Integrantes: Ronny Ibarra, Angelo Sánchez,Carlos Rivera

Fecha: 08-07-2025
TRABAJO EN GRUPO (5ptos)

Ejemplo de SRP (Single Responsibility Principle) en Java PATRÓN SOLID

1. ***¿Qué es SRP?***

SRP significa Single Responsibility Principle (Principio de Responsabilidad Única) que establece **que cada clase debe encargarse de una única responsabilidad o función específica dentro del sistema. Esto permite que los cambios en una parte del sistema afecten solo a la clase correspondiente, haciendo que el código sea más mantenible, claro y modular.**

(Analiza las clases del ejercicio CRUD del estudiante propuesto)

2. ***Ejemplo que NO cumple SRP***

La siguiente clase mezcla varias responsabilidades

En el siguiente ejemplo, la clase Estudiante mezcla responsabilidades que no le corresponden: gestiona los datos del estudiante, imprime su información y guarda en la base de datos.

(Identifica cuales son las responsibilities ej presenter datos,)

```
public class Estudiante {
    private int id;
    private String nombre;

    public Estudiante(int id, String nombre) {
        this.id = id;
        this.nombre = nombre;
    }

    public void guardarEnBD() {
        // Código que guarda en la base de datos
        System.out.println("Guardando en BD...");
    }

    public void imprimir() {
        // Código que imprime en consola
        System.out.println("Estudiante: " + nombre);
    }
}
```

Responsabilidades mezcladas:

- **Representar los datos del estudiante.**
- **Persistencia: guardar en base de datos.**

- **Presentación: imprimir en consola.**

3. Anàlisis del Problema

Esta clase tiene tres responsabilidades distintas:

- Gestionar los datos del estudiante (modelo con atributos id y nombre).
- Guardar en la base de datos (persistencia).
- Mostrar información por consola (presentación).
- Estas funciones deberían estar separadas en clases distintas.

(Ejemplo Presentar los datos del estudiante)

4. Ejemplo que SÍ cumple SRP

Ahora se separan las responsabilidades en clases distintas:

1. Clase Modelo (solo datos):

```
public class Estudiante {
   private int id;
   private String nombre;

   public Estudiante(int id, String nombre) {
      this.id = id;
      this.nombre = nombre;
   }

   // Getters y Setters
   public int getId() { return id; }
   public String getNombre() { return nombre; }
}
```

2. Clase DAO (persistencia):

```
public class EstudianteDAO {
   public void guardar(Estudiante e) {
      // Código para guardar en BD
      System.out.println("Guardando estudiante en la BD...");
   }
}
```

3. Clase Vista (presentación):

```
public class EstudiantePrinter {
   public void imprimir(Estudiante e) {
      System.out.println("Estudiante: " + e.getNombre());
   }
}
```

5. Ventajas de aplicar SRP

(Analiza para responder las siguientes preguntas.

a. Si cambias cómo se imprimen los datos, no tocas la clase Estudiante ni el DAO.

b. Si cambias cómo se guardan los datos, no afecta la vista (Main) ni la clase Estudiante.

c. Cada clase tiene una única razón para cambiar, lo que facilita el mantenimiento, mejora la claridad del código y reduce errores.

Rúbrica de Evaluación

Criterio	Puntaje Máximo
Explica con claridad qué es SRP	4 puntos
Identifica correctamente las responsabilidades mezcladas	4 puntos
Muestra correctamente el ejemplo que sí cumple SRP	4 puntos
Analiza ventajas de aplicar SRP	4 puntos
Presenta el contenido de forma clara y ordenada	4 puntos
TOTAL SOBER 20 PTOS	