

Actividad PostContenido 1 - Unidad 1

Esteban Mauricio Calderón Bayona

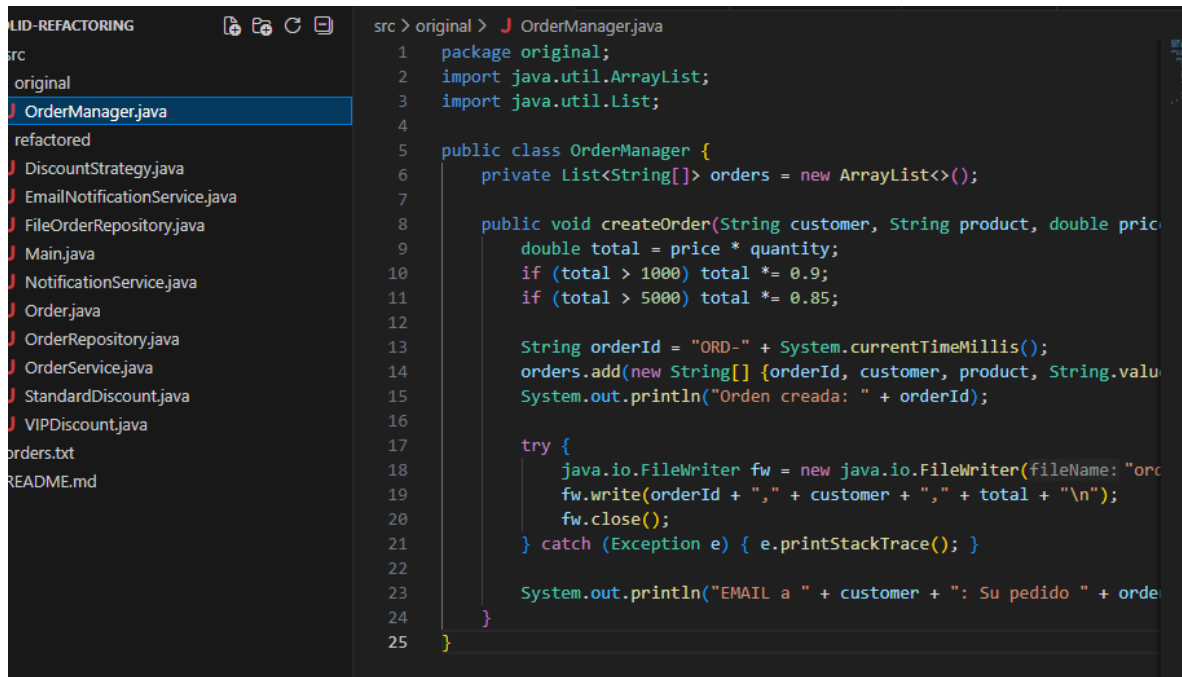
Universidad de Santander

Patrones de Diseño

Jhonatan Castillo

21/02/2026

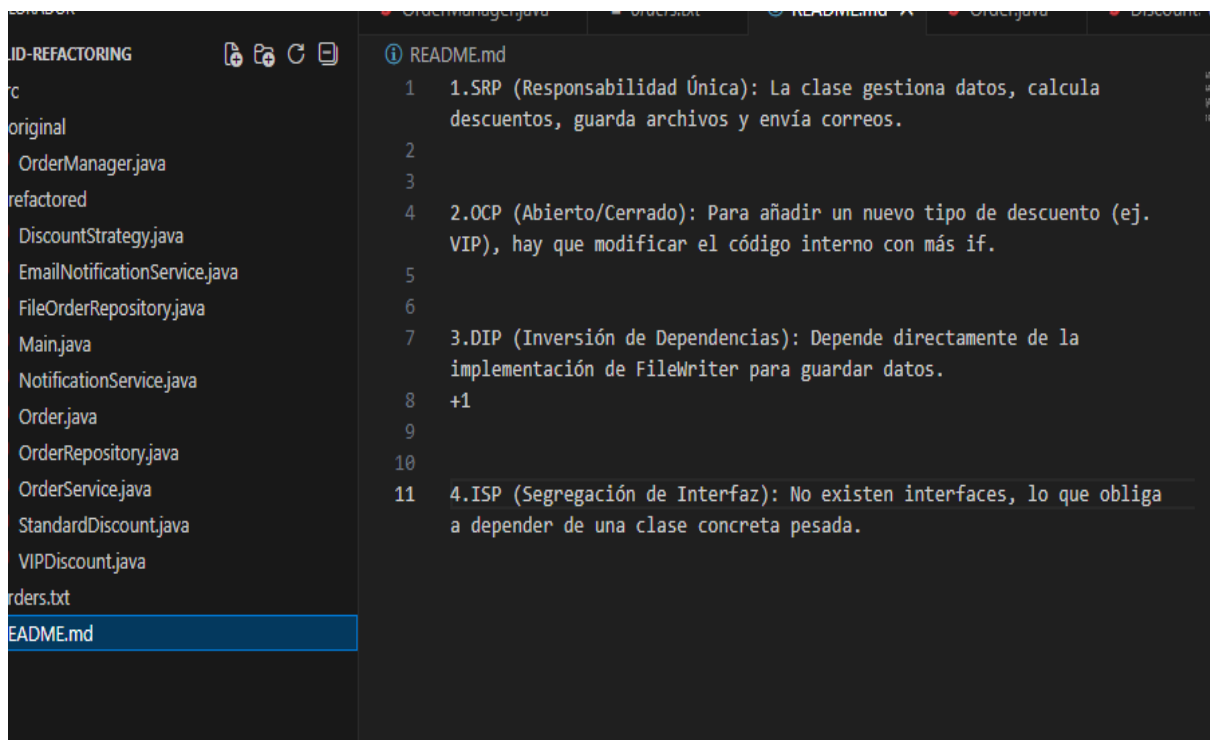
Capture 1: Cap de la Estructura del Proyecto



The screenshot shows an IDE with a project structure on the left and the code of `OrderManager.java` in the main editor. The project structure includes a `src` directory with `original` and `refactored` subdirectories. The `original` subdirectory contains `OrderManager.java`, `DiscountStrategy.java`, `EmailNotificationService.java`, `FileOrderRepository.java`, `Main.java`, `NotificationService.java`, `Order.java`, `OrderRepository.java`, `OrderService.java`, `StandardDiscount.java`, and `VIPDiscount.java`. The `refactored` subdirectory contains `DiscountStrategy.java`, `EmailNotificationService.java`, `FileOrderRepository.java`, `Main.java`, `NotificationService.java`, `Order.java`, `OrderRepository.java`, `OrderService.java`, `StandardDiscount.java`, and `VIPDiscount.java`. The `src` directory also contains `orders.txt` and `README.md`.

```
src > original > OrderManager.java
1  package original;
2  import java.util.ArrayList;
3  import java.util.List;
4
5  public class OrderManager {
6      private List<String[]> orders = new ArrayList<>();
7
8      public void createOrder(String customer, String product, double price,
9                              double total = price * quantity;
10         if (total > 1000) total *= 0.9;
11         if (total > 5000) total *= 0.85;
12
13         String orderId = "ORD-" + System.currentTimeMillis();
14         orders.add(new String[] {orderId, customer, product, String.valueOf(total)});
15         System.out.println("Orden creada: " + orderId);
16
17         try {
18             java.io.FileWriter fw = new java.io.FileWriter(fileName: "orders.txt");
19             fw.write(orderId + "," + customer + "," + total + "\n");
20             fw.close();
21         } catch (Exception e) { e.printStackTrace(); }
22
23         System.out.println("EMAIL a " + customer + ": Su pedido " + orderId);
24     }
25 }
```

Cap 2 del Análisis SOLID (README.md)



The screenshot shows an IDE with a project structure on the left and the content of `README.md` in the main editor. The project structure is the same as in the previous screenshot. The `README.md` file contains the following text:

```
1  1.SRP (Responsabilidad Única): La clase gestiona datos, calcula
2     descuentos, guarda archivos y envía correos.
3
4  2.OCP (Abierto/Cerrado): Para añadir un nuevo tipo de descuento (ej.
5     VIP), hay que modificar el código interno con más if.
6
7  3.DIP (Inversión de Dependencias): Depende directamente de la
8     implementación de FileWriter para guardar datos.
9  +1
10
11  4.ISP (Segregación de Interfaz): No existen interfaces, lo que obliga
    a depender de una clase concreta pesada.
```

3. Cap de Compilación (Sin errores)

The screenshot shows the Eclipse IDE interface. On the left, the 'Project Explorer' displays the 'solid-refactoring' project structure, including 'src' and 'refactored' folders. The 'OrderService.java' file is selected. The main editor shows the refactored code for the 'OrderService' class, which includes private final fields for 'repository', 'notifier', and 'discountStrategy', and a constructor that initializes them. The 'Terminal' view at the bottom shows the command prompt output, indicating a successful compilation and execution of the 'refactored.Main' class. The output includes messages about creating orders for Mauricio and Andrés, and a final summary of 2 orders saved.

```
src > refactored > J OrderService.java > OrderService
1 package refactored;
2
3 public class OrderService {
4     private final OrderRepository repository;
5     private final NotificationService notifier;
6     private final DiscountStrategy discountStrategy;
7
8     public OrderService(OrderRepository repository, NotificationService notifier, DiscountStrategy discountStrategy) {
9         this.repository = repository;
10        this.notifier = notifier;
11    }
12 }

PROBLEMAS SALIDA TERMINAL ...
Debug: Main + - [] [] | [] X

PS C:\Users\USUARIO\Desktop\solid-refactoring> & 'C:\Program Files\Eclipse Adoptium\jdk-25.0.2.1
0-hotspot\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:
57139' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\USUARIO\AppData\Roaming\Code\User\workspaceStorage\e31a372e7e5e0f652d5b2963f71ddf9b\redhat.java\jdt_ws\solid-r
efactoring_f66a480b\bin' 'refactored.Main'
EMAIL a Mauricio: Su pedido ORD-1771730706253 ha sido creado.
EMAIL a Andrés: Su pedido ORD-1771730706310 ha sido creado.

--- Proceso Finalizado con Éxito ---
Total de pedidos guardados: 2
PS C:\Users\USUARIO\Desktop\solid-refactoring>
```

CAP 4: El Archivo de Datos

The screenshot shows the Eclipse IDE interface. The 'Project Explorer' on the left shows the 'solid-refactoring' project structure. The 'orders.txt' file is selected and opened in the main editor. The file contains a list of orders, each with an order ID, a customer name, and a total amount. The orders are listed as follows:

```
1 ORD-1771730706253,Mauricio,2250.0
2 ORD-1771730706310,Andrés,800.0
3 ORD-1771731705510,Mauricio,2250.0
4 ORD-1771731705552,Andrés,800.0
5
```

