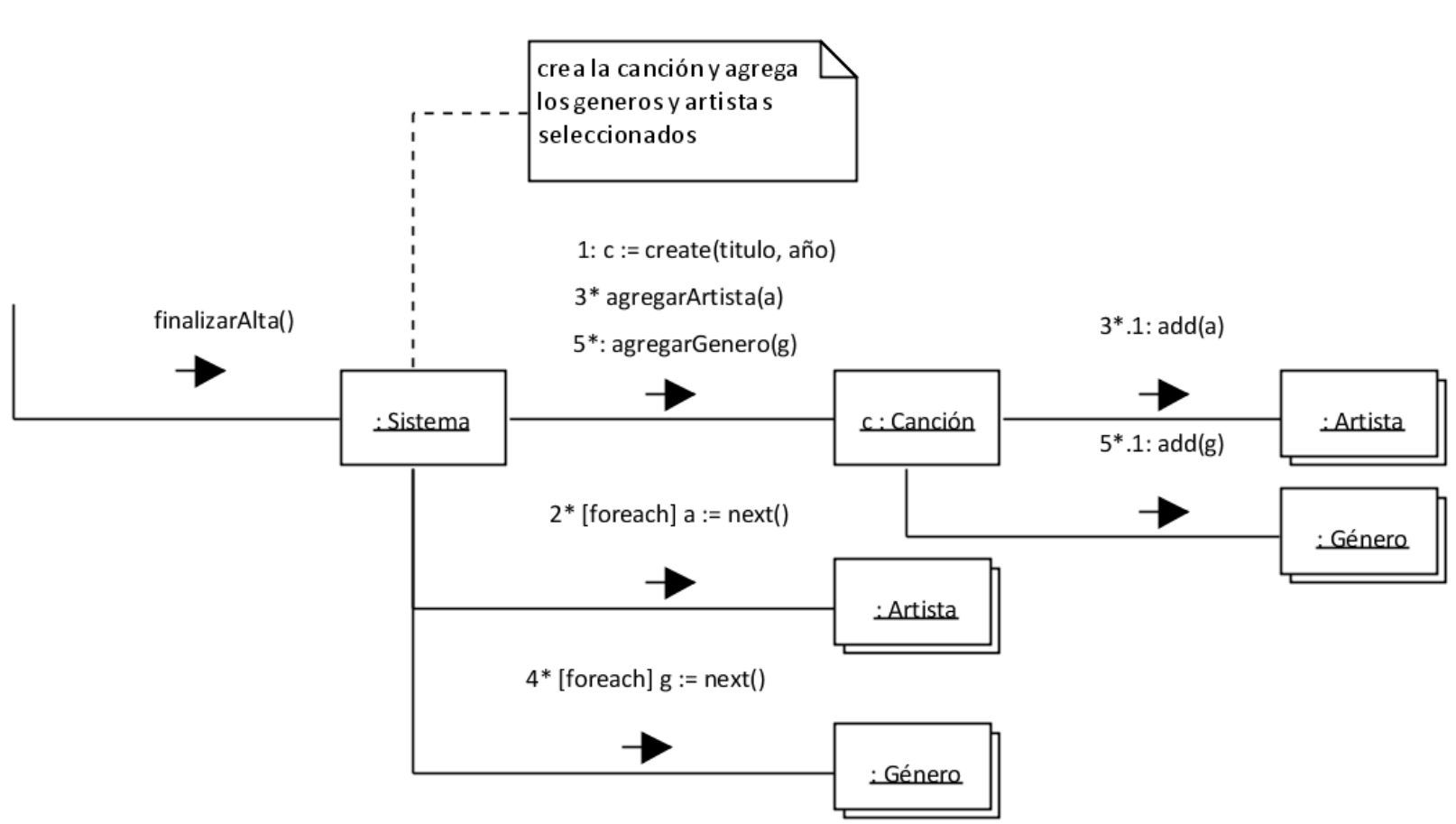
Diseño OO :: Actividades

- Diseño de Interacciones
  - definir cómo se comunican los objetos para resolver las operaciones del sistema
  - Herramienta: Diagrama de Comunicación
- Diseño de Estructura
  - especificar la estructura necesaria para que todas las interacciones puedan ocurrir
  - Herramienta: Diagrama de Clases de Diseño

Diseño OO :: Interacciones

- Se realiza un Diagrama de Comunicación por operación del sistema
- Los objetos protagonistas aparecen “sugeridos” en el Modelo de Dominio
- El resultado esperado es el especificado en el contrato de la operación a diseñar

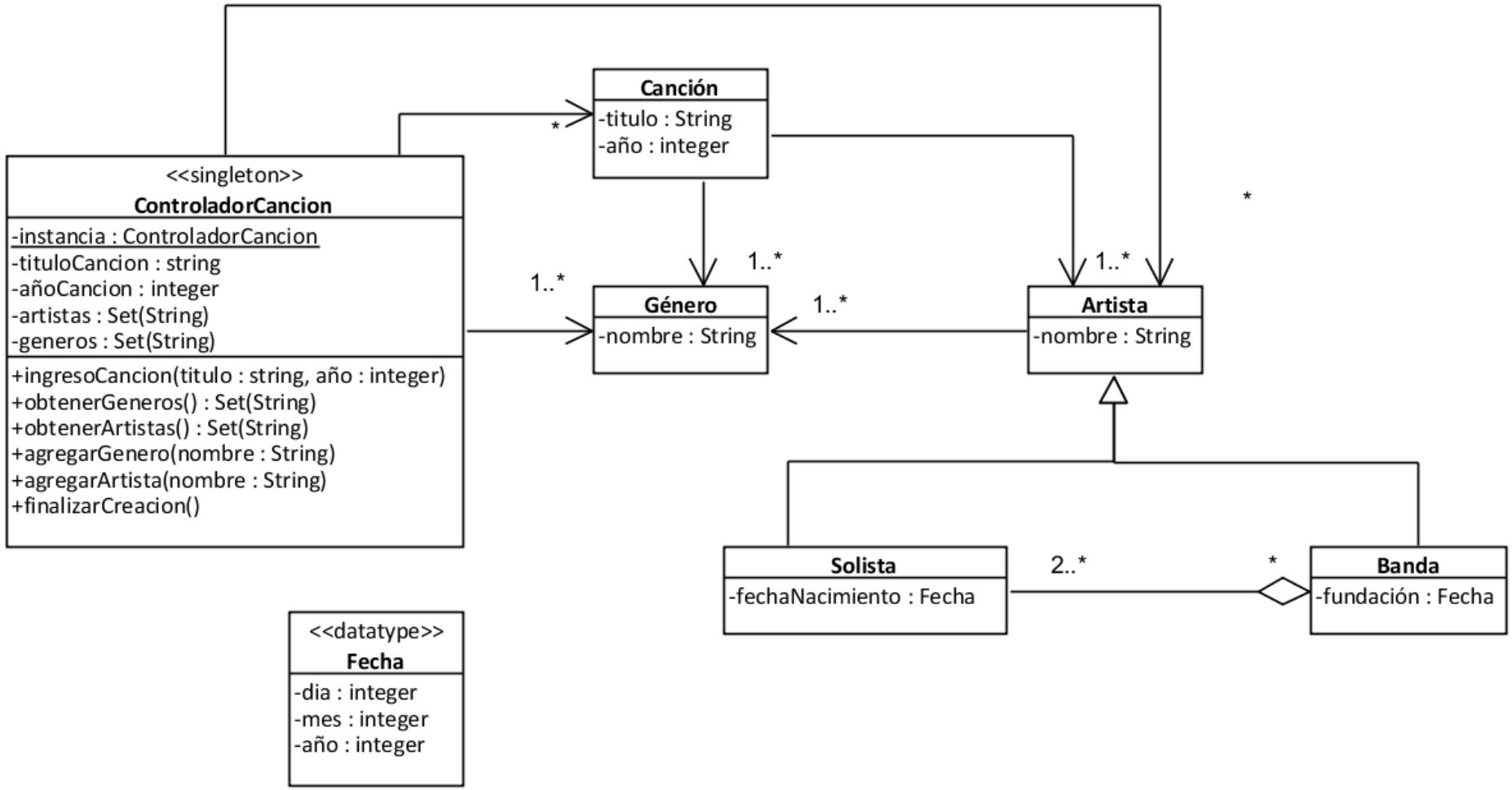
Diagramas de Comunicación



Diseño OO :: Estructura

- Se resume la información provista por los Diagramas de Comunicación
- La estructura está guiada por el Modelo de Dominio
- Generalmente se realiza un único Diagrama de Clases que resume toda la información

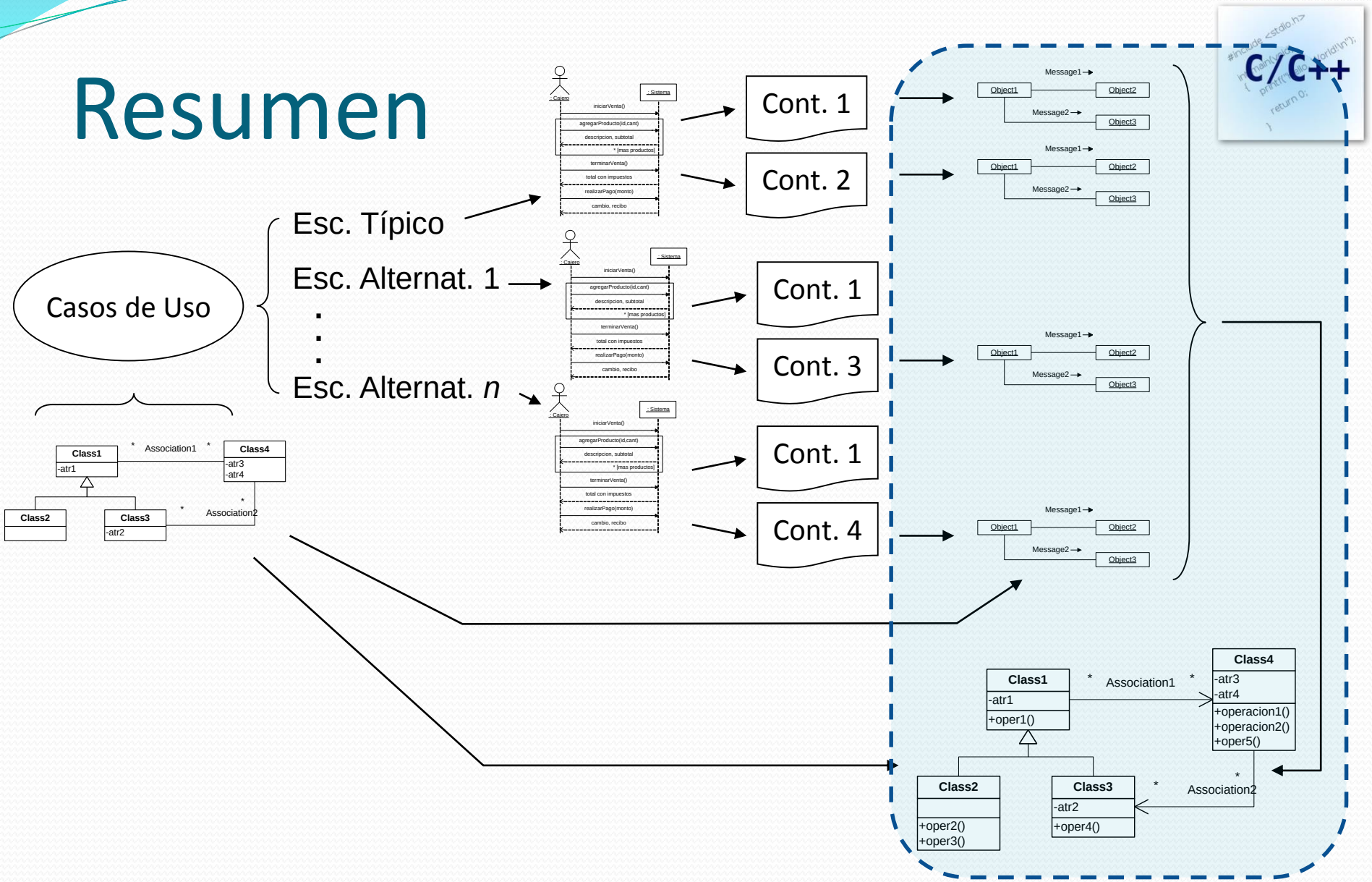
Diagrama de Clases de Diseño



Implementación OO

- Su objetivo es codificar en un lenguaje de programación orientado a objetos las construcciones definidas en el diseño
- La definición de los objetos y el intercambio de mensajes requieren construcciones particulares en el lenguaje a utilizar

Resumen


Casos de Uso
Esc. Típico
Esc. Alternat. 1
Esc. Alternat. n
Cont. 1
Cont. 2
Cont. 1
Cont. 3
Cont. 1
Cont. 4
C/C++
Class1
-atr1
Association1
Class4
-atr3
-atr4
Class2
Class3
-atr2
Association2
+oper1()
+operacion1()
+operacion2()
+oper5()
+oper2()
+oper3()
+oper4()
Message1
Message2
Object1
Object2
Object3