Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

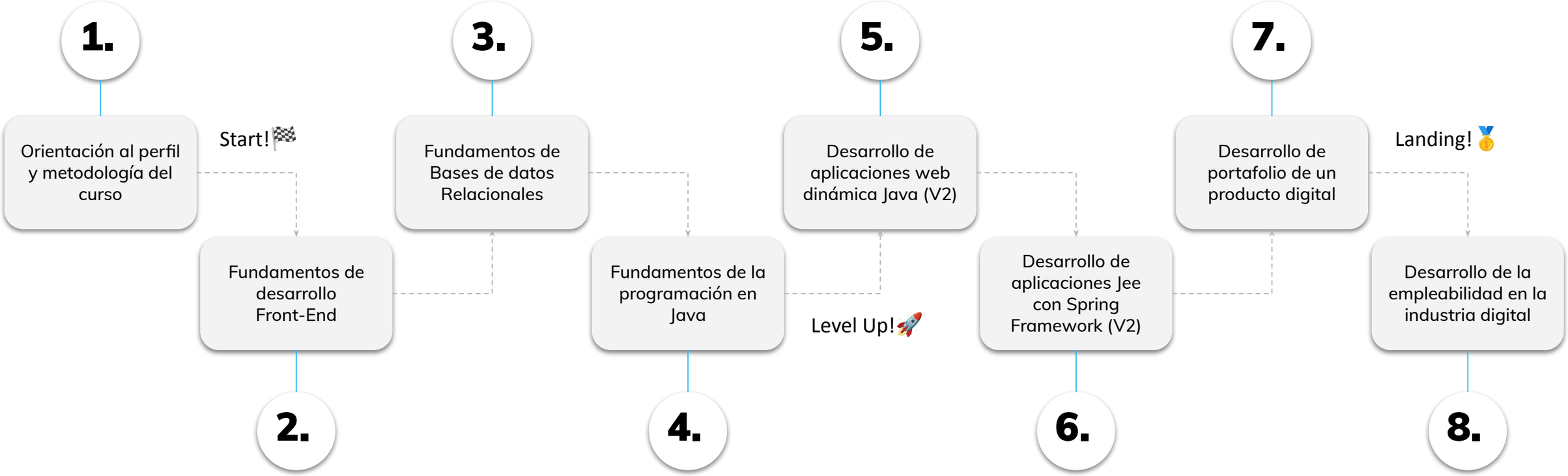
Polimorfismo y principios básicos de diseño

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 📚:

- ✓ *Herencia y Métodos*

English Always | alkemy

LEARNING PATHWAY

4.6 Start!🏁

Polimorfismo y principios básicos de diseño → Polimorfismo mediante Herencia → Veterinaria 3.0

Polimorfismo.

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Comprender el concepto de polimorfismo en programación orientada a objetos*
- *Aprender a implementar el polimorfismo mediante herencia*

Rompehielo🥶

¿Qué ven en esta imagen?: 🙌
Respondan en el chat,
¿Coincidieron?

Claramente, podemos referirnos a distintos tipos de objetos usando una clase padre común. A fines prácticos, ¿cómo lo definirías en programación?

Polimorfismo

Polimorfismo

¿Qué es el polimorfismo?:

En programación orientada a objetos, polimorfismo **es la capacidad que tienen los objetos de una clase en ofrecer respuesta distinta e independiente en función de los parámetros** (diferentes implementaciones) **utilizados durante su invocación**.
Dicho de otro modo el objeto como entidad puede contener valores de diferentes tipos durante la ejecución del programa.

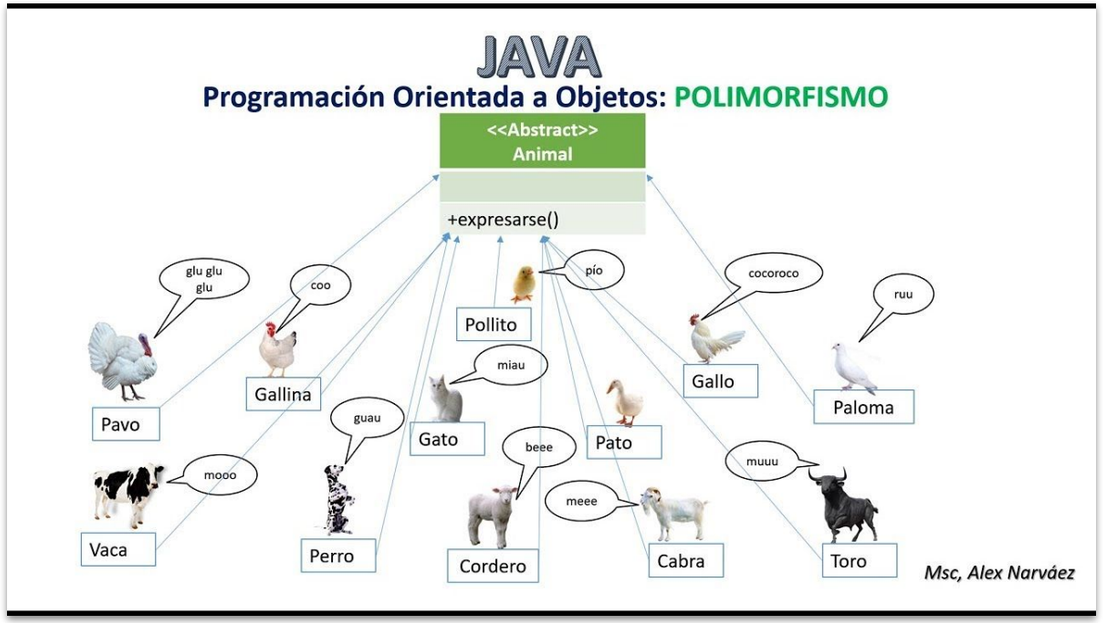
Basicamente, el polimorfismo en Java alude al **modo en que se pueden crear y utilizar dos o más métodos con el mismo nombre para ejecutar funciones diferentes**.

Polimorfismo

Veamos un ejemplo:

Imagina que tienes una mascota. Puede ser un perro, un gato o un pájaro. Cada uno de ellos tiene sus propias características y comportamientos específicos, como ladrar, maullar o volar. Ahora, piensa en cómo podrías representar todas estas mascotas en un programa de Java.

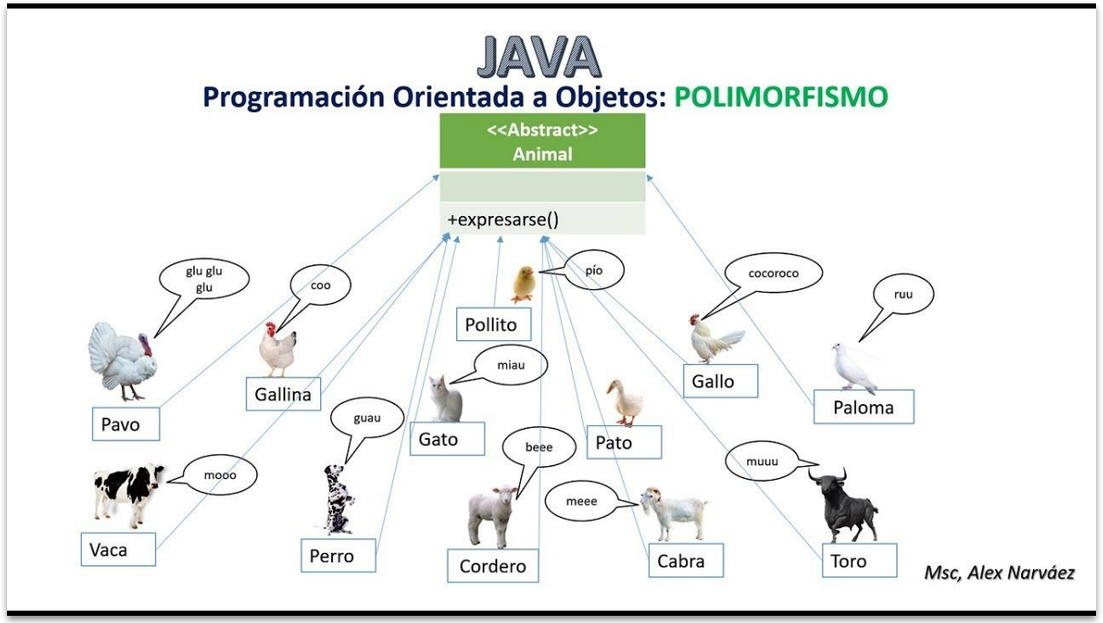
En lugar de crear una clase separada para cada tipo de mascota, podemos utilizar el polimorfismo para tratar a todas las mascotas de manera general. Esto significa que podemos tener una clase genérica llamada "Mascota" y luego crear subclases como "Perro", "Gato" y "Pájaro" que heredan de la clase "Mascota".



Polimorfismo

¿Pero, es igual a la clase Animal de los ejercicios?:

En realidad en este caso, por ejemplo, **la magia del polimorfismo ocurre cuando creamos un arreglo o una lista de mascotas**.
Podemos agregar instancias de diferentes subclases en la misma lista y **tratarlas como si fueran del tipo genérico "Mascota"**.
Cuando llamamos al método "hacerSonido()" en una mascota particular, Java se encarga de invocar el comportamiento correspondiente según el tipo de mascota real. Por ejemplo, si la mascota es un perro, se ladrará; si es un gato, se maullará; y así con todas las mascotas que agregues.



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

Apliquemos lo aprendido en el proyecto de Billetera Virtual:

1- Declararemos una superclase FormaDePago con el método void realizarPago():

2- Luego se crean subclases TarjetaDeCrédito y Moneda que heredan de FormaDePago. La particularidad del pago con tarjeta es poder elegir cantidad de cuotas, mientras que Moneda puedes elegir el tipo de moneda.

3- En la clase Billetera se debe declarar una referencia de tipo FormaDePago: FormaDePago metodoPago;

LIVE CODING

Ejemplo en vivo

4- De esta manera podremos asignar el método de pago según se necesite:

metodoPago = new TarjetaDeCredito();

metodoPago = new Moneda();

Y utilizar polimorfismo al llamar: metodoPago.realizarPago();

Tiempo: 30 minutos

Ejercicio N° 1

Veterinaria 3.0

Veterinaria 3.0

Seguiremos trabajando sobre la clase Animal: 🙌

Vamos a manipular la clase Animal que contenga un método hacerRuido() que devuelva un saludo “Hola”.

Luego, las clases Perro y Gato (que extienden de Animal) deben sobreescribir el método hacerRuido() con el ruido que corresponda a cada uno.

Consigna: ✍️ En el main vamos a crear un ArrayList de animales y los siguientes animales:

- Animal a = new Animal();
- Animal b = new Perro();
- Animal c = new Gato();

Agregar a la lista a cada uno y luego, con un for each, recorrer la lista llamando al método hacerRuido() de cada ítem.

Tiempo🕐: *30 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender el uso del polimorfismo en herencia*
- ✓ *Reconocer la correcta implementación del polimorfismo en programación orientada a objetos*

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 6: página 4*
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.