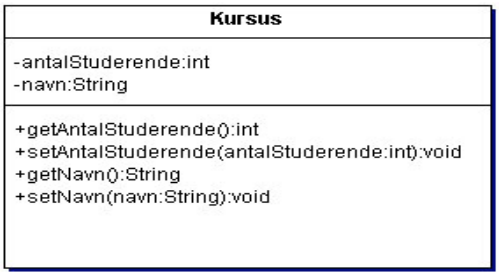
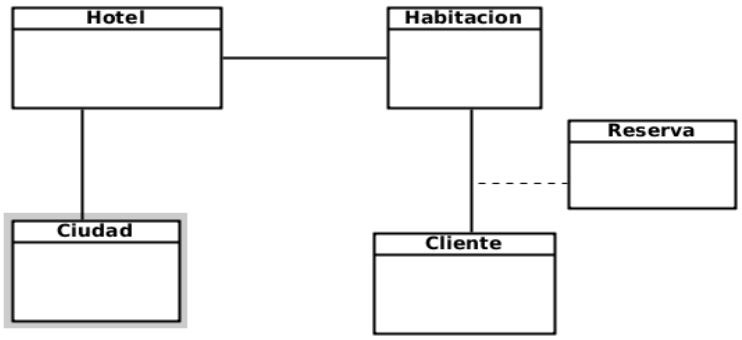
Relación 2

Ejercicio 1. UML es un lenguaje “universal”. Escribe el código Java y Ruby correspondiente a la declaración de la siguiente clase en danés (incluyendo la declaración de atributos y la cabecera de los métodos).

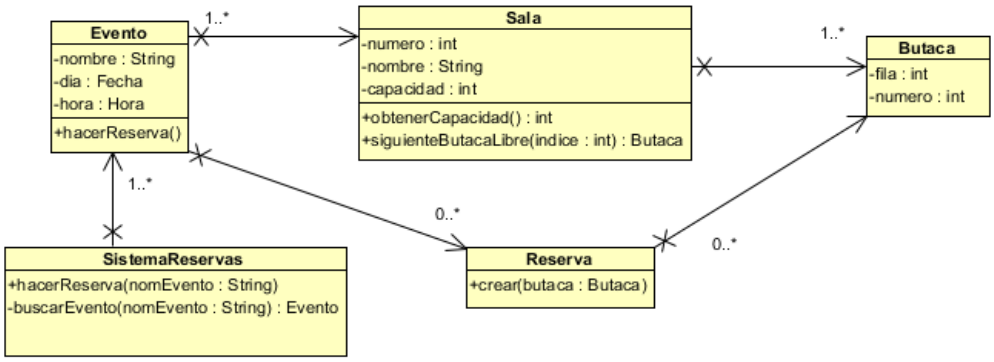


Ejercicio 2. El siguiente diagrama de clases representa hoteles con sus habitaciones y las reservas hechas por sus clientes:



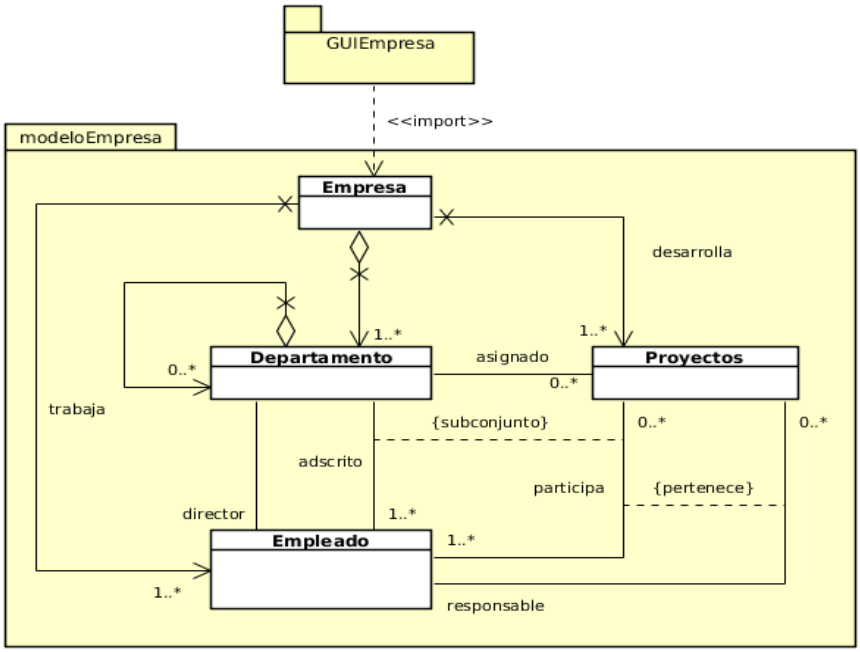
A) Complétalo con los atributos de las clases y el nombre, roles, multiplicidad y navegabilidad de las asociaciones, haciendo las suposiciones que consideres oportunas.
B) Implementa en Java el resultado obtenido.
C) Implementa en Ruby el resultado obtenido.

Ejercicio 3. A partir del siguiente diagrama de clases.



A) Indica el nombre o roles de las asociaciones.
B) ¿La asociación que existe entre Sala y Butaca podría ser una agregación?¿Y una composición?
C) Partiendo de una sala, ¿podría saberse para qué eventos está siendo usada?
D) Escribe el código Java correspondiente.
E) Escribe el código Ruby correspondiente.

Ejercicio 4. A partir del siguiente diagrama de clases, responde a las cuestiones plateadas:



A) La asociación que hay entre Empresa y Departamento, ¿es de tipo agregación o composición? ¿podría ser del otro tipo? Razona por qué.

B) ¿Cuál es la multiplicidad de dicha asociación en el lado de Empresa? ¿podría ser distinta? Razona por qué.

C) La asociación que hay entre Departamentos, ¿es de tipo agregación o composición? ¿podría ser del otro tipo? Razona por qué

D) ¿Qué son GUIEmpresa y modeloEmpresa?

E) La relación que hay entre GUIEmpresa y Empresa ¿de qué tipo es?, es decir ¿qué significa <<import>> sobre ella?

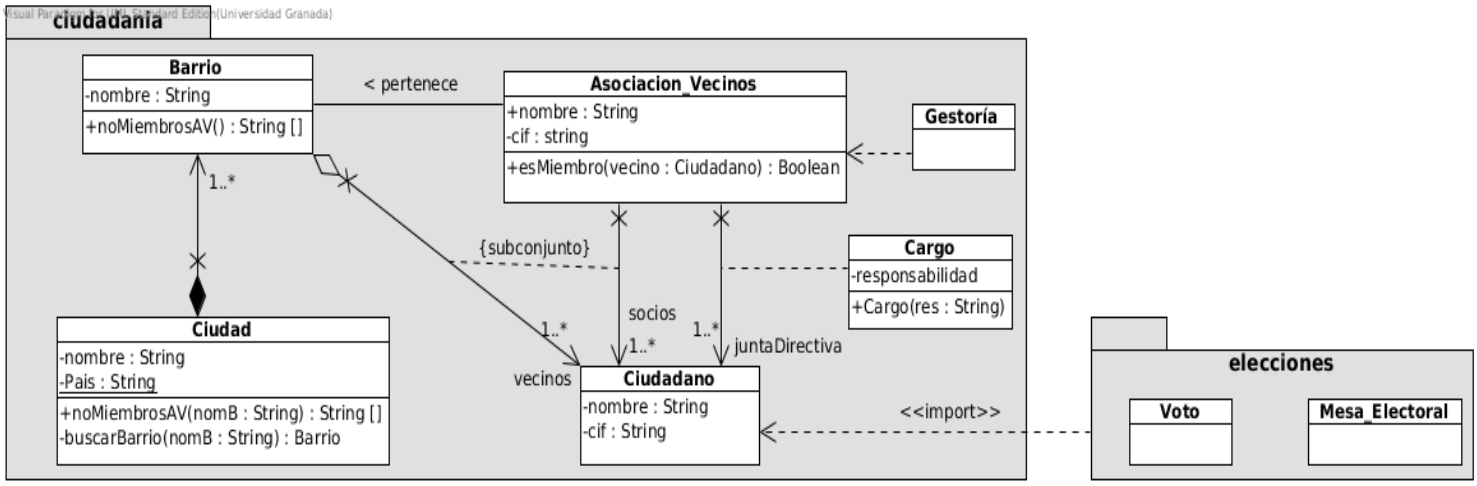
F) ¿Cuál es la navegabilidad de la asociación adscrito entre Empleado y Departamento? ¿y de la asociación desarrolla entre Empresa y Proyectos?

G) ¿Qué representa la línea discontinua entre las asociaciones adscrito y participa y cuál es su significado en este diagrama?

H) Hay dos asociaciones entre Proyectos y Empleado, ¿qué representan?

I) ¿Por qué hay 0..* y no 1..* en la multiplicidad de la asociación entre Empleado y Proyectos?

Ejercicio 5 (Examen 13/14). Partiendo del siguiente diagrama de clases de UML, resolver las cuestiones planteadas a continuación.



A) ¿Qué cambio habría que hacer en el diagrama de clases para modelar que un barrio puede tener varias asociaciones de vecinos?

B) ¿Desde un objeto de la clase Ciudadano puede saberse si es presidente de la asociación de vecinos de su barrio?¿Por qué?

C) ¿De qué tipo es la relación entre Ciudad y Barrio? ¿qué significa?

D) ¿De qué tipo es la relación entre Barrio y Ciudadano? ¿qué significa? ¿podría ser de otro tipo?

E) ¿Qué significa la línea discontinua con la indicación "{subconjunto}" entre la asociaciones de Barrio---Ciudadano y Asociacion_Vecinos---Ciudadano?

F) Si la junta directiva de una asociación de vecinos está formada por exactamente 6 ciudadanos, ¿cómo lo indicarías en el diagrama de clases?

G) ¿Qué significa que la clase Cargo esté ligada a la asociación entre Asociacion_Vecinos y Ciudadano?

H) Implementa en Ruby los consultores/modificadores básicos de la clase Asociación_Vecinos.

I) ¿Cómo debería ser el constructor de Ciudadano para que inicialice el estado completo de una instancia? Impleméntalo en Java y Ruby.

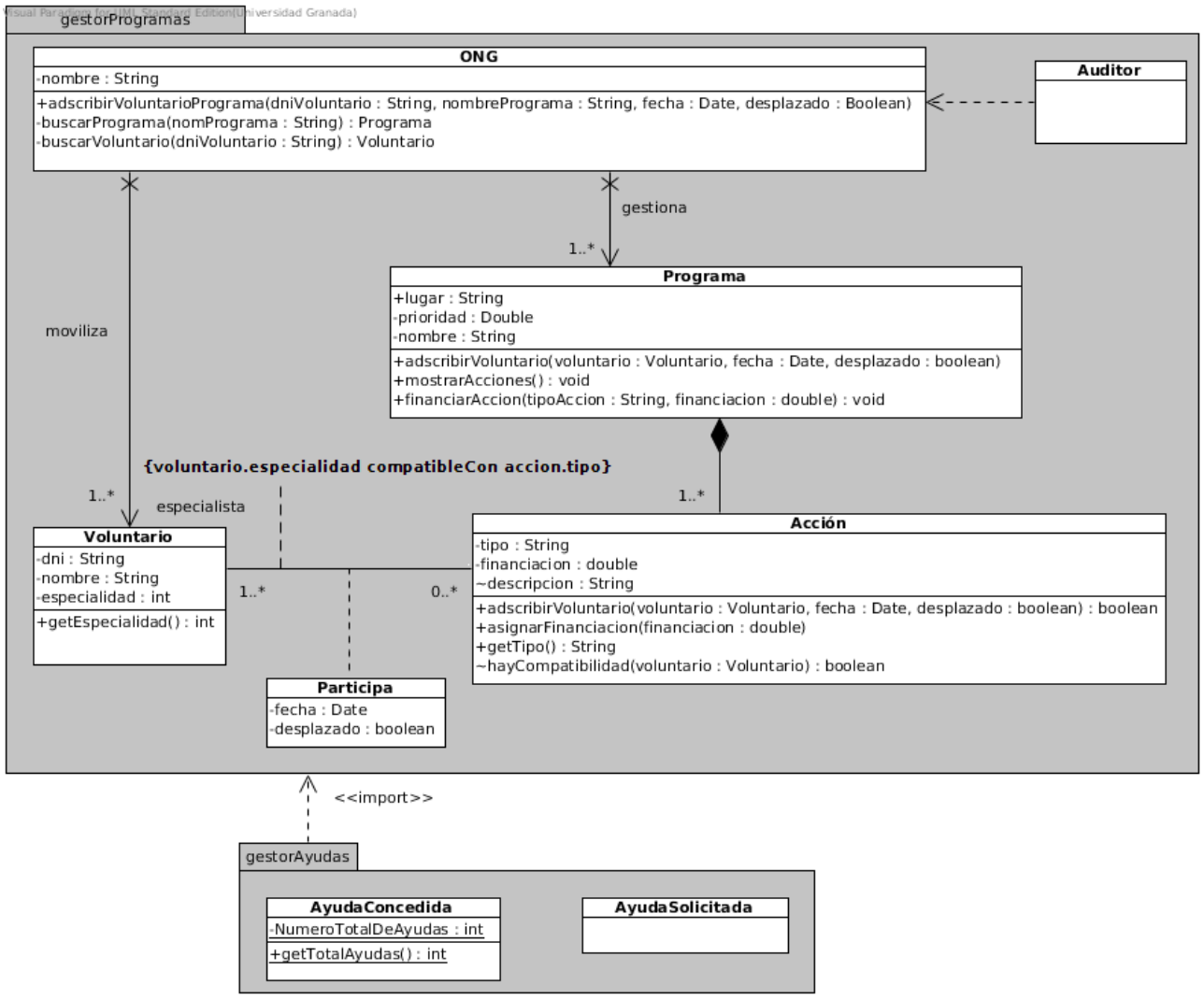
J) Suponiendo que la visibilidad de todas las clases del diagrama es pública y que estamos codificando la clase Voto en Java ¿A qué otras clases puede acceder ésta usando directamente su nombre, sin indicar el nombre del paquete al que pertenecen?

K) Define en Java los atributos de referencia de la clase Barrio.

L) ¿Desde qué clases es accesible el atributo nombre de la clase Asociacion_Vecinos?

M) Implementa en Java y Ruby las clases Asociacion_Vecinos y Mesa_Electoral completas (Ojo: no incluir nada que no esté en el diagrama de clases proporcionado).

Ejercicio 6 (Examen 12/13). Partiendo del siguiente diagrama de clases de UML, responde verdadero (V) o falso (F) a las cuestiones:



Desde la clase AyudaSolicitada se puede acceder a todos los elementos públicos del paquete GestorProgramas.	
Un voluntario puede participar en cualquier acción de un programa sin ningún tipo de restricción.	
El estado de un objeto de la clase Auditor viene determinado por el estado de un objeto de la clase ONG.	
Un voluntario podría participar en acciones de distintos programas.	
Un voluntario puede pertenecer a varias ONG.	
Cuando se define un objeto de la clase Acción, éste tiene que asociarse a un determinado objeto de la clase Programa.	

En una acción puede participar más de un voluntario como especialista.	
Desde un objeto de la clase ONG se puede llegar a conocer a todos los especialistas de una determinada acción en un programa.	
El estado de un objeto Voluntario está exclusivamente determinado por su dni, nombre y especialidad.	
Todos los métodos de la clase Acción pueden ser accedidos desde la clases AyudaConcedida.	

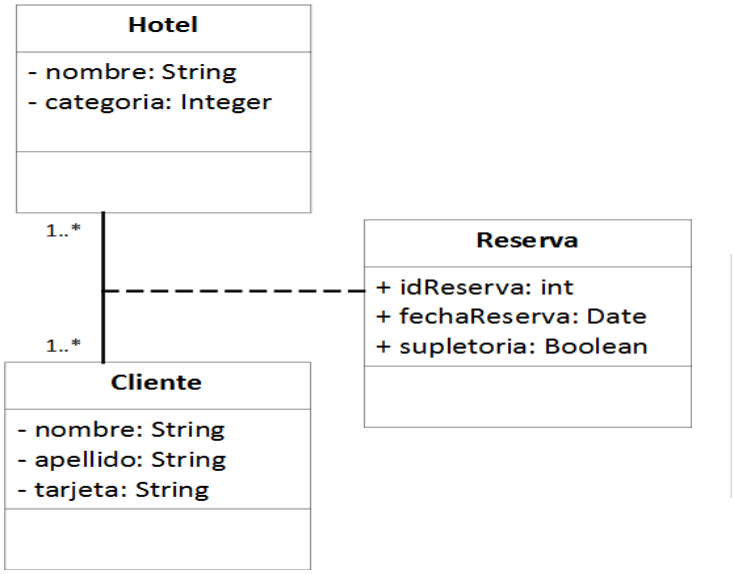
Ejercicio 7 (Examen 13/14). Partiendo del diagrama de clases del **ejercicio 6** responde a las siguientes preguntas:

A) Usando la siguiente nomenclatura: AS = Asociación, CO = Composición, AG = Agregación, DE = Dependencia, CA = Clase Asociación y RE = Restricción, etiqueta los elementos correspondientes en el propio diagrama de clases.

B) Implementa en Java y Ruby las clases Accion y AyudaConcedida.

Ejercicio 8. Partiendo del diagrama de clases del **ejercicio 6**, implementa en Java y en Ruby los atributos de referencia de todas las clases.

Ejercicio 9 (Examen 12/13). Cuando se implemente en Java el siguiente diagrama de clases, responde si las afirmaciones son verdaderas (V) o falsas (F)



Hotel tendrá dos atributos de referencia, uno relativo a las reservas y otro relativo a los clientes	

Reserva tendrá los atributos de referencia:
`private Hotel hotel;`
`private Cliente cliente;`	

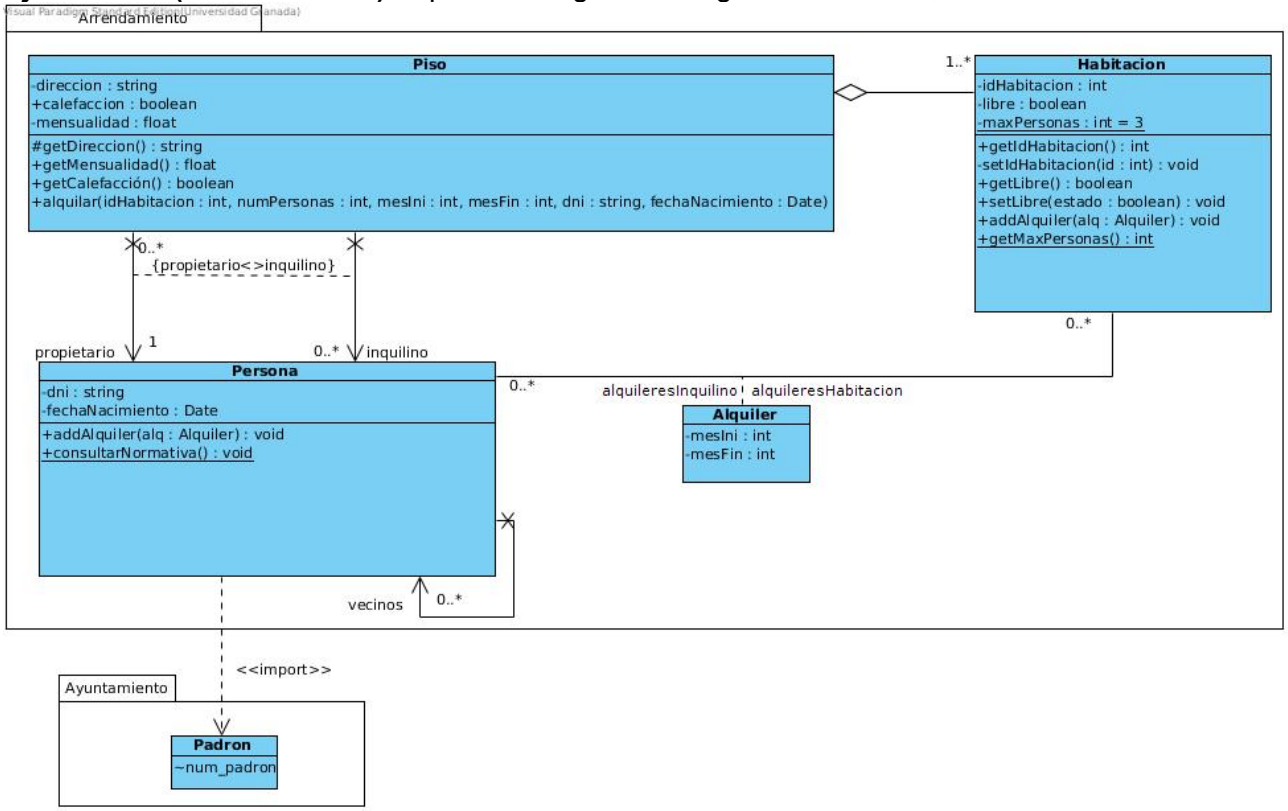
Ejercicio 10. Modela (obtén el diagrama de clases) de los siguientes supuestos, incluyendo en el modelo todo lo que conozcas del tema:

A) En un restaurante se dispone de un Gestor de Menú encargado de elaborar los menús para cada mes, cada menú está compuesto por tres platos: un primero, un segundo y un postre. A cada plato le corresponde una receta. Los ingredientes de la receta lo componen una lista de productos y las cantidades necesarias de cada producto. Para elaborar un menú para un día concreto hay que proporcionar un primero, un segundo y un postre.

B) En una carrera de relevos se inscriben equipos de atletas, cada equipo lo forman 4 deportistas, cada uno corre una distancia determinada tras la cual entrega el testigo al siguiente corredor del equipo. Para participar hay que inscribirse en la carrera. Una vez finalizada, reciben medalla (oro, plata y bronce) los tres equipos que primero lleguen a la meta, también se proporciona medalla (oro, plata y bronce) a los corredores que hayan obtenido los tres mejores tiempos.

C) Un alumno está matriculado en una serie de asignaturas de una determinada titulación, las asignaturas están formadas por grupos, de tal forma que cada alumno asiste a clase en un determinado grupo. Al comienzo del curso se le asigna a cada alumno un grupo de cada asignatura en la que esté matriculado.

Ejercicio 11 (Examen 15/16). A partir del siguiente diagrama de clases,



Responde verdadero (V) o falso (F) a las siguientes cuestiones:

La asociación *Piso-Habitación* es de agregación	
{propietario<>inquilino} es una restricción	
Una persona puede saber de qué pisos es propietaria	
Alquiler es una clase asociación	
vecinos es un rol	
Puede haber una habitación sin inquilinos	
Piso tiene un atributo de referencia de la clase *Persona*	

El método *getDireccion* de *Piso* es método de clase, no de instancia	
Desde la clase *Piso* se puede acceder directamente al atributo *dni* de un inquilino	
Desde la clase *Persona* se puede acceder directamente al atributo *num_padron* de un objeto de la clase *Padron*	
Los siguientes objetos de Piso tienen la misma identidad: *Piso piso1= new Piso("Calle Alta", true, 600)* *Piso piso2= new Piso("Calle Alta", true, 600)*	

Ejercicio 12. Escribe el código necesario para hacer que piso1 y piso2 sean idénticos.

Ejercicio 13. ¿Desde la clase Padrón se puede acceder al método getCalefacción de Piso? Justifica tu respuesta.

Ejercicio 14. Indica si es correcto implementar en Ruby para la clase Padrón lo siguiente:

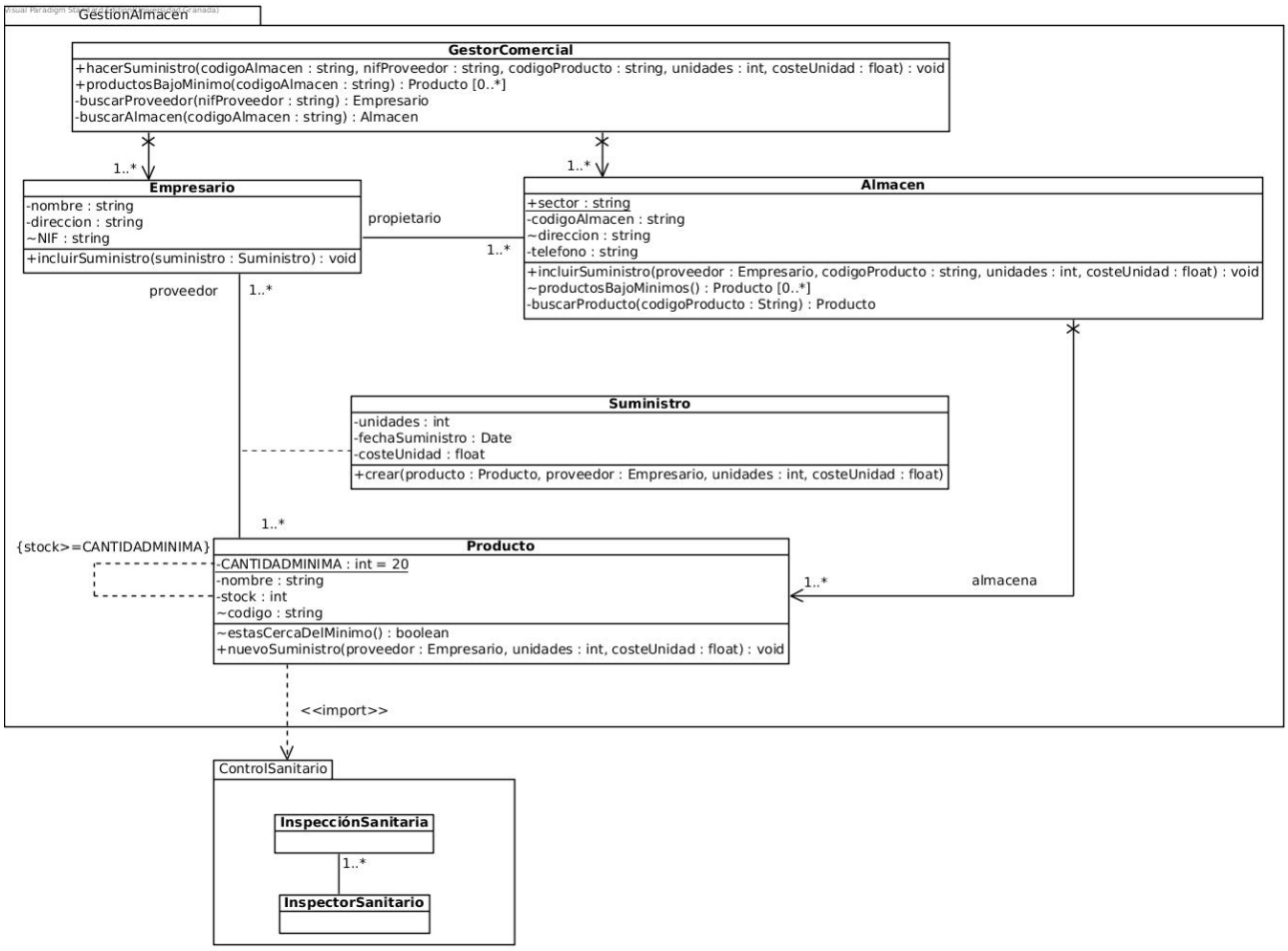
a) *attr_reader :dirección, :mensualidad*
b) *attr_accessor :calefaccion*

Justifica tu respuesta.

Ejercicio 15. Teniendo en cuenta sólo la información presentada en el diagrama de clases del **ejercicio 11**, implementa en Java y Ruby las clases *Persona, Alquiler y Padrón*. Añade sus constructores para inicializar los atributos y métodos consultores para todos los atributos. No te olvides de incluir también el código necesario para indicar que cada clase está dentro de un paquete o módulo, y si precisa algo de otro paquete o módulo. Supón que en Ruby cada clase se implementa dentro de un fichero con el mismo nombre que la clase y que están todas en la misma carpeta. Dado que no tenemos información sobre los métodos de Persona, implementa solo sus cabeceras.

Relación de problemas (PDOO) **Curso: 2023-24**

Ejercicio 16 (Examen 16/17). Dado el siguiente diagrama de clases:



1) Dibuja cómo representarías en UML la relación entre la clase **Almacen** y una nueva clase **Nave**, sabiendo que un almacén está compuesto por varias naves.

2) Responde verdadero (V) o falso (F).

Si la clase **InspeccionSanitaria** tuviese atributos privados, éstos serían accesibles desde la clase **Producto** puesto que hemos hecho el import del paquete.	
El **stock** de un **Producto** puede ser de menos de 20 unidades.	
Un mismo **Producto** lo pueden suministrar varios **Empresarios**.	
El atributo **sector** de la clase **Almacen** es accesible desde los métodos de la clase **Producto**.	

3) Teniendo en cuenta sólo la información presentada en el diagrama de clases, implementa en Java la clase **Producto** y en Ruby la clase **Almacen**. Añade e implementa sus constructores para inicializar los atributos. Cuando estos atributos sean colecciones, inicialízalos con la lista vacía. Incluye los métodos consultores (no los modificadores) para todos los atributos.