Modelado de Software

Modelado/Modelamiento

- Técnica que ayuda al equipo de desarrolladores a “visualizar” el sistema que se va a construir, a través del uso de modelos y de la abstracción.
  - Ayuda a ver cómo es, o cómo queremos que sea.
  - Permite especificar estructura y/o comportamiento.
  - Proporciona plantillas para la construcción.
  - Documenta en si mismo las decisiones tomadas.

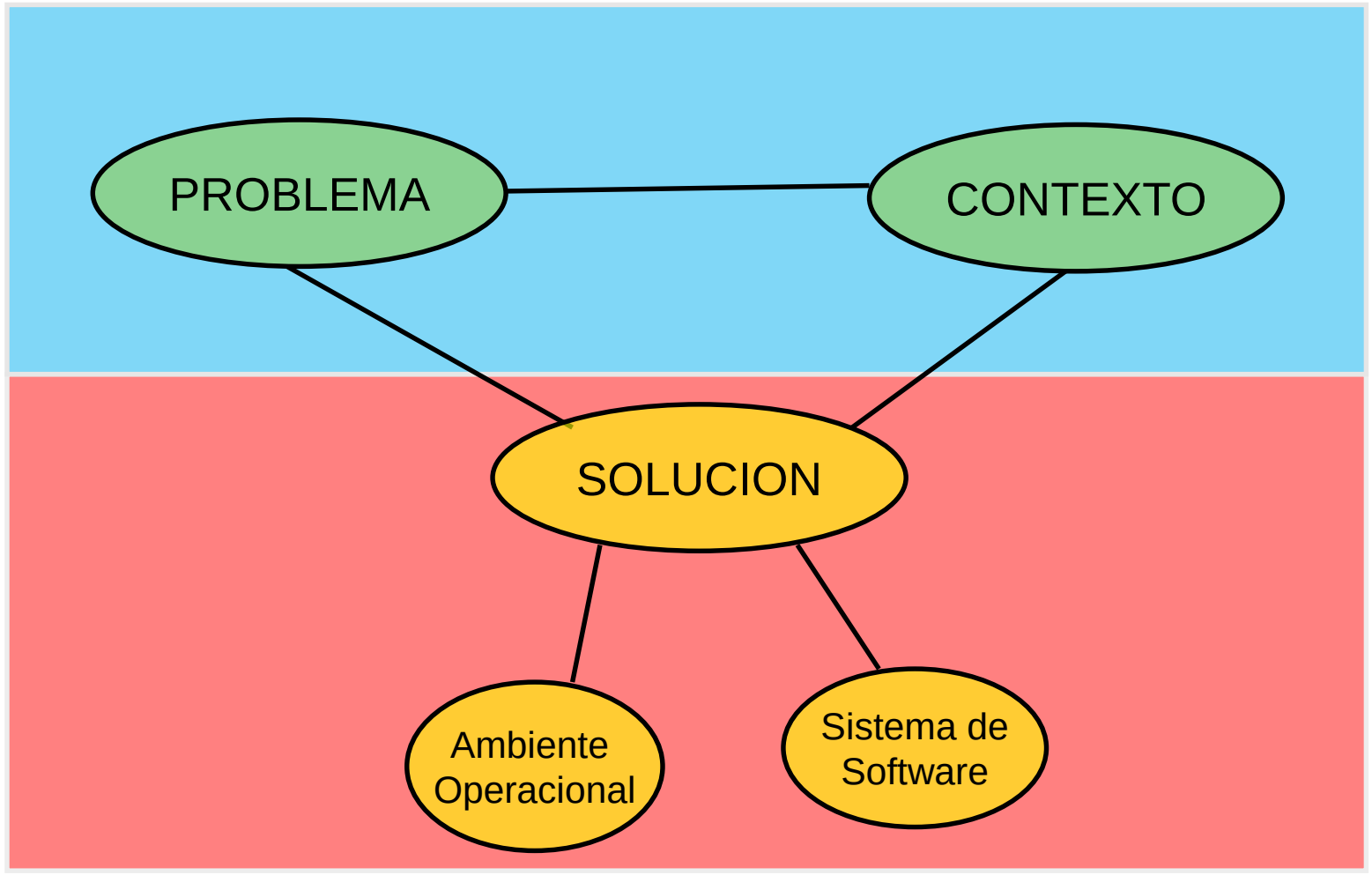
Importancia

- Comprender la realidad
  - Modelo → Simplificación (abstracción) de la realidad.
- Comprender el sistema a construir
  - Modelo → elementos + relaciones
- Reducir la complejidad de comprensión
  - Permite “ver el todo” de manera simple
- Facilitar la comunicación (cliente y equipo)
  - Todos tienen claro qué se va a hacer

Principios de Modelamiento

- Cómo vemos el problema influye directamente en cómo lo solucionamos.
- Para sistemas complejos, se requieren múltiples modelos para cada vista de sistema, tal que éstas interactúen entre si.
- Los modelos pueden representarse con distintos grados de precisión.
- Los modelos deben estar ligados estrechamente con la realidad.

Modelo Lógico



Lenguaje de Modelado

- Es cualquier lenguaje artificial que puede ser usado para expresar información, conocimiento, o sistemas en uan estructura definida por un conjunto consistente de reglas, que son usadas para interpretar los componentes dentro de la estructura.

- Puede ser gráfico o textual.

Lenguaje de Modelado: Ejemplos

- Architecture description language (ADL)
- Business Process Modeling Notation (BPMN)
- Extended Enterprise Modeling Language (EEML)
- Fundamental Modeling Concepts (FMC)
- Jackson Structured Programming (JSP)
- Unified Modeling Language (UML)
- Systems Modeling Language (SysML)

UML

- El Unified Modeling Language (UML) es un lenguaje que permite modelar, construir y documentar elementos de sistema software orientados a objetos.

- Uno de sus objetivos principales es posibilitar el intercambio de modelos entre las distintas herramientas CASE del mercado.
  - Define una notación y semántica común.
  - Permite visualizar las diferentes vistas de un sistema.

Vistas del UML

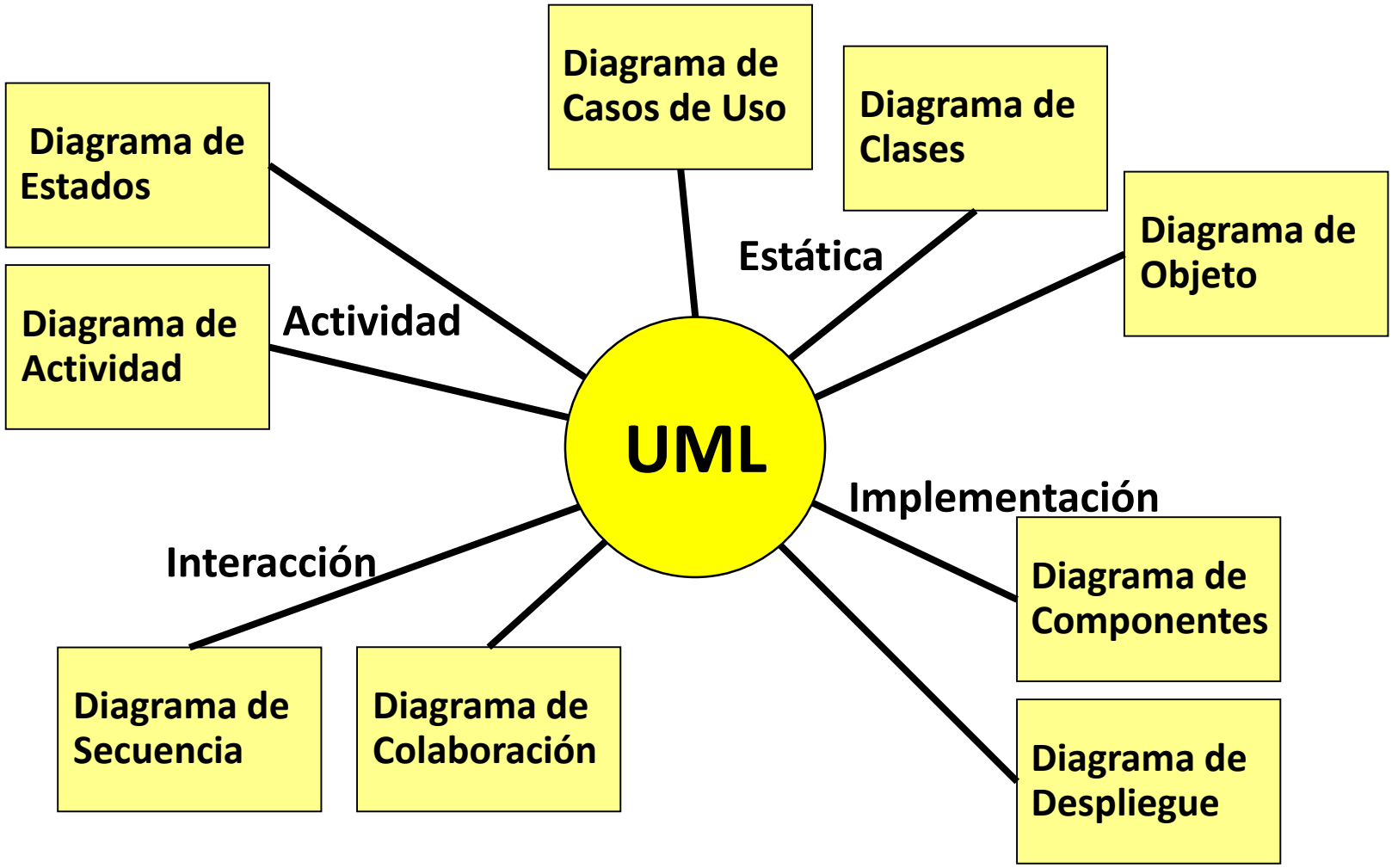


Diagrama de Clases

- Describe la estructura estática del sistema.

- Presenta el conjunto de clases, interfaces y colaboraciones, así como sus relaciones, cubriendo la vista de diseño estática del sistema.

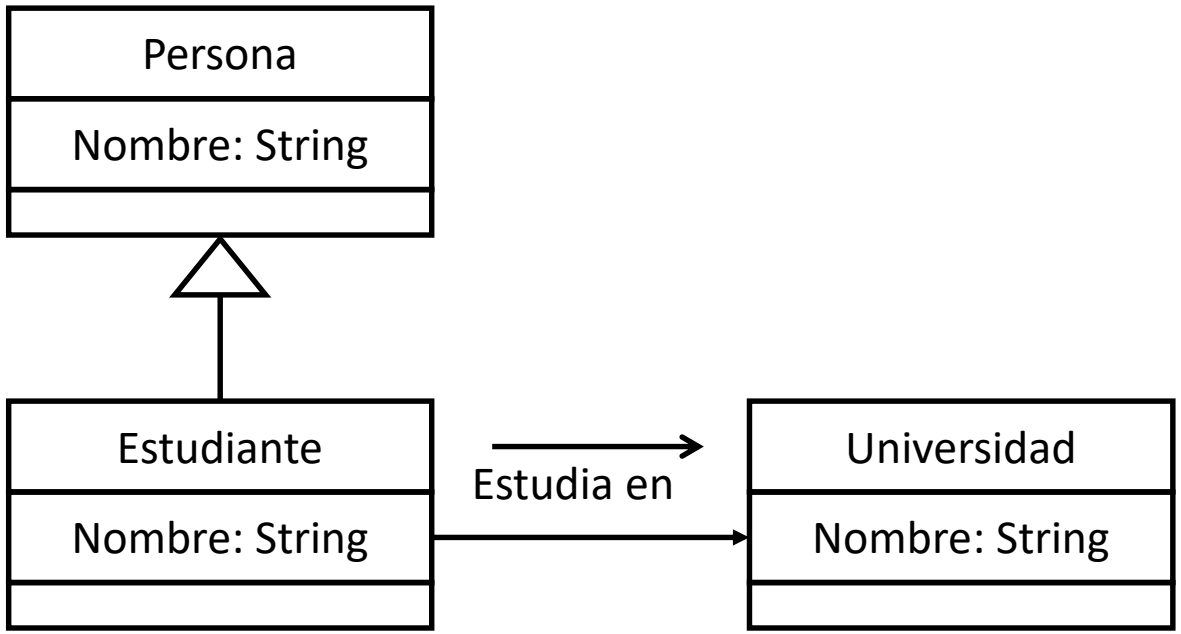


Diagrama de Objetos

- Análogo al diagrama de clases, muestra un conjunto de objetos y sus relaciones, en un instante dado.

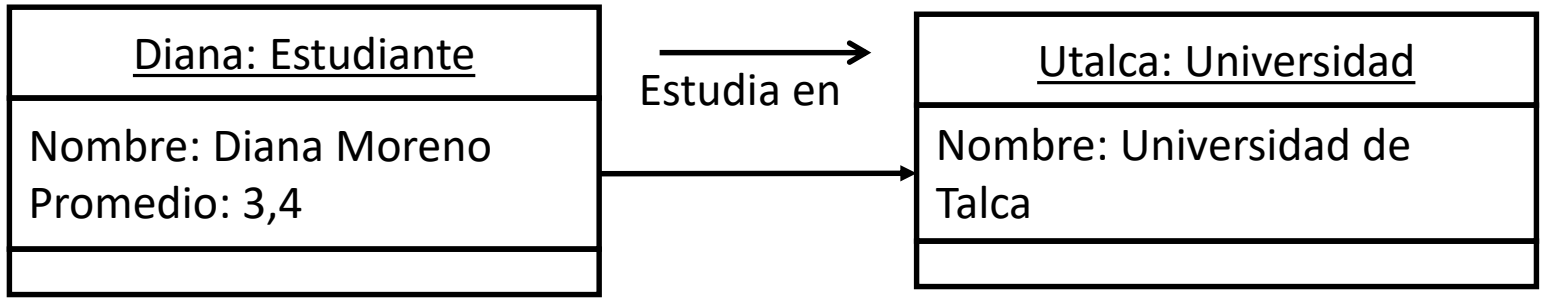


Diagrama de Componentes

- Muestra la organización y dependencias de un conjunto de componentes.

- Cubren la vista de implementación de un sistema, y describen la interacción entre componentes de Software.

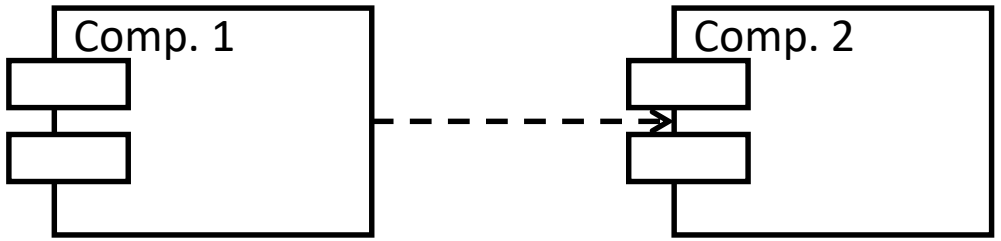


Diagrama de Despliegue

- Describe la disposición del hardware.

- Muestra la configuración del hardware del sistema, los nodos de proceso y los componentes empleados por éstos.

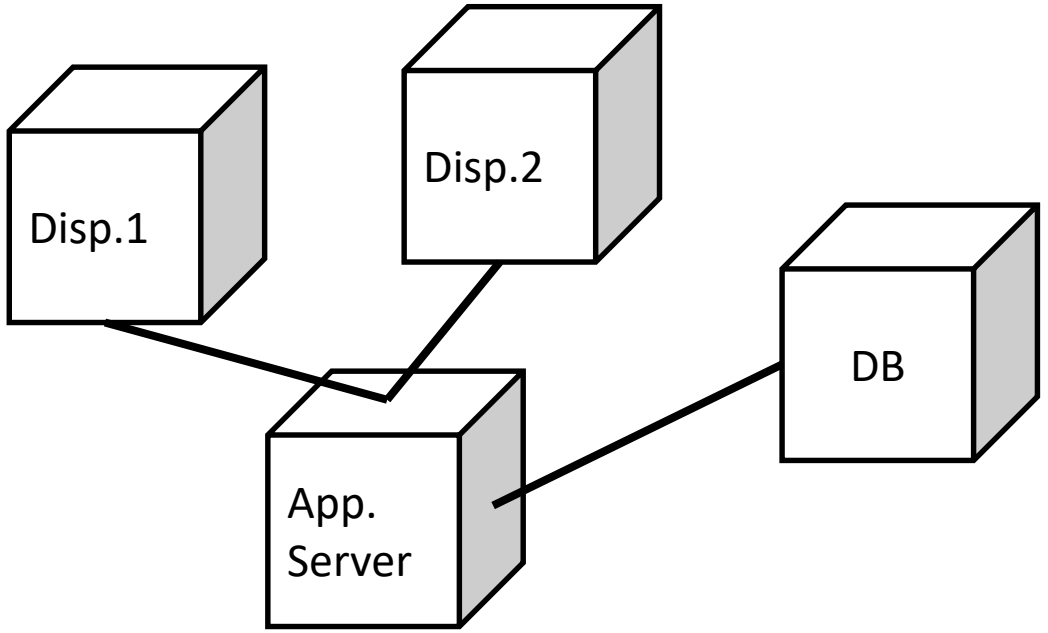


Diagrama de Casos de uso

- Describe las funcionalidades del sistema a partir de las interacciones del usuario.

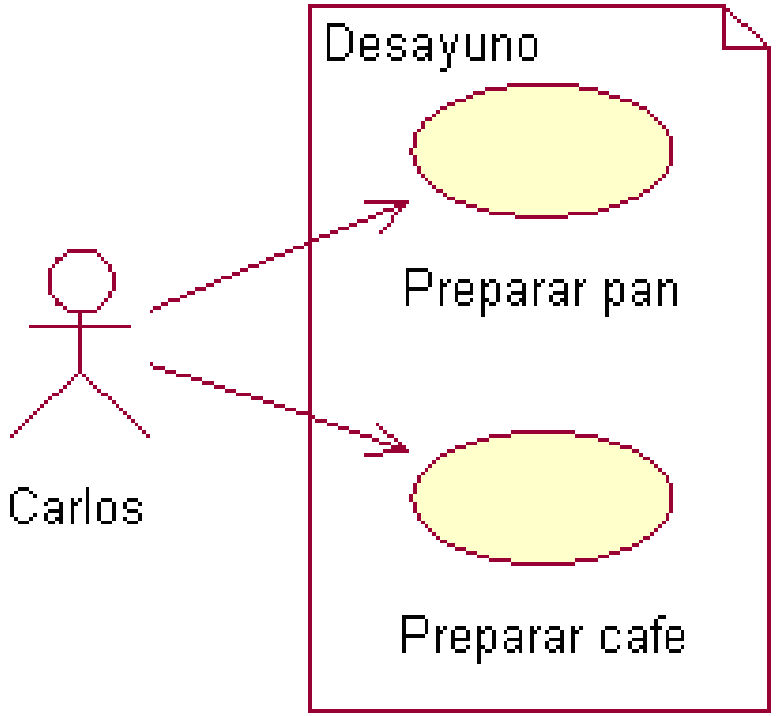


Diagrama de Secuencia

- Diagrama de interacción que muestra un conjunto de objetos, sus relaciones, y los mensajes que se intercambian entre ellos.

- Resalta la ordenación temporal de los mensajes.

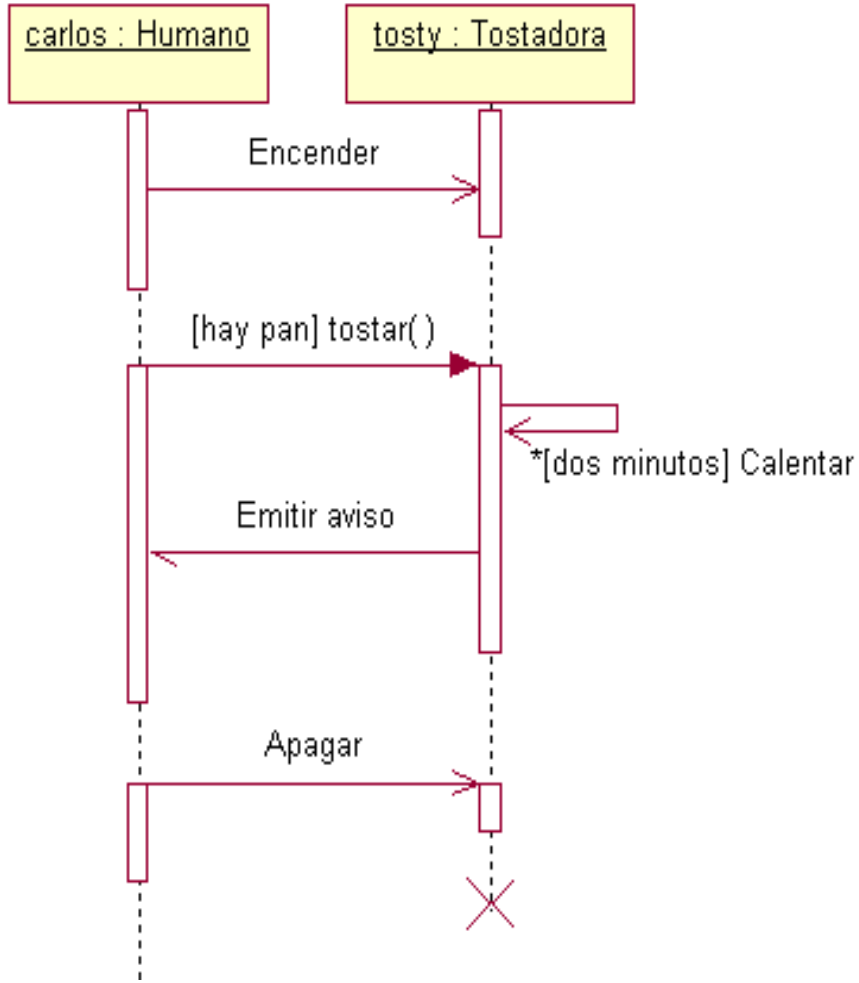


Diagrama de Colaboración

- También es un diagrama de interacción.
- Resalta la organización estructural de los objetos.
- Es equivalente (isomorfo) con respecto al diagrama de Secuencia.

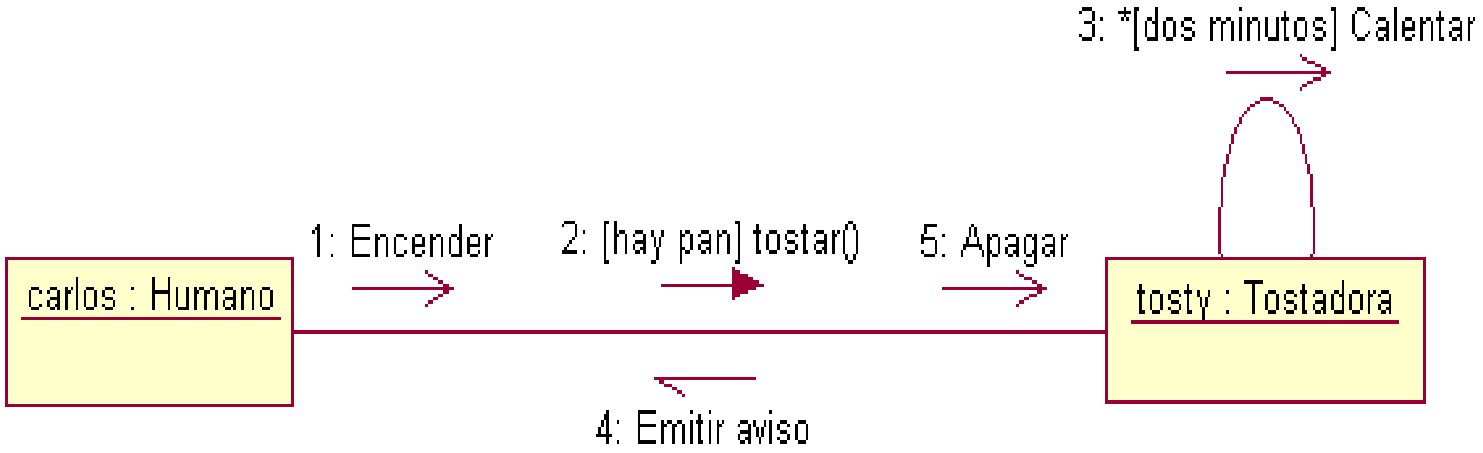


Diagrama de Estados

- Muestra una máquina de estados de un objeto, con sus estados, transiciones, eventos y actividades.

- Modela comportamientos reactivos en base a eventos.

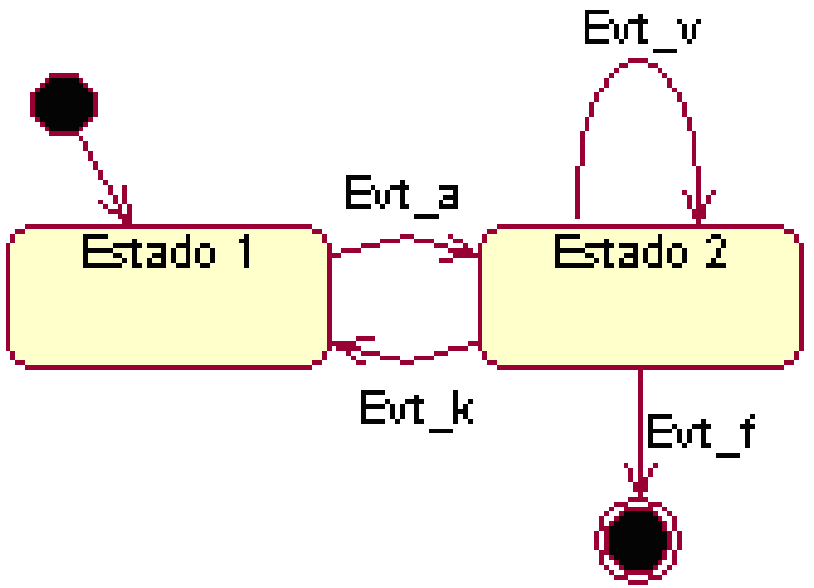


Diagrama de Actividades

- Describe el flujo de trabajo, muestra las actividades, su secuencia y coordinación

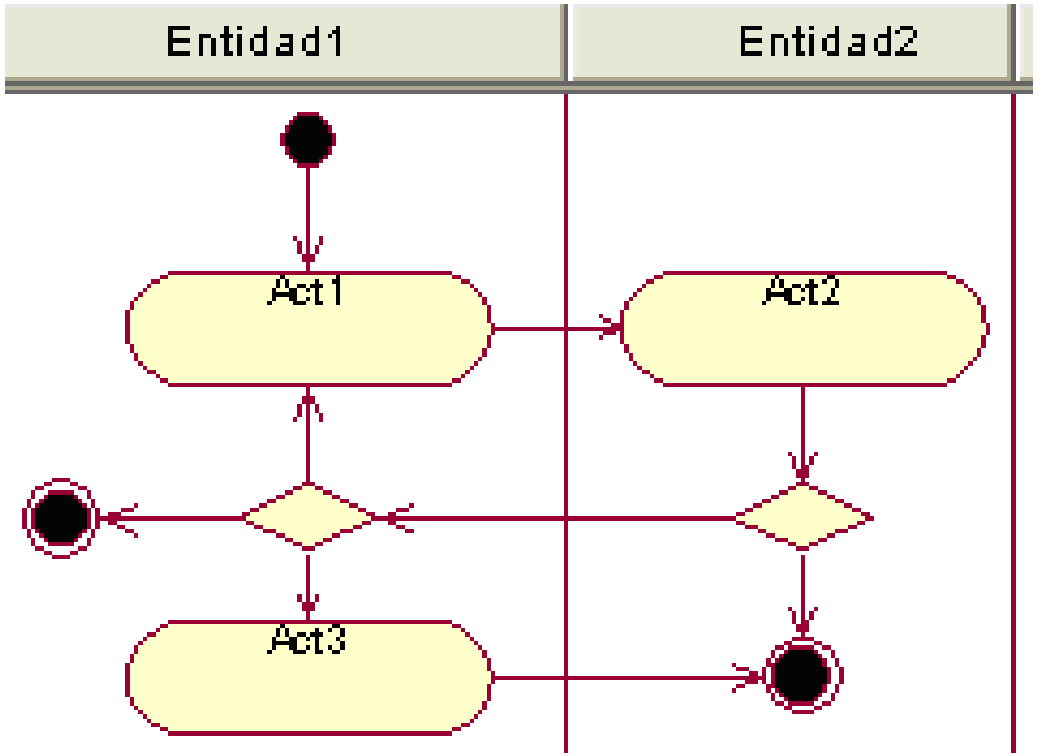
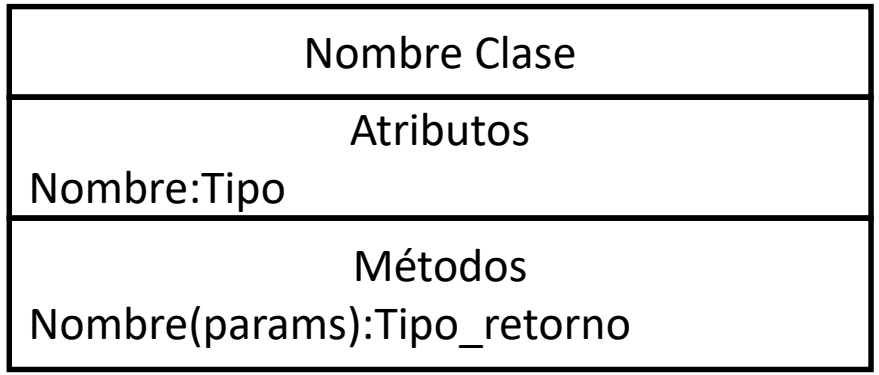


Diagrama de Clases

Diagramas de Clases I

- Describe la estructura estática del sistema, mostrando sus clases y las relaciones entre estas.

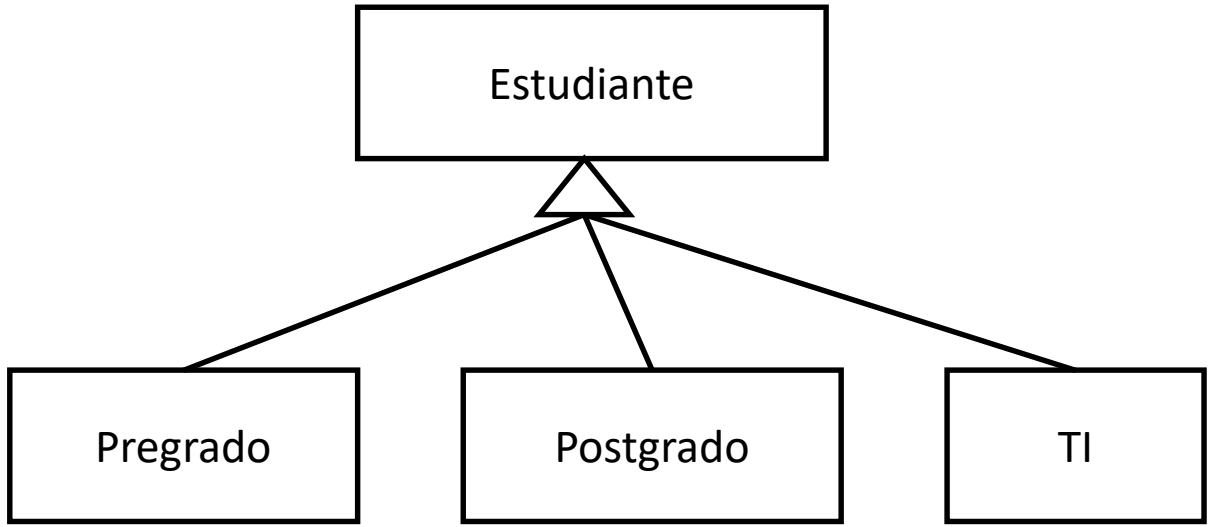
- Una clase es la definición de un conjunto de objetos con características y comportamiento similares.



Diagramas de Clases II

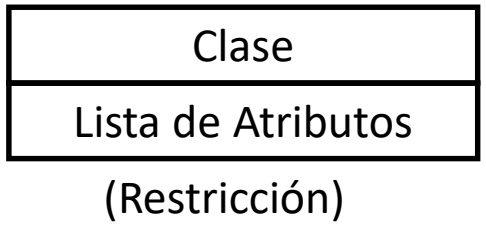
- **Clases Abstractas:** No tienen instancias directas, pero sus clases descendientes si.

- **Clases Concretas:** Son instanciables.



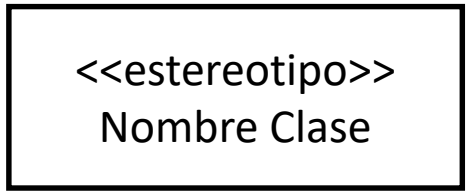
Diagramas de Clases III

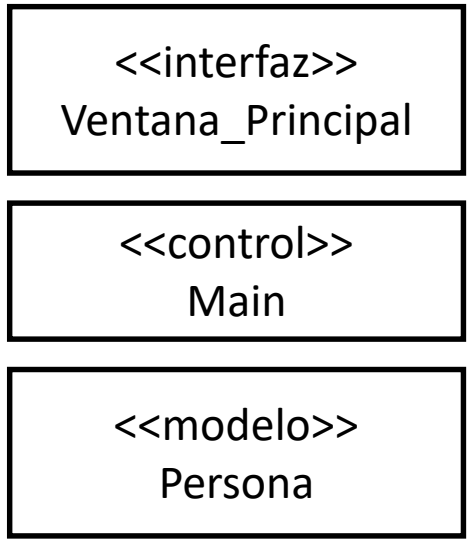
- Restricciones (constraints):
  - Restricciones de Atributos: Permiten adicionar reglas en atributos.
  - Restricciones en relaciones: Reglas a las relaciones.



Diagramas de Clases IV

- Estereotipos: <<estereotipo>>
  - Es un *metatipo*, cuya utilidad es definir a otros tipos.
  - Define el propósito del conjunto de elementos a modelar con el mismo estereotipo.
  - Es opcional.





Diagramas de Clases V

- Relaciones entre clases
  - Conexión semántica entre elementos del modelo.

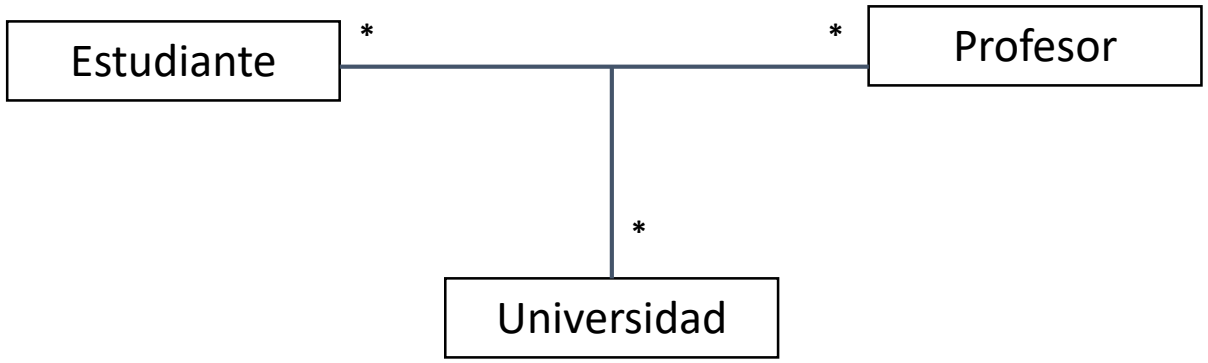
- Tipos de Relaciones entre clases:
  - Asociación.
  - Agregación.
  - Composición.
  - Generalización / Especialización.
  - Dependencia.

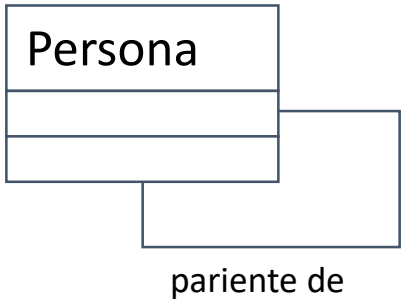
Diagramas de Clases VI

- Asociación:
  - Relación o invocación significativa entre 2+ clases.
  - Según notación UML, la asociación comprende:
    - Descripción, o nombre de la relación.
    - Rol: Responsabilidad de la clase en la relación.
    - Multiplicidad: Indica cuantos objetos pueden participar en la relación.
      - 0 ó más: *
      - 1 o más: 1..*
      - De 2 a 4: 2..4
      - Sólo 7: 7

Diagramas de Clases VII

- *Grado de la Asociación*: Se determina por el número de clases conectadas por la misma asociación. Las asociaciones puedes ser binarias ternarias o de mayor grado.
- Las *asociaciones* pueden ser *reflexivas*, es decir pueden relacionar distintos objetos de una misma clase.





Diagramas de Clases VIII

- Agregación: (“es parte de”, “contiene”)
  - Asociación que especifica relación Parte de entre el agregado (Todo) y el componente (Parte).

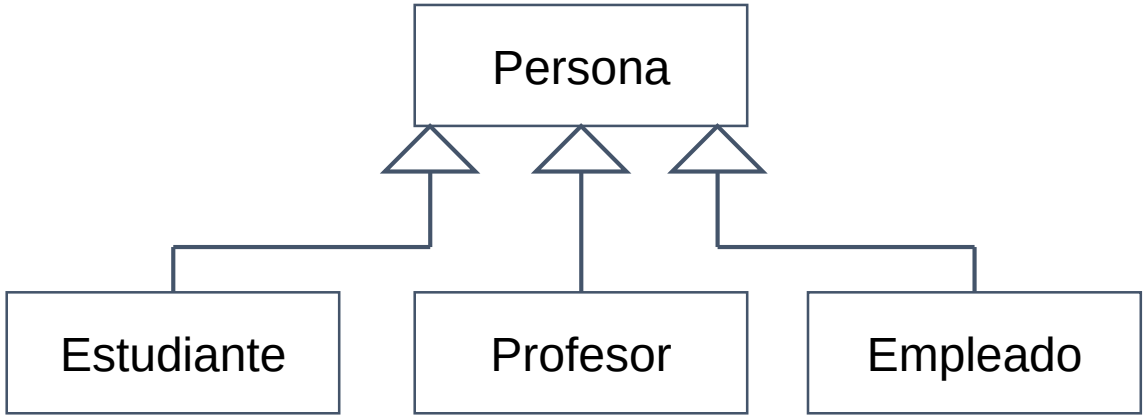
Diagramas de Clases IX

- Composición: (“compuesto por”)

- Relación de agregación especial donde las partes no pueden existir sin que exista el objeto todo.

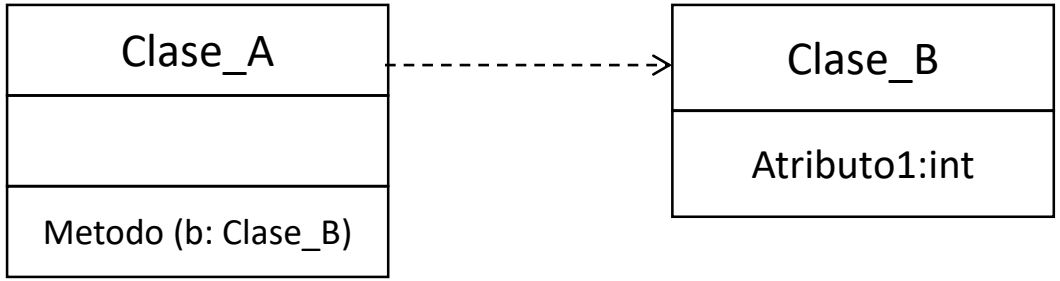
Diagramas de Clases X

- Generalización / Especialización: (“es un”)
  - Generalización: Se crea una superclase, que generaliza propiedades comunes de varias clases.
  - Especialización: Dada una clase, se crean subclases que especializan la clase dada, agregando las diferencias.



Diagramas de Clases XI

- Dependencia
  - Es una conexión entre clases que indica que un cambio en una clase B puede afectar a otra clase A que la usa.



Diagramas de Clases XII

- Clase Asociativa:
  - Asociación entre clases, donde la relación posee atributos propios.
  - Cada enlace es una instancia de clase.

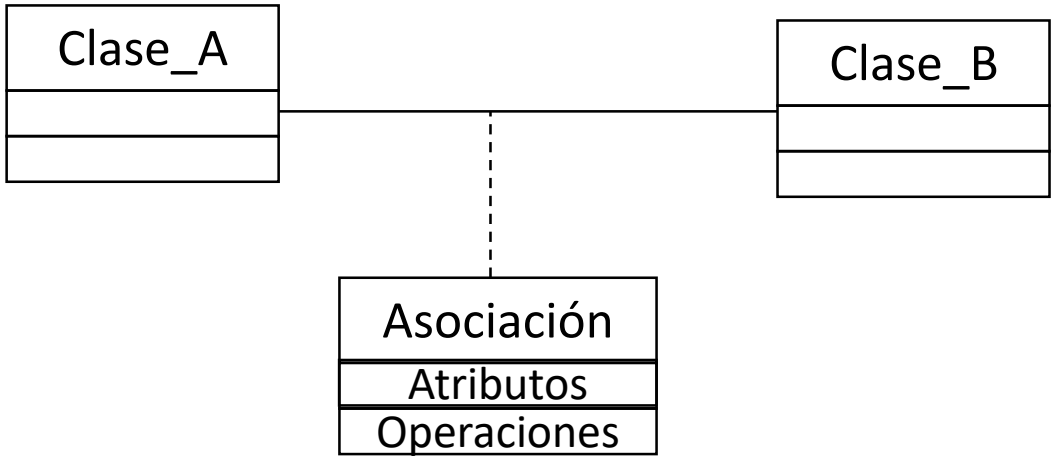
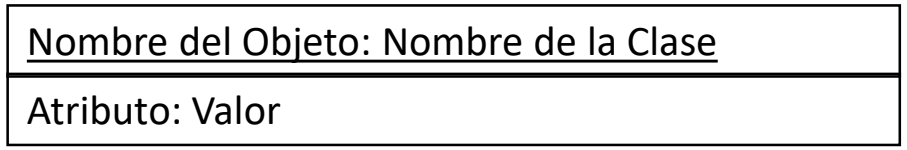


Diagrama de Objetos

Diagrama de Objetos

- Un diagrama de objetos muestra un conjunto de objetos y sus relaciones en un momento determinado.



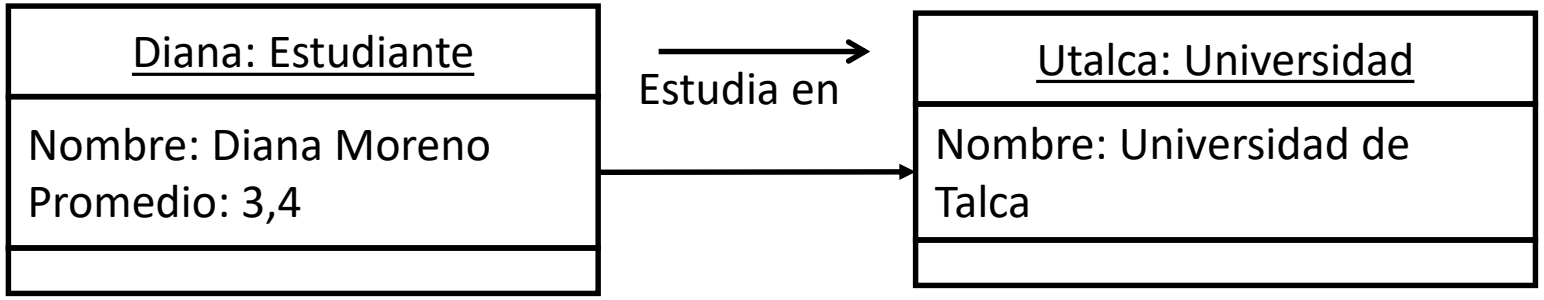


Diagrama de Casos de Uso

Diagrama de Casos de uso I

- Describen lo que hace un sistema, enfatizando el qué en vez del cómo.

- Describen las funcionalidades del sistema a partir de las interacciones del usuario; esto es, describen un uso del sistema y cómo este interactúa con el usuario.

- Se emplean para visualizar el comportamiento del sistema.

Diagrama de Casos de uso II

- **Actores**
  - Entidad externa que interactúa con el sistema.
  - Entidades distintas a los usuarios de sistema.
  - En algunos casos, representan cierta función que un usuario va a realizar en el sistema.

    - Persona
    - Componente de SW
    - Organización

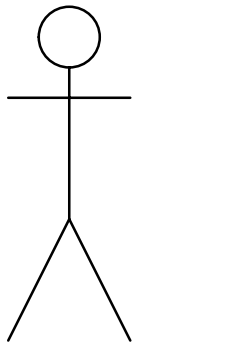
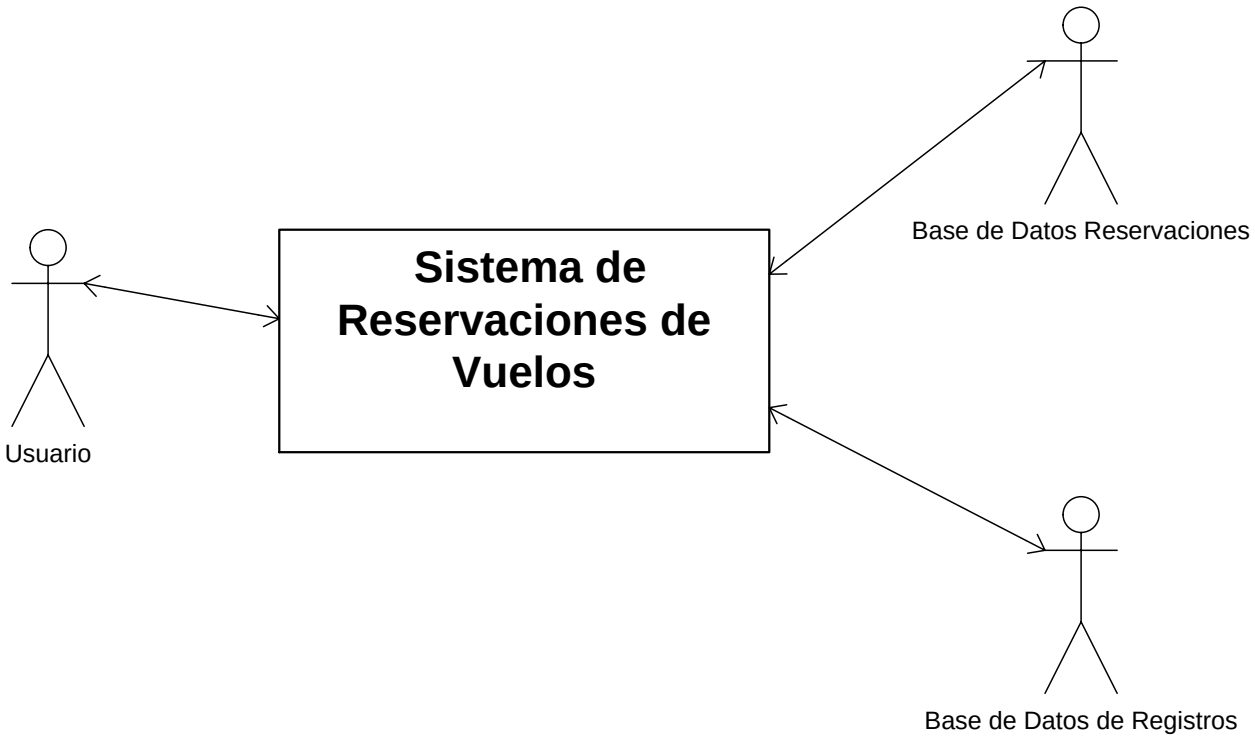


Diagrama de Casos de uso III

- Relaciones entre actores:
  - **Generalización:** Cuando diferentes actores realizan roles similares, pueden heredar de un actor común.



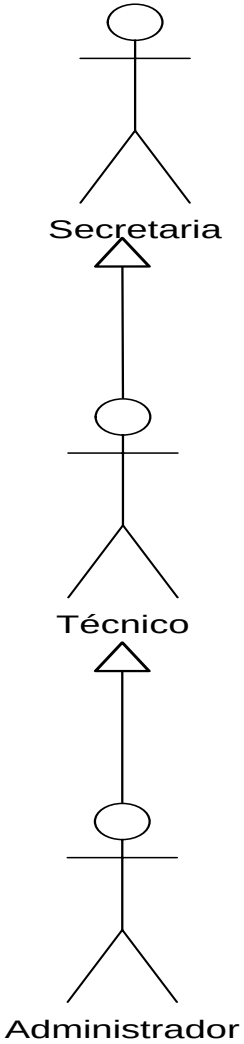


Diagrama de Casos de uso IV

- Casos de Uso
  - Un caso de uso define una funcionalidad del sistema.
  - Cada caso de uso constituye un flujo de eventos, que especifican la interacción que toma lugar entre el actor y el sistema.
  - Cada caso de uso produce un resultado observable y válido para el actor involucrado en la secuencia de acciones.

    - Verbos
    - Acciones

Diagrama de Casos de uso V

- Relaciones entre Casos de Uso:
  - **Generalización**
    - Relación que define la especialización de un caso de uso.
    - Los casos de uso abstractos describirán las partes similares y no podrán ser instanciados independientemente.
    - Los casos de uso concretos describirán el comportamiento específico.

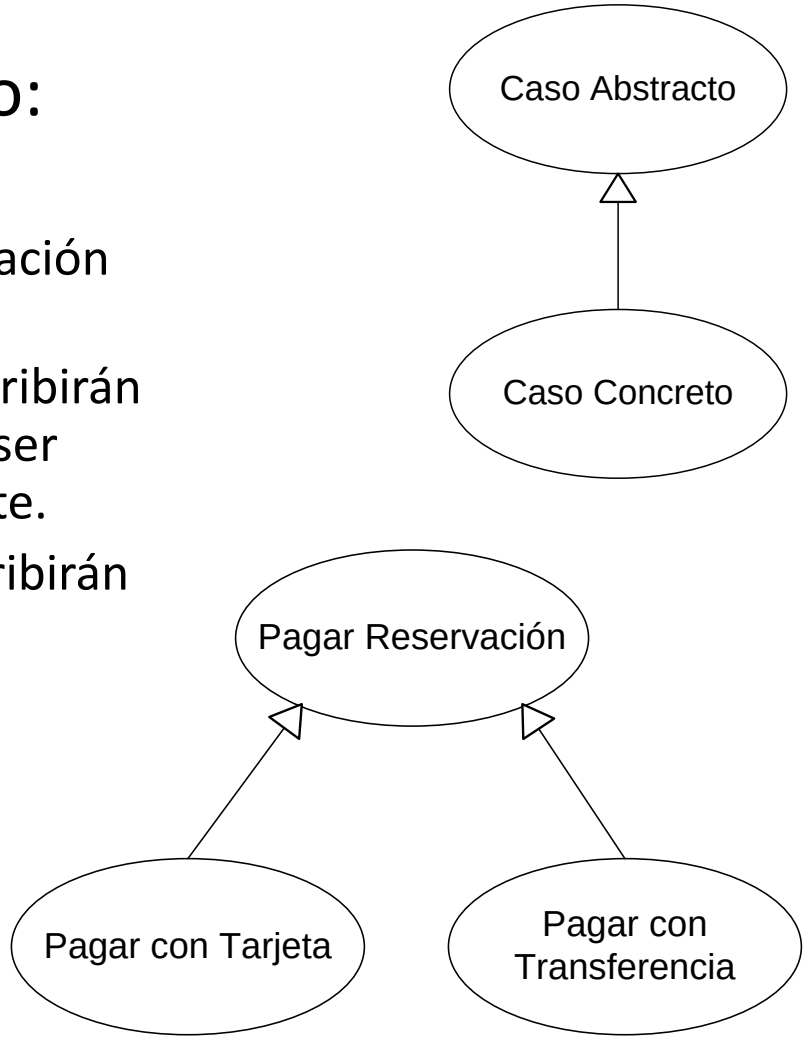


Diagrama de Casos de uso VI

- **Extensión <<extend>>**
  - Especifica como un caso de uso puede insertarse en otro para extender la funcionalidad de un caso de uso base.
  - El Caso Opcional es una extensión del Caso Base:
  - Una instancia del caso de uso Base puede incluir el comportamiento especificado por el Caso Opcional.

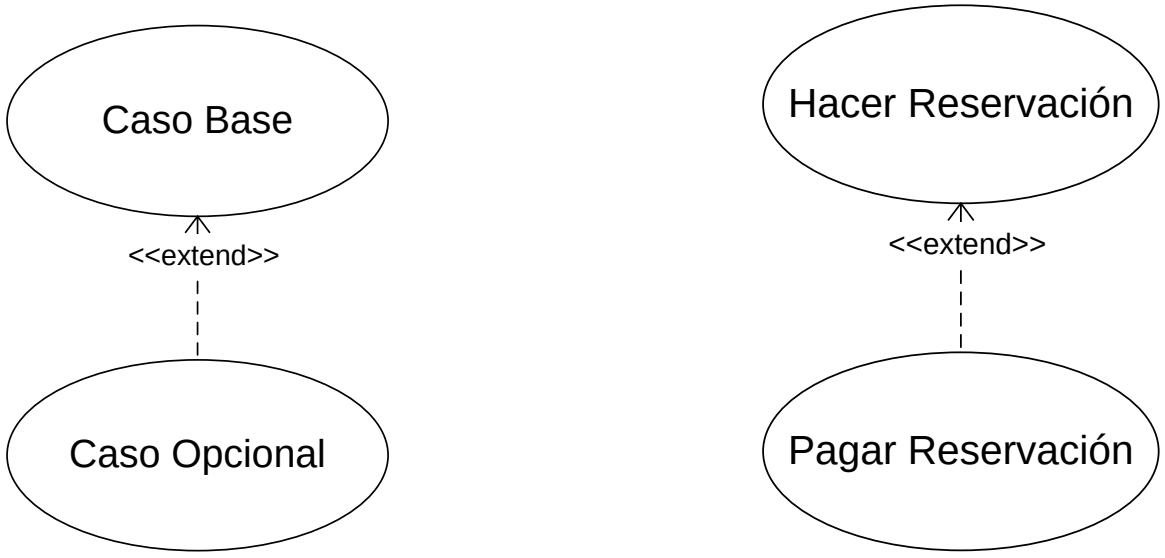


Diagrama de Casos de uso VII

- **Inclusión <<include>>**
  - La inclusión define como un caso de uso es parte obligatoria de un caso de uso base.
  - Un Caso Base incluye un Caso Obligatorio.
  - Una instancia de un caso base siempre incluye el comportamiento especificado por un caso de uso obligatorio.

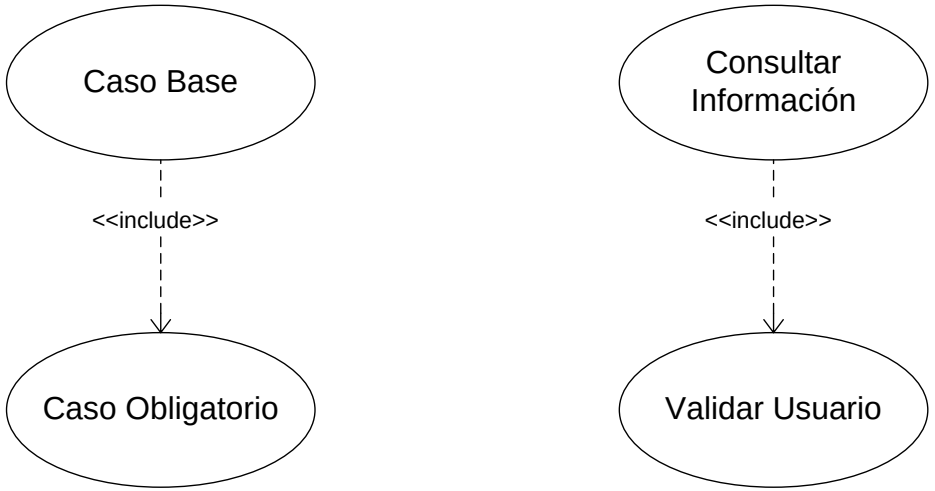


Diagrama de Casos de uso VIII

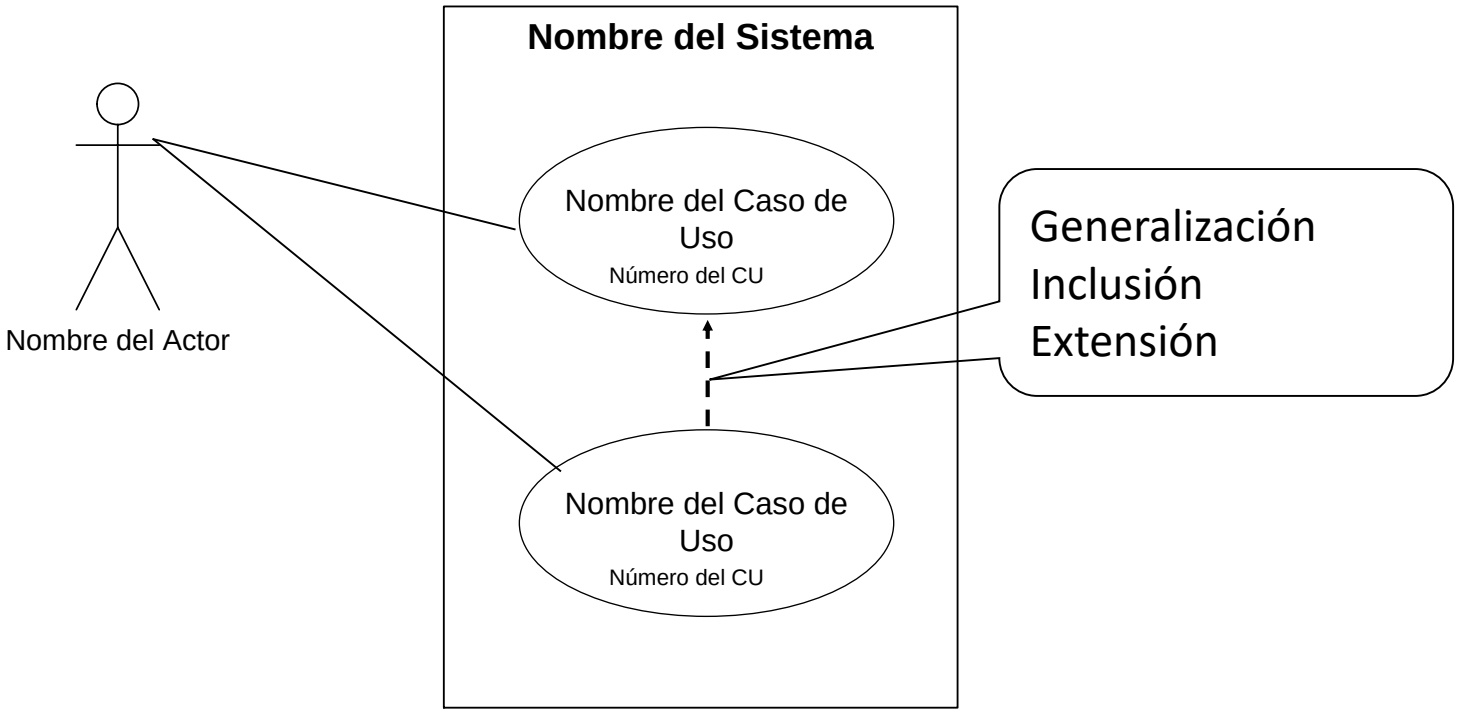


Diagrama de Casos de uso IX

- Especificación de un Caso de Uso: (Documentación)
  - Nombre del Caso de Uso.
  - Actores.
  - Propósito.
  - Precondiciones.
  - Flujo de Eventos Principal.
  - Sub Flujos.
  - Excepciones.
  - Postcondiciones.

Diagrama de Actividades

Diagrama de Actividades

- Define la lógica de los procedimientos, los procesos del negocio y flujos de trabajo del sistema.

- Demuestra la serie de actividades que deben ser realizadas en un caso de uso, así como las distintas rutas que pueden irse desencadenando en el caso de uso.

Diagrama de Actividades

- **Actividad**
- Representa una acción que será realizada por el sistema.
  - Actividad *Inicial*
  - Actividad *Final*

Diagrama de Actividades

- **Sub Actividad**
- Una acción puede ser descompuesta en varias actividades.

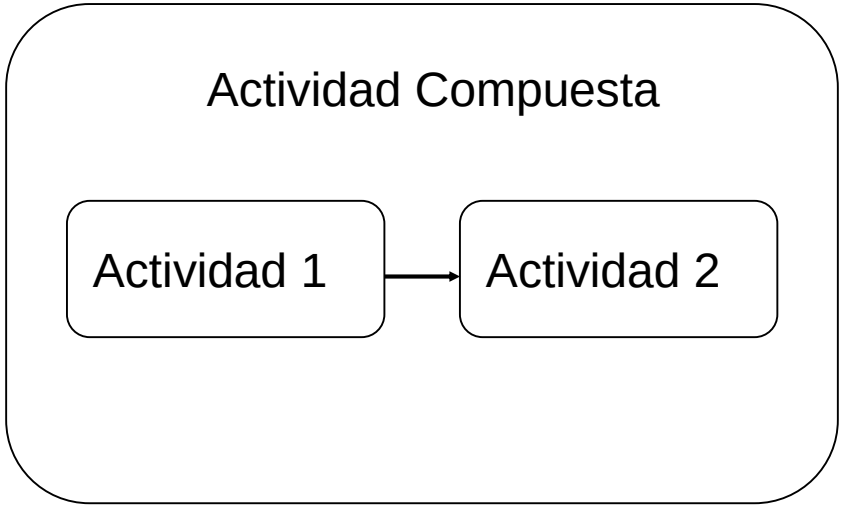


Diagrama de Actividades

- **Ramificación** (Branch):
- Una ramificación surge cuando existe la posibilidad que ocurra más de una transición (resultado) al terminar determinada actividad.

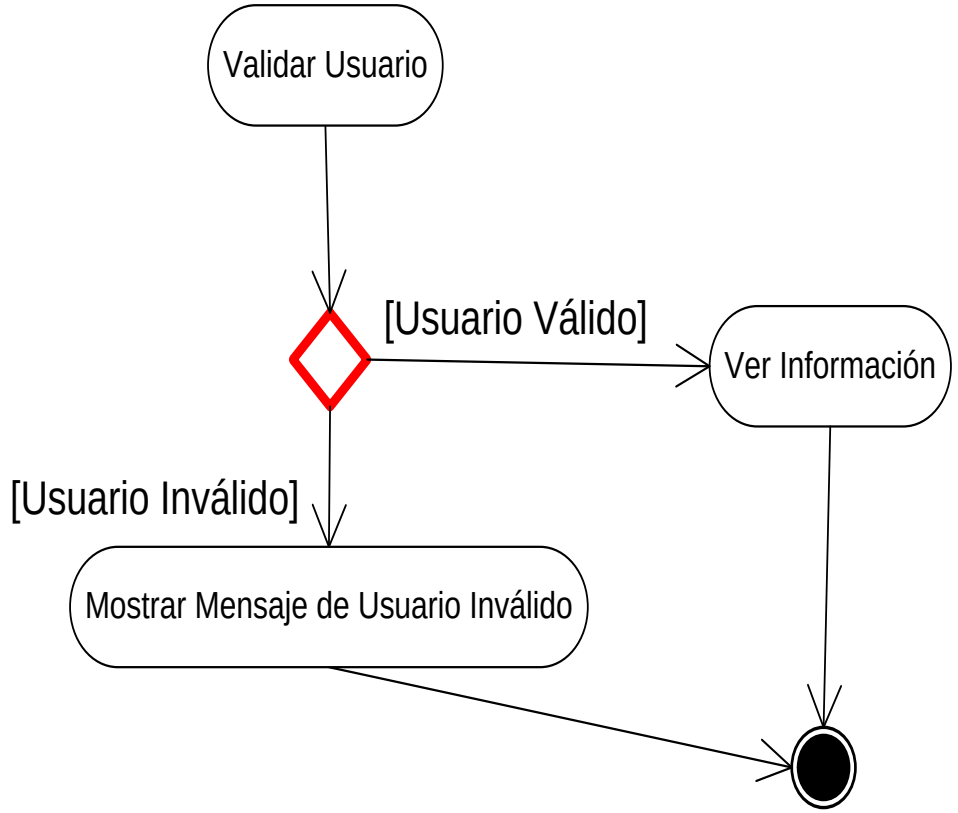
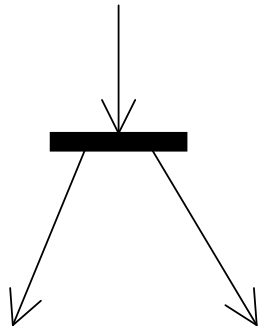


Diagrama de Actividades

- **Especificaciones Join**
  - *División*



  - *Unión*

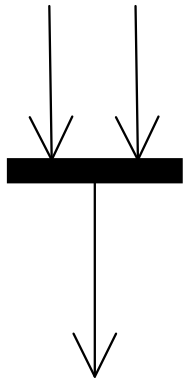
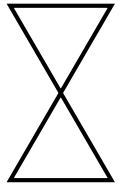


Diagrama de Actividades

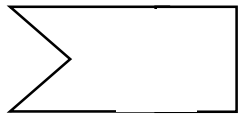
- **Señales**

  Algunas acciones responden a señales

  - Señales de tiempo



  - Envío de señal



  - Recepción de señal

Diagrama de Actividades

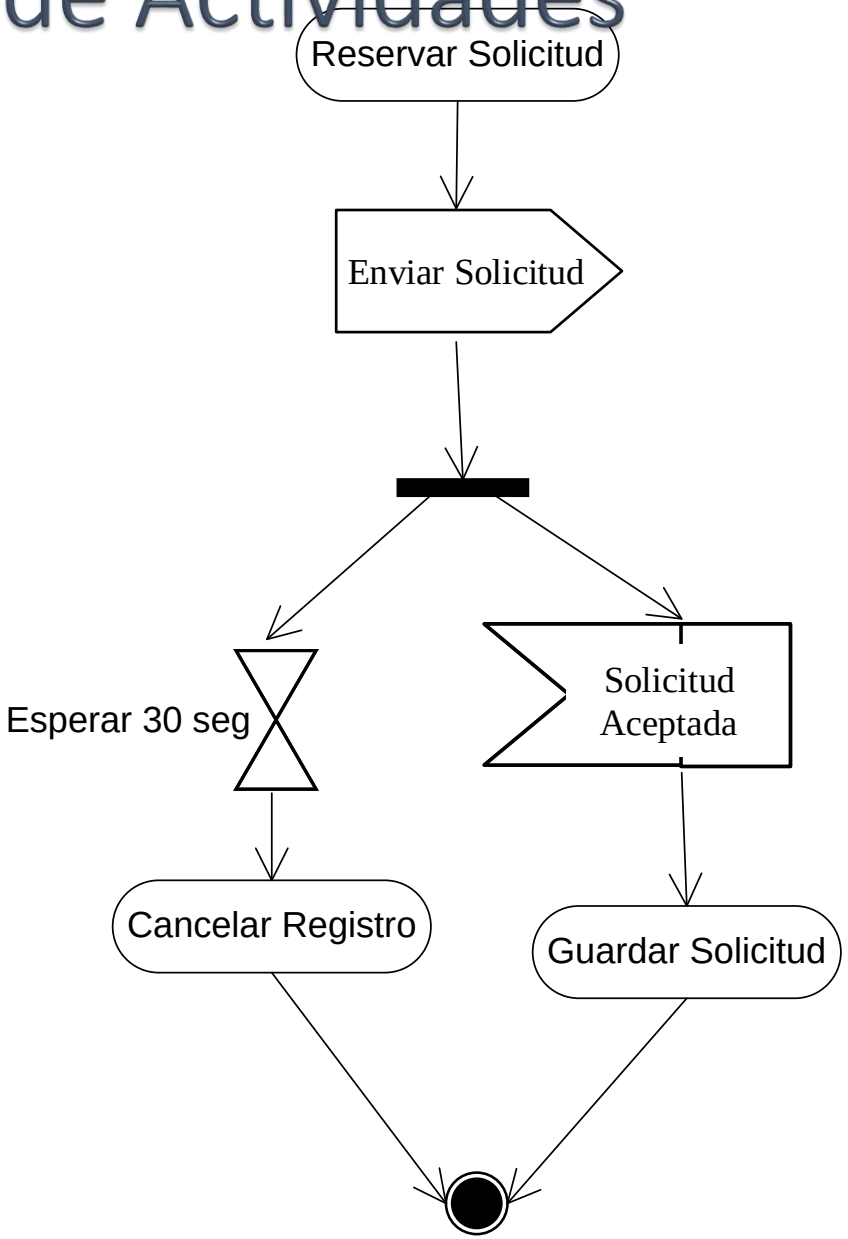


Diagrama de Secuencia

Diagrama de Secuencia

- Describen como colaboran y se comunican (interaccionan) los objetos del sistema.

- Muestra los objetos que participan en una interacción, el intercambio de mensajes y su ordenamiento en el tiempo.

- Son una representación que muestra, para un escenario de un caso de uso, los *eventos* que generan los actores, su *orden* y posibles eventos internos en el sistema

Diagrama de Secuencia

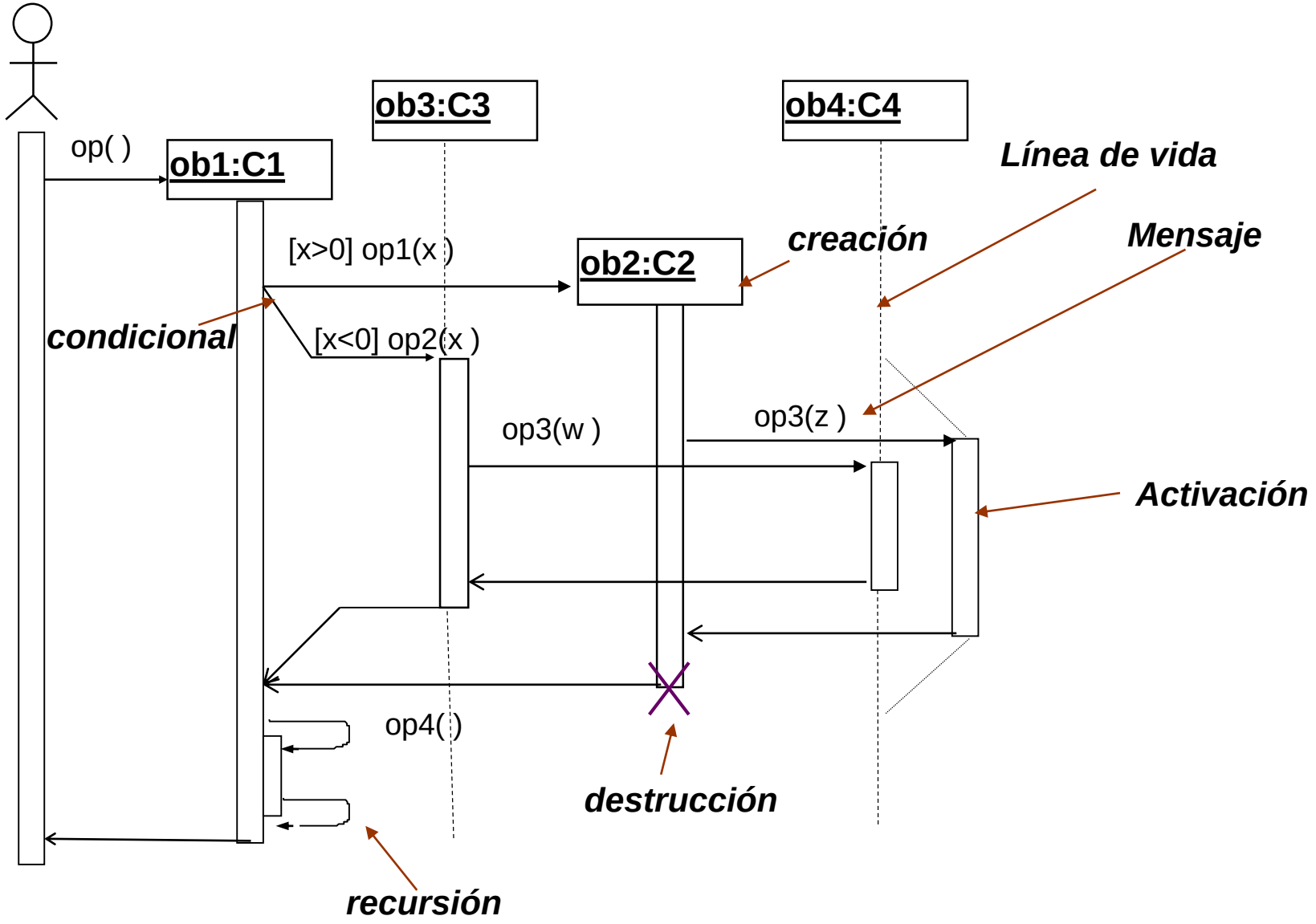


Diagrama de Secuencia

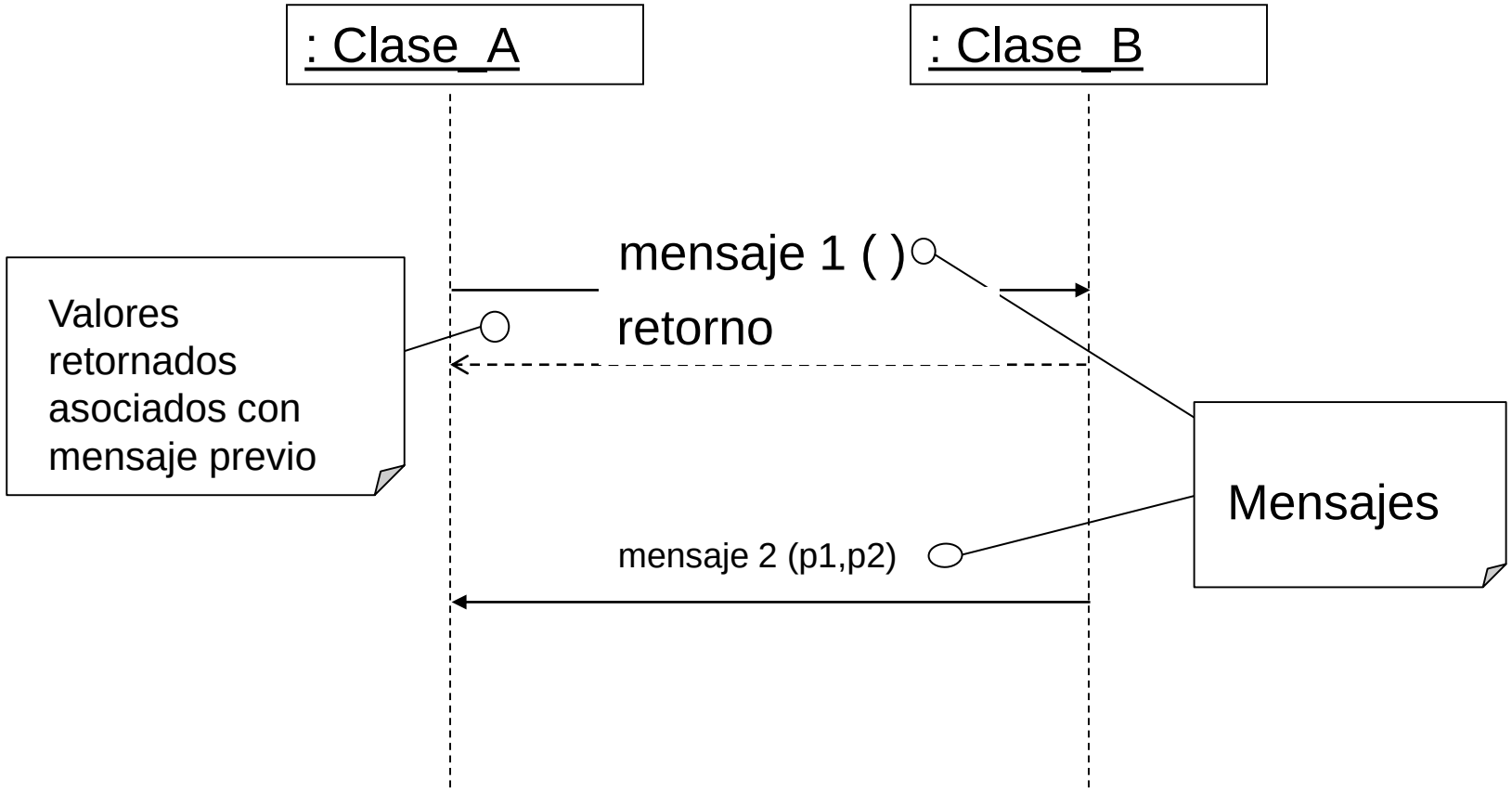


Diagrama de Secuencia

- Crear Objetos: **<<create>>**

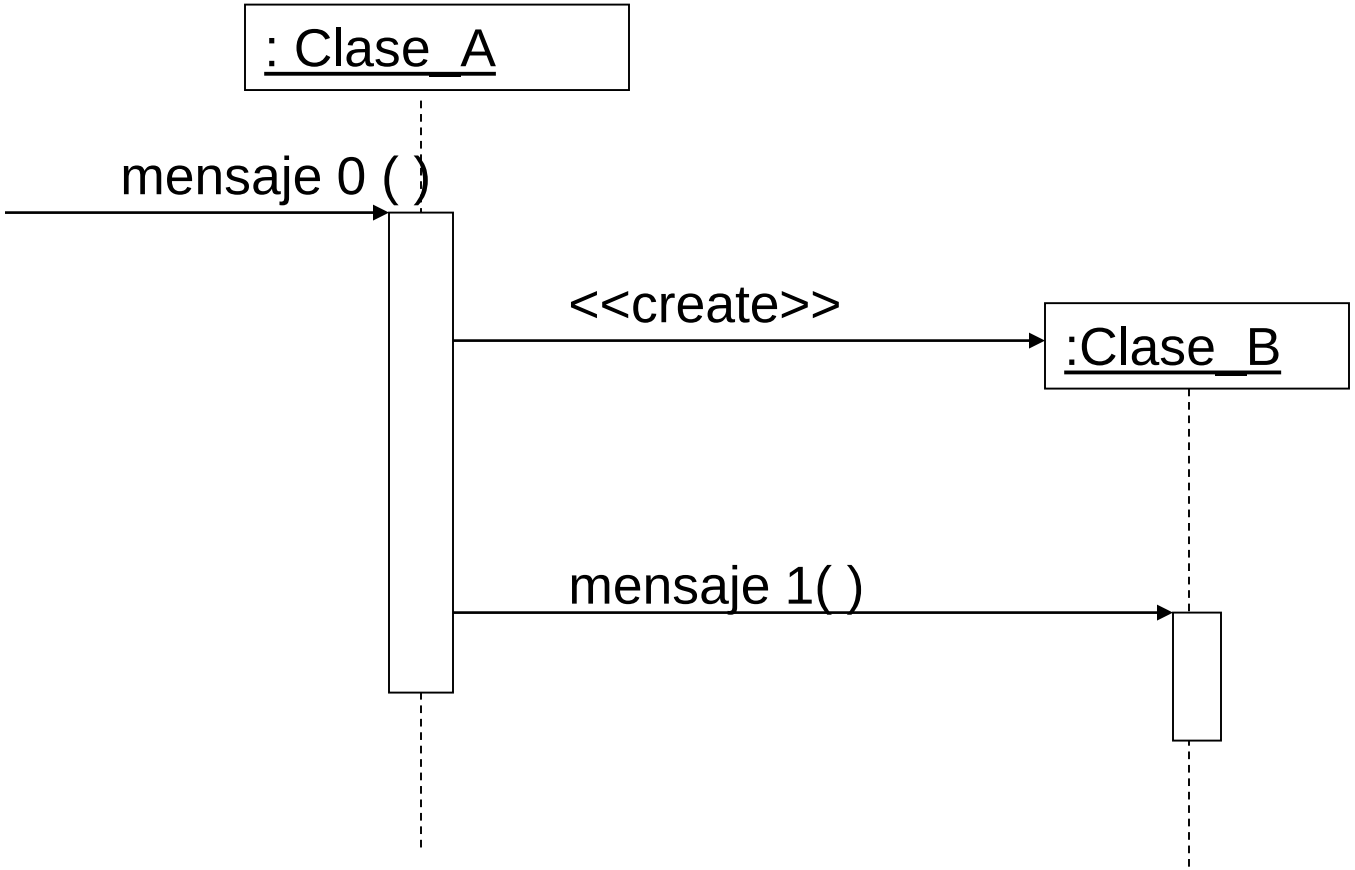


Diagrama de Secuencia

- Destruir Objetos: **<<destroy>>**

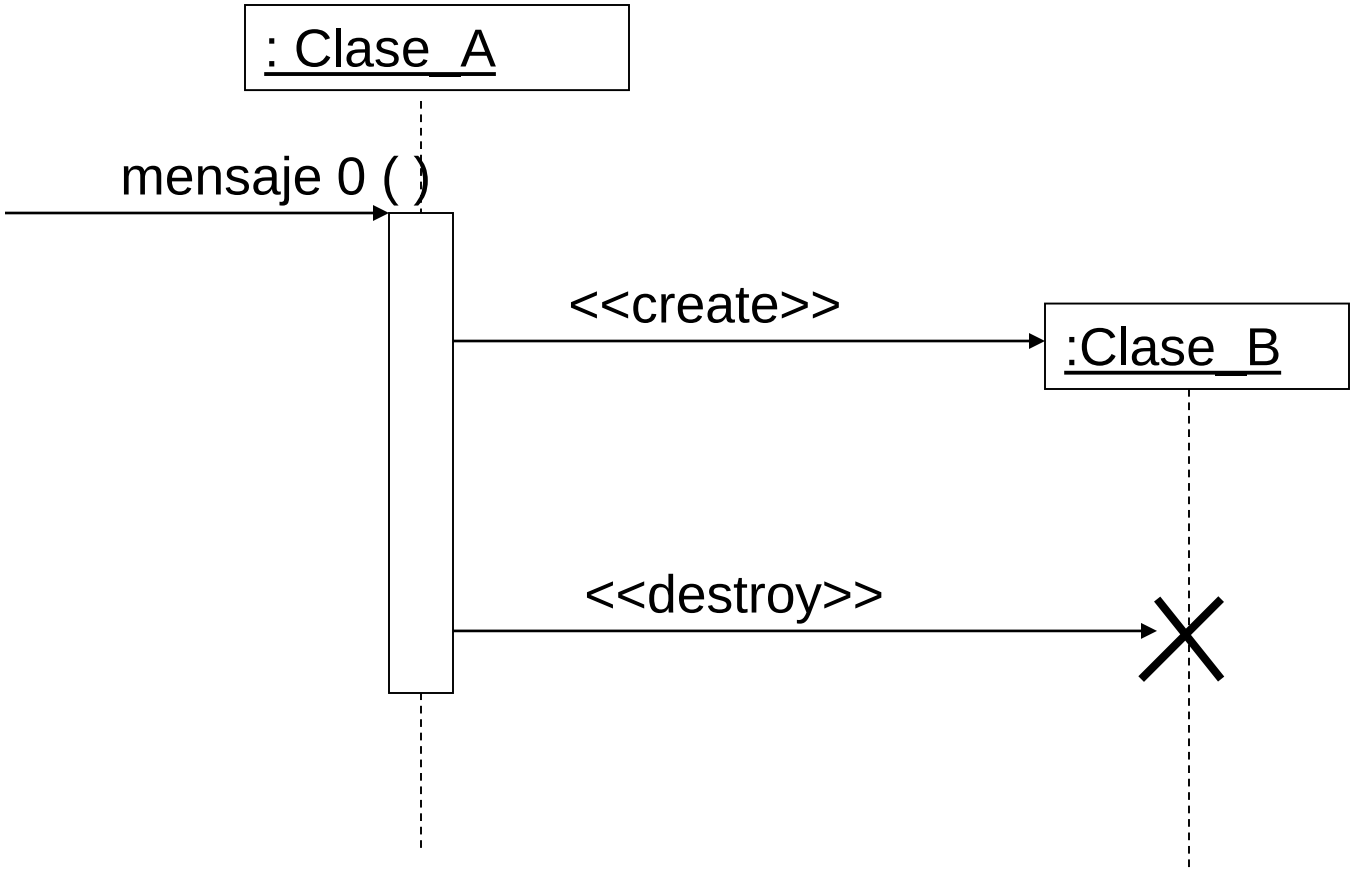


Diagrama de Secuencia

- Mensajes Condicionales

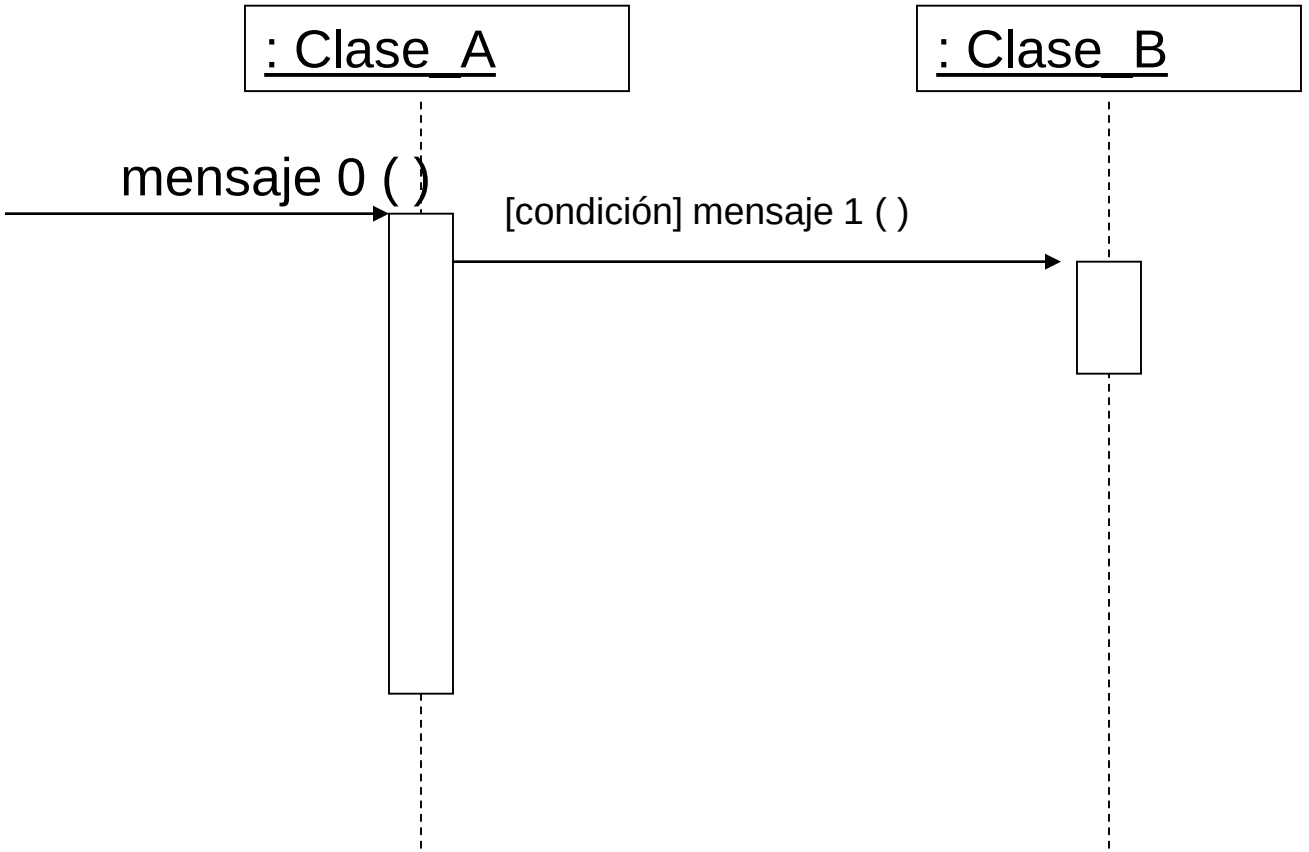


Diagrama de Secuencia

- Mensajes Condicionales Excluyentes

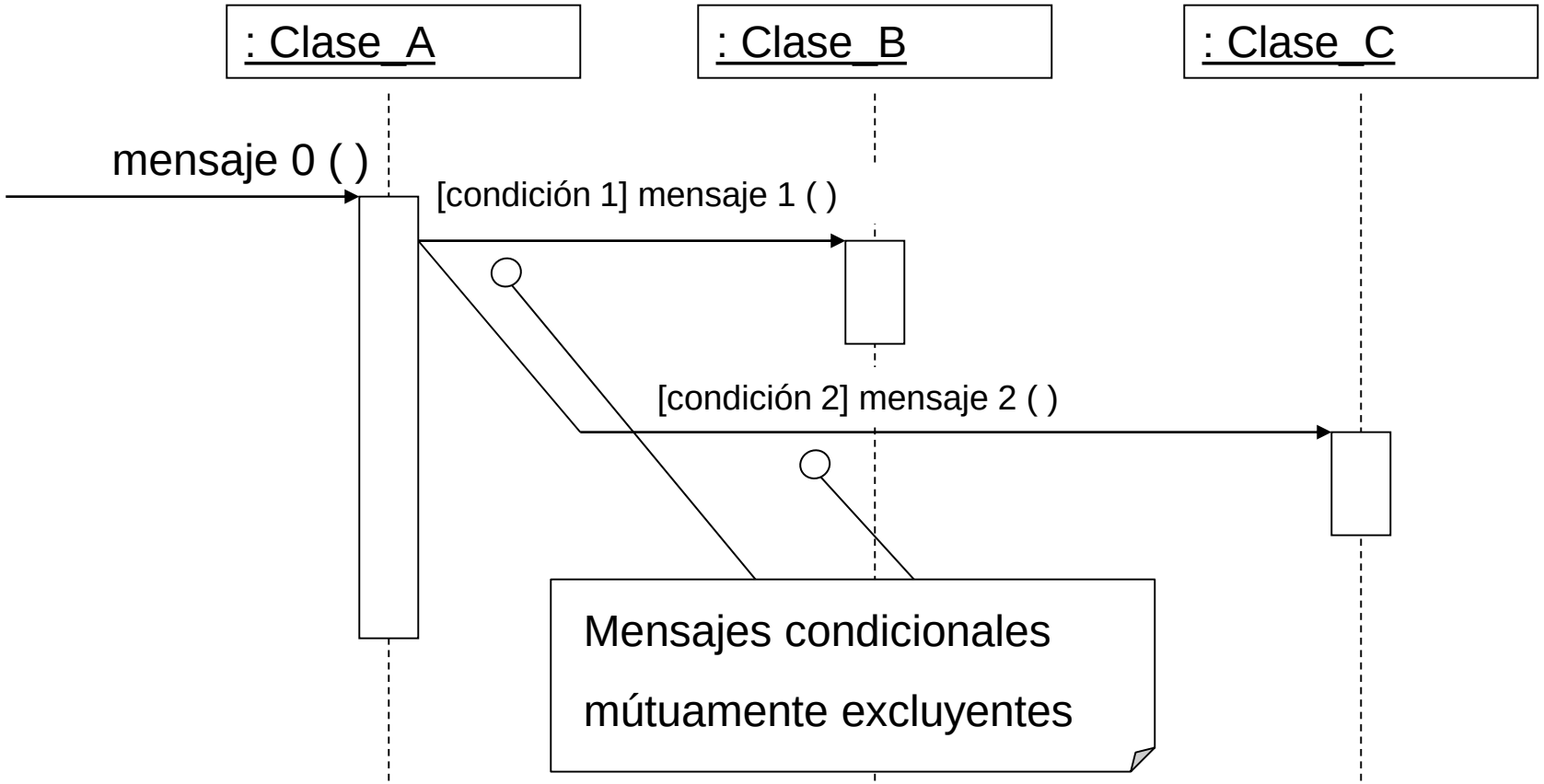


Diagrama de Secuencia

- Caja de Iteración:

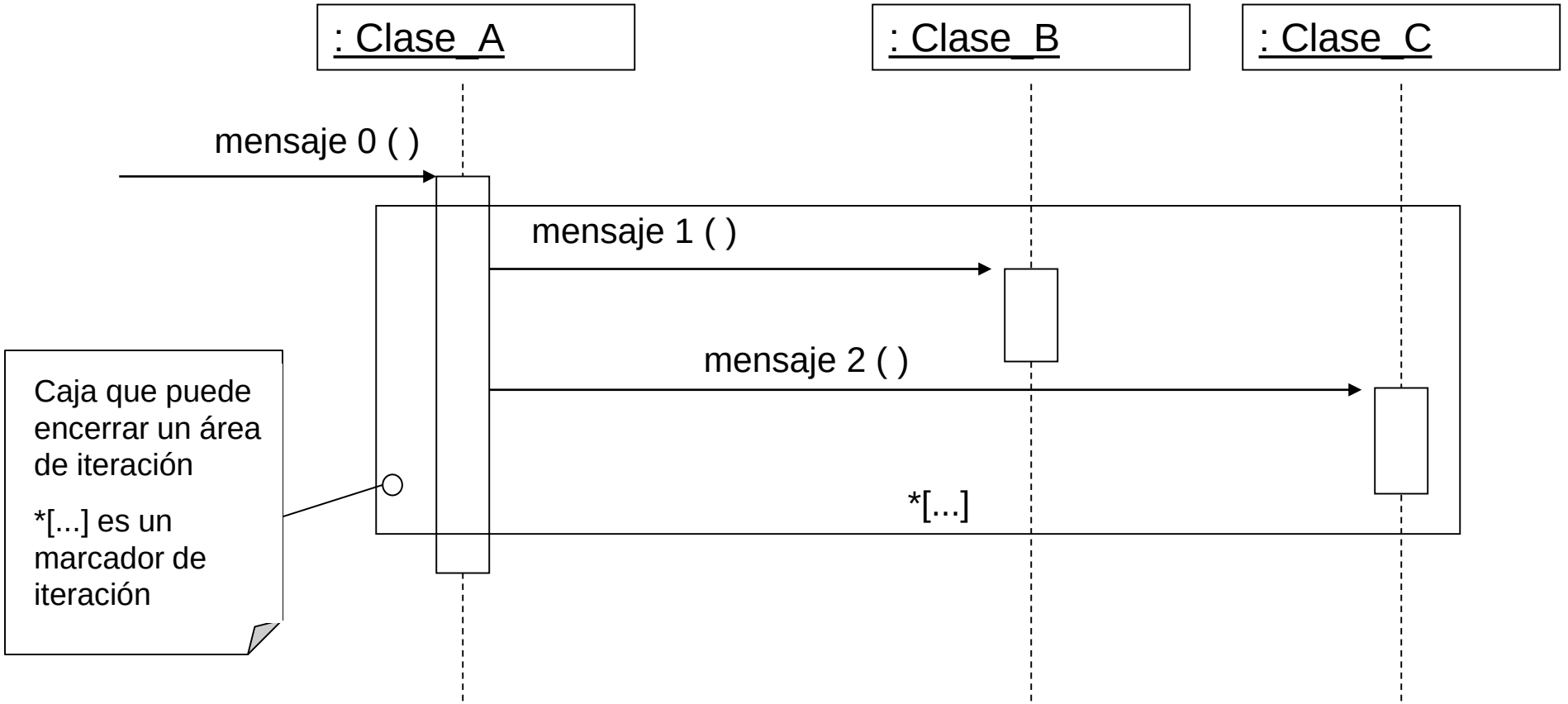


Diagrama de Secuencia

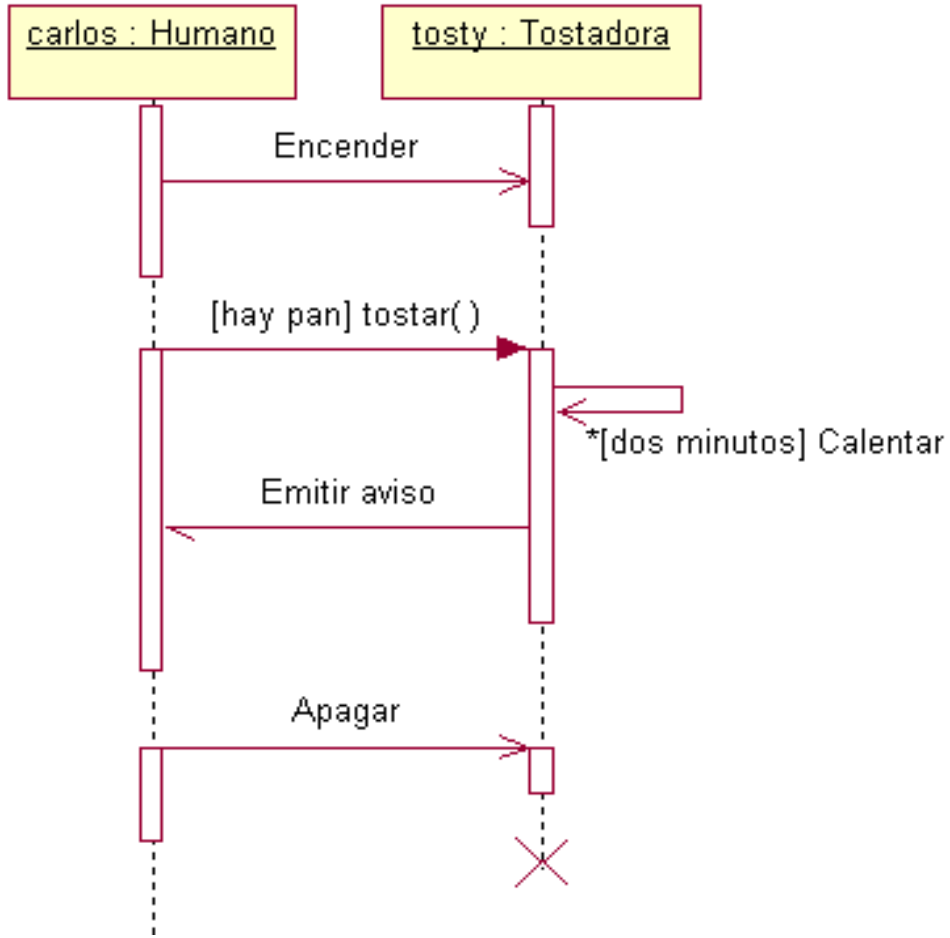


Diagrama de Colaboración

Diagrama de Colaboración

- Describe la **interacción** entre los objetos, numerando la secuencia de mensajes.

Diagrama de Colaboración

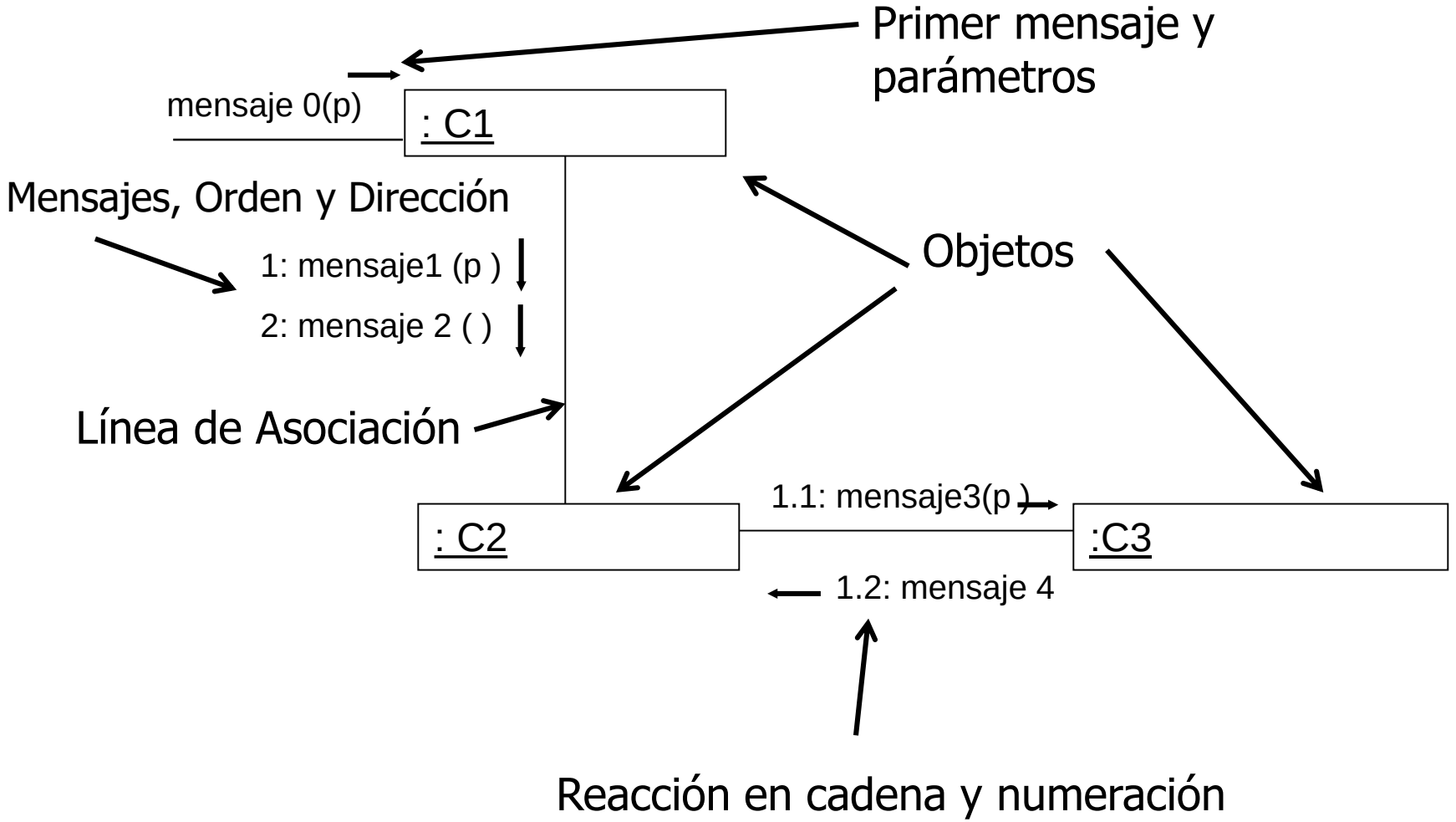


Diagrama de Colaboración

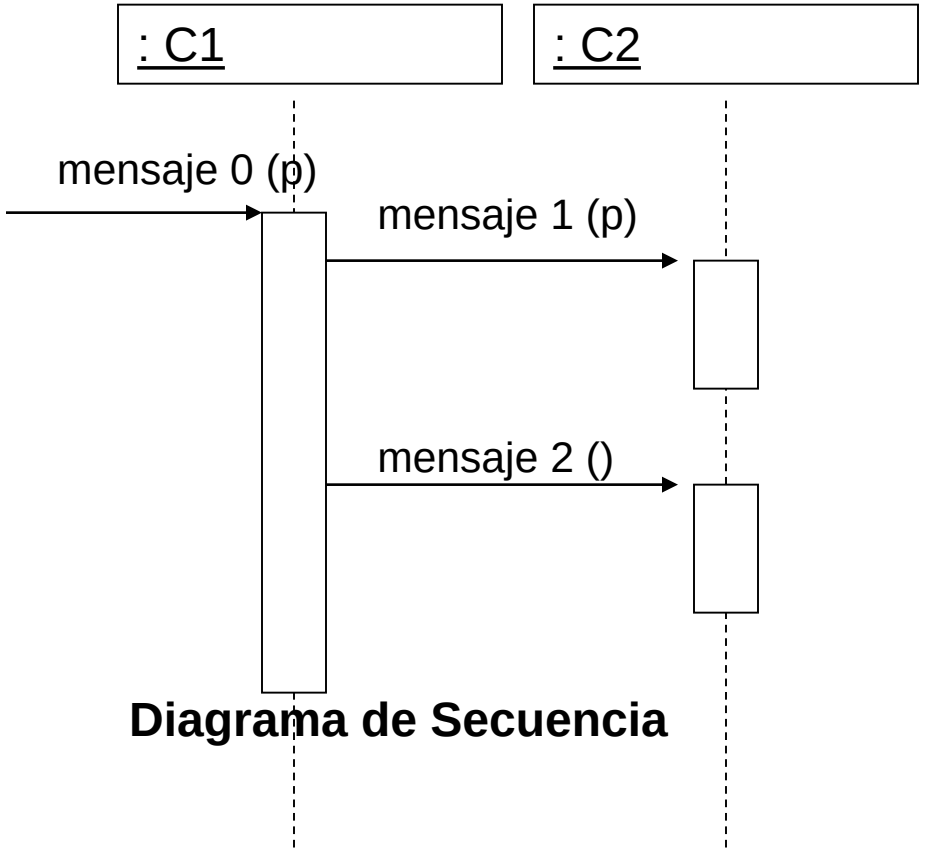


Diagrama de Secuencia

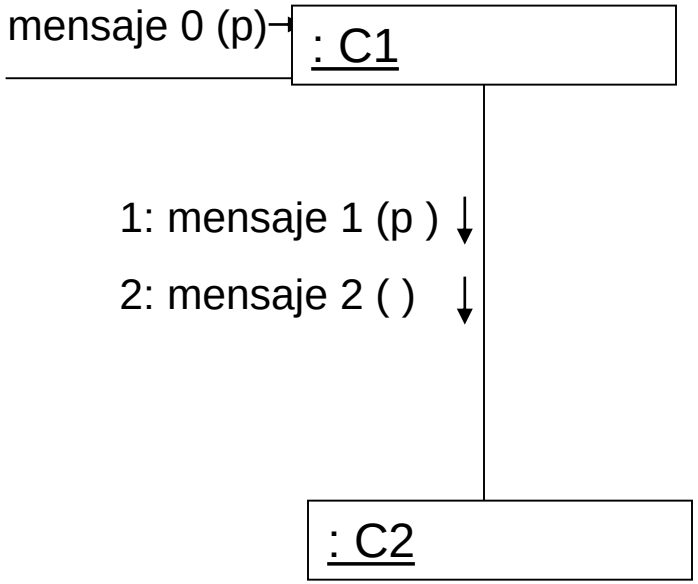


Diagrama de Colaboración

Diagrama de Colaboración

- Mensajes Excluyentes

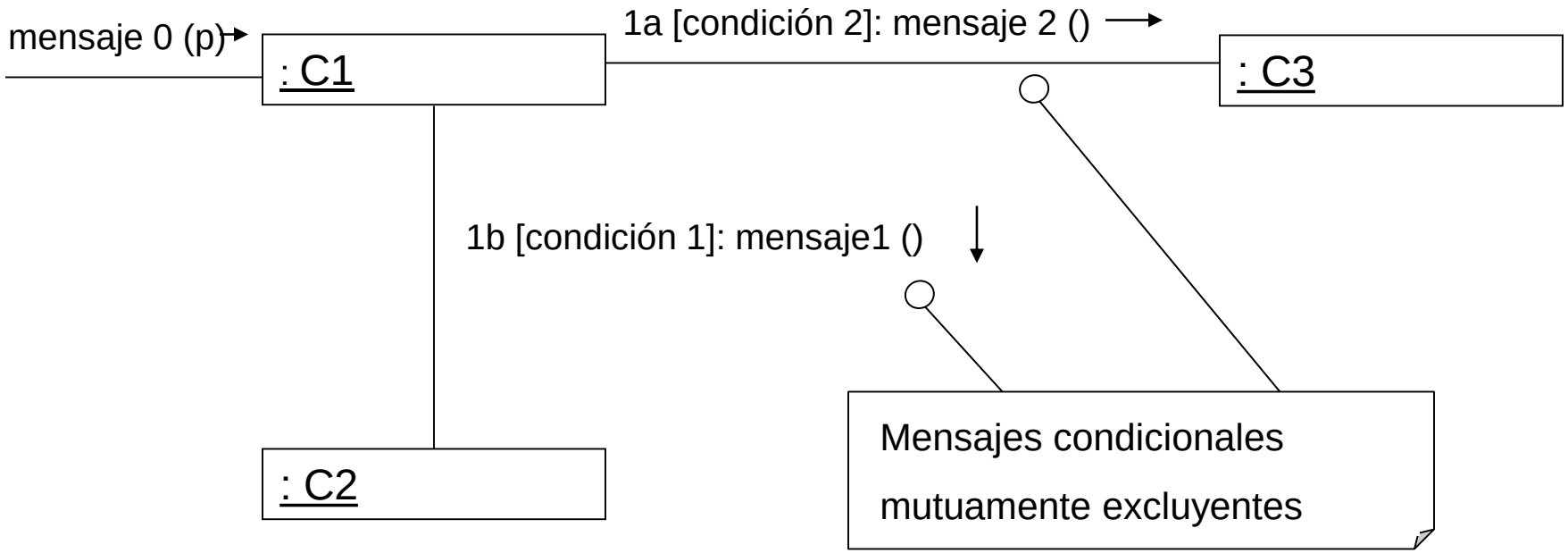


Diagrama de Colaboración

- Iteración

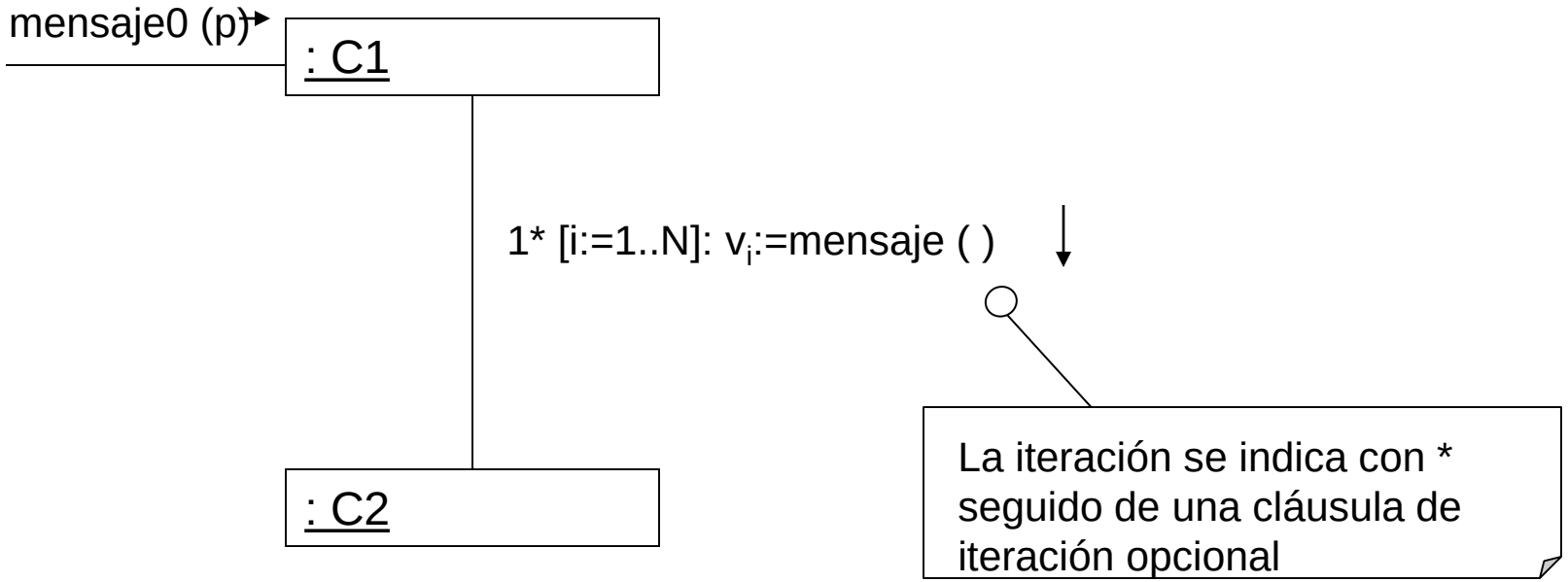


Diagrama de Colaboración

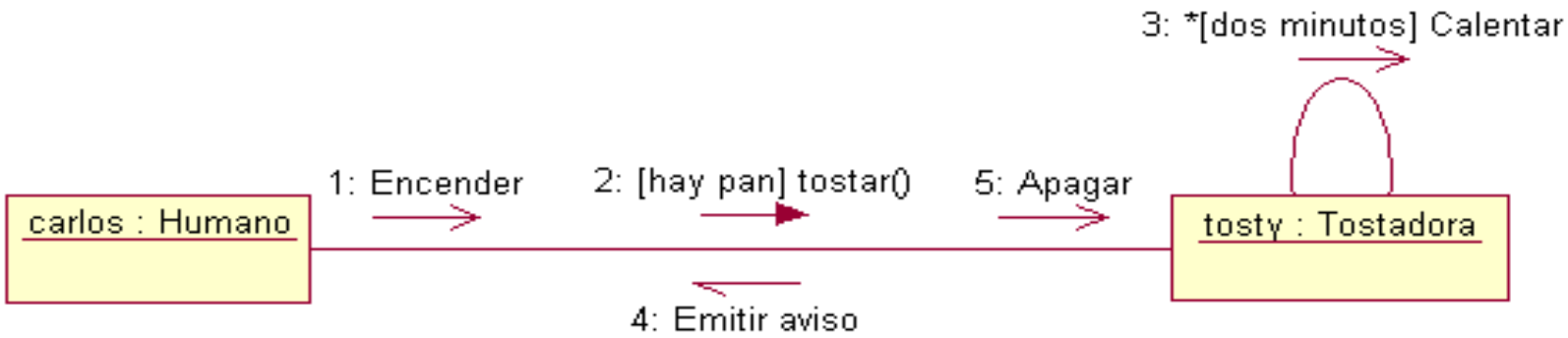


Diagrama de Despliegue

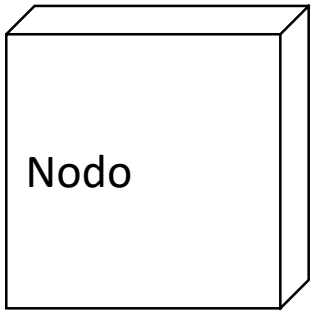
Diagrama de Despliegue

- Muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

- En este diagrama se indica la situación física de los componentes lógicos desarrollados.

Diagrama de Despliegue

- **Nodo**: Elemento donde se ejecutan los componentes, representan el despliegue físico de estos componentes.



- **Asociación**: Representa el tipo relación que soporta la comunicación entre nodos

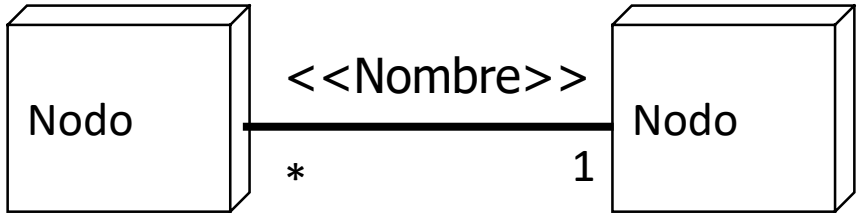


Diagrama de Despliegue

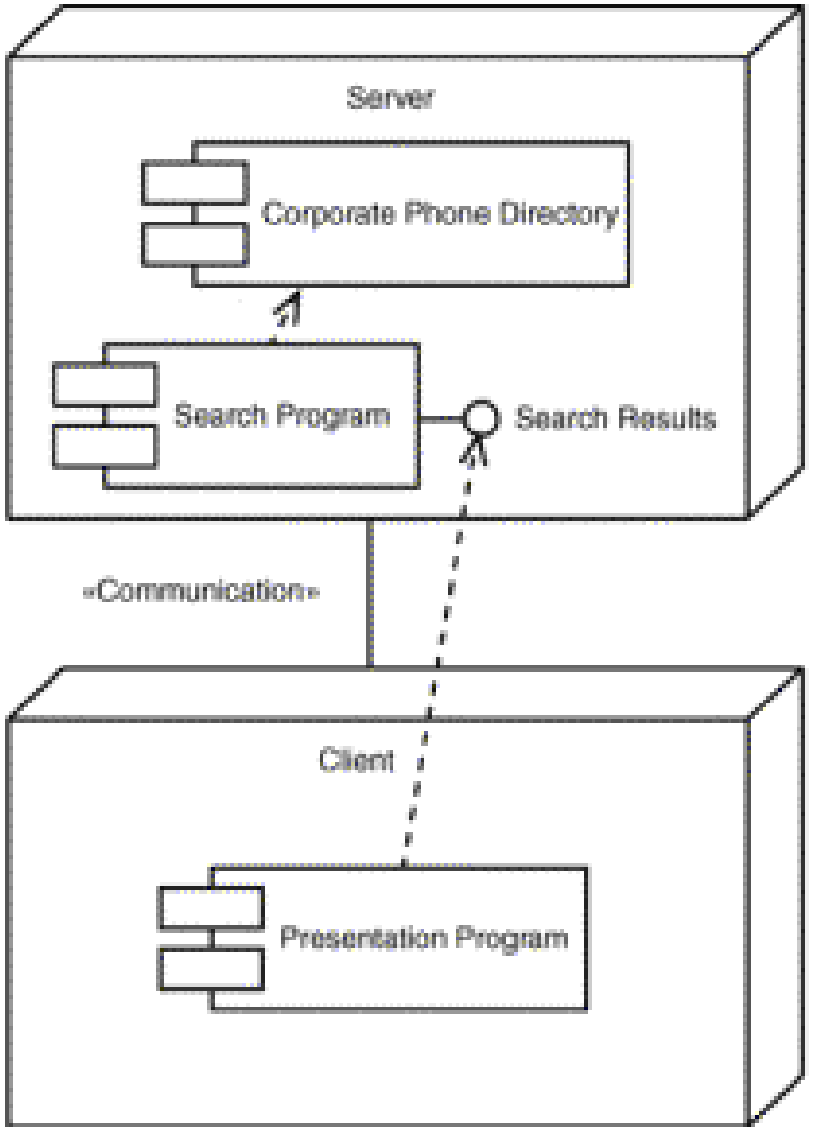


Diagrama de Componentes

Diagrama de Componentes

- Muestra la relación entre componentes de software: Dependencias, comunicación, ubicación, etc.

- Muestra organizaciones y dependencias lógicas entre componentes *software*, sean éstos componentes de código fuente, binarios o ejecutables.

- Define los módulos físicos del software y sus relaciones.

Diagrama de Componentes

- **Componente**
  - Es un tipo de contenedor.
  - Provee una vista encapsulada de las funcionalidades definidas en las clases.
  - Por ejemplo, un **paquete** en un diagrama de componentes representa un división física del sistema.

Diagrama de Componentes

- **Interfaces**
  - Las interfaces son los **puntos visibles** de entrada o los servicios que un componente está ofreciendo y dejando disponibles a otros componentes de software y clases.

  - Exponen funcionalidades para otros componentes y las requeridas de otros.

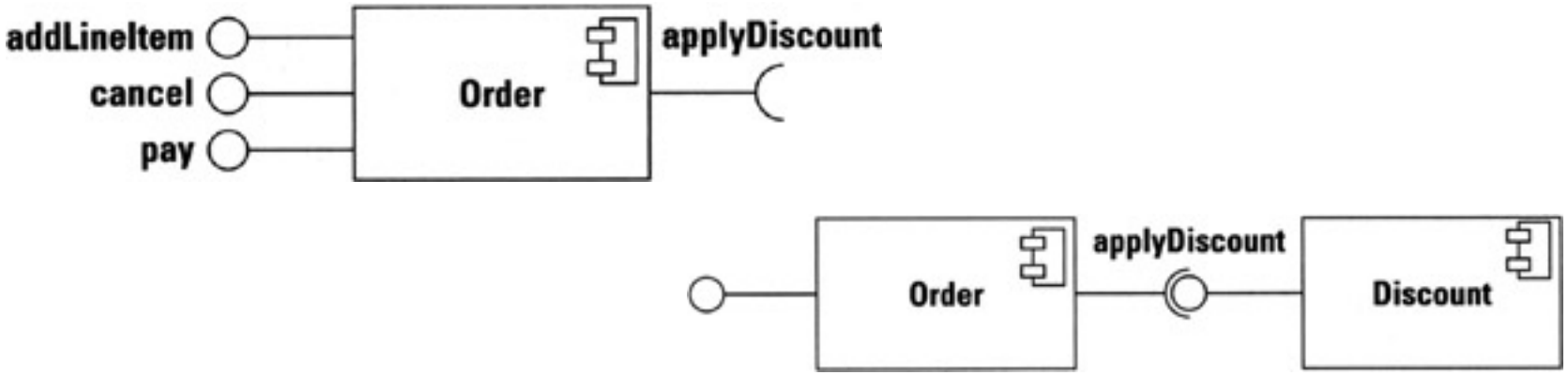


Diagrama de Componentes

- **Dependencia de módulos**
  - Abstrae la implementación de la interfaz e indica la dependencia entre módulo

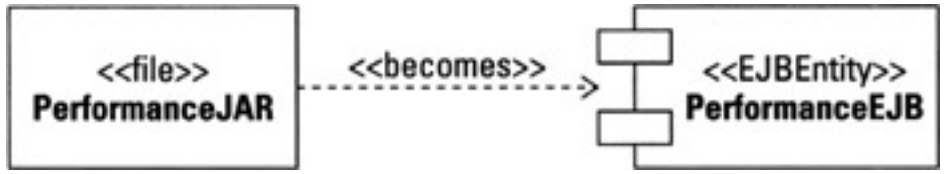


Diagrama de Estado

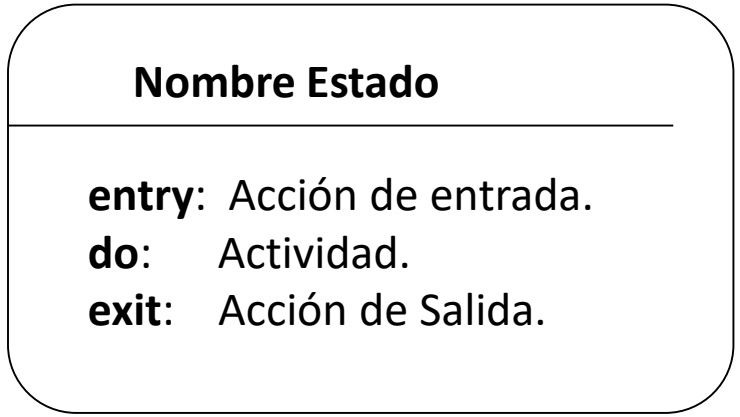
Diagrama de Estado

- Muestra los diferentes estados de un objeto durante su vida, y los estímulos que provocan los cambios de estado en un objeto.

- Funcionan como máquinas de estado o autómatas finitos, que pueden estar en un conjunto de estados finitos y que pueden cambiar su estado a través de un estímulo perteneciente a un conjunto finito.

Diagrama de Estados

- **Estado:**

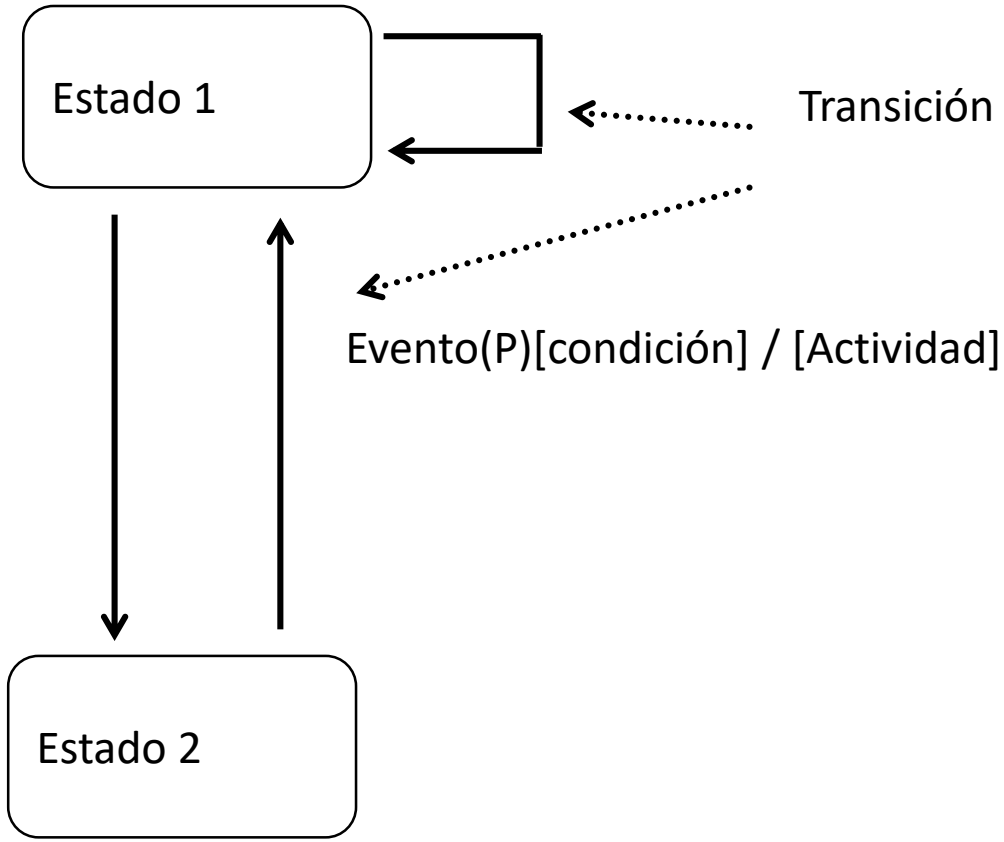


- Estado Inicial

- Estado Final

Diagramas de Estados

- **Transición**:



Diagramas de Estados

- Diagrama de Estados de un Libro: