INYECCIÓN DE DEPENDENCIAS

Danna Valeria Morales Aguilar

¿QUÉ ES INYECCIÓN DE DEPENDENCIAS?

Es un patrón de diseño y una técnica en el desarrollo de software, tiene como objetivo resolver el problema del acoplamiento entre clases.
En lugar de que un objeto cree sus propias dependencias, la inyección de dependencias permite que estas dependencias sean administradas por un componente externo.

FORMAS DE REALIZAR LA INYECCIÓN DE DEPENDENCIAS

- Inyección por constructor

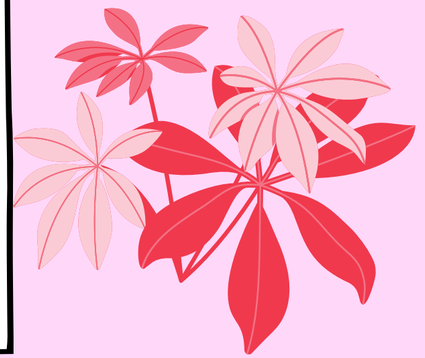
Las dependencias se pasan por el constructor del objeto cuando se crea una instancia de este.

- Inyección por método/setter

Se establecen a través de métodos setter en el objeto después de que se crea una instancia de este.

- Inyección por campo

Se establecen directamente en los campos del objeto después de que se crea una instancia de este.



ESCENARIO

Para explicar la inyección de dependencias, escogí un sistema sencillo de compras en una tienda de belleza llamada Sephora. La idea de este ejemplo es que el cliente pueda agregar productos a su carrito de compras de diferentes categorias como Maquillaje, cuidado de la piel y articulos para el cabello.

La inyección de dependencias se utilizaría para seleccionar y agregar el producto correcto según la categoría elegida.

```java
interface Sephora{
    void agregarCarrito();
}
```

Es una interfaz que define el método agregarCarrito(). Este será implementado por las diferentes categorías de productos como maquillaje, cuidado de la piel y productos para el cabello.

```java
class Makeup implements Sephora{
    @Override
    public void agregarCarrito() {
        System.out.println("Se agrego maquillaje al carrito");
    }
}
```

Las clases Makeup, Skincare y Cabello son implementaciones de la interfaz Sephora, cada una tiene su propia implementación del método agregarCarrito().

La clase ShoppingCart representa el carrito de compras. Tiene un atributo producto de tipo Sephora que representa el producto que se agregará al carrito.

```java
public class ShoppingCart {

    private String cart;
    private int id;
    private Sephora producto;

    public ShoppingCart(String cart, int id) {
        this.cart=cart;
        this.id=id;
    }
```

```java
public boolean addPro() {
    System.out.println("El carrito: "+cart+" con id: "+id);
    //DELEGACION
    producto.agregarCarrito();
    return true;

}
```

```java
public class Inyector {
public static void inyectarProducto(ShoppingCart cart, String producto) {

        Sephora makeup = new Makeup();
        Sephora skincare = new Skincare();
        Sephora hair = new Cabello();

        switch(producto){

            case "Makeup":
                cart.setProducto(makeup);
                break;
            case "Skincare":
                cart.setProducto(skincare);
                break;
            default:
                cart.setProducto(hair);
        }
    }
}
```

Contiene un método inyectarProducto(), toma un objeto de ShoppingCart y un string producto. Se encarga de inyectar la dependencia del producto correcto en el carrito según el producto dado.

```java
public class Main {

    public static void main(String[] args) {
        ShoppingCart cart = new ShoppingCart("Carrito 1", 1);
        Inyector.inyectarProducto(cart, "Makeup");

        boolean addProd = cart.addPro();

        if(addProd)
            System.out.println("El produto se agrego a su carrito.");
        else {
            System.out.println("El producto no se pudo agregar a su carrito.");
            System.out.println("Pruebe con otro producto.");
        }
```

En el método main, se crea un objeto ShoppingCart llamado cart, luego se llama al método inyectarProducto() del inyector para agregar un producto al carrito.
Por último se llama al método addPro() para agregar el producto y se imprime un mensaje indicando que se agrego correctamente.

CONCLUSIÓN

La inyección de dependencias se logra mediante el método inyectarProducto(), que selecciona y establece la implementación correcta de Sephora en el carrito.
El cliente puede agregar diferentes productos sin preocuparse por la lógica interna de como se agrega cada producto.