Universidad de Los Andes
Escuela de Ingeniería de Sistemas
Departamento de Computación

# PROGRAMACIÓN 2

## Clase 04

Junior Altamiranda
altamira@ula.ve

# AGENDA

- Tipos de Datos Abstractos (TDA)
- Programación Orientada a Objetos (POO)
  - Clases, objetos (Instancias de objetos).
  - Estado (Atributos), comportamiento (Métodos), identidad.
  - Encapsulamiento, interfaz.
- Ejemplos.

# Tipos de Datos Abstractos

Un **tipo de dato abstracto** es una interpretación, que describe un conjunto de objetos en términos de una estructura de datos encapsulada u oculta y las operaciones sobre esta estructura.

Son tipos de datos definidos por el programador

Corresponden a un conjunto tipificable

Es abstracto porque interpretamos al objeto real a través de una cantidad finita de **propiedades** (Capturando la esencia o parte que nos interesa)

Al igual que en el caso de los tipos de datos básicos del lenguaje; tienen un conjunto de **operaciones** asociadas.

# Tipos de Datos Abstractos

Exporta un **conjunto de operaciones**. Este conjunto es llamado **interfaz**.
Las operaciones de la interfaz son el único y exclusivo mecanismo de acceso a la estructura de datos del TDA.

**Tipos de Operaciones**:
**Constructor** (Crea una instancia del TDA)
**Destructor** (Libera la memoria ocupada por la instancia)
**Lectura** (Permiten leer el valor de las propiedades)
**Asignación** (Permiten asignar valores a las propiedades)
**Transformación** o cálculo

**Encapsulamiento**: Principio de esconder la estructura de los datos usada y solamente proveer una interfaz bien definida

# Programación Orientada a Objetos

- Paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computador
- Los objetos son entidades que combinan *estado*, *comportamiento* e identidad:

  El **estado** está compuesto de datos, serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).

  El **comportamiento** está definido por los procedimientos o **métodos** con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.

  La **identidad** es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador

- Componentes: Clase, Objeto, Propiedades o atributos, Métodos, Estado, Identidad

# Clases y Objetos

**Clase:** Colección o plantilla de objetos que comparten los mismos:

Atributos: Representan un tipo de característica que se desea conocer o guardar sobre la clase.

Métodos: Operaciones (Funciones) que se pueden realizar a través de las instancias de la clase.

Relaciones

Semántica

**Objeto:** Instancia de una clase

Representa a un ente real.

Se caracteriza por tener:

- Identidad: Propiedades que permiten identificar (de forma única) al objeto.
- Estado: Definido por el valor de sus propiedades.
- Comportamiento: Definido por sus métodos.

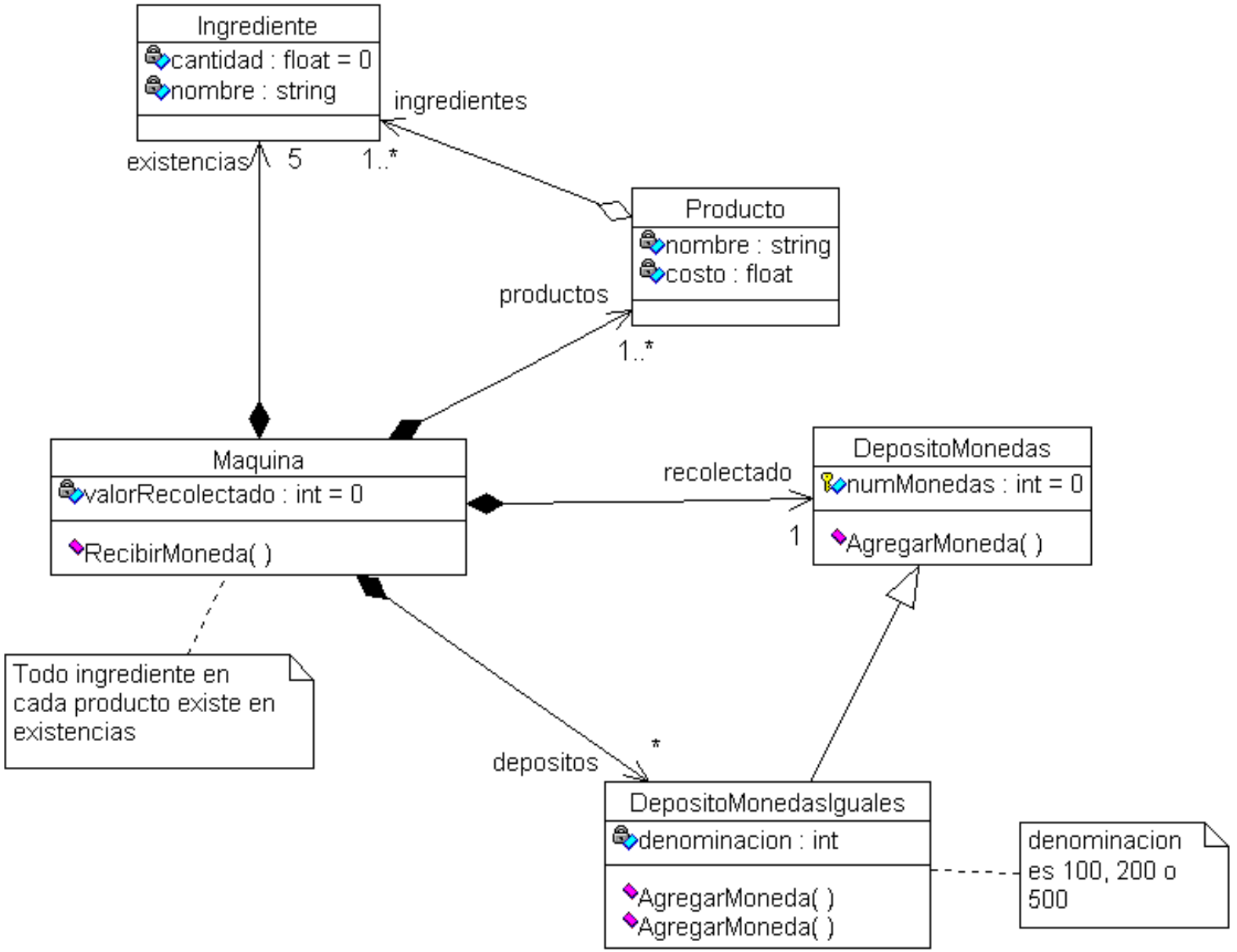
# Diagramas de Clases

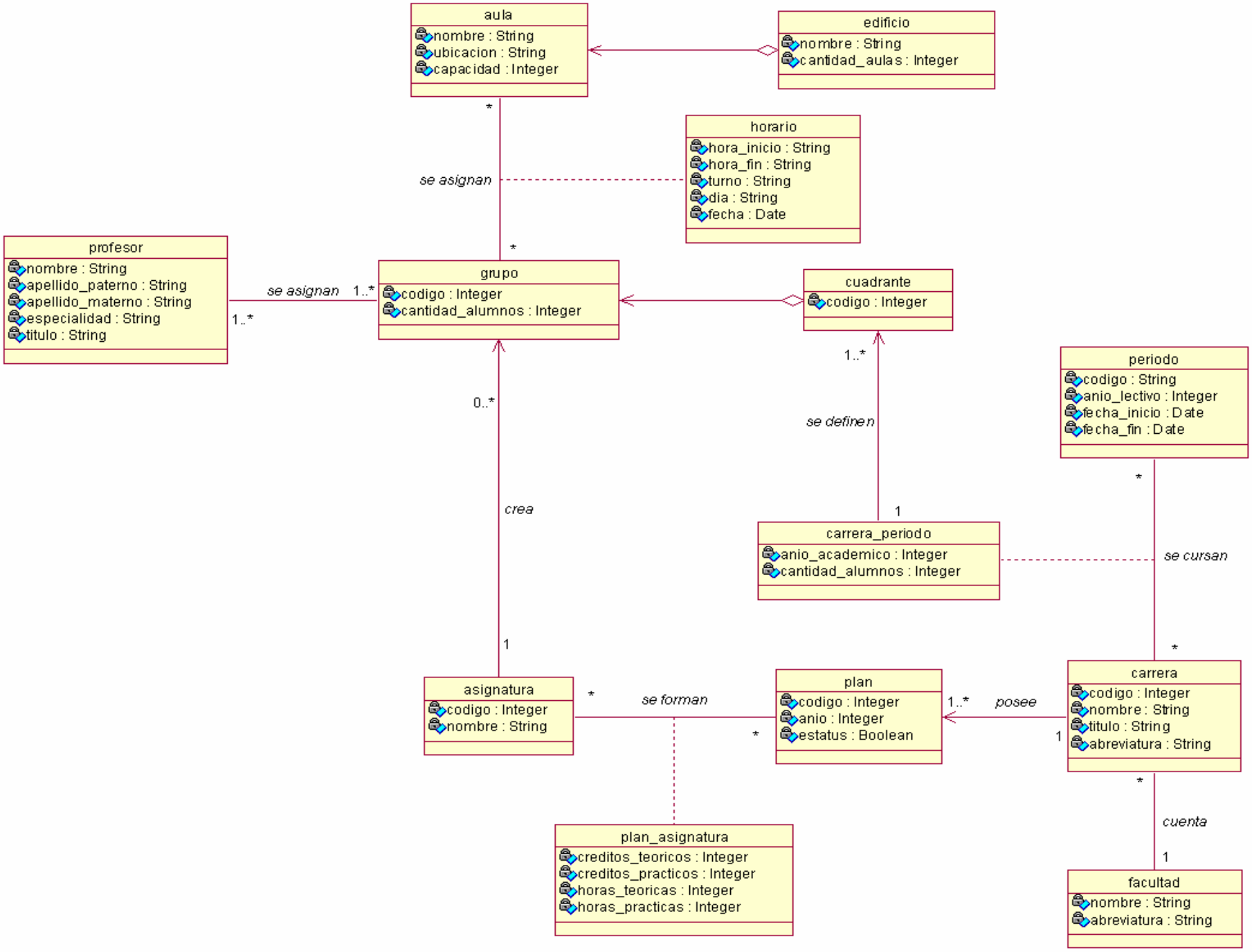
Diagrama estático que describe la estructura de un sistema, mostrando los atributos y métodos de las clases y las relaciones entre estas.
Una clase es representada a través de un rectángulo, con tres divisiones internas:

- Nombre de la clase
- Propiedades o atributos
- Métodos

Son utilizados durante el proceso de análisis y diseño de un sistema.

Veamos algunos ejemplos de diagramas de clases…


Ingrediente
cantidad : float = 0
nombre : string
ingredientes
existencias
5
1..*
Producto
nombre : string
costo : float
productos
1..*
Maquina
valorRecolectado : int = 0
RecibirMoneda( )
recolectado
DepositoMonedas
numMonedas : int = 0
AgregarMoneda( )
1
Todo ingrediente en cada producto existe en existencias
depositos
*
DepositoMonedasIguales
denominacion : int
AgregarMoneda( )
AgregarMoneda( )
denominacion es 100, 200 o 500


aula
nombre : String
ubicacion : String
capacidad : Integer
edificio
nombre : String
cantidad_aulas : Integer
horario
hora_inicio : String
hora_fin : String
turno : String
dia : String
fecha : Date
se asignan
profesor
nombre : String
apellido_paterno : String
apellido_materno : String
especialidad : String
titulo : String
se asignan
grupo
codigo : Integer
cantidad_alumnos : Integer
cuadrante
codigo : Integer
periodo
codigo : String
anio_lectivo : Integer
fecha_inicio : Date
fecha_fin : Date
se definen
crea
carrera_periodo
anio_academico : Integer
cantidad_alumnos : Integer
se cursan
asignatura
codigo : Integer
nombre : String
se forman
plan
codigo : Integer
anio : Integer
estatus : Boolean
posee
carrera
codigo : Integer
nombre : String
titulo : String
abreviatura : String
cuenta
plan_asignatura
creditos_teoricos : Integer
creditos_practicos : Integer
horas_teoricas : Integer
horas_practicas : Integer
facultad
nombre : String
abreviatura : String

# Diagrama de Clases

- Visibilidad de propiedades y métodos:

  + Público: Visible a todas las clases
  # Protegido: Visible a la clase y a subclases
  - Privado: Visible sólo a la clase

# Asociaciones

- Una *asociación* representa la relación entre dos o más clases.

  Una *asociación binaria* es una relación entre dos clases.

  Existe una asociación binaria si un objeto de una clase requiere un objeto de otra clase para hacer su trabajo

# Asociaciones

- Una asociación binaria está representada por una línea sólida que conecta dos clases


**Figura 1** *Asociación entre las clases que representan estudiantes y profesores*

# Asociaciones

- Una *asociación de una vía* indica la dirección en la que se puede navegar de un objeto de una clase a un objeto de otra clase.

  Una *asociación de dos vías* indica una navegación bidireccional entre objetos de dos clases.

# Asociación una vía

- Una *asociación de una vía se indica con una flecha al final de la línea de asociación.*

  El atributo de la primera clase que contiene una referencia a un objeto de la segunda clase también está escrito al final de la línea.

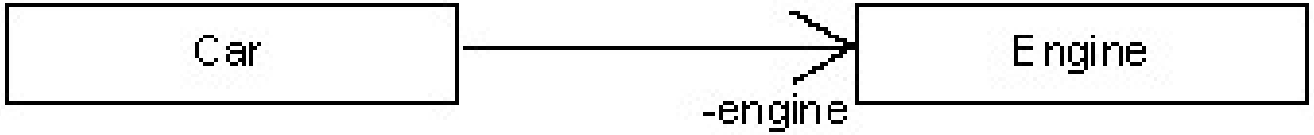



**Figura 3** *Asociación de una vía entre las clases que representan carros y motores*

# Asociación una vía

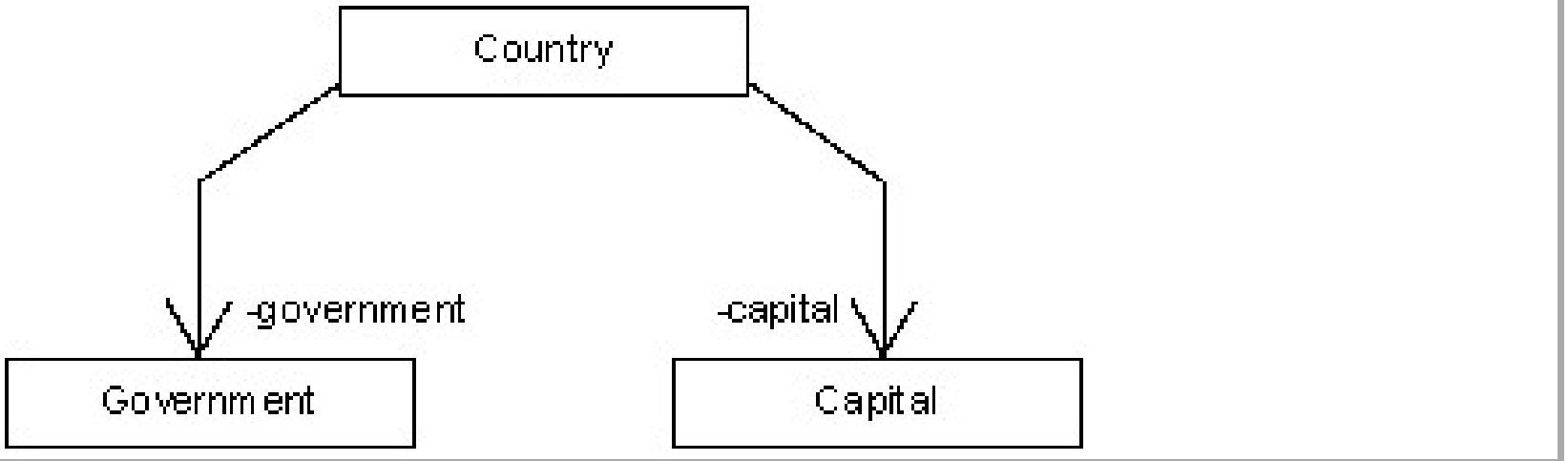



**Figura 4** *Asociaciones de una vía entre las clases que representan países, gobiernos y capitales*

El ejemplo muestra la relación entre las clases Country (país), Government (gobierno) y Capital. Cada país tiene un gobierno y una capital.

# Asociación una vía

- Una clase puede contener una o más asociaciones con otra clase.
  Por ejemplo, el siguiente diagrama de clase muestra dos asociaciones entre las clases Flight (*vuelo*) y Pilot (*piloto*), una asociación con el atributo pilot(*piloto*) y otra con el atributo coPilot (*copiloto*):

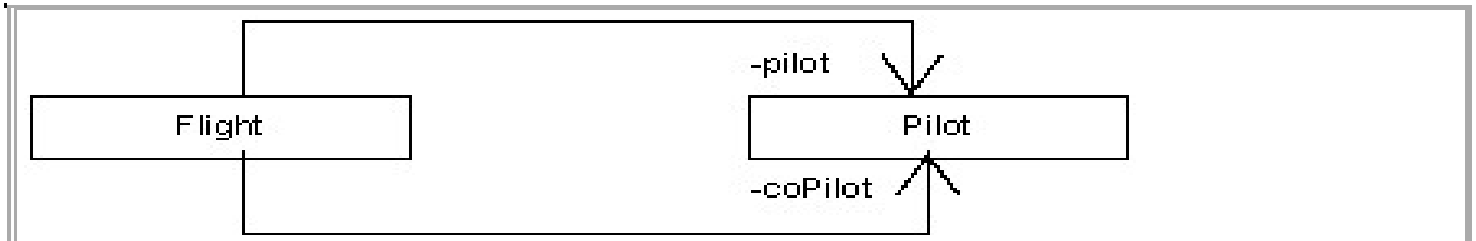



**Figura 5** *Dos asociaciones entre las clases que representan vuelos y pilotos*

# Multiplicidad

La *multiplicidad* indica el número de instancias de una clase que pueden ser asociadas a una sola instancia de otra clase.

La multiplicidad puede especificarse con un solo entero o como un rango *n..m*, donde *n* es el límite inferior y *m* es el límite superior. Podemos utilizar un asterisco ( * ) para denotar que no existe un límite superior

# Multiplicidad

.

| | |
|---|---|
| *0..1* | Cero o una instancia |
| *0..** ó * | Cero o más instancias |
| 1 | Exactamente una instancia |
| 1..* | Una o más instancias |
| **Tabla 1** *Multiplicidades comunes* | |

Las asociaciones pueden clasificarse de acuerdo a su multiplicidad.

En este curso discutiremos tres tipos: uno a uno, uno a muchos y muchos a muchos

# Asociación uno a muchos

- En una *asociación uno a muchos* entre las clases A y B, una instancia de la clase A puede estar relacionada con muchas instancias de la clase B, pero una instancia de la clase B está relacionada solamente con una instancia de la clase A.


**Figura 8** *Asociación uno a muchos entre las clases que representan clientes y cuentas bancarias*

# Asociación uno a muchos

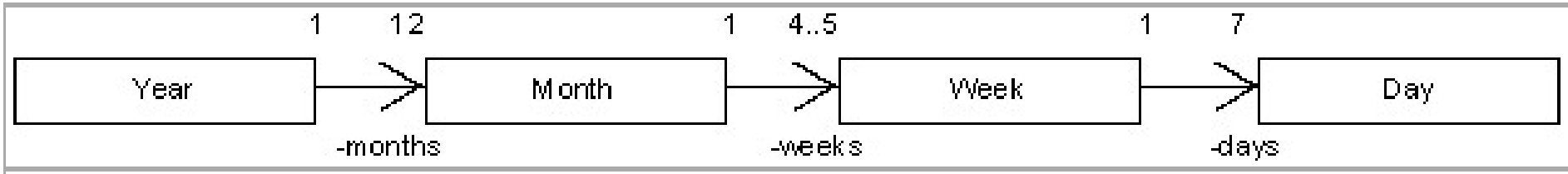



**Figura 9** *Asociación entre las clases de un calendario*

En este diagrama, un año (Year) contiene doce meses (Month) y cada mes está asociado con solo un año.

De la misma forma, un mes contiene cuatro o cinco semanas (Week) y cada semana está asociada con solo un mes.

Finalmente, una semana contiene siete días (Day) y cada día está asociado con una sola semana.

# Asociación muchos a muchos

.

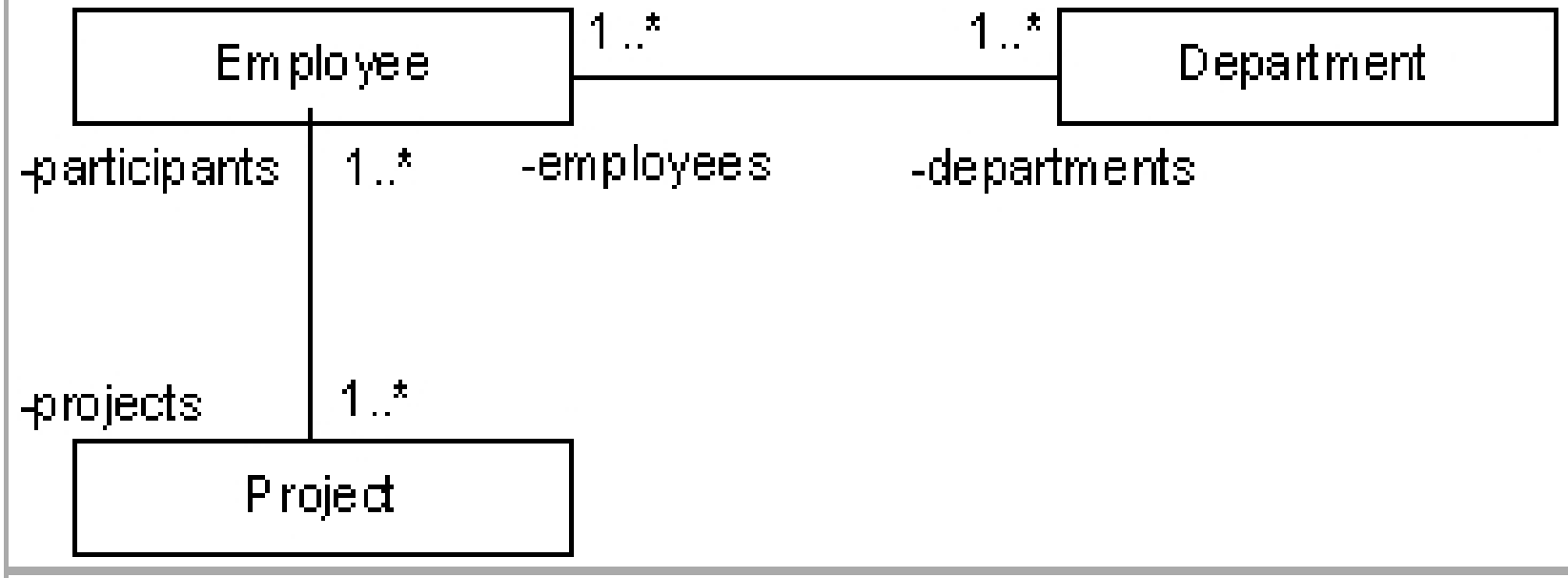



**Figura 11** *Asociación entre las clases que representan empleados, departamentos y proyectos*

# Agregación

La *agregación* es una forma especial de asociación.

Una agregación es una asociación entre las clases A y B, donde cada instancia de la clase A contiene, o está compuesta por, instancias de la clase B

# Agregación

- En este sentido, una instancia de la clase B es parte de una instancia de la clase A.

  A la instancia de la clase A se le conoce como *agregada (aggregate)* y a la instancia de la clase B se le conoce como *componente (component)*.

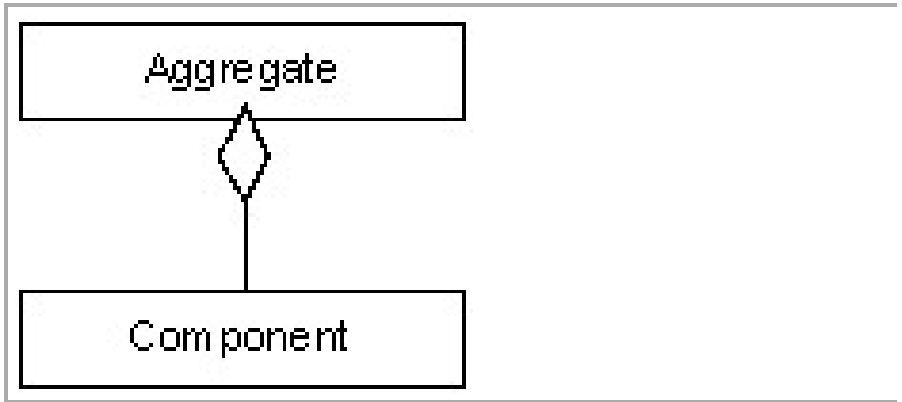



**Figura 12** *Notación de agregación*

# Agregación

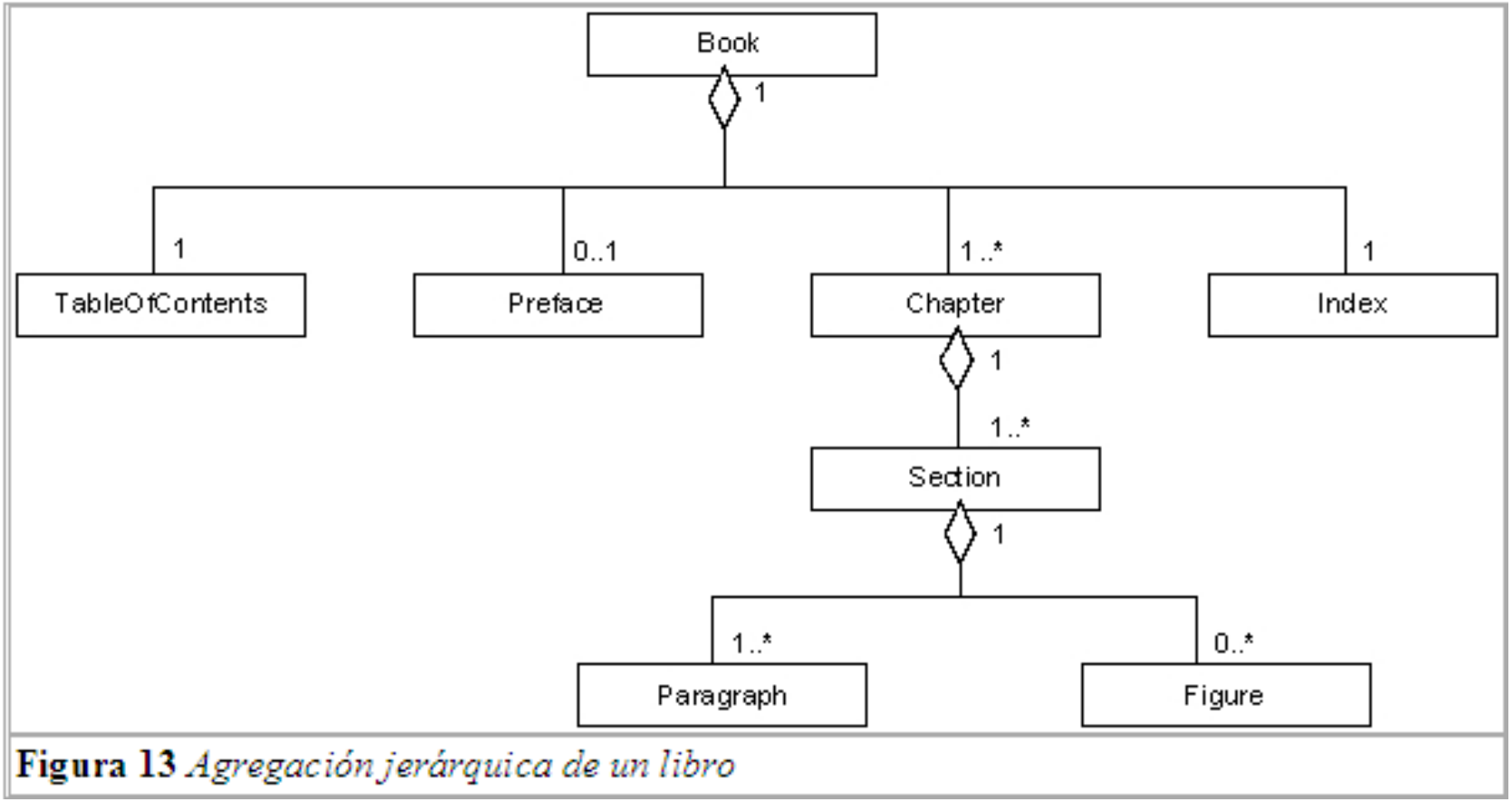
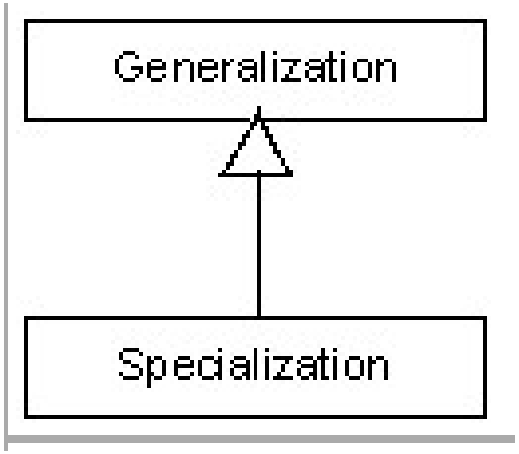



**Figura 13** *Agregación jerárquica de un libro*

# Especialización/Generalización

- La *Especialización/Generalización* representa a la relación *es un*. Por ejemplo, una ballena *es un* mamífero y un cliente *es una* persona.

  La especialización/generalización permite que la clase A sea definida como especialización de otra clase B, más general.



# Especialización/Generalización

- Una consecuencia importante de esta relación es que la clase A *hereda* todas las características de la clase B.

  Esto significa que todos los atributos y métodos de la clase B son también atributos y métodos de la clase A.

# Especialización/Generalización

.

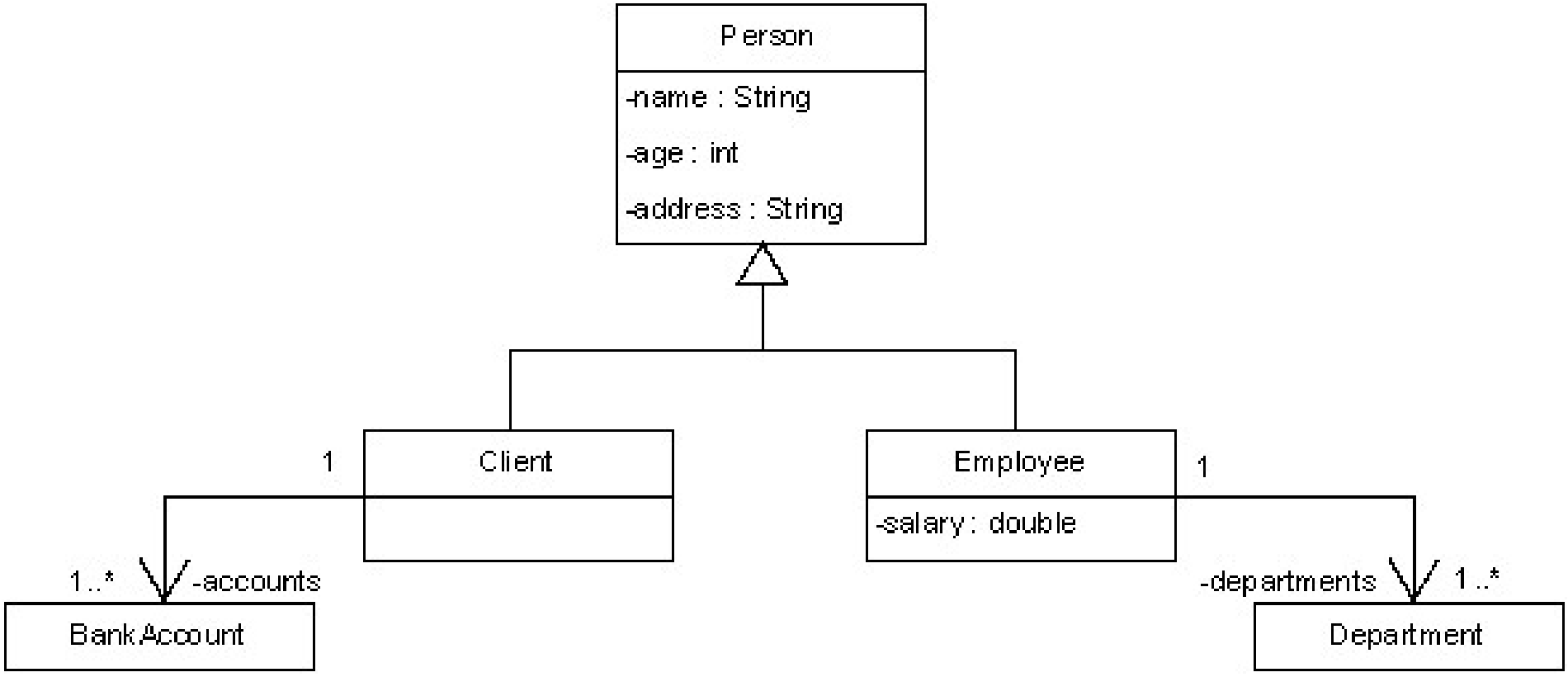



**Figura 15** *Relación de especialización/generalización entre clases que representan clientes, empleados y personas*

# Ejemplo

Realizar un programa en lenguaje C++ que defina las clases necesarias para realizar un sistema de venta de los pasajes de autobús del terminal de Mérida. Realizarlo de la forma más sencilla posible: No tomar en cuenta ni horario ni rutas.

**Resolución**:
Pensar en el diseño:

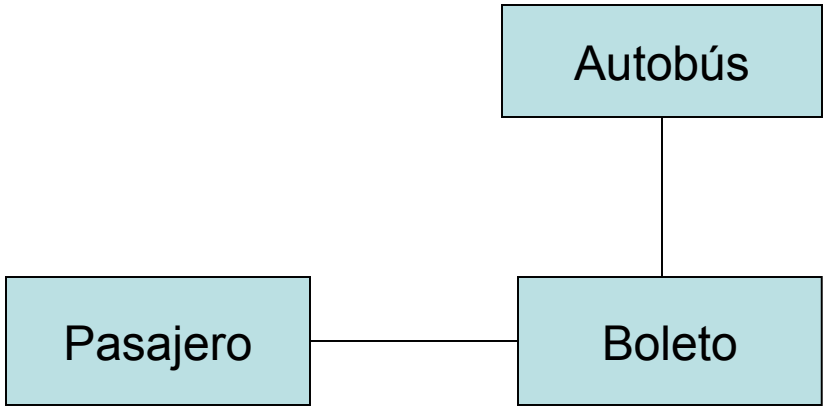
¿Qué clases son necesarias?

¿Qué atributos y métodos tendrá cada una?

# Diseño sencillo

3 clases:

- Autobús
- Pasajero
- Boleto




Campos por clase:

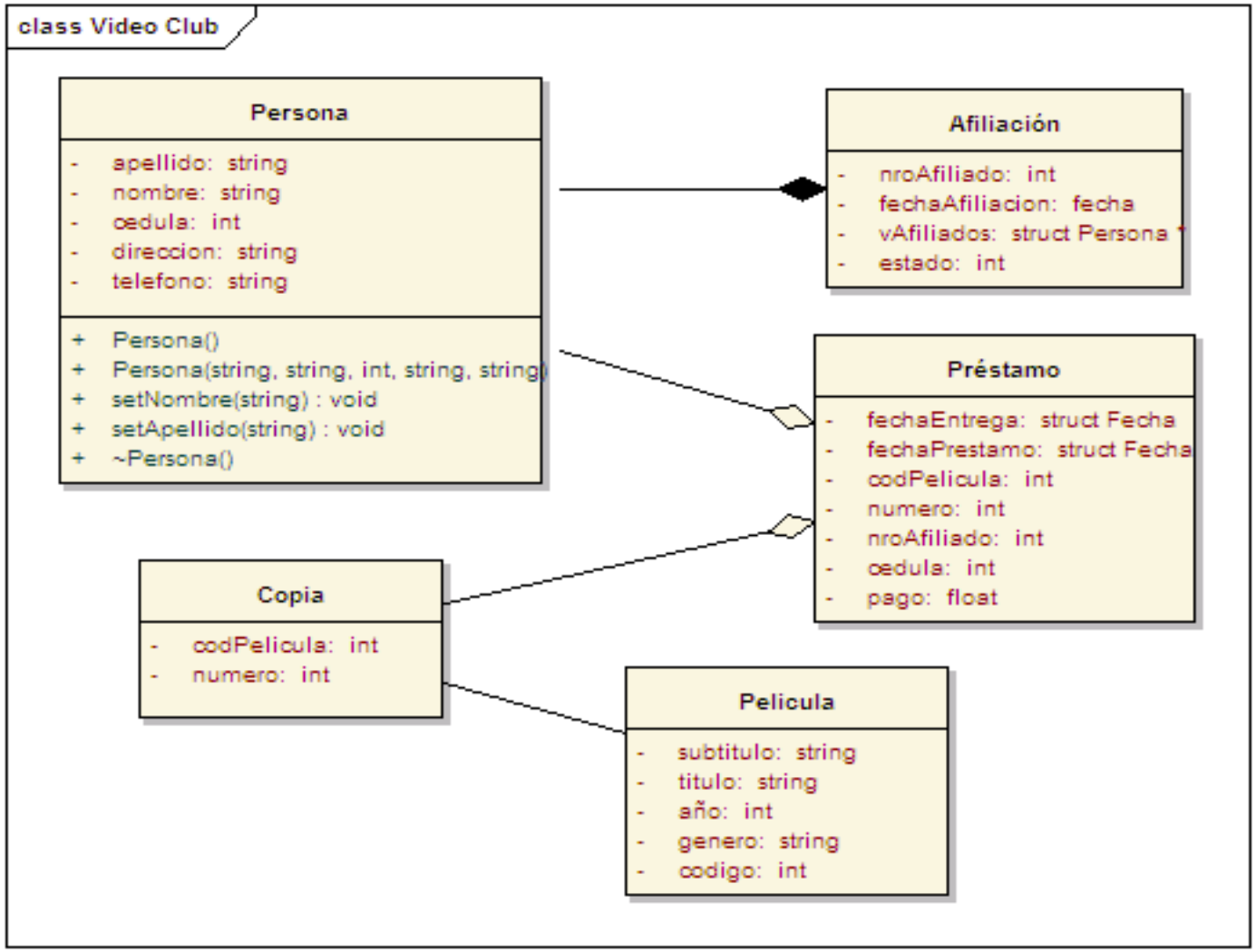
**Autobús**: número, idEmpresa, capacidad, idTipo
**Pasajero**: cédula, nombre, apellido, dirección, teléfono
**Boleto**: pasajero, bus, dia, mes, año, destino, precio

¿Métodos?

# Ejercicios

Video Club

# Ejercicios


class Video Club
Persona
- apellido: string
- nombre: string
- cedula: int
- direccion: string
- telefono: string
+ Persona()
+ Persona(string, string, int, string, string)
+ setNombre(string) : void
+ setApellido(string) : void
+ ~Persona()
Afiliación
- nroAfiliado: int
- fechaAfiliacion: fecha
- vAfiliados: struct Persona *
- estado: int
Préstamo
- fechaEntrega: struct Fecha
- fechaPrestamo: struct Fecha
- codPelicula: int
- numero: int
- nroAfiliado: int
- cedula: int
- pago: float
Copia
- codPelicula: int
- numero: int
Pelicula
- subtitulo: string
- titulo: string
- año: int
- genero: string
- codigo: int

# Definición de una Clase en C++

Declaración:

```
class <nombre> {
    public:
        <nombre>();        //constructor
        ~<nombre>();       //destructor
        <lista_propiedades_públicas>;
        <lista_métodos_públicos>;
    private:
        <lista_propiedades_privadas>;
        <lista_métodos_privados>;
};
```

# Ejemplo de Clase en C++

```
class Alumno {
    private:
        int ci;
        char nombre[20];
        char apellido[20];
        int cod_carrera;
    public:
        Alumno();
        inicializa(int ci, char *nombre, char *apellido,
                   int carrera);
        mostrar_datos();
        modificar_nombre(char *n);
        ~Alumno();
};
```