Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

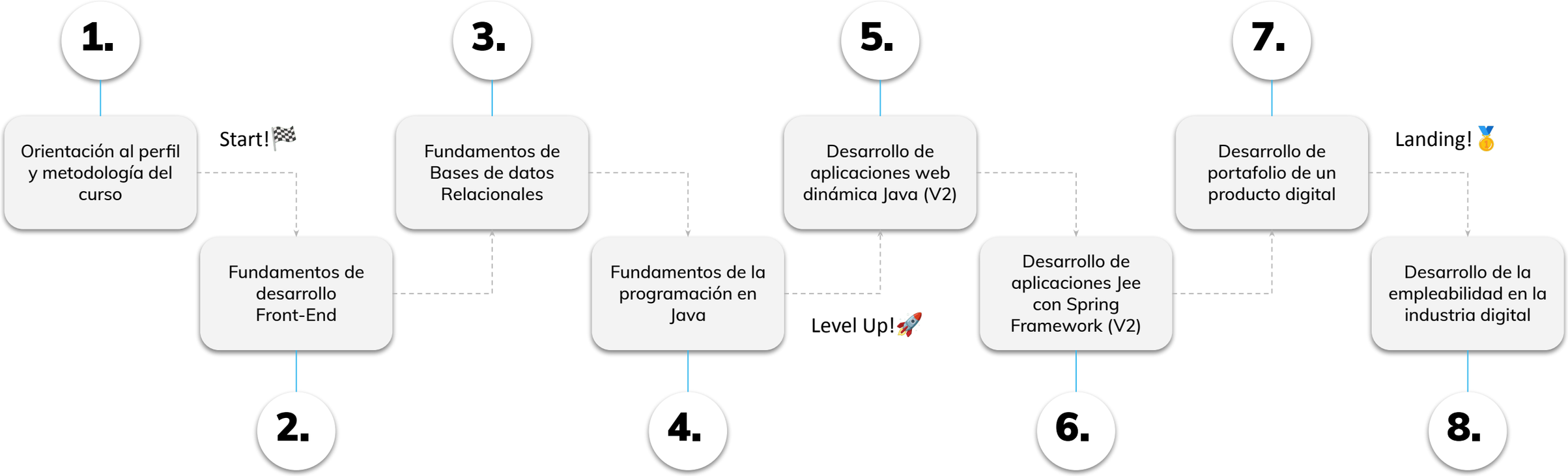
El Paradigma de Orientación a Objetos

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✓ *Relaciones entre clases*

English Always | alkemy

LEARNING PATHWAY

4.5 Start! 🏁

El Paradigma de Orientación a Objetos → Diagrama de clases → Diagramando

Diagrama de clases

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- ***Comprender la representación de clases a través de diagramas de flujo***

Rompehielo🥶

¿Qué crees que representa la imagen?:
🙌

Respondan levantando la mano o en el chat!

- ¿Para qué crees que sirve este tipo de gráfico?
- ¿La información puede reflejarse en otro lado?
- ¿Que funcion le darias en programación?

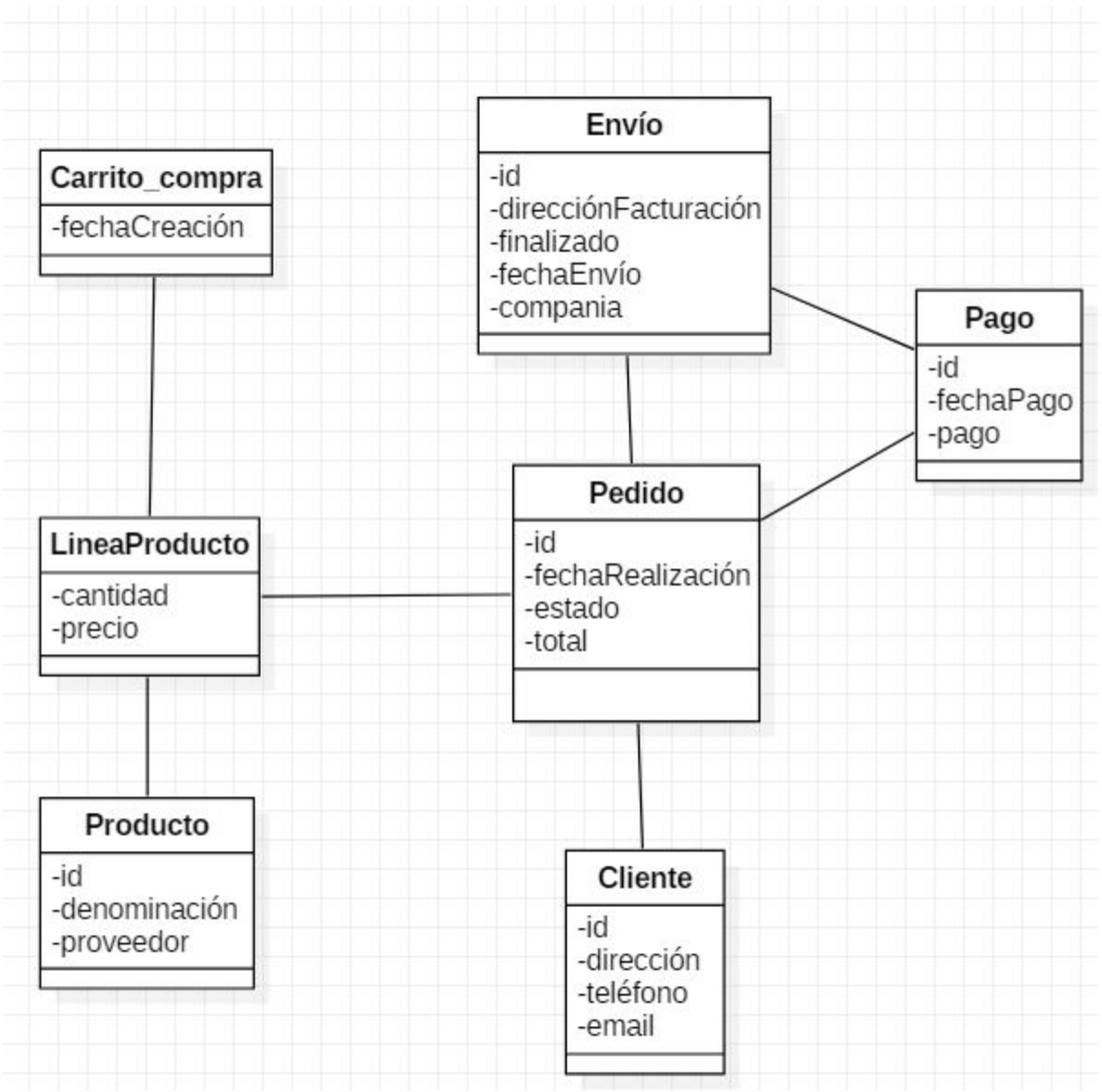


Diagrama de Clases

Diagrama de Clases

¿Qué es y para qué se utiliza?

El diagrama de clases es un diagrama puramente orientado al modelo de programación orientado a objetos, ya que **define las clases que se utilizarán** cuando se pase a la fase de construcción **y la manera en que se relacionan** las mismas. Su utilidad es la representación de datos y su interacción. Este diagrama **muestra el modelo lógico de los datos de un sistema.**

Diagrama de Clases

Cada clase está representada por un rectángulo que tiene una subdivisión de tres zonas: **nombre**, **atributos** y **métodos**.

- La primera de las zonas se utiliza para el nombre de la clase. En caso de que la clase sea abstracta se utilizará su nombre en cursiva.

- La segunda de las zonas se utiliza para escribir los atributos de la clase, uno por línea.

- La última de las zonas incluye cada uno de los métodos que ofrece la clase.

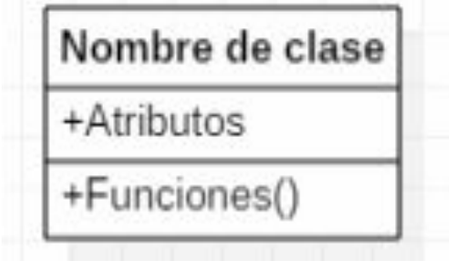


Diagrama de Clases

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

- **(+) public**: Representa que se puede acceder al atributo o función desde cualquier lugar de la aplicación.
- **(-) private**: Representa que se puede acceder al atributo o función únicamente desde la misma clase.
- **(#) protected**: Representa que el atributo o función puede ser accedida únicamente desde la misma clase o desde las clases que hereden de ella (clases derivadas).

Estos tres tipos de visibilidad son los más comunes. No obstante, pueden incluirse otros en base al lenguaje de programación que se esté usando (no es muy común). Por ejemplo: (/) Derivado o (~) Paquete.

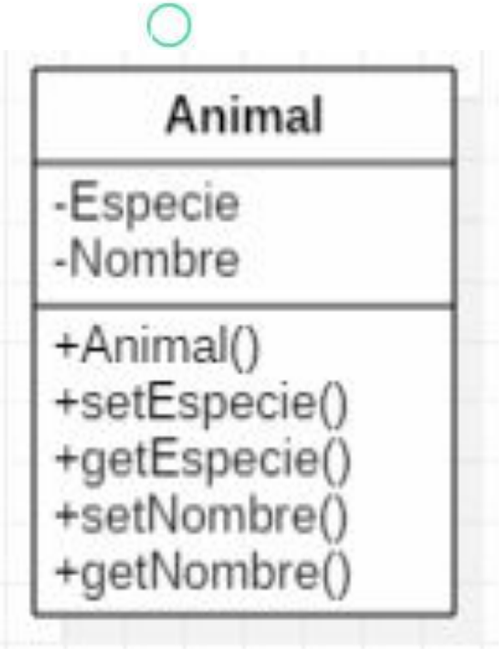


Diagrama de Clases

Relaciones:
Las relaciones en el diagrama de clases tienen varias propiedades, que dependiendo la profundidad que se quiera dar al diagrama se representarán o no. Estas propiedades son las siguientes:

- **Multiplicidad**. Es decir, el número de elementos de una clase que participan en una relación. Se puede indicar un número, un rango... Se utiliza **n** o ***** para identificar un número cualquiera.

- **Nombre de la asociación**. En ocasiones se escriba una indicación de la asociación que ayuda a entender la relación que tienen dos clases. Suelen utilizarse verbos como por ejemplo: «Una empresa **contrata** a n empleados»

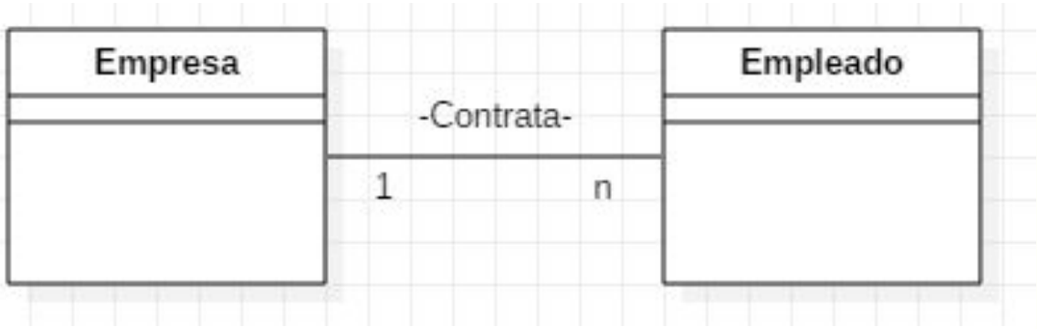
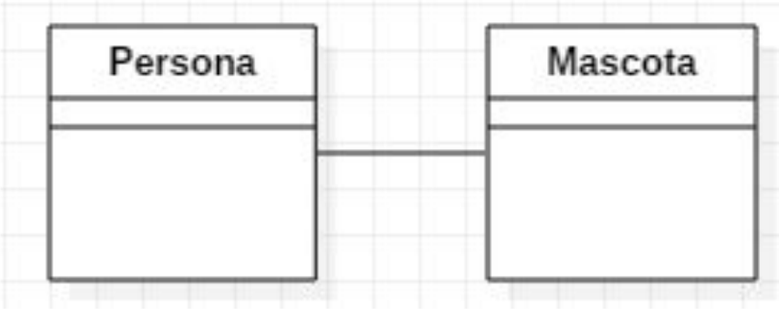


Diagrama de Clases

Relación de Asociación:

Este tipo de relación es el más común y 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:

«Una mascota **pertenece** a **una** persona**»**.

Diagrama de Clases

Relación de Agregación:

Es una representación jerárquica que indica a un objeto y las partes que componen ese objeto. Es decir, **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).

Un ejemplo de esta relación podría ser: «Las mesas **están formadas por** tablas de madera y tornillos o, dicho de otra manera, los tornillos y las tablas **forman parte de** una mesa».

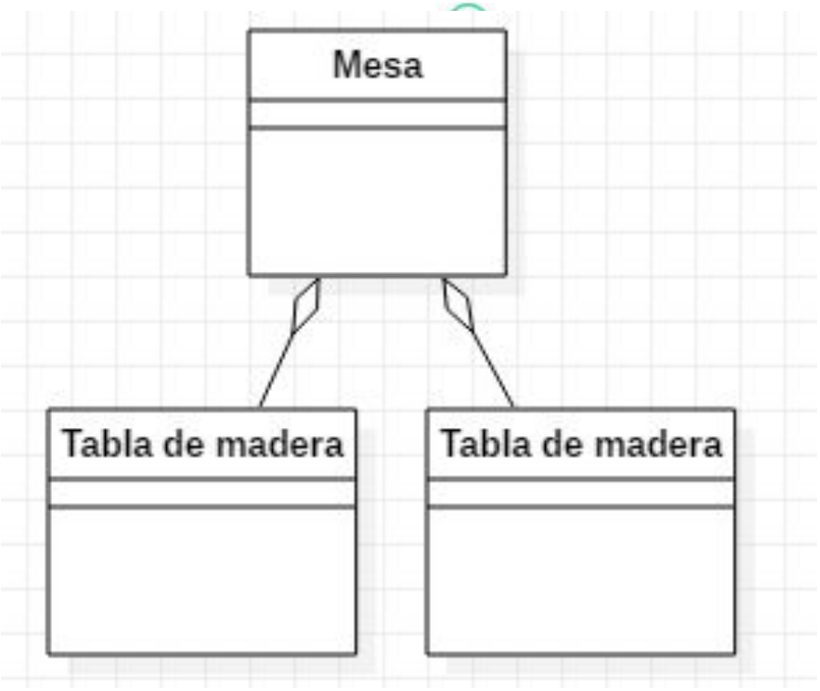


Diagrama de Clases

Relación de Composición:

En este caso, l**os elementos que forman parte no tienen sentido de existencia cuando el primero no existe**. Es decir, cuando el elemento que contiene los otros desaparece, deben desaparecer todos ya que no tienen sentido por sí mismos sino que dependen del elemento que componen.

Se representa con una **línea continua con un rombo relleno** en la clase que es compuesta.

Un ejemplo de esta relación sería: «Un vuelo de una compañía aérea **está compuesto por** pasajeros, qué es lo mismo que decir que un pasajero **está asignado a** un vuelo»

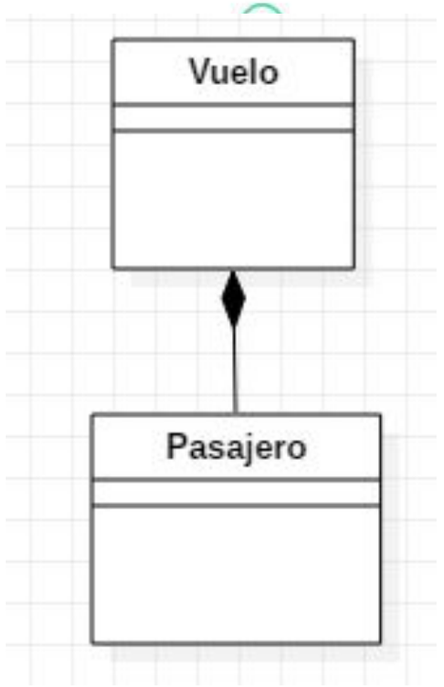


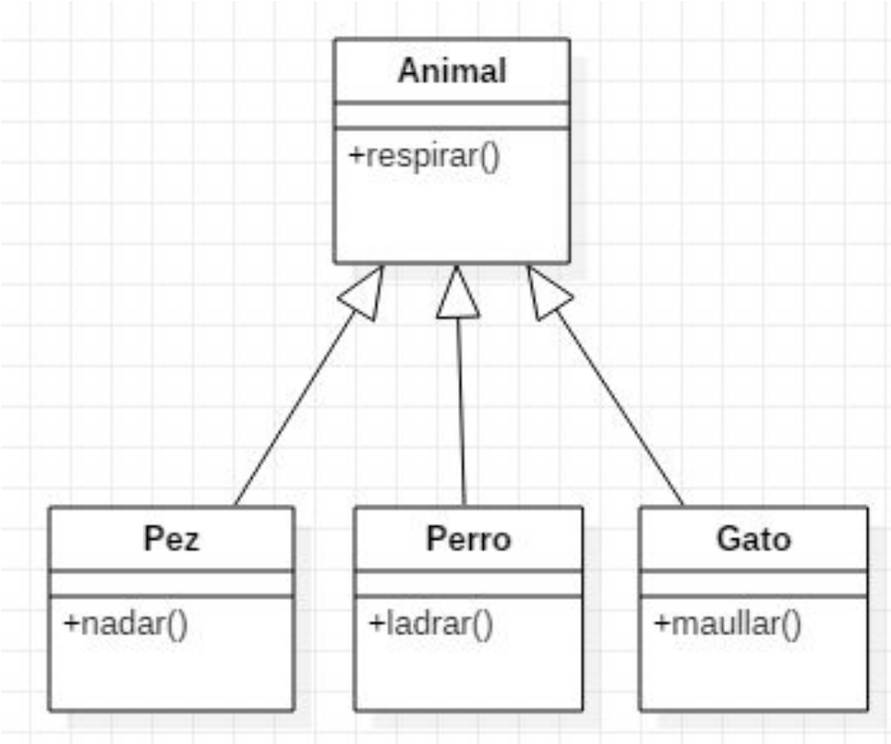
Diagrama de Clases

Relación de Herencia:

Este tipo de relaciones **permiten que una clase** (clase hija o subclase) **reciba los atributos y métodos de otra clase** (clase padre o superclase). Estos atributos y métodos recibidos se suman a los que la clase tiene por sí misma. Se utiliza en relaciones **«es un»**.

Un ejemplo de esta relación podría ser la siguiente: Un pez, un perro y un gato **son** animales.

En este ejemplo, las tres clases (Pez, Perro, Gato) podrán utilizar el método **respirar**, ya que lo heredan de la clase animal, pero solamente la clase Pez podrá nadar, la clase Perro ladrar y la clase Gato maullar.



Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro....

Trabajo Integrador del Módulo 💪🏽

Iremos completándolo progresivamente clase a clase.

English Always | alkemy

LIVE CODING

Ejemplo en vivo

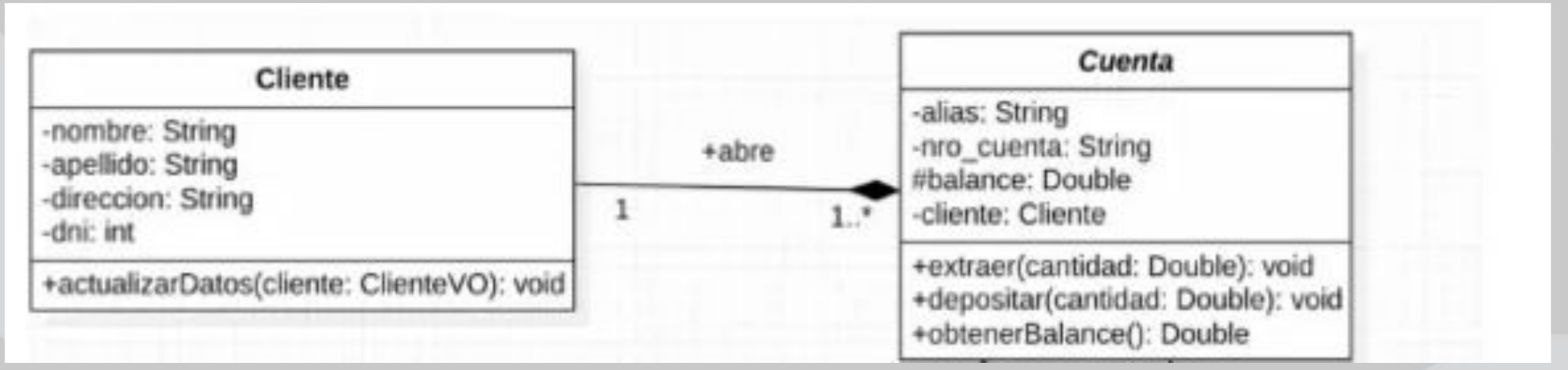
Creando programas a partir del diagrama:

En programación es habitual crear primero la estructura del programa que resuelve una necesidad determinada en un diagrama de clases y luego codificar. Es por esto que en este primer ejemplo realizaremos el siguiente ejercicio.

1. *Crear un programa que respete el diagrama de clases dado.*

LIVE CODING

Ejemplo en vivo



Tiempo: 30 minutos

English Always | alkemy

Ejercicio N° 1

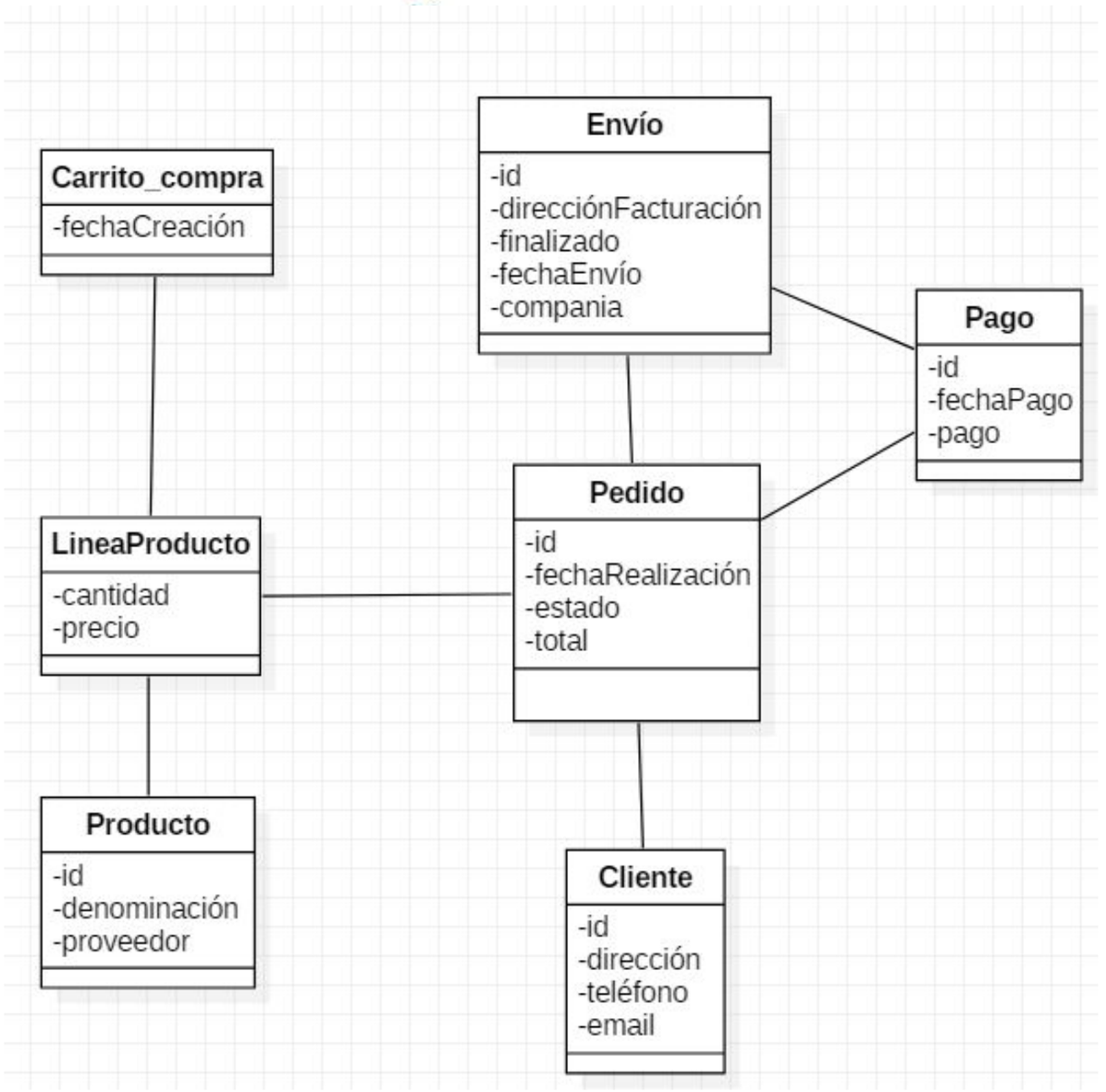
Diagramando

Diagramando

Consigna ✍️:

Vamos a practicar la creación de programas basándonos en el siguiente diagrama de clases:

Tiempo🕐: *30 minutos*



¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ ***Comprender la utilidad de los diagramas de clase para la consecuente codificación***

English Always | alkemy

#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 👇👇👇

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
   a. Material 1 *(Foro)*
   b. *Lectura Módulo 4, Lección 5: páginas 10 - 12*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 👋🏻

English Always | alkemy

Momento:

Time-out!

⏰5 min.