Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

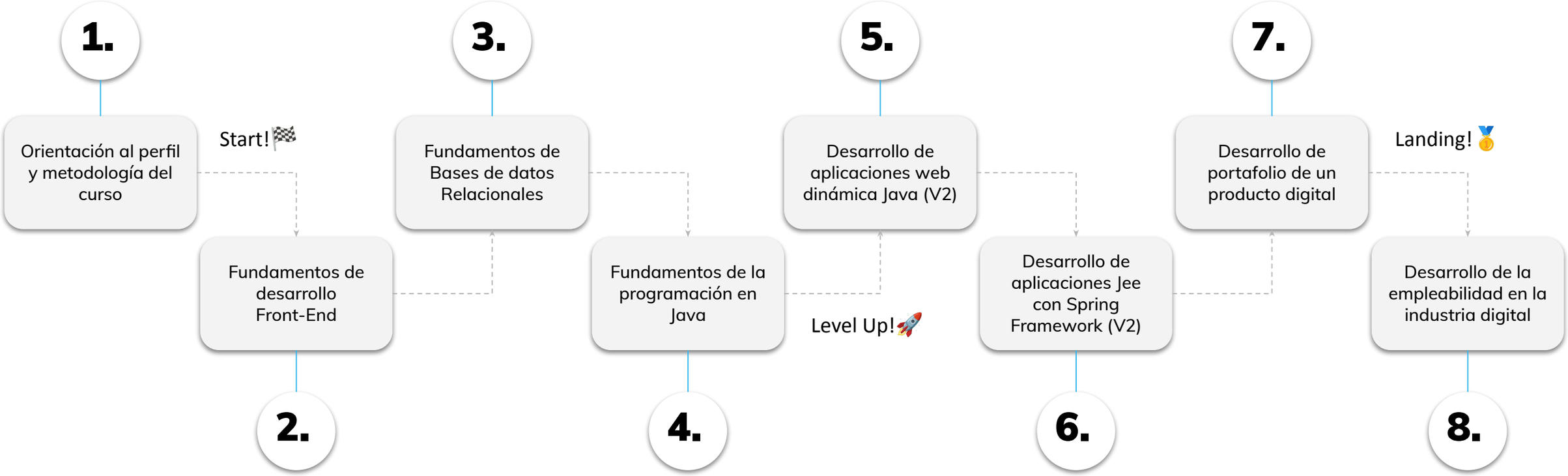
Polimorfismo y principios básicos de diseño

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 📚:

- ✔ *Polimorfismo y herencia*

English Always | alkemy

LEARNING PATHWAY

4.6 Start! 🏁

Polimorfismo y principios básicos de diseño → Clases y métodos abstractos → Figura

Clases y métodos abstractos.

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Comprender los casos de uso e implementación de las clases abstractas*
- *Aprender a manipular métodos abstractos*

Rompehielo🥶

Respondan en el chat o levantando la mano: 🙌

Imaginen que están en un restaurante y piden un plato combinado. El mesero **solo les trae la receta** con los ingredientes y pasos a seguir, pero no la comida preparada.

1. ¿Podríamos comer la receta?
2. Si la receta es la estructura, ¿debería ser extendida (preparada) para poder ser utilizada?
3. ¿Qué ejemplo relacionado pueden pensar en programación?

Clases abstractas

Clases abstractas

¿Qué son y para qué sirven?:

Las clases abstractas son una característica importante de la programación orientada a objetos en Java. Una clase abstracta es una clase que **no se puede instanciar** directamente y se utiliza como base para otras clases.

Habrá ocasiones en las cuales necesitemos crear una clase padre donde únicamente coloquemos la estructura de una abstracción, una estructura muy general, dejando que sean las clases hijas quienes definan los detalles. En estos casos haremos uso de las clases abstractas.

Clases abstractas

¿Qué son y para qué sirven?:

Usualmente las clases abstractas suelen ser las **superclases**, esto sucede porque creemos que la superclase o clase padre no debería poder instanciarse. Por ejemplo, si tenemos una clase Animal, el usuario no debería poder crear un Animal, sino que sólo debería poder instanciar objetos de las subclases (Perro, Gato, etc).

public **abstract** class Animal { }

Una clase abstracta es practicamente identica a una clase convencional; las clases abstractas **pueden poseer atributos, métodos, constructores, etc** ... La principal diferencia entre una clases convencional y una clase abstracta es que **la clase abstracta debe poseer por lo menos un método abstracto.**

Métodos abstractos

Métodos abstractos

¿Qué son y para qué se utilizan?:

Un método abstracto para Java es un método que **nunca va a ser ejecutado porque no tiene cuerpo**. Simplemente, un método abstracto referencia a otros métodos de las subclases.

¿Qué utilidad tiene un método abstracto? Podemos ver un método abstracto como una palanca que fuerza dos cosas: la primera, q**ue no se puedan crear objetos de una clase**. La segunda, que **todas las subclases sobreescriban el método** declarado como abstracto.

Métodos abstractos

Los métodos abstractos **deben ser declarados en clases abstractas y deben ser implementados por las clases que heredan** de la clase abstracta.

Los métodos abstractos se escriben **sin llaves {} y con ; al final** de la declaración.

Por ejemplo:

public **abstract** double area**();**

Un método se declara como abstracto porque en ese momento (en esa clase) no se conoce cómo va a ser su implementación.

Clases y métodos abstractos

Veamos un ejemplo:

En este caso la clase Figura posee una atributo, un constructor y un método, a partir de esta clase podré generar la n cantidad de figuras que necesite, ya sean cuadrados, rectangulos, triangulos, circulos etc...

Dentro de la clase encontramos el **método área**, método que se encuentra pensado para obtener el área de cualquier figura, sin embargo cómo sabemos todas las figuras poseen su propia fórmula matemática para calcular su área.

```
public class Figura {

    private int numeroLados;

    public Figura() {

        this.numeroLados = 0;
    }

    public float area() {
        return 0f;
    }
}
```

Clases y métodos abstractos

Si comenzamos a **heredar de la clase Figura,** todas las clases hijas tendrían que **sobreescribir el método área** e implementar su propia fórmula para así poder calcular su área. En estos casos en los que la clase hija siempre deba sobreescribir el método, es cuando debemos convertir al método convencional en un método abstracto, un método que defina qué hacer, pero no cómo se deba hacer. 😀

```
public abstract float area();
```

Clases y métodos abstractos

Ahora que el método área es un método abstracto la clase se convierte en una clase abstracta.

Es importante recordar que las clases abstractas pueden ser heredadas por la n cantidad de clases que necesitemos, pero **no pueden ser instanciadas**.

Al heredar de una clase abstracta **es obligatorio implementar todos sus métodos abstractos**, es decir debemos definir comportamiento, definir cómo se va a realizar la tarea.

```
public abstract class Figura {
```

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

Vamos a practicar la codificación para implementar clases y métodos abstractos:

1. *Declarar una clase abstracta CuentaDigital*
2. *Declarar un método abstracto verificarFondos()*
3. *Crear las subclases CuentaCL y CuentaUSD extendiendo de CuentaDigital.*
4. *Crear la lógica necesaria para implementar en cada método.*

Tiempo: 30 minutos

Ejercicio N° 1

Figura

Figura

Calculando el área: 🙌

Como vimos en el ejemplo de la clase, las clases y métodos abstractos nos facilitan la implementación de particularidades dentro de las subclases. Pongamos en práctica lo aprendido!

Consigna: ✍️

Declarar una clase abstracta **Figura** con un método abstracto **calcularArea()**.

Extenderla en clases Rectangulo, Triangulo y Circulo implementando el método y mostrando los resultados por pantalla.

Tip: ⚙️

- **Rectangulo**: area = base * altura
- **Triangulo**: (base * altura) / 2
- **Circulo**: PI * radio * radio

Tiempo🕐**:** *30 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ ***Comprender la implementación de las clases abstractas.***
- ✓ ***Reconocer la funcionalidad de los métodos abstractos en herencia.***

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: páginas 4 - 5*
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.