Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

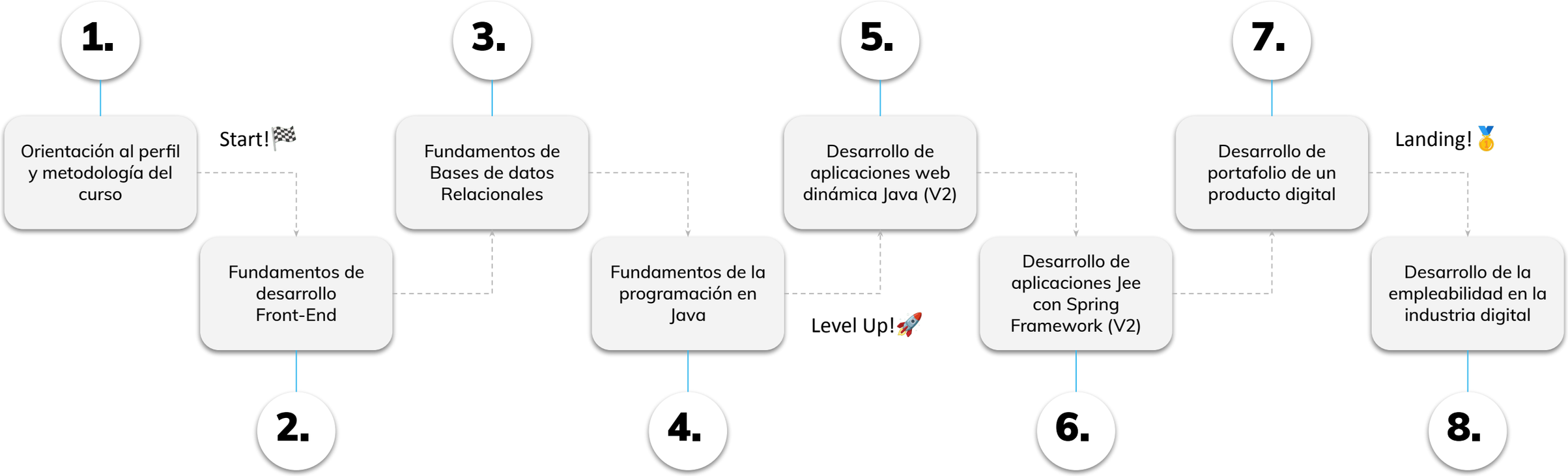
Polimorfismo y principios básicos de diseño

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 📚:

- ✓ *Principio de Liskov*

English Always | alkemy

LEARNING PATHWAY

4.6 Start!🏁

Polimorfismo y principios básicos de diseño

Principio de Segregación de Interfaces

Principio de Segregación de Interfaces

Segregando

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Comprender la implementación del principio de segregación de interfaces*

Rompehielo🥶

Imaginen un supermercado con cajas para diferentes tipos de cliente: una para menos de 15 productos, otra para carros grandes. ¿Sería eficiente tener una única caja para todos?

Respondan en el chat o levantando la mano: ✍️

- ¿Por qué existen cajas diferentes en el supermercado?
- ¿Qué pasaría si solo hubiera una caja para todos?
- Al tener cajas especializadas, ¿quién se beneficia?
- Si se necesitan cajas rápidas para clientes con ticket de comida, ¿se debería agregar esta funcionalidad a todas las cajas o crear una nueva especializada?

Con esto introducimos el concepto de que es mejor tener **interfaces pequeñas y especializadas** según tipos de clientes/contextos diferentes, en lugar de una interfaz grande con todo junto.

Principio de Segregación de Interfaces

Segregación de Interfaces

¿Qué es?:

El principio de segregación de interfaces (ISP, por sus siglas en inglés: Interface Segregation Principle) es uno de los cinco principios SOLID de la programación orientada a objetos (POO). Este principio fue propuesto por Robert C. Martin y se centra en la estructuración adecuada de las interfaces en una aplicación. El ISP se formula de la siguiente manera:

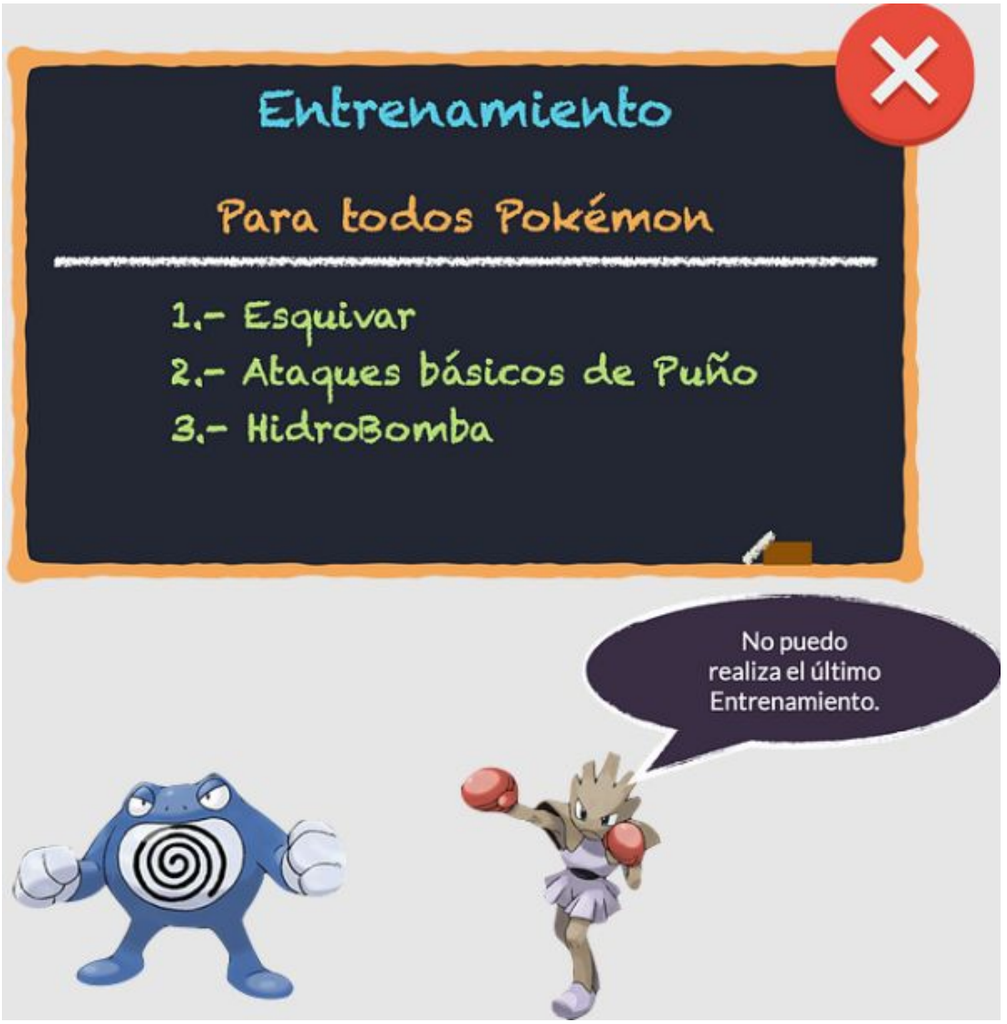
"**Ninguna clase debe verse forzada a implementar métodos que no utiliza, y las interfaces deben ser específicas para las necesidades de las clases que las utilizan.**"

El ISP se basa en la idea de que las **interfaces deben ser pequeñas, cohesivas y específicas** para cada clase que las implementa.

Segregación de Interfaces

Algunos conceptos clave:

- **Interfaces Coherentes**: Los métodos en una interfaz deben estar relacionados en función del contexto y de las responsabilidades de la clase que la implementa.
- **Evitar Interfaces Monolíticas**: Las interfaces grandes y complejas pueden obligar a las clases a implementar métodos que no necesitan, lo que va en contra del ISP.
- **División de Interfaces**: Si una clase necesita diferentes funcionalidades o comportamientos, es preferible dividir las interfaces en interfaces más pequeñas y específicas.
- **Aplicación en Herencia Múltiple**: Especialmente relevante en lenguajes de programación que admiten implementación de múltiples interfaces. En estos casos, las clases pueden seleccionar las interfaces que desean implementar de manera más granular.



Segregación de Interfaces

Beneficios de Cumplir con el ISP:

- **Mantenimiento Simplificado**: Las interfaces específicas y cohesivas facilitan el mantenimiento del código, ya que los cambios en una interfaz afectan solo a las clases que realmente la implementan.
- **Reutilización Mejorada**: Las interfaces específicas hacen que sea más fácil reutilizar clases en diferentes contextos, ya que una clase puede implementar solo las interfaces relevantes para una situación particular.
- **Flexibilidad**: Permite una mayor flexibilidad en la composición de objetos, ya que las clases pueden seleccionar las interfaces que necesitan.

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 DEMO

Ejemplo en vivo

Aplicando principio de Segregación de Interfaces:

Supongamos que eres el desarrollador a cargo del sistema de gestión de la nueva línea de impresoras multifuncionales de Xerox.

Este nuevo sistema se desarrollará desde cero, por lo tanto estarás a cargo de los futuros desarrollos de las diferentes impresoras que se originen de la nueva línea de multifuncionales.

Tiempo: 30 minutos

LIVE DEMO

Ejemplo en vivo

Esta nueva línea cuenta con las siguientes capacidades:

- *Impresión.*
- *Escaneo.*
- *Faxeo.*

```java
public interface MultifuncionalLX {

    /**
     * Método encargado de imprimir el siguiente documento
     * activo en la cola de impresión.
     *
     * @return      Retorna boolean indicando el resultado.
     */
    public boolean imprimirDocumento();

    /**
     * Método encargado de escanear el documento cargado
     * en la bandeja de escaneo.
     *
     * @return      Retorna boolean indicando el resultado.
     */
    public boolean escanearDocumento();

    /**
     * Método encargado de faxear el documento cargado
     * en la bandeja de faxeo.
     *
     * @return      Retorna boolean indicando el resultado.
     */
    public boolean faxearDocumento();
}
```

LIVE DEMO

Ejemplo en vivo

Para llevar a cabo el desarrollo se define la siguiente interfaz. La cual establece los métodos básicos que tendrán las multifuncionales de la nueva línea.

```
public class ImpresoraLX8578 implements MultifuncionalLX {

    @Override
    public boolean imprimirDocumento() {
        //Lógica de impresión de documentos
        return true;
    }

    @Override
    public boolean escanearDocumento() {
        //Lógica de escaneo de documentos
        return true;
    }

    @Override
    public boolean faxearDocumento() {
        //Lógica de faxeo de documentos
        return true;
    }

}
```

LIVE DEMO

Ejemplo en vivo

Durante varios meses se lanzan diferentes versiones de la multifuncional, las cuales utilizan la misma interfaz para definir sus funcionalidades principales.

Gracias a la aceptación de la nueva línea de impresoras, la gerencia te solicita la integración del sistema de gestión de impresiones ***a una nueva impresora económica****, que será lanzada bajo la línea de impresoras que está bajo tu responsabilidad.* ***La diferencia es que esta nueva impresora no contará con la capacidad de enviar faxes****, únicamente impresión y escaneo.*

Tomando en cuenta que esta nueva impresora pertenece a la misma línea que las impresoras anteriores y también es categorizada como multifuncional, ***decides utilizar la interfaz definida para dicha línea****.*

LIVE DEMO

Ejemplo en vivo

En este punto es donde se da la violación al Principio de Segregación de Interfaces, debido a que la impresora ImpresoraLX8590 es forzada a depender de métodos que no utiliza.

```java
public class ImpresoraLX8590 implements MultifuncionalLX {

    @Override
    public boolean imprimirDocumento() {
        //Lógica de impresión de documentos
        return true;
    }

    @Override
    public boolean escanearDocumento() {
        //Lógica de escaneo de documentos
        return true;
    }

    @Override
    public boolean faxearDocumento() {
        throw new UnsupportedOperationException("Impresora no cuenta con la capacidad de faxeo.");
    }

}
```

LIVE DEMO

Ejemplo en vivo

¿Cómo solucionamos ésta problemática?

La solución al ejemplo anterior, es la ***segregación de la interfaz que agrupa todos los métodos en diferentes interfaces especializadas****. Por lo tanto, la interfaz MultifuncionalLX se convertirá en las interfaces:*

- *MultifuncionalLXImpresion*
- *MultifuncionalLXEscaneo*
- *MultifuncionalLXFaxeo*

LIVE DEMO

Ejemplo en vivo

```java
public interface MultifuncionalLXEscaneo {

    /**
     * Método encargado de escanear el documento cargado
     * en la bandeja de escaneo.
     *
     * @return      Retorna boolean indicando el resultado.
     */
    public boolean escanearDocumento();
}
```

```java
public interface MultifuncionalLXFaxeo {

    /**
     * Método encargado de faxear el documento cargado
     * en la bandeja de faxeo.
     *
     * @return      Retorna boolean indicando el resultado.
     */
    public boolean faxearDocumento();
}
```

```java
public interface MultifuncionalLXImpresion {

    /**
     * Método encargado de imprimir el siguiente documento
     * activo en la cola de impresión.
     *
     * @return      Retorna boolean indicando el resultado.
     */
    public boolean imprimirDocumento();
}
```

LIVE DEMO

Ejemplo en vivo

Luego se modifican las clases de las impresoras anteriores para que utilice las interfaces especializadas.

```
public class ImpresoraLX8578 implements MultifuncionalLXEscaneo,
        MultifuncionalLXFaxeo,
        MultifuncionalLXImpresion{

    @Override
    public boolean imprimirDocumento() {
        //Lógica de impresión de documentos
        return true;
    }

    @Override
    public boolean escanearDocumento() {
        //Lógica de escaneo de documentos
        return true;
    }

    @Override
    public boolean faxearDocumento() {
        //Lógica de faxeol de documentos
        return true;
    }

}
```

LIVE DEMO

Ejemplo en vivo

Se define la clase ImpresoraLX8590 únicamente con las interfaces de escaneo e impresión.

```
public class ImpresoraLX8590 implements MultifuncionalLXEscaneo,
        MultifuncionalLXImpresion{

    @Override
    public boolean imprimirDocumento() {
        //Lógica de impresión de documentos
        return true;
    }

    @Override
    public boolean escanearDocumento() {
        //Lógica de escaneo de documentos
        return true;
    }

}
```

Las modificaciones realizadas permiten que el código se encuentre en cumplimiento con el Principio de Segregación de Interfaces.

Ejercicio N° 1

Segregando

Segregando

A trabajar con la clase Animal: 🙌

Vamos a hacer que la superclase Animal con la que trabajamos anteriormente, se transforme en una clase abstracta.

Esta clase va a heredar sus componentes a las clases: Perro, Gato, Condor y Pingüino.

Consigna: ✍️

1- Crear los métodos; hacerRuido(), volar(), picotear().

2- Refactorizar el código agregando las interfaces necesarias para respetar el principio de segregación.

Tiempo🕐: *25 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ ***Comprender la importancia e implementación del principio de segregación de interfaces.***

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. *Lectura Módulo 4, Lección 6: página 9*
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.