

Programador de Videojuegos

CLASE 2/07/25



Índice de temas de la clase

- UML
 - Diagrama de secuencia
 - Diagrama de actividad
 - Diagrama de clases

Diagrama de Caso de Uso

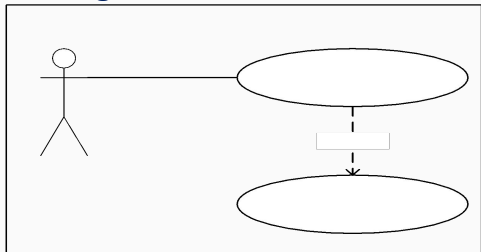
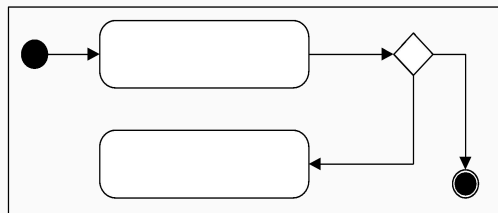


Diagrama de Actividades



Escenario del caso de uso

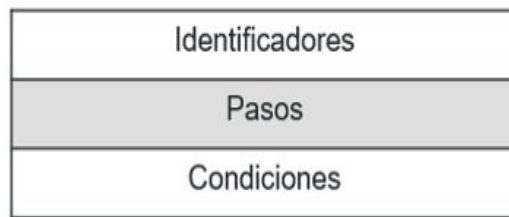


Diagrama de secuencias

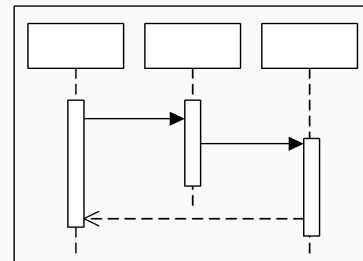
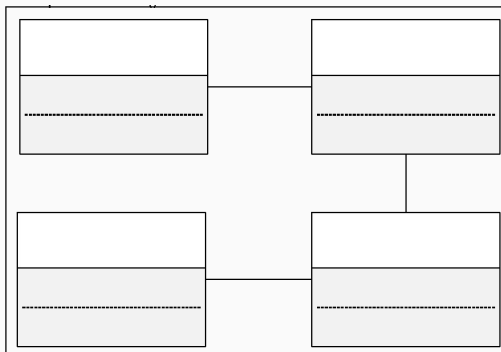


Diagrama de clases



3. Diagrama de secuencia

Diagrama de secuencia

Un diagrama de secuencia (en UML) es un tipo de diagrama que muestra cómo interactúan los objetos o componentes de un sistema a lo largo del tiempo para cumplir con un proceso o caso de uso.

Muestran las actividades e interacciones dentro de una funcionalidad. Suelen utilizarse para representar el proceso de los escenarios de casos de uso, cada escenario puede tener un diagrama de secuencia. Los escenarios simples no suelen tener diagrama de secuencia.

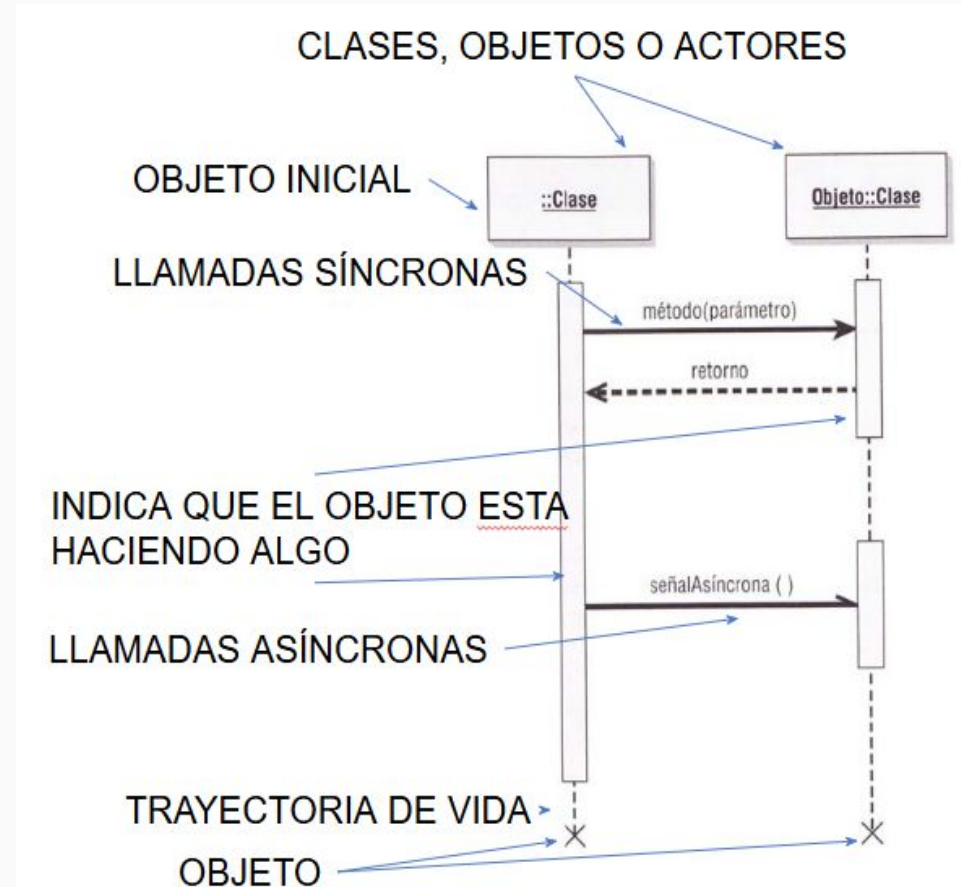
¿Para qué sirve?

- Modelar el flujo dinámico de un sistema: qué ocurre primero, qué ocurre después.
- Mostrar interacciones entre objetos o actores y el sistema.
- Detallar la lógica detrás de un caso de uso.
- Comunicar cómo se da un proceso o funcionalidad paso a paso.
- Útil en la fase de diseño para que los programadores comprendan la secuencia de operaciones.

IMPORTANTE: En estos diagramas se muestran las interacciones en ORDEN. Se podría decir entonces, que es un diagrama para visualizar la secuencia de los eventos que ocurren en un sistema.

Diagrama de secuencia: Elementos

- Actor: siempre son externos al sistema e interactúan con él. Son los actores de los casos de uso.
- CLASES u OBJETOS: partes del sistema que realizan acciones.
- MENSAJES (métodos): Comunicación dentro del sistema. Una línea continua muestra una llamada y una punteada muestra una respuesta.
- ACTIVIDAD: una barra lateral o un rectángulo vertical en la línea de vida muestran el foco de control cuando el objeto está ocupado haciendo cosas.
- LÍNEA DE VIDA línea de tiempo del objeto / clase. Corresponde al tiempo a partir del que se creó hasta el momento en que se destruye.




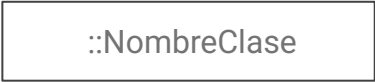
Elemento	Función	Cuestiones a tener en cuenta	Representación
Actor	Indica quién participa en la secuencia. Puede ser un actor externo o un objeto del sistema.	Se colocan horizontalmente en la parte superior del diagrama. Su nombre debe estar claro y único. No se colocan objetos que no intervienen en la interacción.	
Clase			
Línea de vida (Lifeline)	Representa la existencia del objeto (cuando está activo) a lo largo del tiempo durante la interacción.	Solo debe extenderse hasta donde el objeto participa.	Una línea vertical punteada que parte del actor u objeto.
Mensajes sincrónicos.	El remitente espera una respuesta (ej: llamada a función).	Flechas horizontales de un objeto a otro. Mostrar qué mensajes se envían, en qué orden y entre qué elementos. Deben ordenarse de arriba hacia abajo, en orden cronológico.	Se representan con flecha llena y continua.
Mensajes asincrónicos	El remitente no espera respuesta inmediata (ej: envío de evento).		Se representan con flecha línea continua y cabeza abierta.
Mensaje de retorno / respuesta	Se usa para mostrar una respuesta a una llamada (devuelve un resultado). Indica valores devueltos, confirmaciones o respuestas.	No siempre es necesaria, pero si se usa debe ir después del mensaje sincrónico .	Representada con línea discontinua con una flecha de vuelta (←).

Diagrama de Secuencia del caso de uso: ALMACENAR AFILIACIÓN

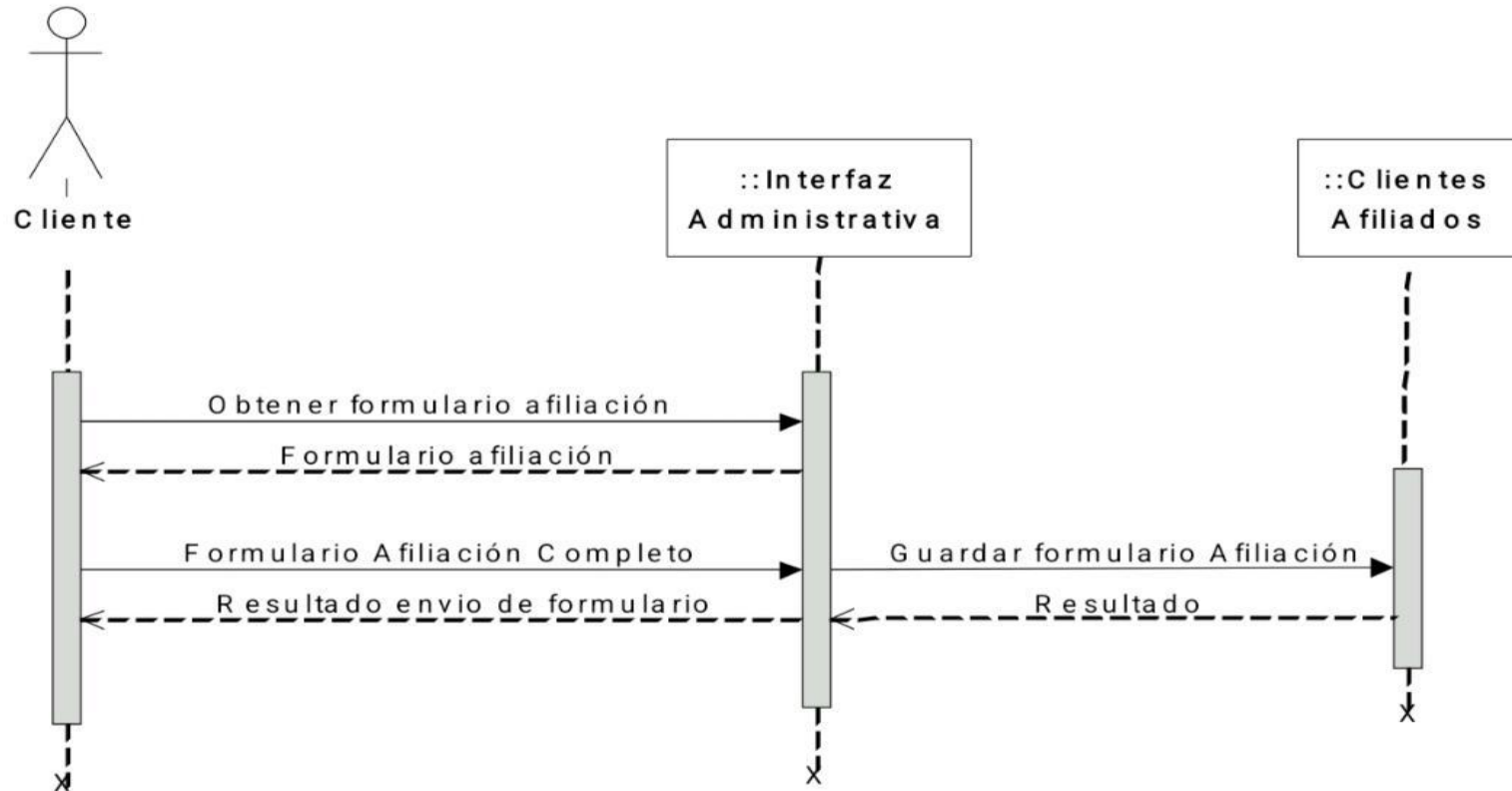
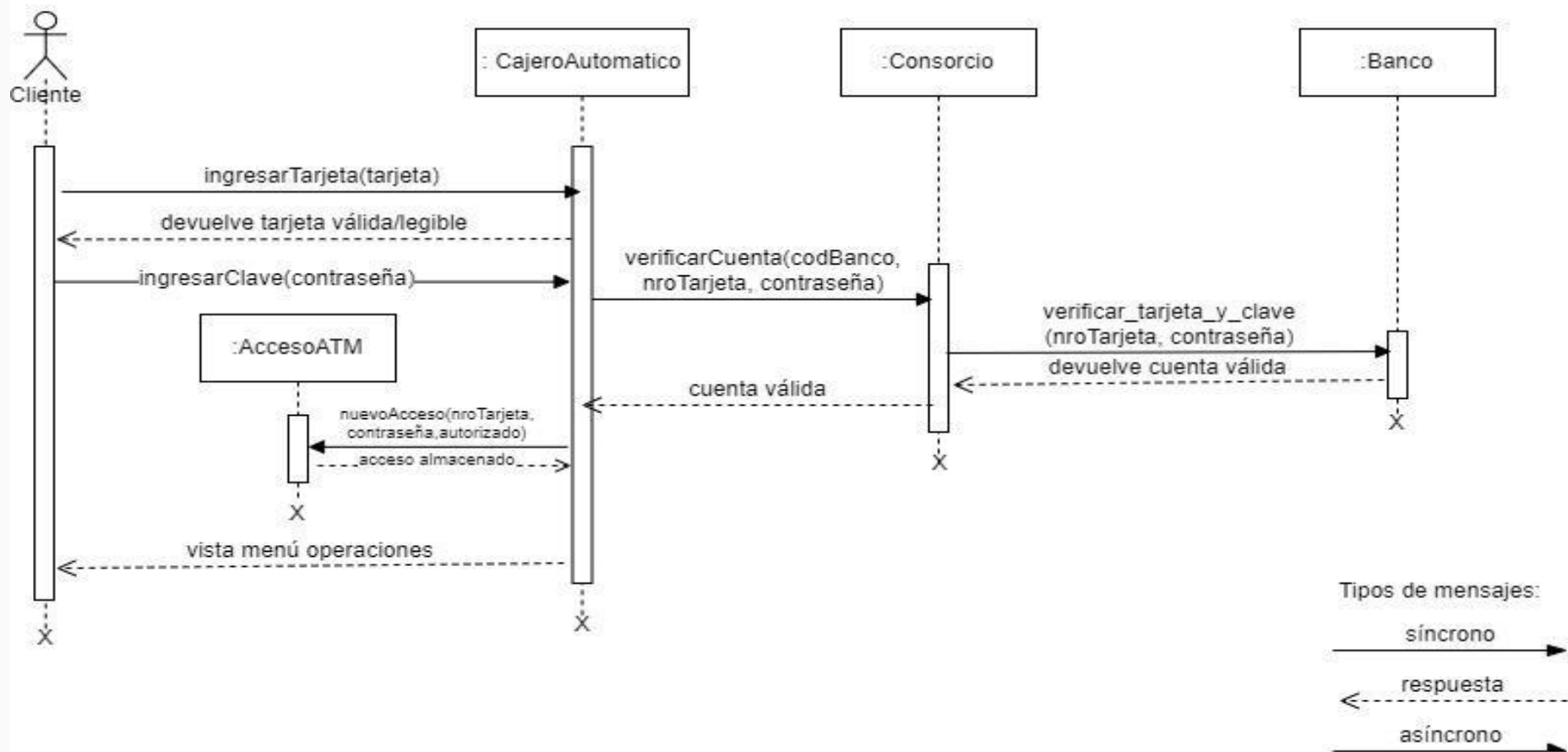
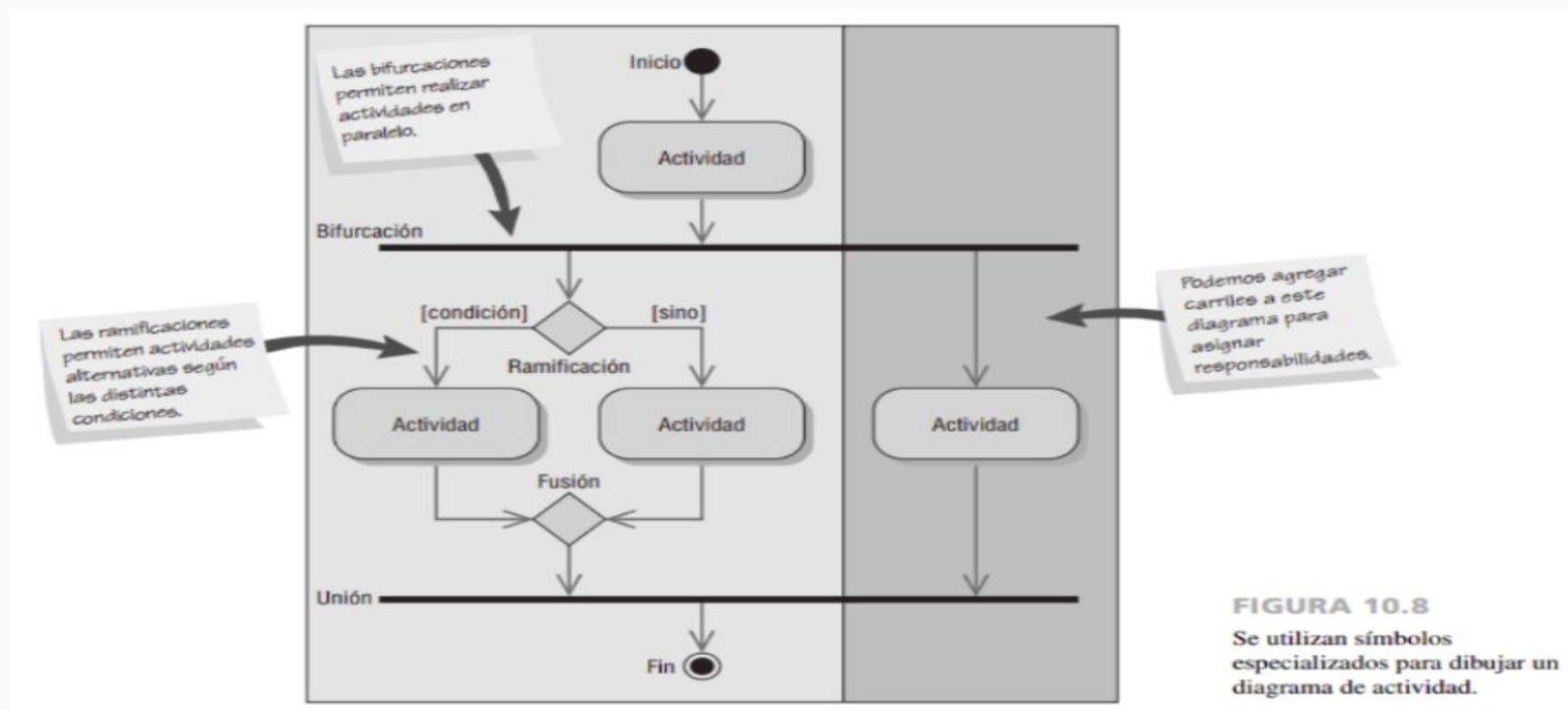


Diagrama de secuencias del caso de uso Realizar operación



4. Diagrama de actividad

Diagrama de actividad



5. Diagrama de clases

Diagrama de clases

EL ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS, SE CENTRA EN LA DEFINICIÓN DE CLASES Y EN EL MODO EN EL QUE COLABORAN UNA CON OTRA PARA CUMPLIR CON LOS REQUERIMIENTOS DEL CLIENTE.

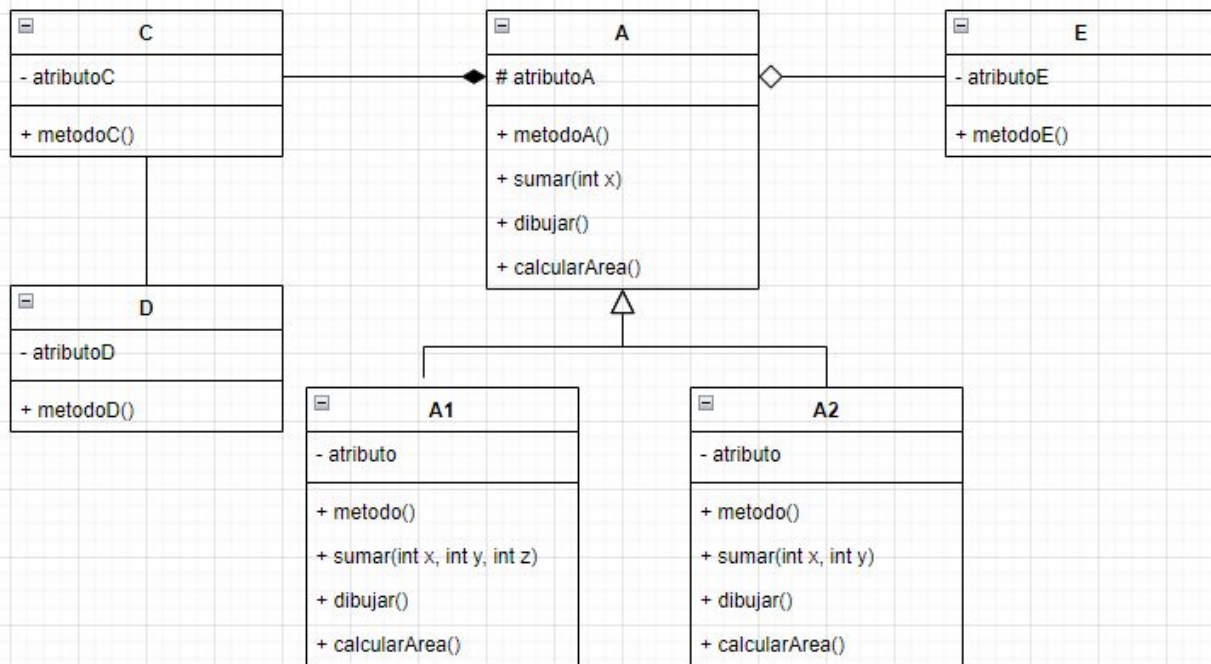


Diagrama de clases

OBJETOS:

- Son todo lo relevante al entorno de análisis.
- Son objetos o cosas de la vida real, pueden ser objetos físicos o conceptuales.
- Ejemplo: personas, cosas, autos, cuentas bancarias, etc.
- Para crearlos es necesario definir la clase.

CLASES:

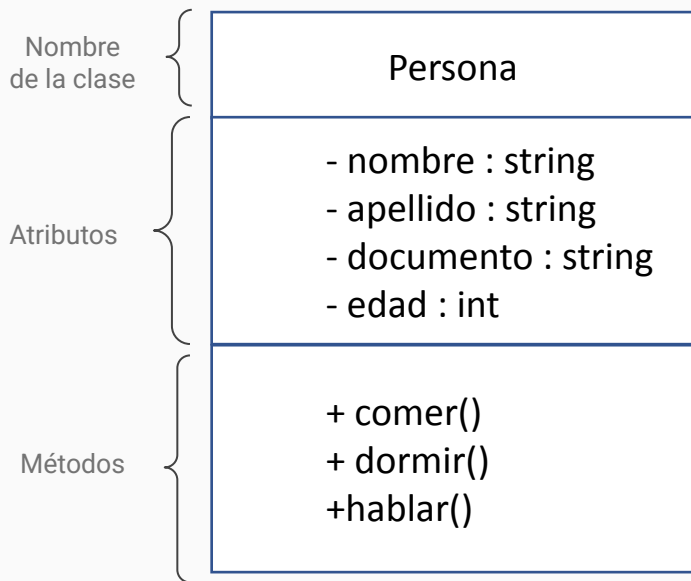
- Son el molde (plantilla) de los objetos.
- Define los atributos y comportamientos (métodos) de los objetos.
- En tiempo de ejecución son instanciadas como objetos.
- Cada instancia (objeto) tiene atributos particulares (valores en sus atributos).
- El comportamiento de cada instancia depende de la lógica y de sus atributos.

ATRIBUTOS:

- Son las propiedades de las clases. Cuando una clase es instanciada toman valores específicos.
- Pueden ser datos u otras clases.
- Suelen tener modificadores de acceso (públicos, privados o protegidos).

MÉTODO:

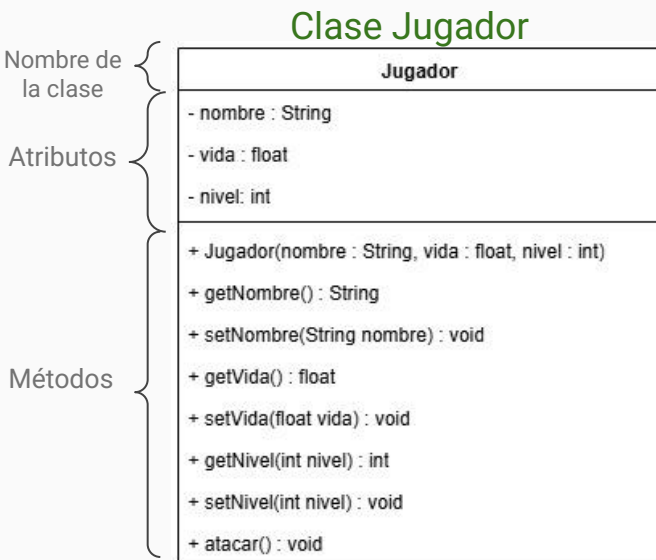
- Implementan el comportamiento (las operaciones o funciones) de las clases.
- Son acciones solicitadas a un objeto.
- Pueden recibir variables del entorno.
- Suelen tener modificadores de acceso (públicos, privados o protegidos).



Representación de una clase

Diagrama de clases: Clase \neq objeto

- La clase define cómo se ve y se comporta algo (estructura y comportamiento). Es decir, es una plantilla o modelo general que define: qué atributos (datos) tiene y qué métodos (acciones o comportamientos) puede ejecutar.
- El objeto es una instancia concreta de una clase, con valores reales en sus atributos y la capacidad de ejecutar los métodos definidos. Es decir, no se representa en el diagrama, se crean a partir de las clases en tiempo de ejecución.



A partir de la clase Jugador, puedo tener varios objetos:

Objeto 1:

nombre = "Pepe"
vida = 100
nivel = 5

Objeto 2:

nombre = "Zelda"
vida = 80
nivel = 4

Ambos objetos son Jugadores, pero con datos distintos. Son la instancia real del jugador corriendo en el juego.

Esta clase **no es un jugador específico**, sino la definición general que usaré para crear jugadores concretos.

Diagrama de clases

Visibilidad (de los atributos y métodos): Define la accesibilidad.

- Privado: Ninguna otra clase o subclase puede acceder a ellos.
- + Público: Cualquier otra clase o subclase puede acceder a ellos.
- # Protegido: La misma clase o subclase puede acceder a ellos.

En general, los atributos son privados o protegidos. Se accede al objeto mediante sus métodos. En cambio, muchos de los métodos son públicos, ya que sirven para interactuar con las clases.

Hacer los atributos privados implica que serán visibles sólo para los objetos externos a través de los métodos de la clase, una técnica que se conoce como encapsulamiento u ocultamiento de la información.

Diagrama de clases

Las relaciones son conexiones entre las clases. Se muestran como líneas que conectan a las clases en un diagrama de clases. Hay dos categorías de relaciones: asociaciones y relaciones entre un todo y sus partes.





Tipos de relaciones	Subtipos de relaciones	Representación	Descripción
Asociaciones			Conexión estructural entre clases
Todo/parte	Agregación		Existencia independiente de las partes (asociación débil)
	Composición		Existencia dependiente de las partes (asociación fuerte)
Generalización			Indica que una clase es una cosa específica de la clase más general.

Diagrama de clases: Relación de asociación

Asociación

Es una relación estructural general entre clases, que indica que una instancia de una clase se vincula con instancias de otra clase. No implica pertenencia, ni dependencia fuerte. Es como decir: “un objeto conoce o utiliza a otro”.

¿Para qué sirve?

- Representar relaciones conceptuales como: usa, tiene, interactúa con.
- Modelar vínculos entre objetos sin suponer propiedad o ciclo de vida conjunto.

Sintaxis gráfica

- Se representa con una línea simple entre las clases.
- Puede tener:
 - Nombre de la asociación
 - Multiplicidad (cuántos objetos de una clase están relacionados con cuántos de la otra)
 - Navegabilidad (flechas que indican en qué dirección va la relación)

Ejemplo:

El Jugador usa un Arma, pero no necesariamente la posee permanentemente. Puede cambiar de arma durante el juego, o compartirla.

Diagrama de clases: Relación de asociación

Multiplicidad: ¿Qué es?

Indica cuántos objetos de una clase pueden estar asociados con cuántos objetos de otra.

Por ejemplo:

Un profesor puede estar en 1 o más Cursos.

Cada Curso debe tener un profesor.

Multiplicidad	Significado
1	Uno y solo uno
0..1	Cero o uno (opcional)
* o 0..*	Cero o muchos
1..*	Uno o muchos
n..m	Rango específico (ej. 2..5)

Profesor
- atributos
+ métodos()

Curso
- atributos
+ métodos()

Diagrama de clases: Relación de asociación

- 1) Un Profesor puede dirigir una carrera y cada carrera es dirigida por un solo Profesor.

Profesor
- atributos
+ métodos()

Carrera
- atributos
+ métodos()

- 2) Cada jugador tiene un avatar. Y el avatar corresponde al jugador.

Jugador
- atributos
+ métodos()

Avatar
- atributos
+ métodos()

Diagrama de clases: Relación de asociación

3) Un jugador puede o no estar en una partida. Y la partida corresponde al jugador.

Jugador
- atributos
+ métodos()

Partida
- atributos
+ métodos()

4) Un equipo tiene muchos jugadores y el jugador pertenece a un equipo.

Equipo
- atributos
+ métodos()

Jugador
- atributos
+ métodos()

Diagrama de clases: Relación de asociación

5) Cada nivel puede tener muchos obstáculos y los obstáculos pertenecen a un nivel.

Nivel
- atributos
+ métodos()

Obstáculo
- atributos
+ métodos()

6) Cada nivel puede tener muchos obstáculos y los obstáculos pueden repetirse por nivel.

Nivel
- atributos
+ métodos()

Obstáculo
- atributos
+ métodos()

Diagrama de clases: Relación de asociación

7) Cada equipo puede estar confirmado de 2 a 5 jugadores y cada jugador pertenece a un equipo.

Equipo
- atributos
+ métodos()

Jugador
- atributos
+ métodos()

8) Cada personaje puede o no tener habilidades y la habilidad puede repetirse por personaje.

Personaje
- atributos
+ métodos()

Habilidad
- atributos
+ métodos()

Diagrama de clases: Relación de asociación

RELACIÓN TODO-PARTE: AGREGACIÓN y COMPOSICIÓN

Las relaciones todo-parte (o relaciones de contención) son relaciones especiales de asociación, donde una clase forma parte de otra.

Hay dos tipos principales:

1. AGREGACIÓN (relación débil)

Una relación todo-parte con bajo acoplamiento. Indica que una clase es parte de otra, pero puede existir independientemente de ella. Es una relación “tiene un”, pero no depende de ella para existir.

¿Para qué sirve?

- Modelar una relación parte-todo sin ciclo de vida compartido.
- Representar que los objetos pueden ser compartidos o vivir fuera del todo.

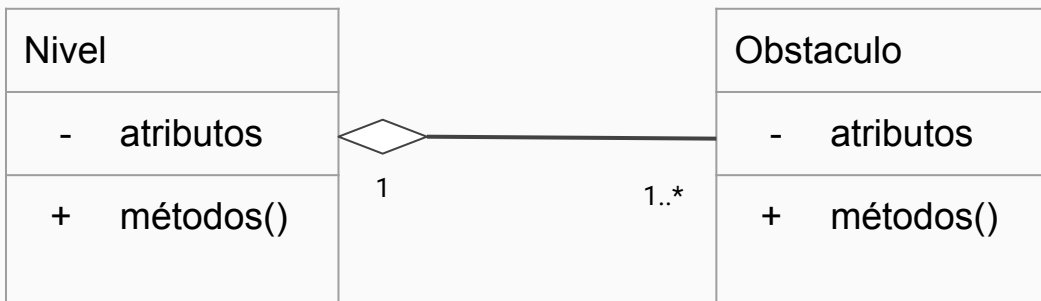


Diagrama de clases: Relación de asociación

2. COMPOSICIÓN (relación fuerte)

Una relación todo-parte más fuerte. El objeto parte no puede existir sin el todo. El ciclo de vida de la parte depende completamente del todo. Si el objeto contenedor muere, las partes también. Se puede leer como "es parte de".

¿Para qué sirve?

- Modelar relaciones de propiedad exclusiva y ciclo de vida compartido.
- Muy útil cuando los objetos se crean y destruyen juntos.

Ejemplo:

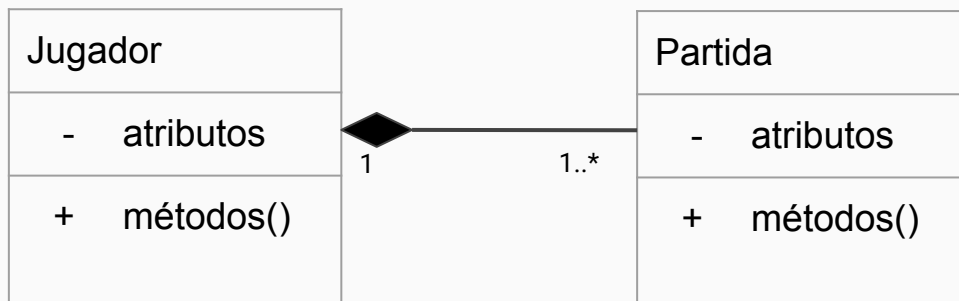


Diagrama de clases

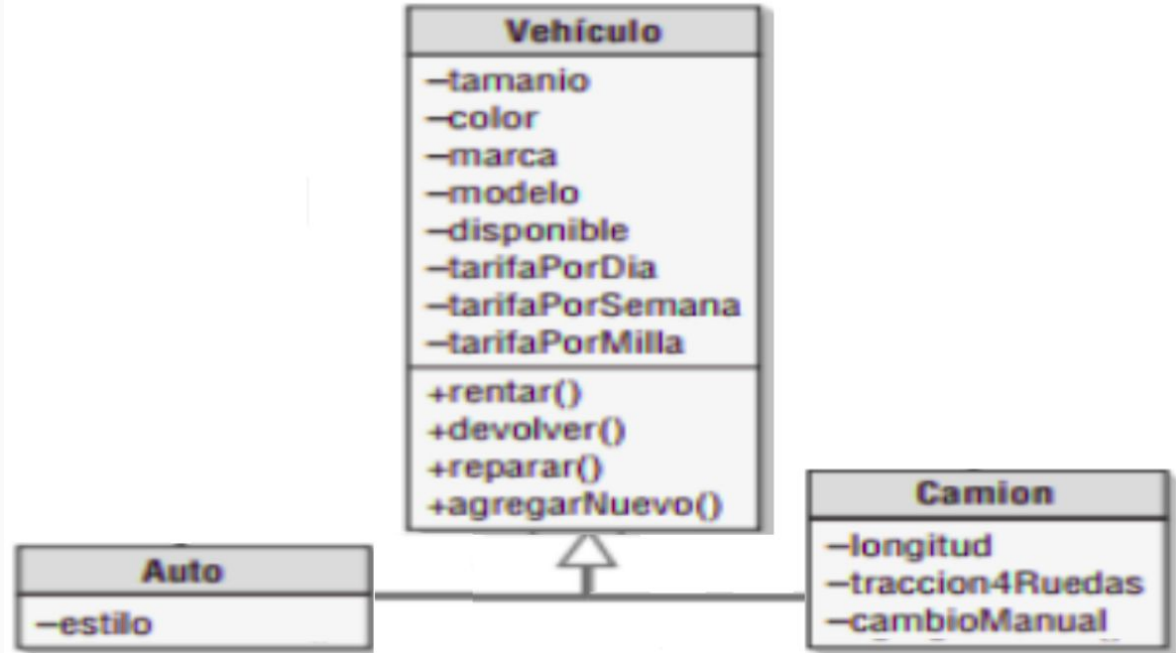
Resumen

Relación	¿Qué es?	¿Para qué sirve?	¿Existe dependencia de vida?	Explicación
Asociación	Relación genérica entre clases	Indicar uso o conocimiento	No	Totalmente independientes
Agregación	Parte-todo débil	Contener sin dependencia	Parcial	La parte puede vivir sola
Composición	Parte-todo fuerte	Contener con dependencia total	Sí	La parte no puede vivir sin el todo

Diagrama de clases

HERENCIA:

- Las clases pueden ser hijas de otras clases.
- Cuando esto ocurre se dice que la clase hija hereda los atributos y métodos de la clase padre.
- La clase padre se suele llamar clase base.
- La clase hija se suele llamar clase derivada.



Ejercitación

Ejercitación: UML

A partir de los siguientes enunciados, se pide generar:

1. el diagrama de casos de uso
2. un escenario de casos de uso (en base a un caso de uso seleccionado)
3. un diagrama de secuencia (en base a un caso de uso seleccionado)
4. el diagrama de clases. En base a este último diagrama:
 - a. Identificar las principales clases del sistema y sus relaciones.
 - b. ¿Qué atributos y métodos podría contener?
 - c. ¿Hay herencia? De ser así, ¿entre qué clases?
 - d. ¿Qué relaciones contiene el diagrama? (asociación, agregación, herencia, composición)

El propósito de esta actividad es que el estudiante:

- Modele un sistema complejo mediante el uso de diagramas UML.
- Practique el análisis de requisitos desde el punto de vista de diseño de videojuegos.
- Aplique principios de la programación orientada a objetos (POO): herencia, composición, abstracción.
- Visualice la interacción entre los distintos componentes de un videojuego multijugador.
- Integre lo aprendido en una propuesta de diseño coherente, escalable y modular.

UML - Desarrollo de un videojuego de plataformas 2D

1er enunciado:

Se busca desarrollar un videojuego 2D de plataformas, el cual se llamará "Aventura Pixelada", donde el jugador controla a un personaje que puede caminar, saltar, recoger monedas y evitar enemigos. El juego incluye un sistema de niveles, un temporizador y un contador de vidas.

El personaje principal puede:

- Caminar hacia la izquierda o derecha.
- Saltar y caer por gravedad.
- Recoger monedas para sumar puntos.
- Chocar con enemigos: si choca, pierde una vida.
- Morir si pierde todas las vidas.
- Superar el nivel al llegar a la meta.

El juego permite registrar y logear usuarios, guardar el progreso (nivel alcanzado y puntaje) y contiene las pantallas de inicio, de juego y fin de partida.

UML - Videojuego de estrategia "Reinos en Guerra"

2do enunciado:

"Reinos en Guerra" es un videojuego de estrategia en tiempo real (RTS) para PC en el que los jugadores deben administrar su propio reino mientras compiten contra otros jugadores en línea. El objetivo es dominar el mapa construyendo estructuras, recolectando recursos, entrenando unidades y atacando los reinos enemigos hasta eliminar a todos los oponentes o conquistar ciertos puntos estratégicos. El juego es multijugador (2 a 4 jugadores por partida).

Cada jugador puede:

- Iniciar sesión o registrarse con un nombre de usuario.
- Unirse a una partida con 2 a 4 jugadores.
- Controlar un Reino donde puede:
 - Construir edificios: Cuarteles (para entrenar unidades), Minas (para obtener oro), Granjas (para generar comida).
 - Recolectar recursos: Oro, madera y comida.
 - Entrenar unidades militares (Soldados, Arqueros, Caballeros).
 - Asignar unidades a grupos de ataque o defensa.
 - Atacar otros reinos, destruyendo sus estructuras y eliminando sus tropas.
 - Defender su base automáticamente o mediante órdenes.
 - Mejorar edificios y unidades (más velocidad, daño, vida, capacidad de producción).

La partida termina cuando un jugador:

- Elimina todos los demás reinos.
- Captura los puntos de control estratégicos del mapa.
- O se alcanza un tiempo límite y se declara ganador por puntaje.

UML - Videojuego de Rol Multijugador (RPG Online) "Crónicas del Reino"

3er enunciado:

"Crónicas del Reino" es un RPG multijugador online ambientado en un mundo medieval fantástico. Cada jugador crea un personaje (guerrero, mago o arquero) y se une a un mundo persistente con otros jugadores. Puede explorar el mundo, combatir enemigos, completar misiones, mejorar su equipamiento, unirse a clanes y participar en eventos de mundo abierto como asaltos a mazmorras o guerras entre clanes.

Cada jugador puede:

- Registrarse e iniciar sesión en el mundo persistente.
- Crear un personaje eligiendo raza y clase (guerrero, mago, arquero).
- Moverse por el mapa del mundo (bosques, ciudades, mazmorras).
- Interactuar con NPCs para recibir misiones o comerciar.
- Completar misiones que otorgan experiencia y objetos.
- Combatir enemigos o monstruos salvajes.
- Usar habilidades especiales, con tiempos de recarga y consumo de maná.
- Recoger objetos (espadas, pociones, oro).
- Equiparse y mejorar su equipo mediante herreros o alquimistas.
- Unirse a un clan para formar alianzas.
- Participar en eventos PvP o PvE (jugador vs jugador o jugador vs entorno).
- Guardar su progreso (nivel, experiencia, inventario).