# Explicación Del Código Caso1

# ¿Qué es la Programación Orientada a Objetos (POO)?

La Programación Orientada a Objetos (POO) es un paradigma de programación que organiza el software en torno a objetos y clases. Cada objeto representa una entidad del mundo real, con atributos (datos) y métodos (acciones).  
  
En este sistema, usamos POO para representar usuarios como objetos, encapsular su información, organizar la lógica en clases separadas, y facilitar la reutilización y mantenimiento del código.

## ¿Cómo se aplica POO en este sistema?

- Usamos la clase Usuario para representar cada usuario como un objeto con su nombre, correo, contraseña, ID y su historial.  
- Usamos la clase GestorUsuario para separar la lógica del programa (como registrar o editar usuarios).  
- En el Main, solo mostramos el menú y llamamos métodos del gestor, lo que demuestra abstracción.  
- Cada clase tiene su propia responsabilidad, cumpliendo con el principio de diseño modular.

# 1. Clase Usuario.java

Representa un objeto del tipo Usuario. Se utiliza para almacenar y agrupar los datos de cada usuario registrado.  
  
Atributos:  
- id: Identificador único autogenerado para cada usuario.  
- nombre: Nombre del usuario.  
- correo: Correo electrónico del usuario.  
- contraseña: Clave del usuario.  
- historial: Lista de cambios que ha realizado (ArrayList).  
  
Método importante:  
- agregarHistorial(String): Guarda un mensaje en el historial del usuario.  
Esto demuestra el principio de encapsulamiento.

# 2. Clase GestorUsuario.java

Es la clase que contiene toda la lógica para manejar usuarios. Permite separar el procesamiento de datos de la interfaz de usuario.  
  
Variables importantes:  
- usuarios: Lista (ArrayList) que guarda todos los objetos de tipo Usuario.  
- sc: Scanner para leer entrada del usuario.  
  
Métodos y para qué sirven:  
- registrarUsuario(): Valida y registra un nuevo usuario.  
- iniciarSesion(): Verifica si un usuario puede acceder.  
- editarPerfil(): Cambia nombre o correo de un usuario por ID.  
- cambiarContrasena(): Cambia la contraseña si la actual es correcta.  
- verHistorial(): Muestra los cambios realizados por un usuario.  
- mostrarUsuarios(): Lista todos los usuarios con su ID, nombre y correo.  
- buscarPorId(int): Encuentra un usuario específico en la lista por su ID.  
  
Esta clase aplica el principio de abstracción.

# 3. Clase Main.java

Contiene el método main(), punto de entrada del programa. Muestra el menú de opciones y llama a los métodos de la clase GestorUsuario.  
  
Variables usadas:  
- opcion: Guarda la opción elegida del menú.  
- sc: Scanner para leer opciones del teclado.  
  
El método main solo se encarga de interactuar con el usuario, dejando la lógica a la clase GestorUsuario. Esto es una demostración de abstracción y buena organización del código.

# 4. Funcionalidades implementadas

|  |  |
| --- | --- |
| Requerimiento | Funcionalidad |
| RF1 | Registrar usuario con validación de correo y contraseña fuerte. |
| RF2 | Iniciar sesión con correo y clave. |
| RF3 | Gestión de perfil nombre y correo. |
| RF4 | Cambiar contraseña al menos 6 Caracteres. |
| RF5 | Ver historial de cambios |
| RF6 | Ver lista de usuarios registrados |
| RNF1 | Validación de datos del registro. |

# 5. Pilares de la Programación Orientada a Objetos en el código

## Encapsulamiento

Se aplica en la clase Usuario, donde se agrupan todos los atributos relacionados al usuario. Los datos del usuario y el historial se manejan desde dentro del objeto Usuario, lo que evita desorden.

## Abstracción

Se aplica en la clase GestorUsuario. Desde el Main solo se llaman métodos como registrarUsuario( ).