



CICL

[FORMACIÓN POR CICLOS]

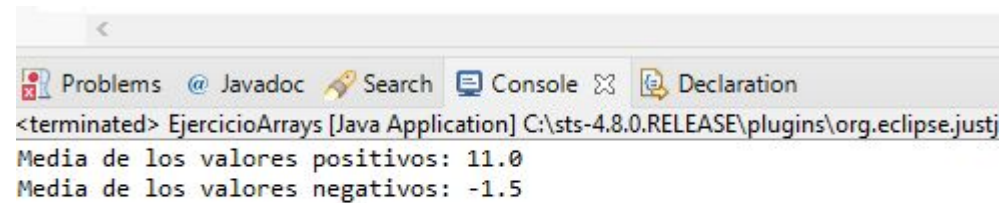
Programación Básic

Programación
Orientada a Objetos



Ejercicio

- Haga un programa que teniendo un array con 10 elementos. Calcule el promedio de los valores positivos y el promedio de los negativos.
- {-1,2,4,-2,3,45,7,8,9,10};



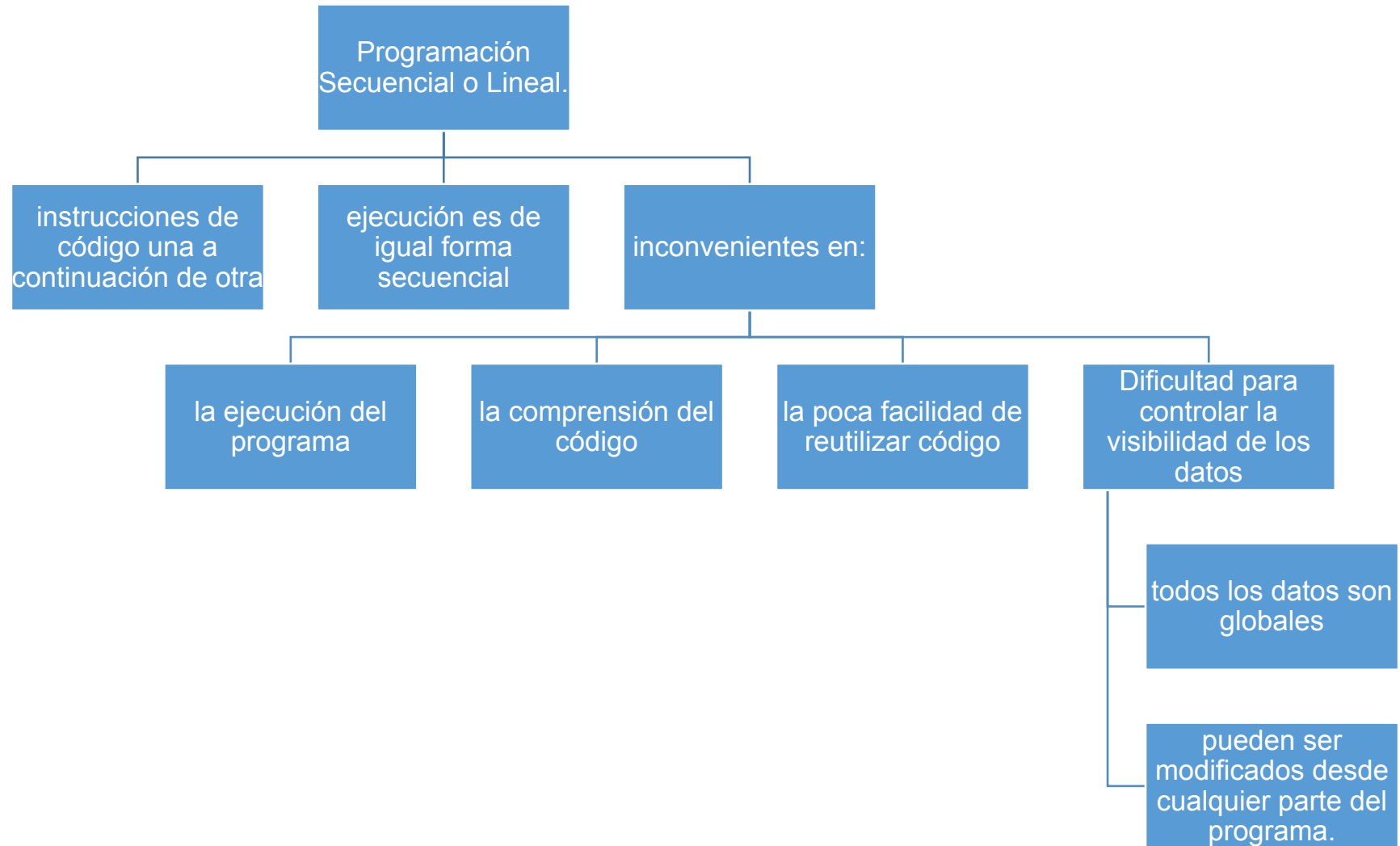
```
<terminated> EjercicioArrays [Java Application] C:\sts-4.8.0.RELEASE\plugins\org.eclipse.justj  
Media de los valores positivos: 11.0  
Media de los valores negativos: -1.5
```

Ejercicio 2

- Estaturas 1.65, 1.80, 1.70, 1.30, 1.68, 1.45
- Calcular el promedio
- Determinar cuantas personas están por encima o por debajo de la media

```
<terminated> EjercicioArrays2 [Java Application] C:\sts-4.8.0.RELEASE\j
persona 1 = 1.65
persona 2 = 1.8
persona 3 = 1.7
persona 4 = 1.3
persona 5 = 1.68
persona 6 = 1.45
Estatura media: 1.5966666666666667
Personas con estatura superior a la media: 4
Personas con estatura inferior a la media: 2
```

Programación Secuencial



Programación Orientada a Objetos

Programación Orientada a Objetos.

estilo de
programación

utiliza objetos
como bloque
esencial de
construcción.

Se parte de la creación de
clases

un objeto

Entidades del
mundo real

tipos abstractos
de datos

definen datos y
métodos.

es instancia de
una clase

identidad
(nombre del
objeto que lo
diferencia del
resto)

posee una copia de la definición de
datos y métodos

Da valores
(atributos o
características)

comportamiento a
los métodos que
trabajan con
estos datos.

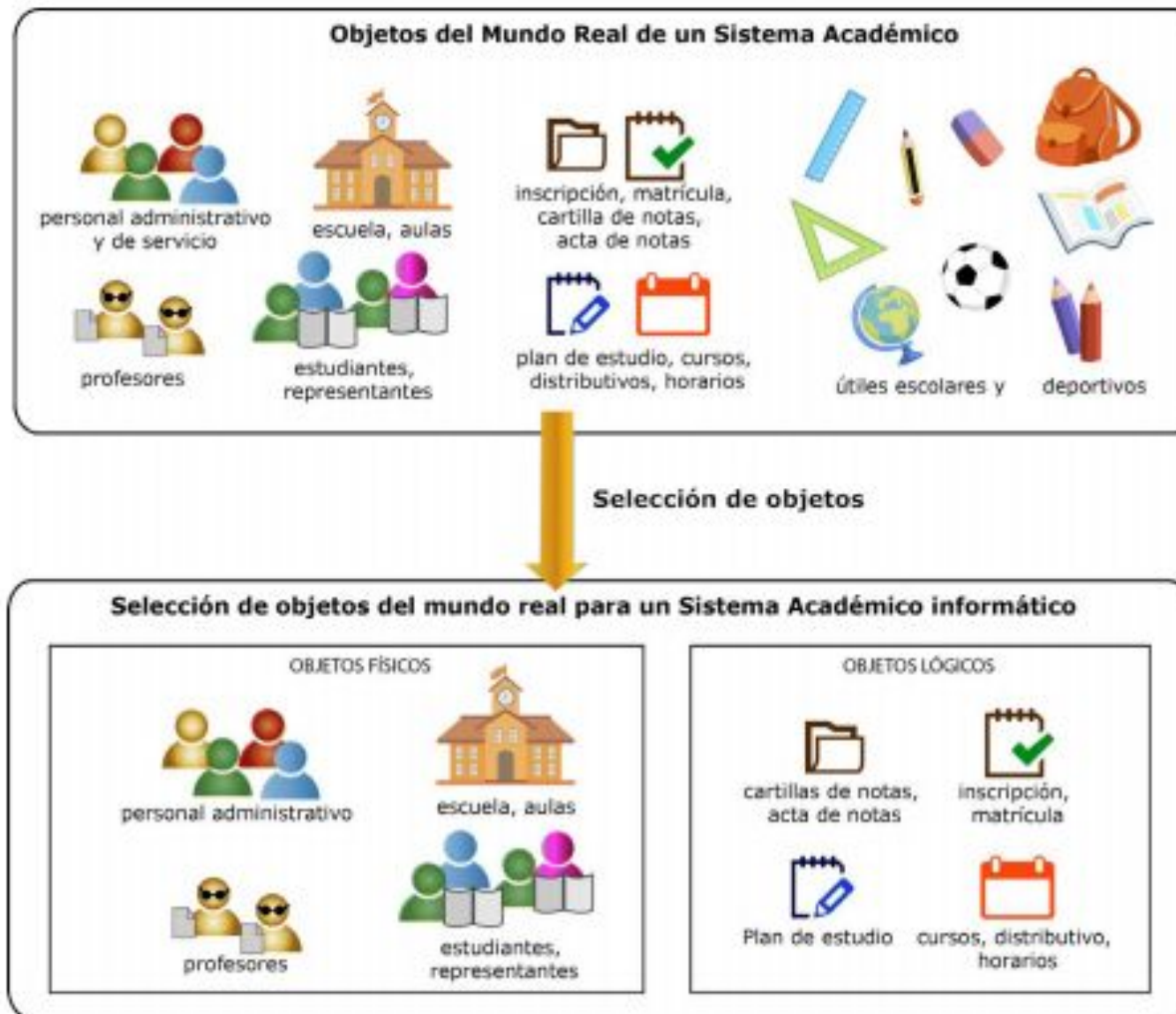
Objetos o entidades reales

Objetos físicos:

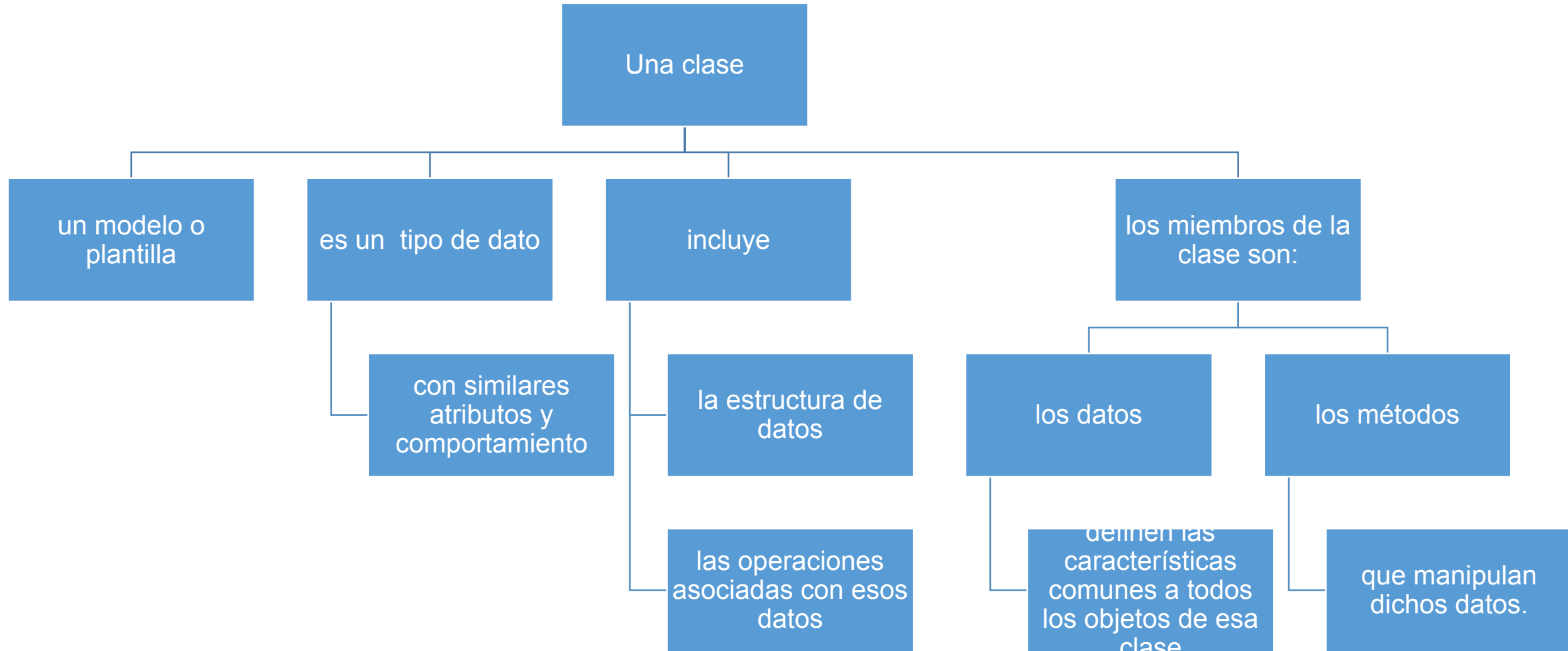
- Son tangibles
 - se puede ver y tocar.
- Ejemplos:
 - Persona
 - Colegio
 - Profesor
 - Automóvil

Objetos Lógicos.

- Son intangibles
- creados para plasmar la información en forma textual o gráfica.
- Ejemplos:
 - Elementos de interfaz gráficos de usuario:
 - ventanas, botones, íconos, menús, etc.
 - objetos lógicos de un sistema académico son:
 - acta de notas, hoja de matrícula, plan de estudios, cursos etc



Clase



Miembros de una clase: atributos y métodos

Los datos

- son considerados también campos
- Tienen asociado un tipo de dato que puede ser
 - Primitivo (int, char, double, etc.)
 - Tipo personalizado (clase).

Los métodos

- determinan cómo tiene que actuar el objeto
- permiten gestionar los datos miembro para dicho objeto.

Un programa orientado a objetos realiza fundamentalmente las siguientes acciones:

- a) Crea los objetos necesarios.
 - Estos son los métodos constructores que llevan el mismo nombre de la clase.
- b) Los mensajes enviados a unos y otros objetos dan lugar a que se procese internamente la información.
 - Estos métodos pueden ser: constructores, de escritura (set), de lectura (get) y otros métodos para realizar tareas personalizadas.
- c) Cuando los objetos no son necesarios, son borrados, liberándose la memoria ocupada por los mismos.

Ejercicio 1 *

Objeto

creado cuando la clase correspondiente recibe un mensaje solicitando su creación

se conocen como instancias de clase.

El primer método que se invoca automáticamente para crear un objeto se denomina constructor.

Un objeto aísla ciertos atributos al exterior

- sólo se acceden o se modifican mediante métodos públicos de lectura (get) o de escritura (set).

Los datos que pueden pertenecer a un objeto son:

- Constantes
- Variables
- Arreglos
- Cadenas
- Estructuras
- objetos de otras clases, etc.

Elementos de un Objeto

Identidad:

Es el identificador o nombre del objeto

Lo distingue de otros objetos.

Tiempo de vida:

La duración de un objeto en un programa siempre está limitada en el tiempo.

La mayoría de los objetos sólo existen durante una parte de la ejecución del programa.

Los objetos son

- creados mediante un mecanismo denominado instanciación
- cuando dejan de existir son destruidos.

Estado:

definido por sus atributos y los valores que almacenan.

Comportamiento:

definido por sus métodos

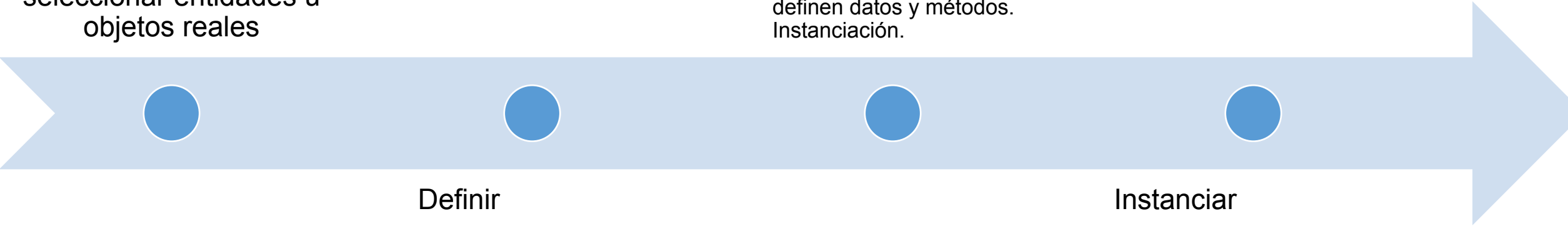
para que el resto de objetos que componen el programa y que pueden interactuar con él.

Procesos de abstracción e instanciación

identificar o se
seleccionar entidades u
objetos reales

crear las clases

- modelos o plantillas donde se definen datos y métodos.
- Instanciación.



Definir

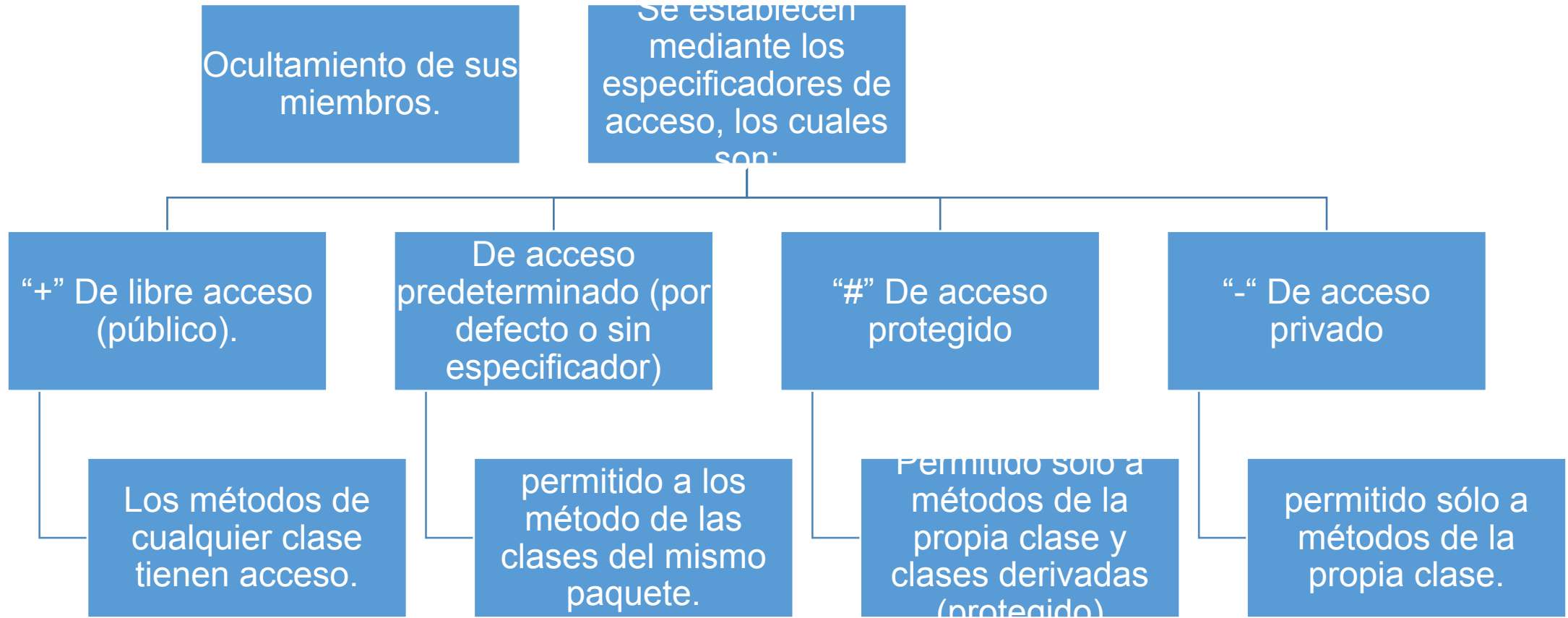
- características (atributos)
- comportamiento;

Instanciar

- Dar identidad
- estado (datos con valores o atributos)
- comportamiento (métodos que actúan sobre los propios datos).



Encapsulamiento



Ejercicio **

Ejemplo Objeto

Por ejemplo

un
auto
es un
objeto

tiene unos atributos:

Tiene un comportamiento:

cantidad de
pasajeros

Color

Capacidad
tanque

Marca

Kpg

tipo.

Acelerar

Frenar

Consumo
combustible

girar
izquierda

girar
derecha,
etc.

Ejercicio

- Tener en cuenta:
 - El recorrido en km por tanque = $\text{capacidad tanque} / \text{kpg}$
 - Crear estos 3 vehículos en un arreglo y con un constructor que asigne los campos y uno por defecto
 - Vehículo [tipo=Automóvil, marca=Mazda, color=Rojo, pasajeros=5, capacidad=7, kpg=50]
 - Vehículo [tipo=Camión, marca=Isuzu, color=Azul, pasajeros=10, capacidad=7, kpg=30]
 - Vehículo [tipo=Autobús, marca=Mercedes, color=Azul, pasajeros=10, capacidad=10, kpg=30]
- El resultado por cada vehículo deberá ser por ejemplo
 - El Autobus, marca Mercedes, de color Azul, para 10 pasajeros, esta acelerando.
 - El Autobus, marca Mercedes, de color Azul, para 10 pasajeros, esta frenando.
 - El Autobus, marca Mercedes, de color Azul, para 10 pasajeros, con capacidad de tanque 10 galones, y consumo de 30 km por galón, recorre 300.0 kpg.
 - El Autobus, marca Mercedes, de color Azul, para 10 pasajeros, esta girando Izquierda.
 - El Autobus, marca Mercedes, de color Azul, para 10 pasajeros, esta girando derecha.

Diagramas UML 2.5

Diseñar, especificar, visualizar, construir y documentar todos los artefactos que componen un Sistema de Información

Crear un diseño previo de una aplicación antes de proceder a su desarrollo e implementación

- en ocasiones concretas puede hacerse posteriormente.

De la misma forma que un arquitecto dibuja y diseña planos sobre el edificio que va a construir

- un analista de software crea distintos diagramas UML que sirven de base para la posterior construcción/mantenimiento del sistema.

El modelado es vital en todo tipo de proyectos

- importancia a medida que el proyecto crece de tamaño.

visualizar y verificar los diseños de sus sistemas de software antes de que la implementación del código haga que los cambios sean difíciles y demasiado costosos.

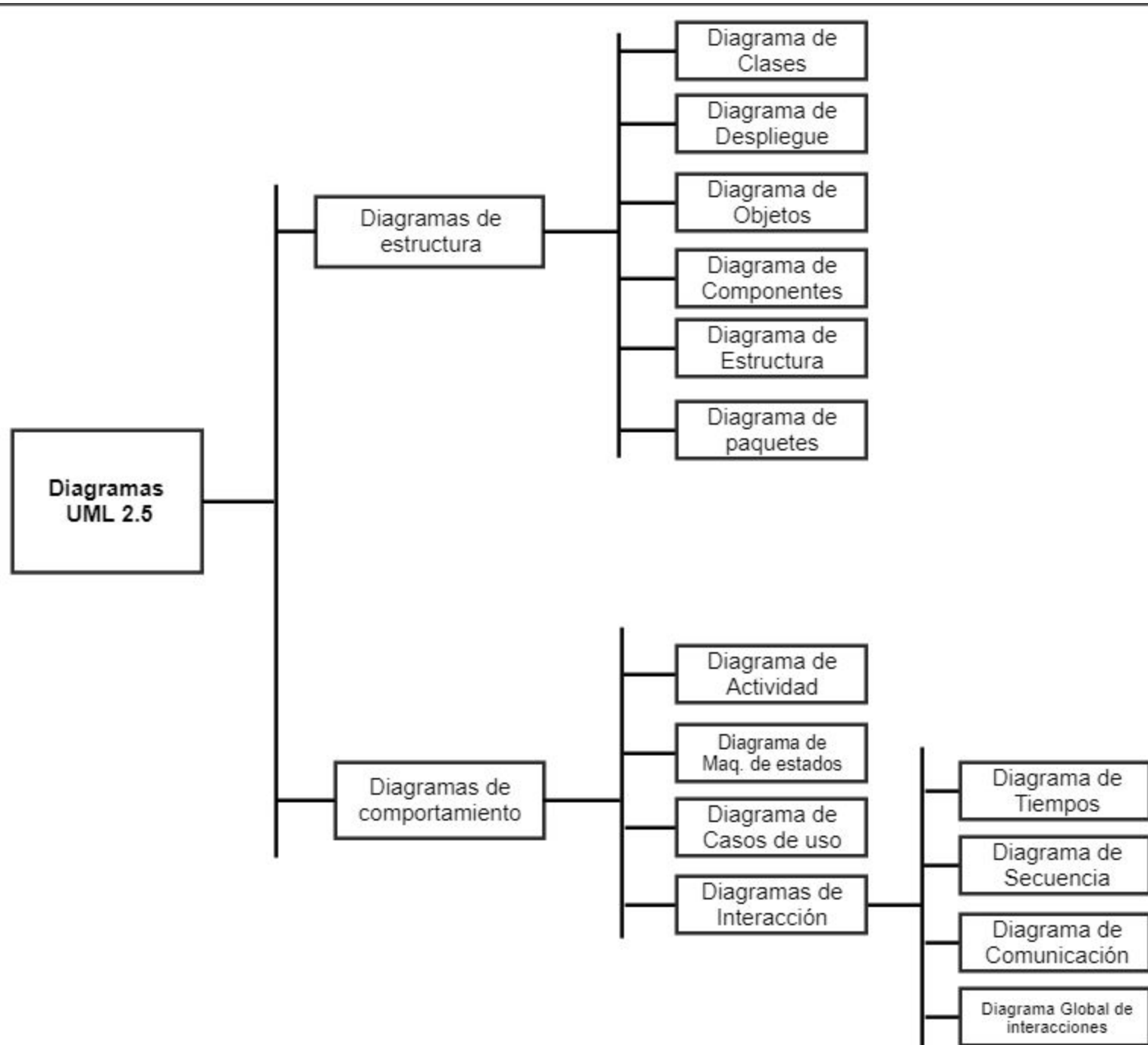


Diagrama de clases

Para representar la clase con estos elementos se utiliza una caja que es dividida en tres zonas:

Nombre de la clase.

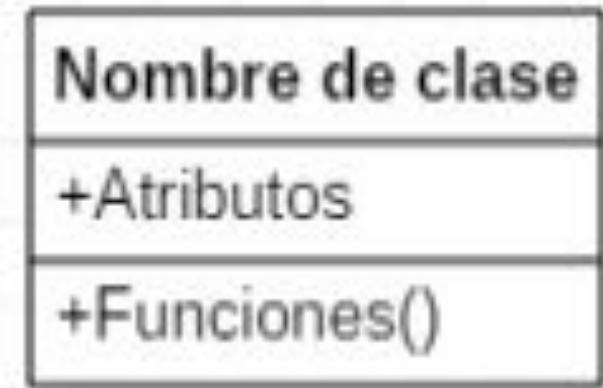
- En caso de que la clase sea abstracta se utilizará su nombre en cursiva. **

Atributos de la clase

- uno por línea y utilizando el siguiente formato:
- *visibilidad nombre_atributo : tipo = valor-inicial { propiedades }*
- Aunque esta es la forma “oficial” de escribirlas, **es común** simplificarlas únicamente poniendo el nombre y el tipo o únicamente el nombre.

Funciones o Metodos

- *visibilidad nombre_funcion { parametros } : tipo-devuelto { propiedades }*
- se suele simplificar indicando únicamente el nombre de la función y, en ocasiones, el tipo devuelto.



Visibilidad

Tanto los atributos como las funciones incluyen al principio de su descripción la visibilidad que tendrá. Esta visibilidad se identifica escribiendo un símbolo y podrá ser:

(+) Pública.

- Representa que se puede acceder al atributo o método desde cualquier lugar de la aplicación.

(-) Privada.

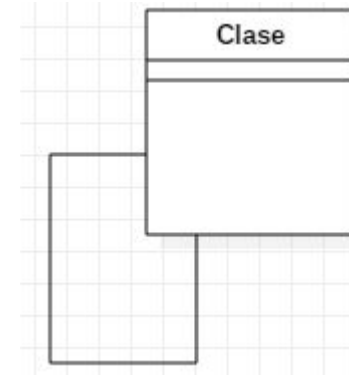
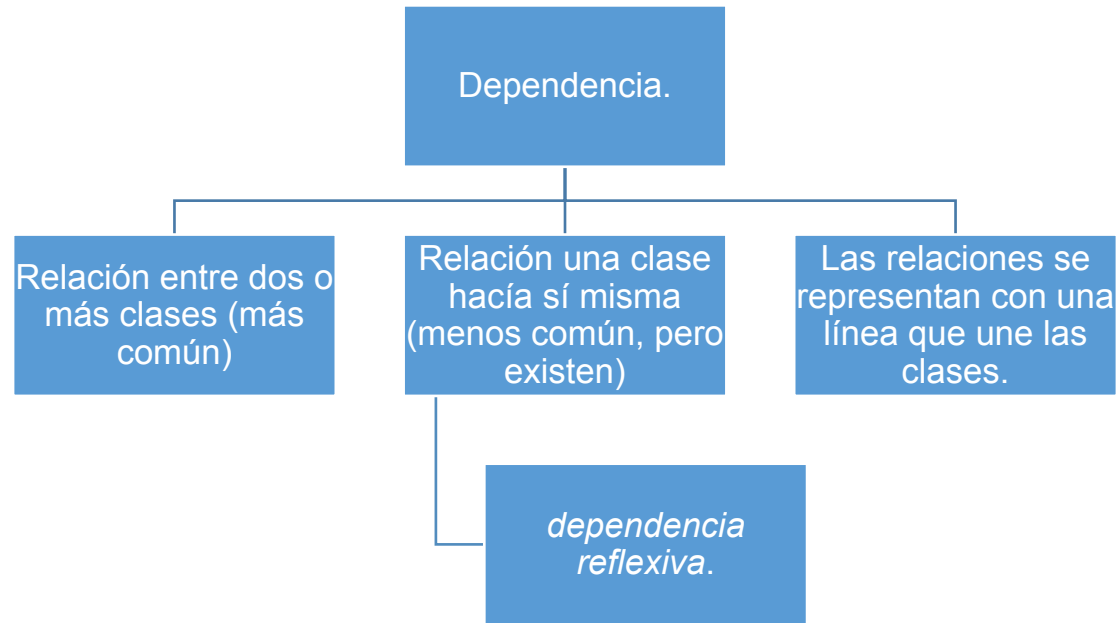
- Representa que se puede acceder al atributo o método únicamente desde la misma clase.

(#) Protegida.

- Representa que el atributo o función puede ser accedida únicamente desde la misma clase o desde las clases que hereden de ella (clases derivadas).

Animal
-Especie -Nombre
+Animal() +setEspecie() +getEspecie() +setNombre() +getNombre()

Relaciones



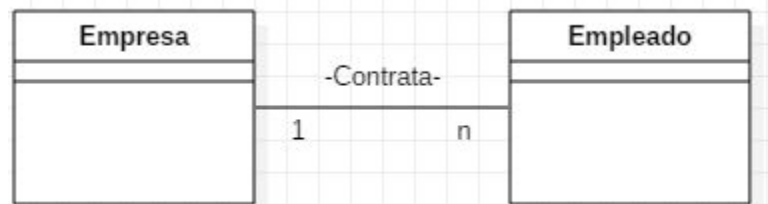
Propiedades de las relaciones

Multiplicidad.

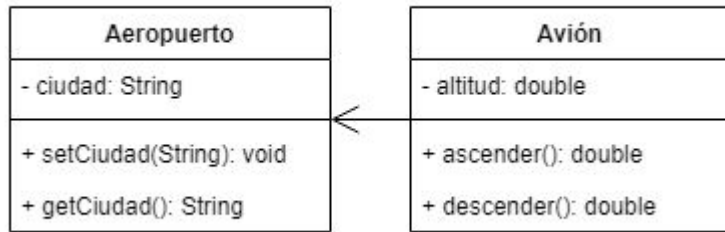
- número de elementos de una clase que participan en una relación.
- Se utiliza n o $*$

Nombre de la asociación.

- Ayuda a entender la relación que tienen dos clases.
- Por ejemplo: “Una empresa contrata a n empleados”



Tipos de relaciones



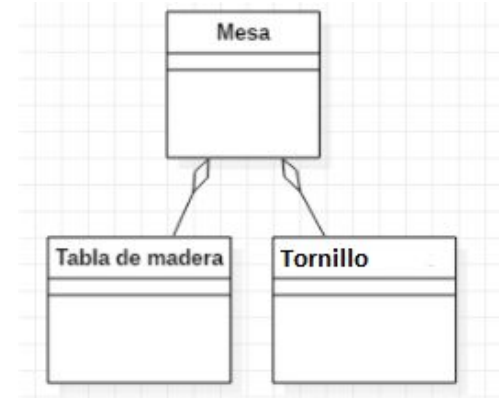
Asociación

- Se utiliza para representar dependencia semántica.
- Se representa con una simple línea continua que une las clases que están incluidas en la asociación.
- Un ejemplo de asociación podría ser: “Un avión pertenece a un aeropuerto”

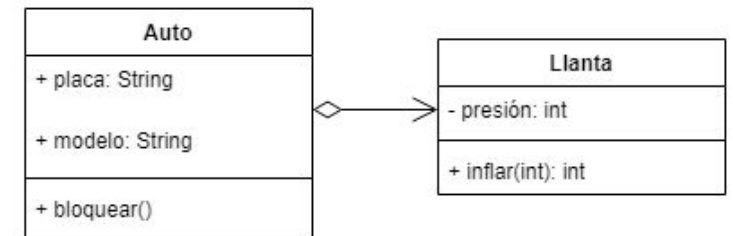
Tipos de relaciones

Agregación

- Es una representación jerárquica que indica un objeto y las partes que componen ese objeto.
- Representa relaciones en las que **un objeto es parte de otro**, pero aun así debe tener **existencia en sí mismo**.
- Se representa con una línea que tiene un **rombo** en la parte de la clase que es una agregación de la otra clase (es decir, en la clase que contiene las otras).



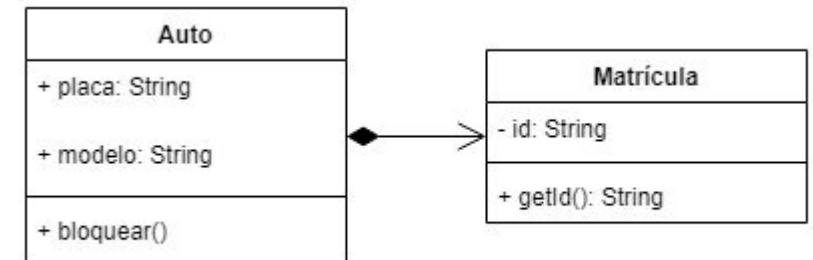
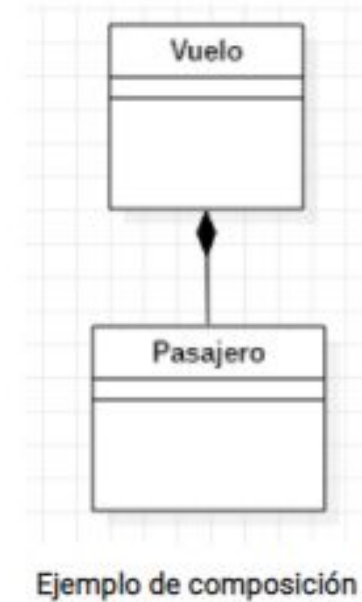
Ejemplo de agregación



Tipos de relaciones

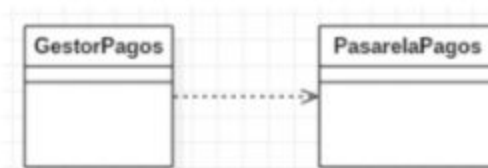
Composición

- Representa una **relación jerárquica entre un objeto y las partes que lo componen, pero de una forma más fuerte.**
- En este caso, los elementos que forman parte no tienen sentido de existencia cuando el primero no existe.
- Los componentes no se comparten entre varios elementos, esta es otra de las diferencias con la agregación.
- Se representa con una línea continua con un rombo relleno en la clase que es compuesta.



Dependencia

- Se utiliza este tipo de relación para **representar que una clase requiere de otra para ofrecer sus funcionalidades**. Es muy sencilla y se representa con una flecha discontinua que va desde la clase que necesita la utilidad de la otra flecha hasta esta misma.

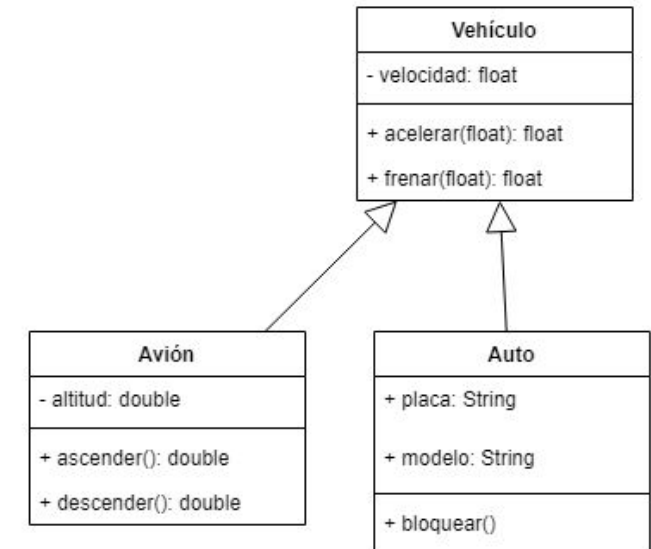


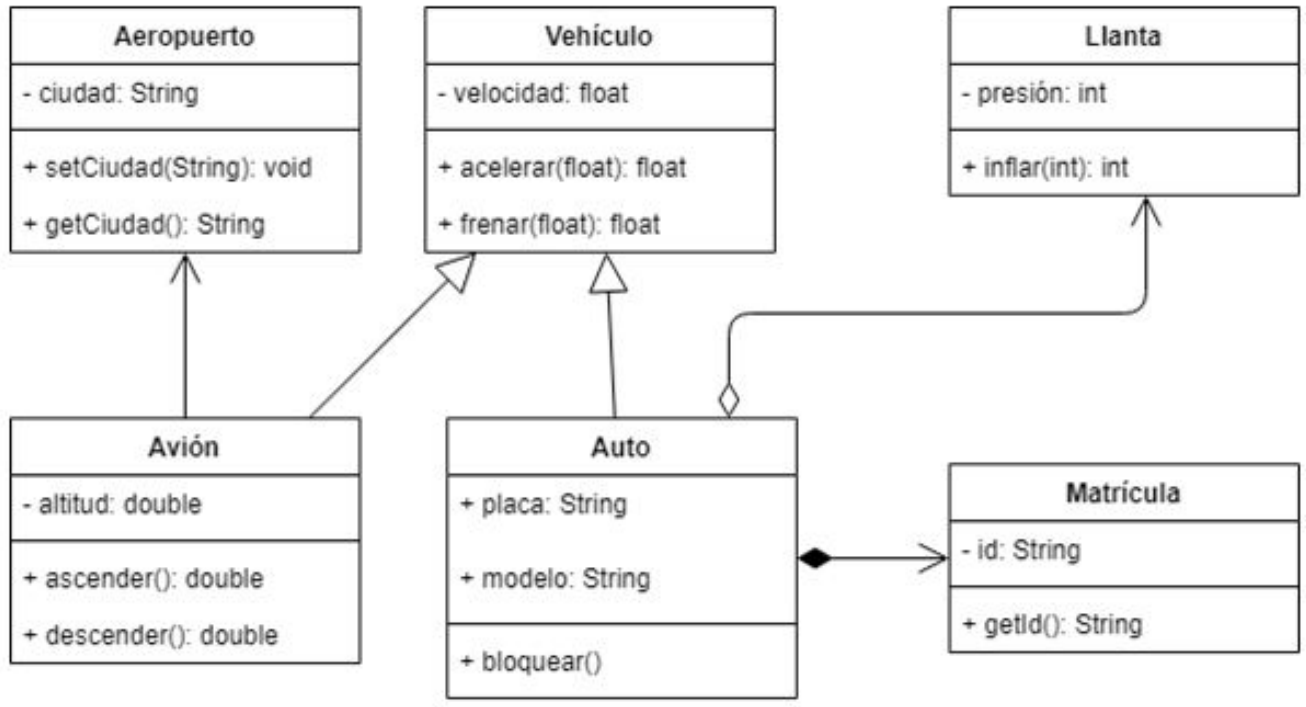
Ejemplo de dependencia

Herencia

herencia.

- Se representa usando una flecha
- Representan la relación entre una o varias clases hijas y su clase padre.
- En una relación de herencia
 - las clases hijas heredan los métodos y atributos de su clase padre
 - pueden agregar a su vez sus propios métodos y atributos.





Ejercicio

- Realiza el diseño de una aplicación para la gestión de empleados y clientes de una empresa. La aplicación deberá:
 - Manejar clientes (se guarda su tipo de documento, identificación, nombre, dirección, teléfono, y puntos por compra. Adicionalmente calcula sus puntos por compra de acuerdo al valor acumulado de una compra)
 - La empresa tiene un nit y un nombre y una ciudad la cual a su vez tiene un nombre y un código.
 - La empresa tiene empleados que tienen un tipo de documento, identificación, nombre, teléfono, dirección, salario y cargo y adicionalmente la opción de calcular el salario por los días trabajados.
 - Un empleado puede tener a cargo otros empleados.

Ejercicio 2

- Teniendo el ejercicio Vehiculo de la clase anterior, añadir una dependencia a la clase Matricula que tenga la siguiente información:
 - Identificación de la Matricula
 - Placa
 - Ciudad
 - Año
- Hacer un constructor de en la clase Vehiculo que tenga como parametro una matricula.
- Crear una clase principal que tenga un arreglo de 3 vehiculos creados con el constructor creado en el paso anterior y que asigne los demás atributos con los metodos “set” de la clase.