Ministerio de Producción
Presidencia de la Nación

Ministerio de Educación y Deportes

Subsecretaría de Servicios Tecnológicos y Productivos

Diagramas de clases y secuencia

Diagrama de clases

- Se utilizan para:
  - Explorar conceptos del dominio
  - Analizar requerimientos
  - Mostrar el diseño detallado de software orientado a objetos

- Muestra una colección de elementos de modelado, tales como clases, tipos y sus contenidos y relaciones.

- Generalmente contiene:
  - Clases
  - Interfaces
  - Relaciones entre clases

Clases

- Una clase es la descripción de un conjunto de objetos que comparten los mismos:
  - Atributos
  - Operaciones
  - Semantica

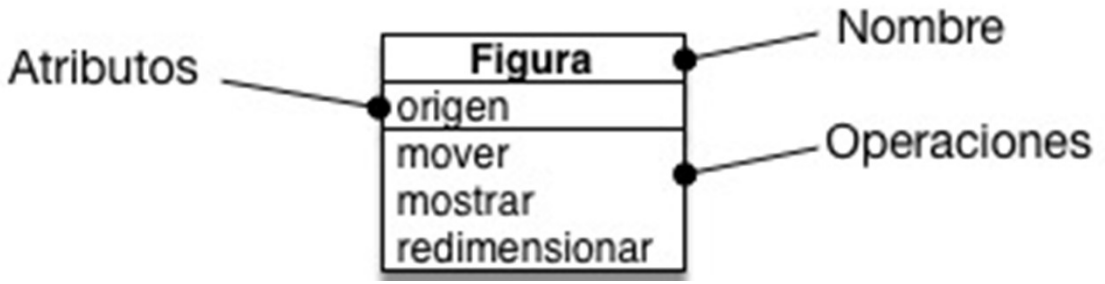
- Con las clases se captura el vocabulario del sistema que se está desarrollando.

- Las clases se pueden utilizar para representar cosas del software, del hardware y cosas puramente conceptuales (por ejemplo una estrategia de ordenamiento).

Representación gráfica

Gráficamente la clase se dibuja como un rectángulo

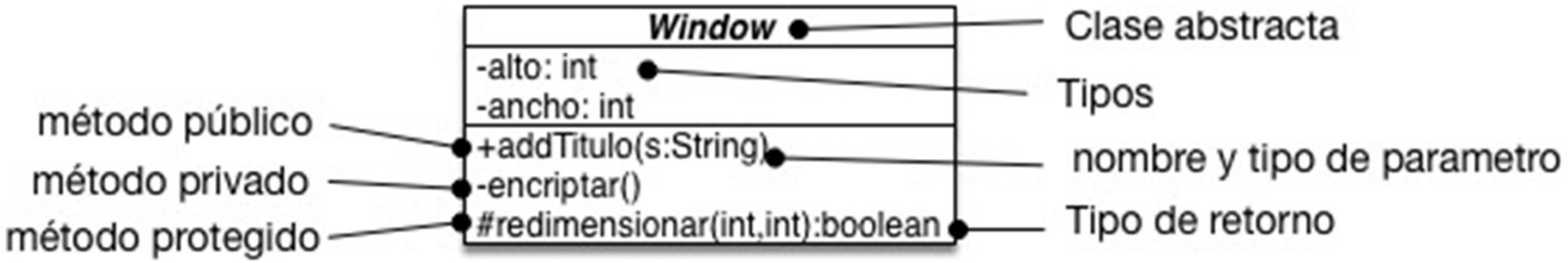
- Debe tener un nombre
- La primera letra de cada palabra del nombre debe capitalizarse (ej. Usuario, SensorDeTemperatura)
- Atributos:
  - Puede o no tenerlos
  - Representan alguna propiedad de lo que se esta modelando
  - Casi siempre son sustantivos
- Operaciones:
  - Representan los servicios que provee la clase
  - Suelen ser verbos



Mayor detalle de la clase

Las clases UML pueden brindarnos mayor información:

- Accesibilidad de atributos y operaciones:
  - Public: +
  - Private: -
  - Protected: #
- Nombres de clases abstractas en cursiva
- Tipo de los atributos
- Parámetros y retornos de los mensajes

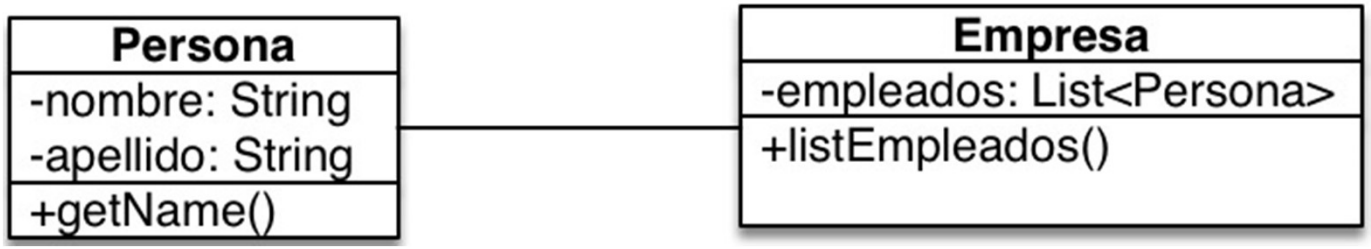


Relaciones entre clases

- Las relaciones son conexiones entre clases
- Modelan la colaboración entre objetos
- Los tres tipos mas comunes de relaciones son:
  - Asociación
  - Generalización
  - Dependencia
- Dos clases A y B estan relacionadas si:
  - Un objeto de la clase A envía un mensaje a un objeto de la clase B
  - Un objeto de la clase A crea un objeto de la clase B
  - Un objeto de la clase A tiene un atributo cuyo tipo es B o que es una colección de objetos de tipo B.
  - Un objeto de la clase A recibe un mensaje con un objeto de la clase B como parámetro
  - La clase A es superclase de B

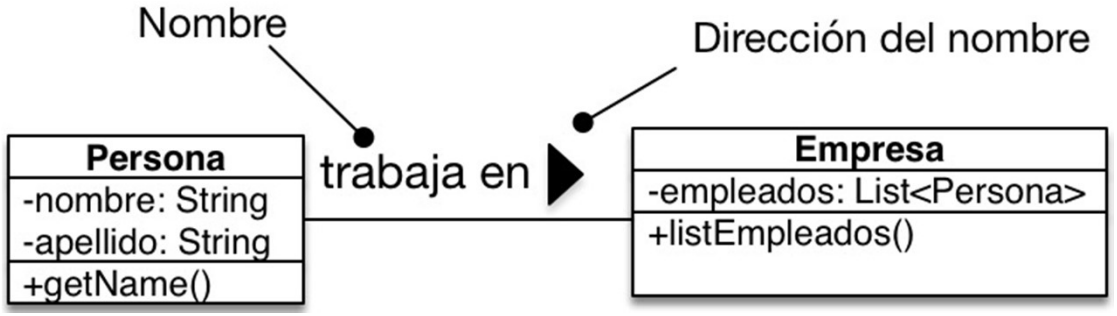
Relación de asociación

- Es una relación estructural.
- Dos clases A y B estan asociaciadas si:
  - Un objeto de la clase A tiene un atributo cuyo tipo es B o que es una colección de objetos de tipo B.
- Usamos asociaciones para modelar conexiones del tipo: “tiene”, “es de”, “conoce”
- Se dibuja como una linea entre dos clases

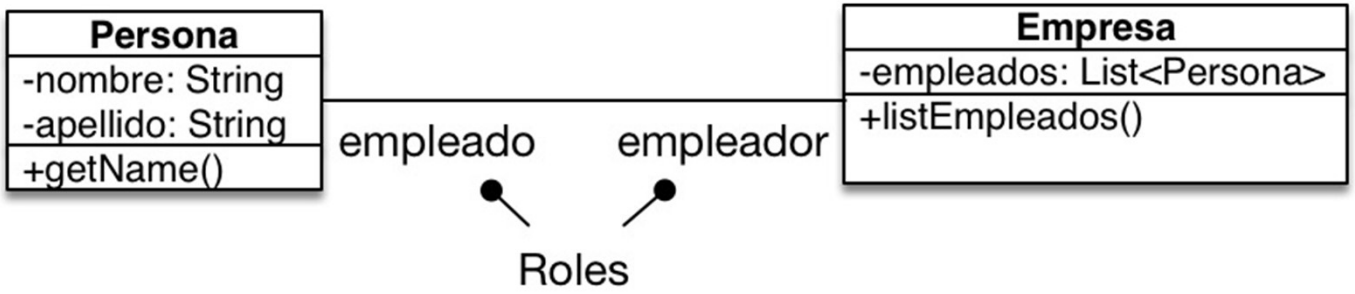


Adornos de asociaciones

- Las asociaciones pueden tener adornos que agregan mas información a la relación
  - Nombre que describe la naturaleza de la relación

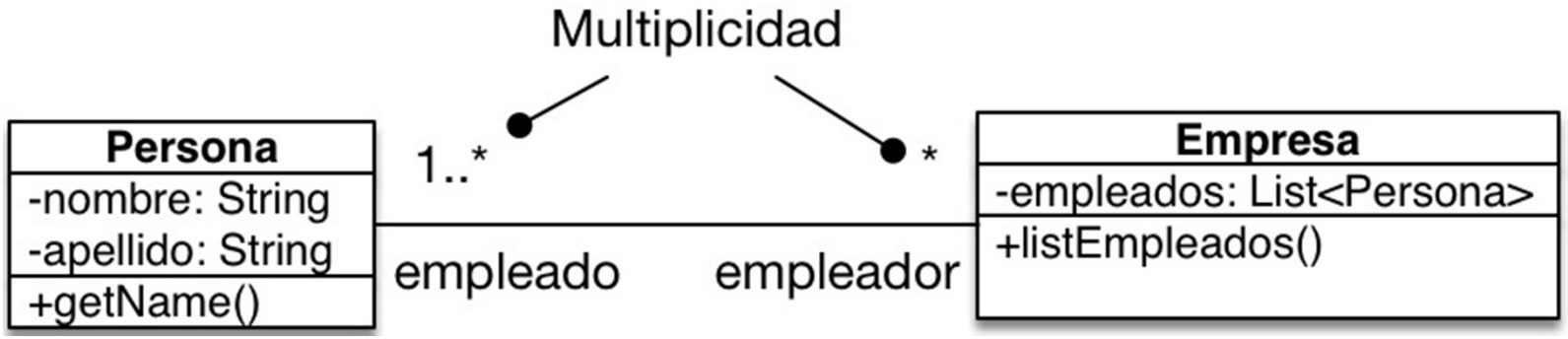


  - Rol que cumple una clase en la relación (La misma clase puede jugar el mismo o diferentes roles en otras asociaciones)



Adornos de asociaciones

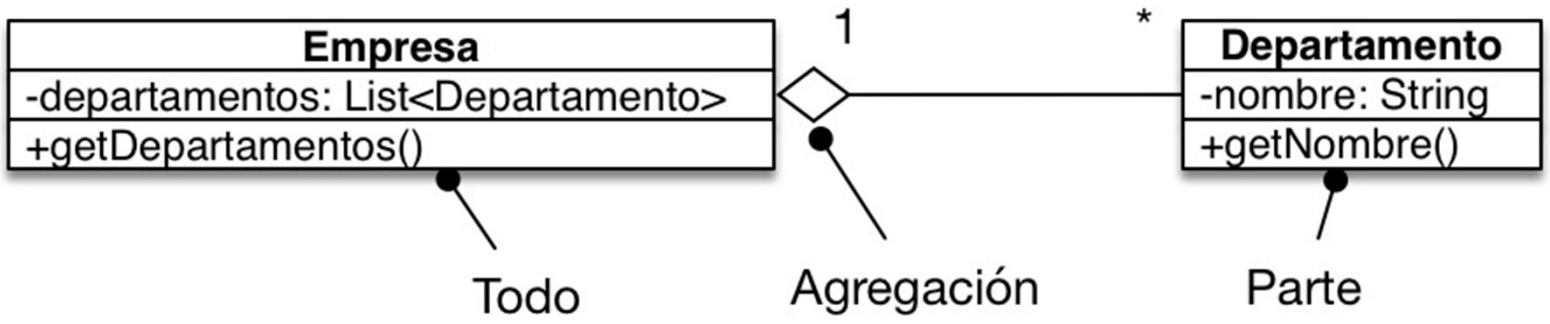
- La multiplicidad indica cuantos elementos de una instancia se relacionan con otra. Por ejemplo:
  - 0..1: entre 0 y 1 objetos
  - 3..4: entre 3 y 4 objetos
  - 6..*: 6 o mas objetos
  - 0..1, 3..4, 6..*: cualquier numero de objetos que no sea 2 o 5.



- Una persona puede trabajar en un numero *N* de empresas. Una empresa puede tener 1 o mas empleados.

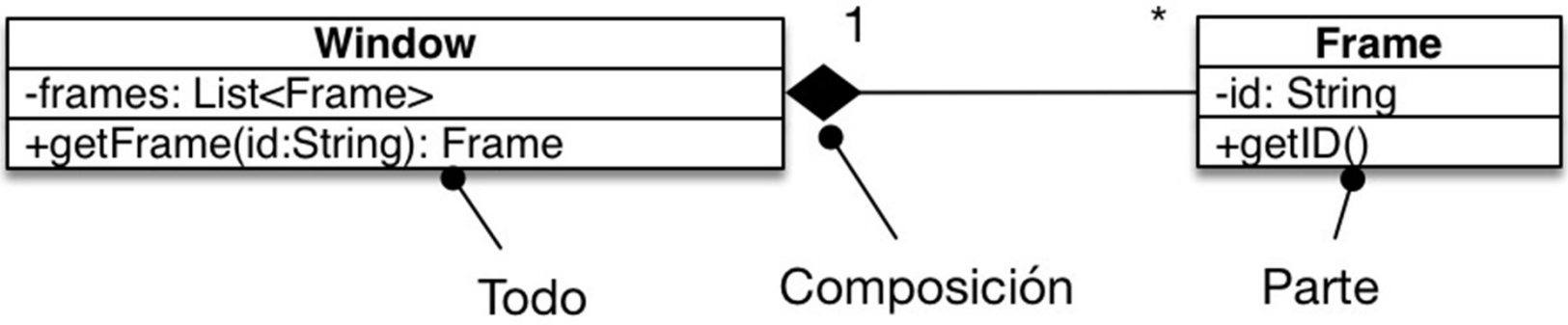
Relación de agregación

- Es un tipo especial de relación de asociación
- Se utiliza solamente cuando una de las clases representa el "todo" y la/s otra/s la/s "partes"
- Modelan conexiones del tipo: "Esta formado por"
- Se dibuja como una línea entre dos clases con un rombo sobre la clase que representa el "todo"
- Puede incluir adornos



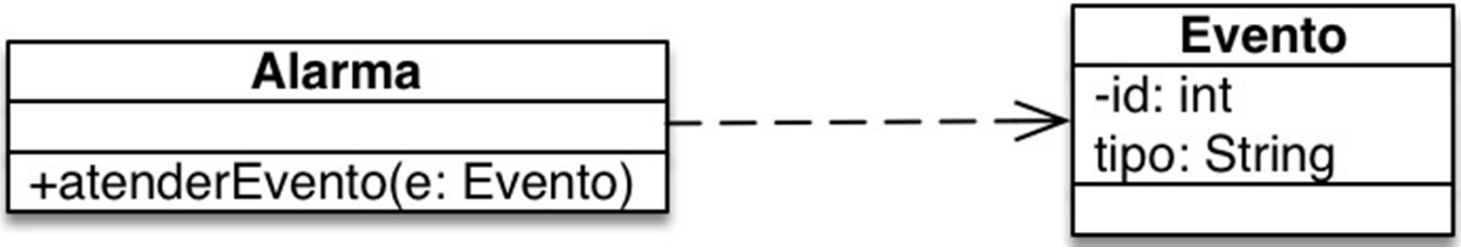
Relación de composición

- Es un tipo especial de relación de composición
- También representa el **“todo”** y la/s otra/s la/s **“partes”** pero el **tiempo de vida de las partes esta ligada al del todo**
  - Las partes se pueden crear después del todo pero cuando se destruye el todo también se destruyen las partes
- Se dibuja como una línea entre dos clases con un rombo **lleno** sobre la clase que representa el “todo”
- Puede incluir adornos



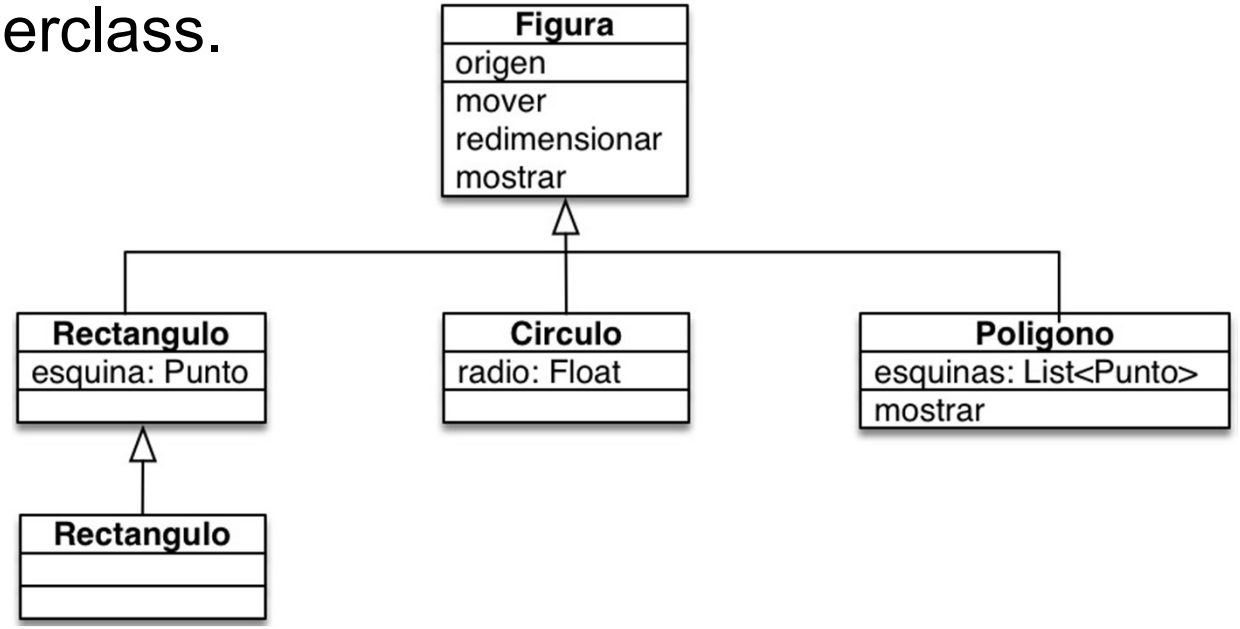
Relación de dependencia

- Es una relación para indicar que una clase usa a otra clase.
- Una clase A depende de otra clase B en alguno de los siguientes casos:
  - Un objeto de la clase A tiene un atributo cuyo tipo es B o que es una colección de objetos de tipo B.
  - Un objeto de la clase A envía un mensaje a un objeto de la clase B
  - Un objeto de la clase A crea un objeto de la clase B
  - Un objeto de la clase A recibe un mensaje con un objeto de la clase B como parámetro
- Usamos dependencias para modelar conexiones del tipo: “usa”
- Se dibuja como una linea punteada entre dos clases con una flecha que indica de que clase se depende



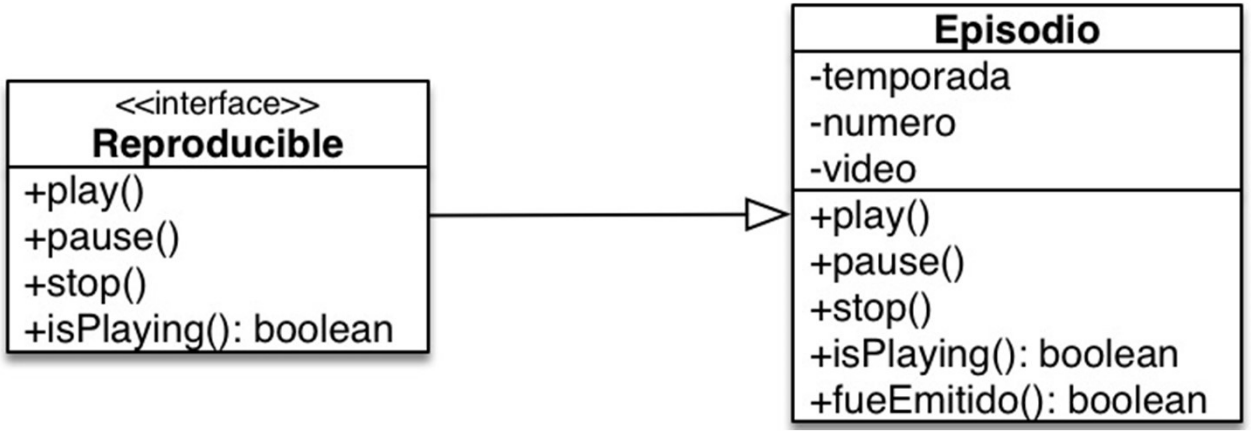
Relación de generalización

- Indica una relación entre una clase general (superclass) y un tipo más específico de esa clase (subclass).
- Usamos generalizaciones para modelar conexiones del tipo: “Es un tipo de ”
- Una clase A es del tipo de otra clase B en caso que:
  - La clase A es subclass de B
- Los atributos, operaciones y relaciones comunes se muestran en la superclass.



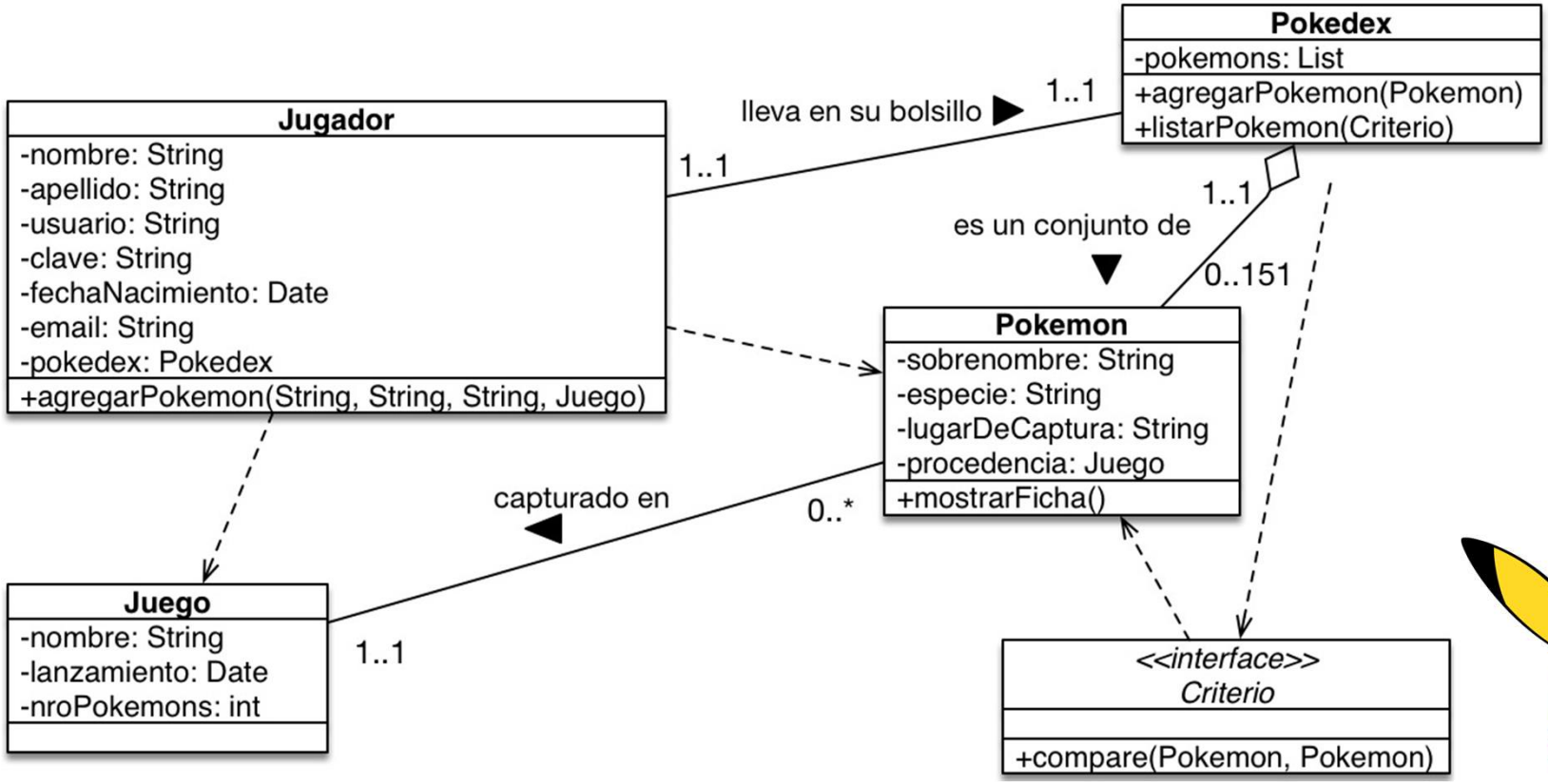
Relación de realización

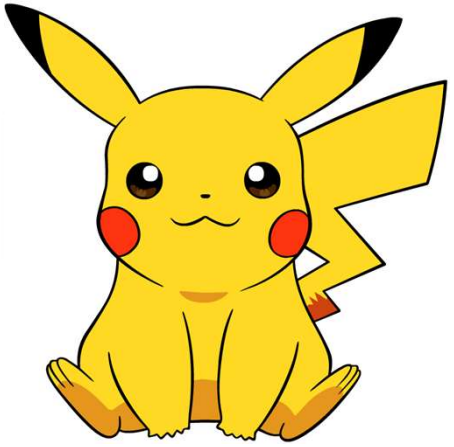
- Indica una relación donde una de las partes especifica un "contrato" y la otra parte garantiza llevarlo a cabo
- Es una mescla entre dependencia y generalización
- Usamos generalizaciones para modelar conexiones del tipo: "Implementa"
- Se usan principalmente para especificar la relación entre una interface y la clase que provee una operación para ella



Ejercicio 1.1

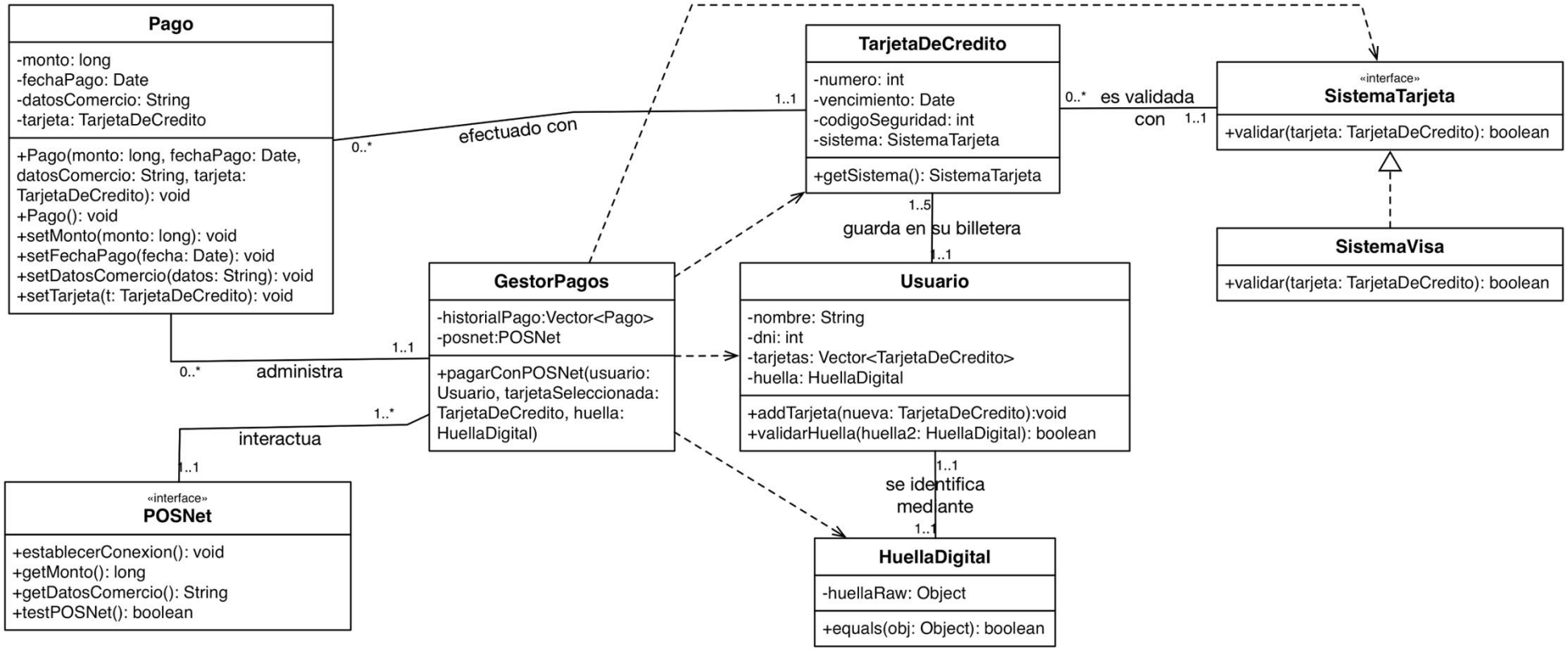
Implemente las clases y los métodos Java que se describen en el siguiente diagrama de clases (no es necesario implementar los cuerpos de los métodos)





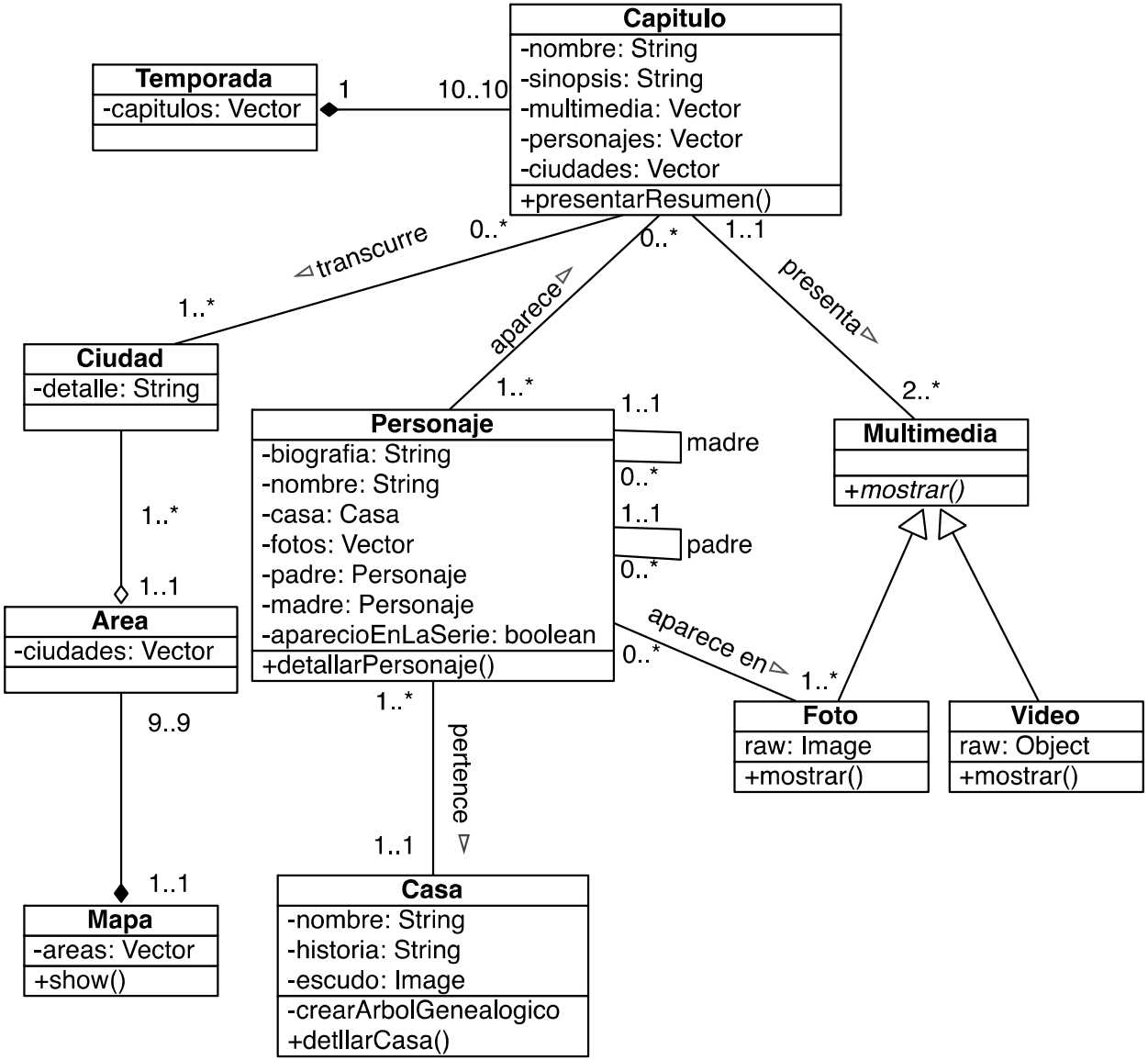
Ejercicio 1.2

Implemente las clases y los métodos Java que se describen en el siguiente diagrama de clases



Ejercicio 1.3

Implemente las clases y los métodos Java que se describen en el siguiente diagrama de clases



Ejercicio 1.4

Implemente las clases y los métodos Java que se describen en el siguiente diagrama de clases

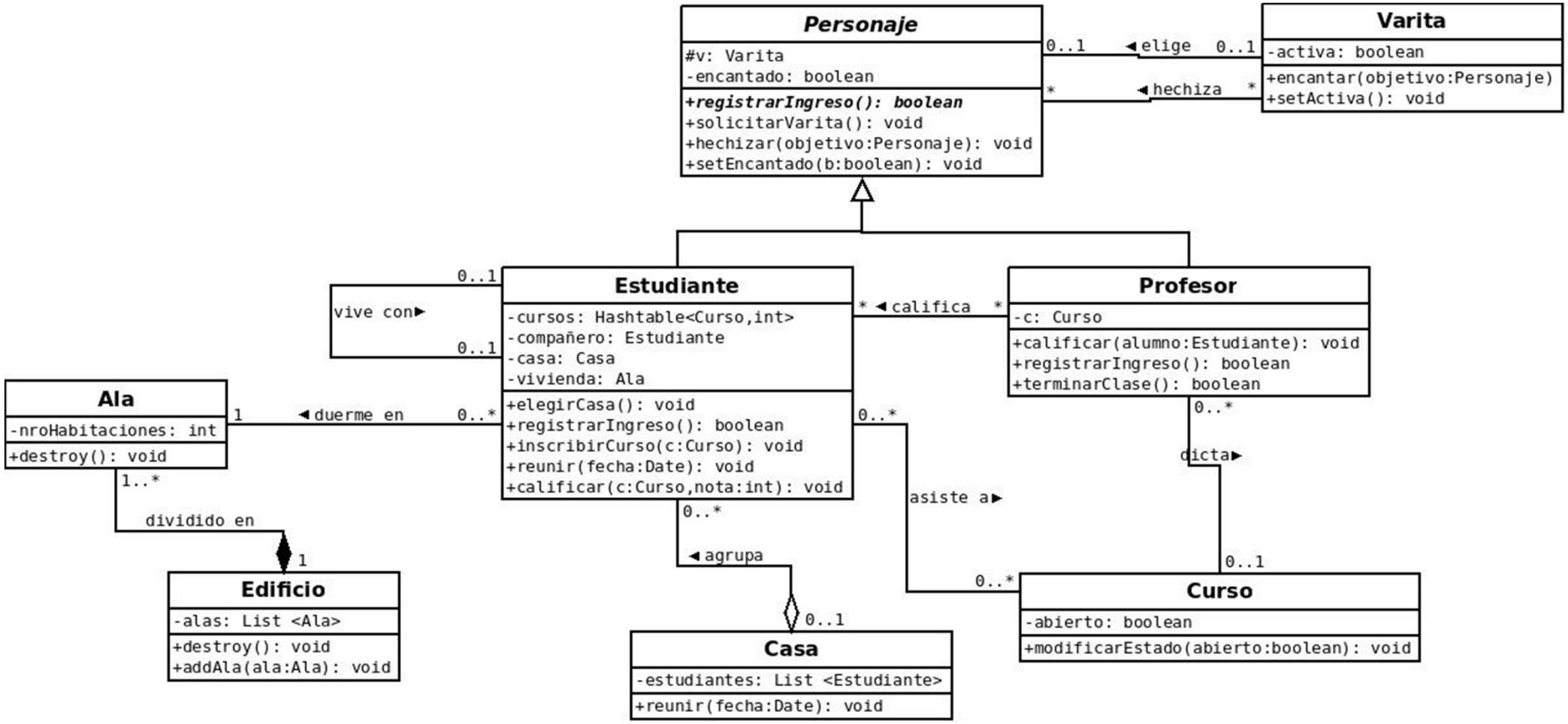
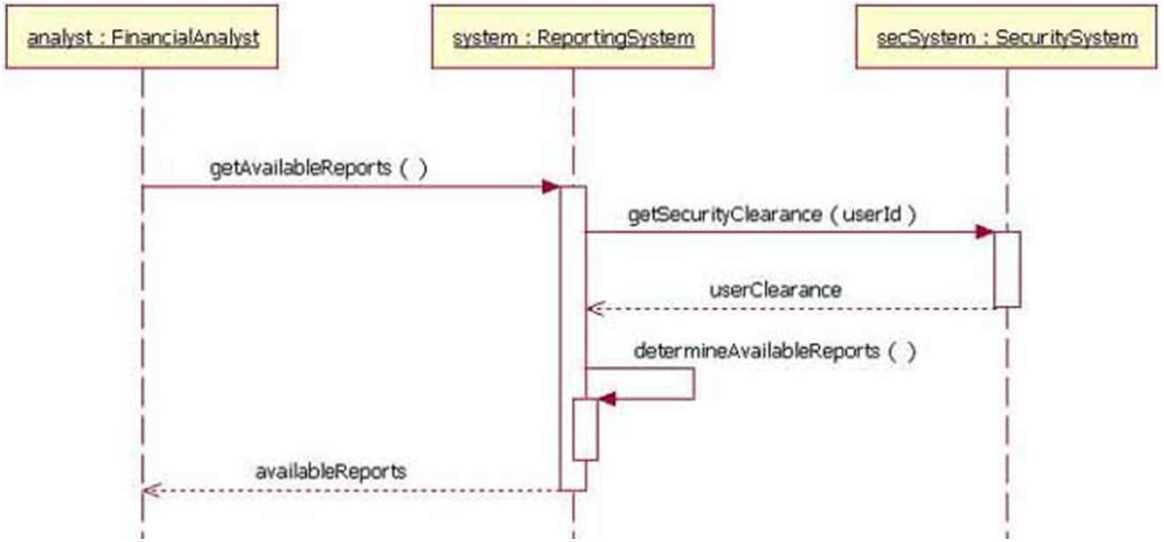


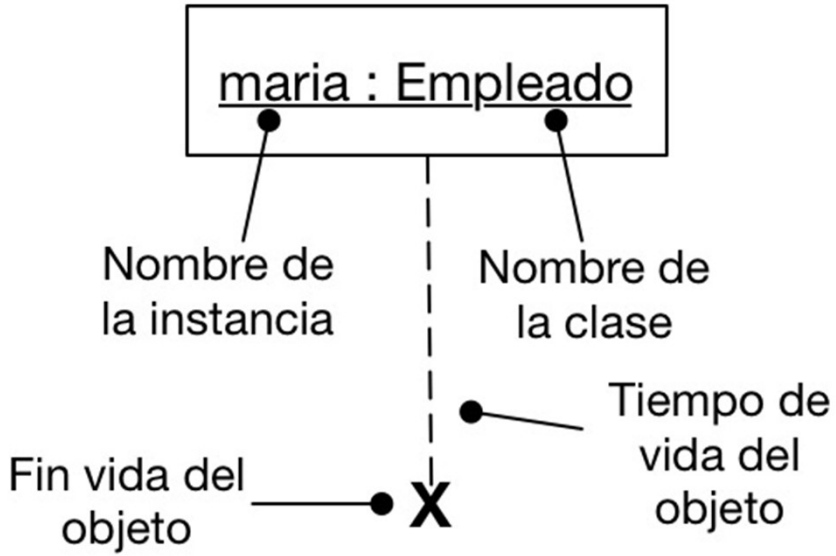
Diagrama de secuencia

- Se usan para modelar el flujo de control de una operación
- Muestran los mensajes intercambiados entre un conjunto de objetos para realizar una tarea especifica
- Hace énfasis en el orden en que se envían los mensajes
- Tiene 2 componentes principales:
  - Objetos
  - Mensajes



Objetos

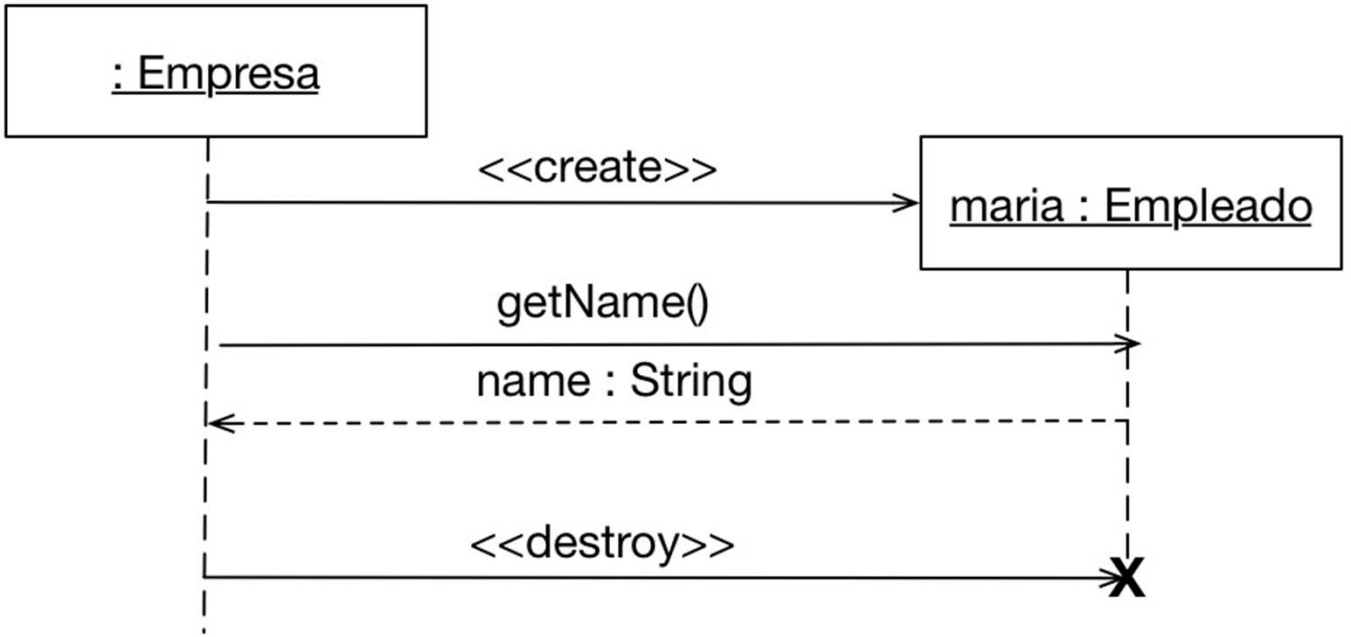
- Son instancias de las clases contenidas en un diagrama de clases
- Gráficamente el objeto se dibuja como un rectángulo con su nombre de instancia y clase subrayados



- La línea de vida indica el tiempo durante el que existe el objeto

Mensajes

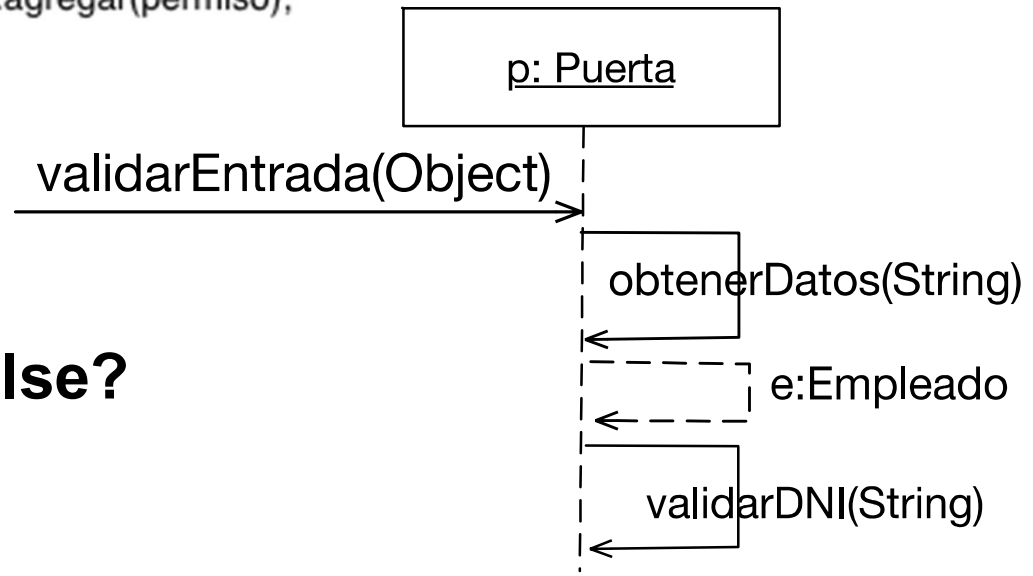
- Los mensajes son la especificación de la comunicación entre objetos
- Permiten modelar distintos tipos de acciones
  - Call: invocación de una operación de un objeto
  - Return: valor de retorno de una operación
  - Create: creación de un objeto
  - Destroy: destrucción de un objeto



Ejemplo diagrama de secuencia

Hagamos el diagrama de secuencia que inicia cuando un objeto **p** de la clase **Puerta** recibe el mensaje **validarEntrada** con un String como parámetro.

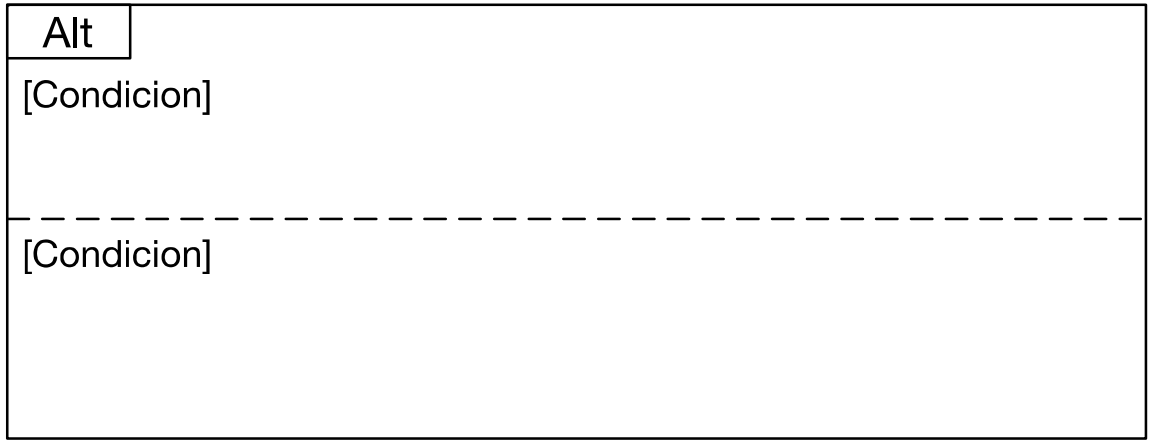
```
public class Puerta {

  public boolean validarEntrada(Object o) {
    String dni = (String) o;
    Empleado e = this.obtenerDatos(dni);
    boolean tienePermiso = this.validarDNI(dni);
    if(tienePermiso) {
      control.abrir(this);
    }
    else {
      this.solicitarPermTemporal(e);
      control.abrir(this);
    }
  }

  public boolean solicitarPermTemporal(Empleado e) {
    PermisoTemporal permiso = new PermisoTemporal();
    permiso.set(this);
    permiso.setDesde(Date.HOY);
    permiso.setHasta(Date.MAÑANA);
    permiso.setDescripcion("");
    e.agregar(permiso);
  }

}
```



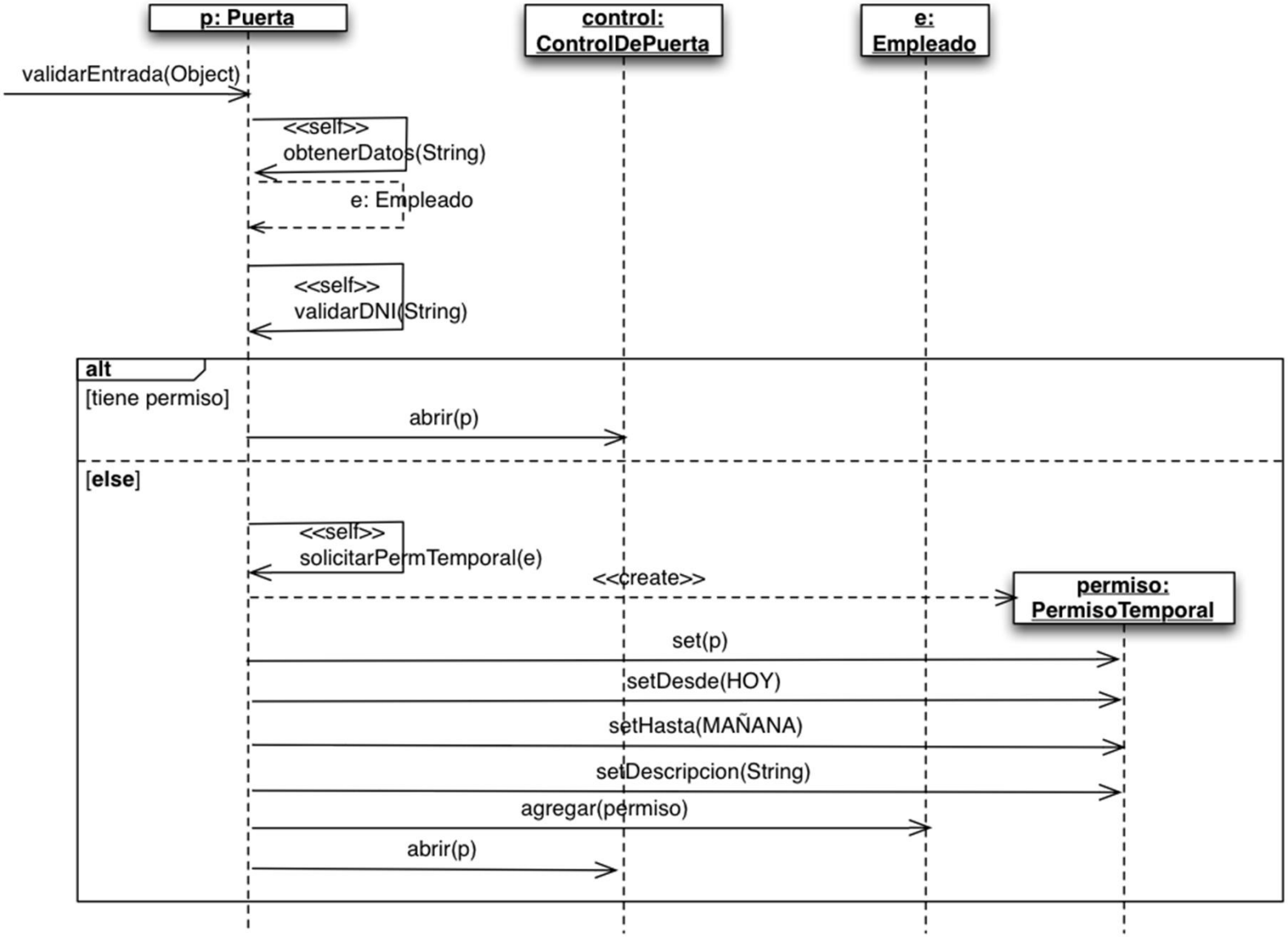
¿cómo modelamos el if/else?

Frames

- Se usan para representar if/else, while, for…
- Operadores
  - Alt:es para multiples fragmentos. Solo ejecuta el de la condición verdadera.
  - Opt: un único fragmento que solo ejecuta cuando la condición es verdadera.
  - Loop: es un fragmento que se ejecuta multiples veces.

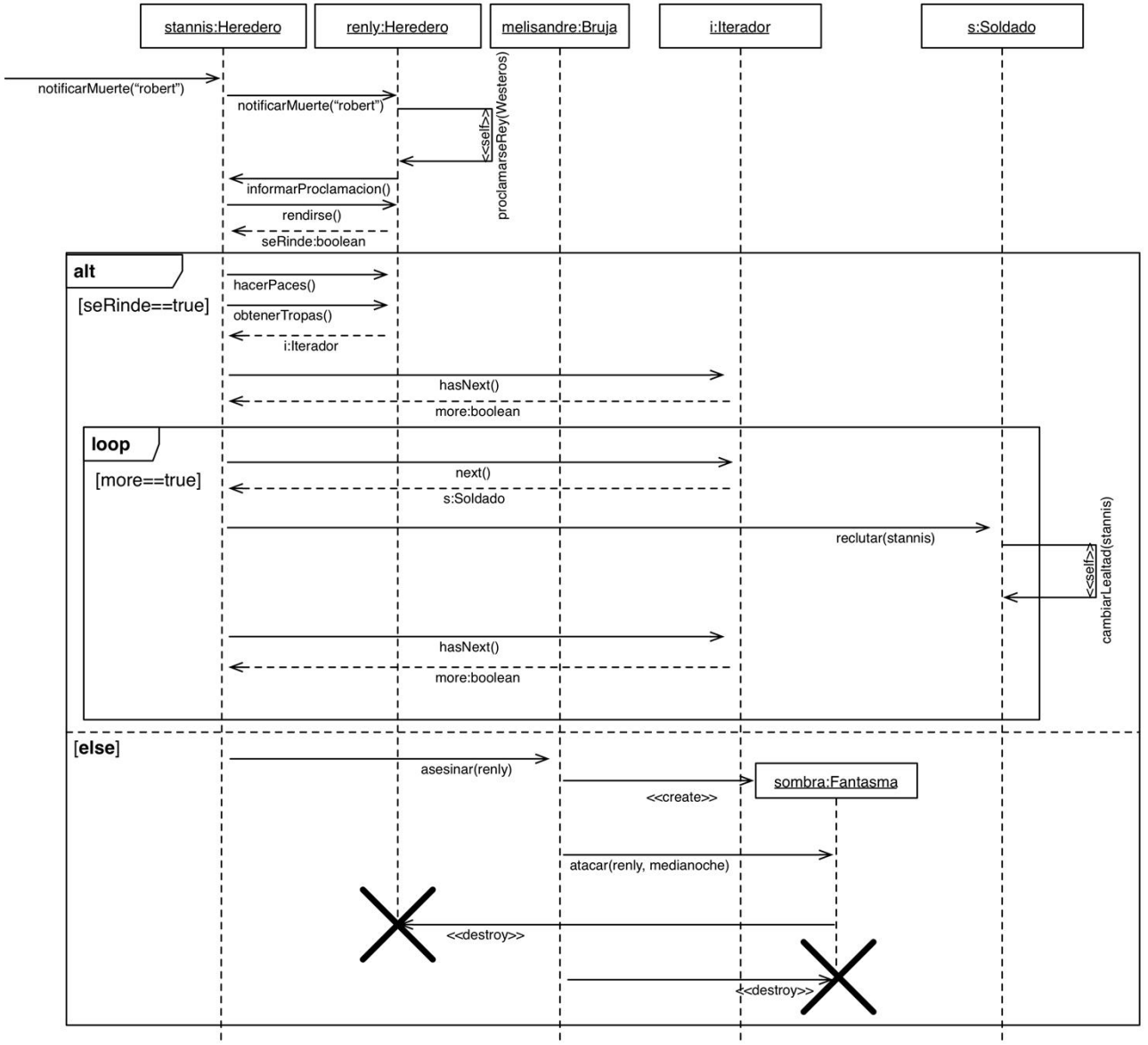


Solución ejemplo apertura de puerta

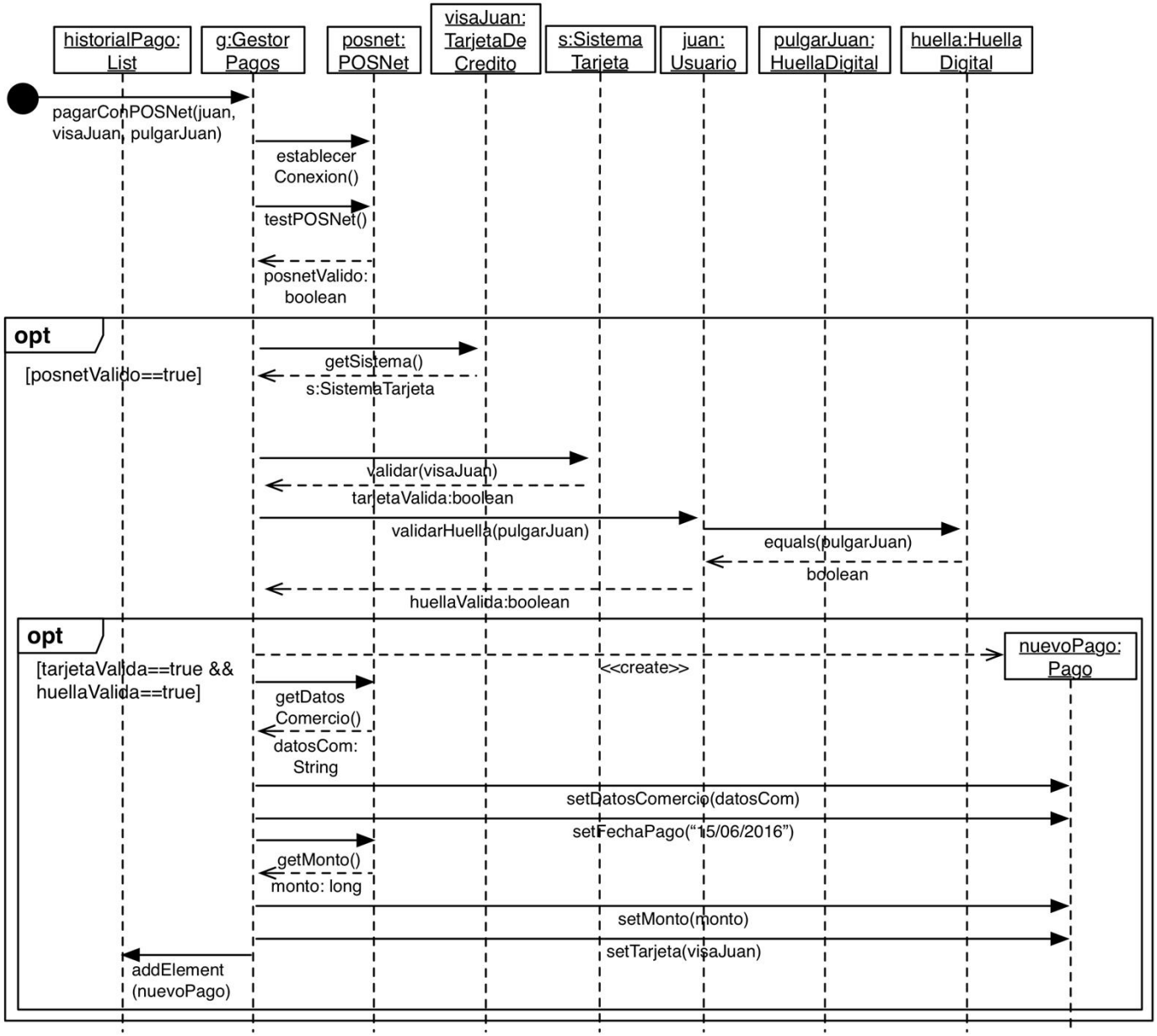


Ejercicio 2.1

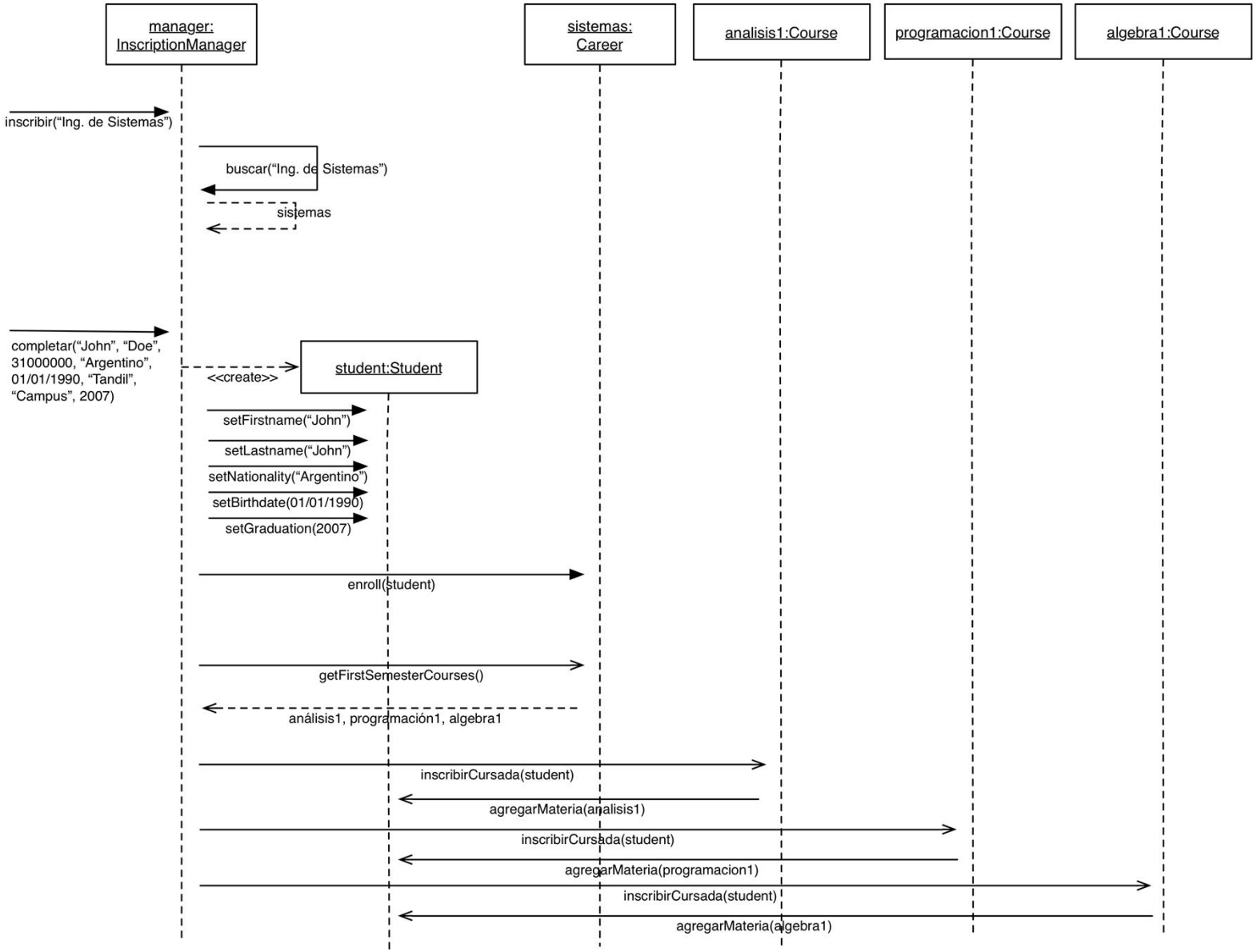
Implemente las clases, métodos y llamados a métodos Java que se describen en el siguiente diagrama de secuencias



Ejercicio 2.2



Ejercicio 2.3



Ejercicio 2.4