ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

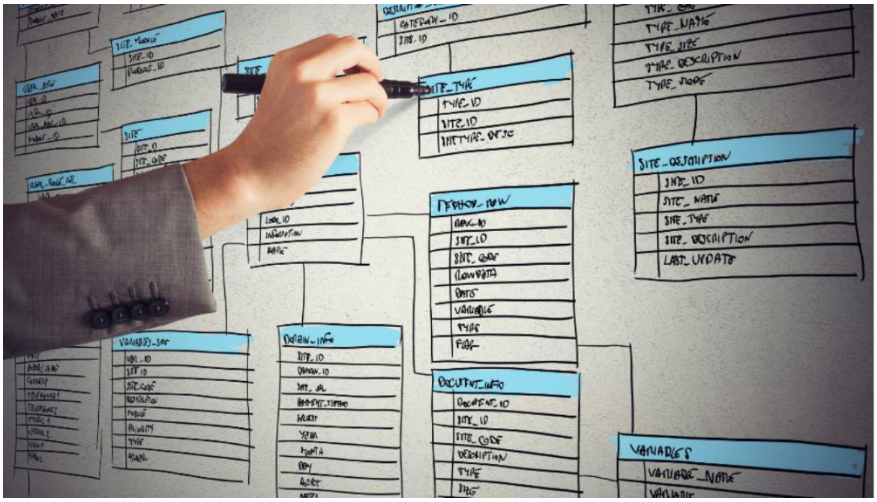
PROGRAMACIÓN ORIENTADA A OBJETOS

PROFESOR:	Ing. Yadira Franco R
PERÍODO ACADÉMICO:	2025-A

TAREA 6 – Grupal 2 integrantes

TÍTULO:

INVESTIGACIÓN Y PRACTICA



Estudiante

María Girón

2025-A

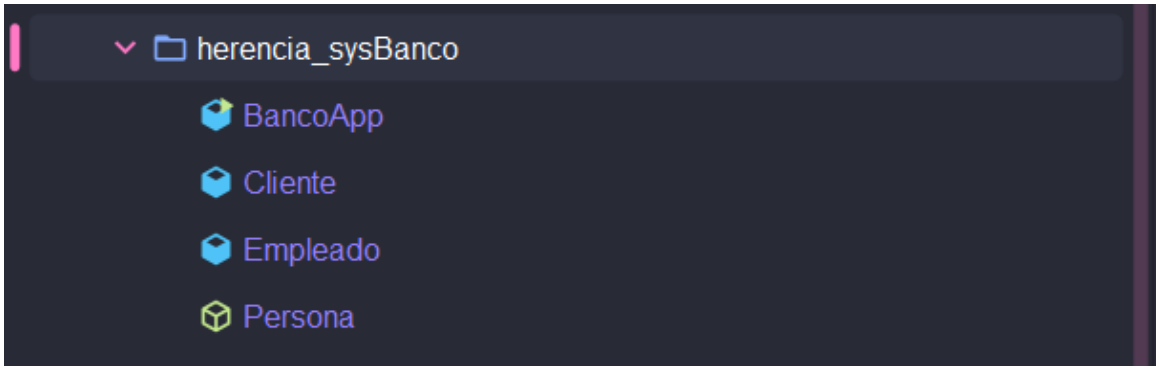
Sistema Bancario: Uso de Clases Abstractas y Jerarquías

Introducción

Este informe explica la implementación de un sistema bancario utilizando el concepto de clases abstractas y herencia en Java. Se sigue una estructura basada en la clase abstracta `Persona`, con subclases `Cliente` y `Empleado` que modelan roles específicos en la gestión bancaria, además de prácticas de autenticación para seguridad dentro del programa.

Se ha estructurado el código en múltiples clases separadas para garantizar un modelo bien organizado y escalable. Además, se han aplicado principios de encapsulamiento y herencia para mantener una arquitectura modular y flexible.

Estructura del Proyecto



2. Uso de la Clase Abstracta `Persona`

La clase `Persona` sirve como modelo base y define atributos comunes como `nombre`, `cedula`, `direccion` y `telefono`. Su propósito es evitar la duplicación de código y garantizar que `Cliente` y `Empleado` compartan características esenciales. Además, incluye un método abstracto `mostrarRol()`, lo que obliga a las subclases a definir su propio comportamiento.

```java
package Herencia.herencia_sysBanco;

abstract class Persona {
    protected String nombre;
    protected String cedula;
    protected String direccion;
    protected String telefono;

    public Persona(String nombre, String cedula, String direccion, String telefono) {
        this.nombre = nombre;
        this.cedula = cedula;
        this.direccion = direccion;
        this.telefono = telefono;
    }

    public void actualizarDatos(String direccion, String telefono) {
        this.direccion = direccion;
        this.telefono = telefono;
    }

    public abstract void mostrarRol();
}
```

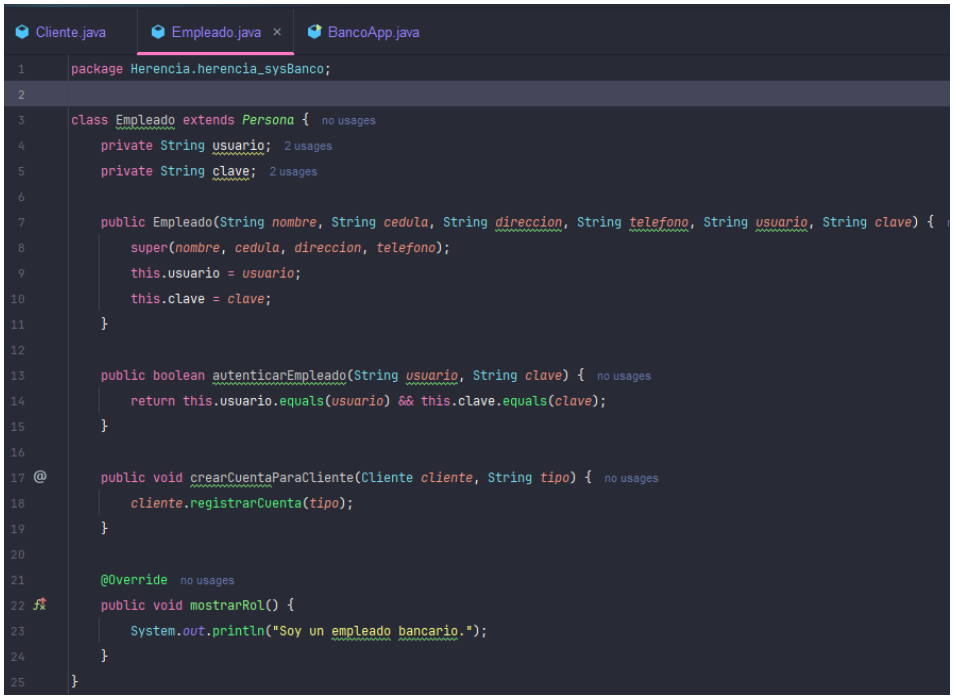
3. Subclase `Cliente`

La clase `Cliente` hereda de `Persona` e implementa funcionalidades específicas para el usuario bancario. Se incluyen métodos como `registrarCuenta()`, `solicitarPrestamo()`, y `ingresarAlSistema()`. Estos métodos permiten a los clientes acceder al sistema, registrar cuentas y solicitar préstamos.

```java
package Herencia.herencia_sysBanco;

class Cliente extends Persona {
    public Cliente(String nombre, String cedula, String direccion, String telefono) {
        super(nombre, cedula, direccion, telefono);
    }

    public boolean ingresarAlSistema() {
        return true;
    }

    public void registrarCuenta(String tipo) {
        System.out.println("Cuenta " + tipo + " registrada.");
    }

    public void solicitarPrestamo(double monto) {
        System.out.println("Solicitando préstamo de $" + monto);
    }

    @Override
    public void mostrarRol() {
        System.out.println("Soy un cliente bancario.");
    }
}
```

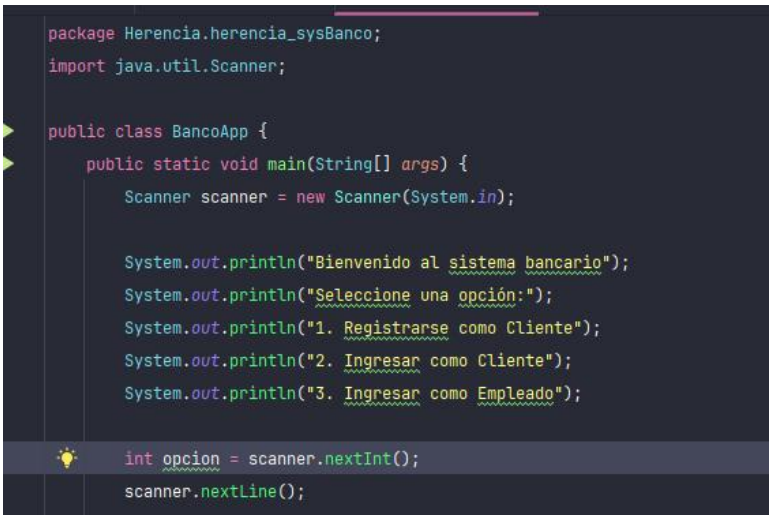
4. Subclase `Empleado`

La clase `Empleado` también extiende `Persona`, pero con métodos propios de gestión bancaria, como `autenticarEmpleado()`, `crearCuentaParaCliente()`, y otros. La encapsulación de atributos privados como `usuario` y `clave` garantiza la seguridad de la autenticación.



5. Implementación del Menú Interactivo

Se utilizó `Scanner` en la clase BancoApp para capturar datos y simular la interacción del usuario con el sistema. Este menú permite registrar clientes y gestionar el acceso a empleados.



```
        switch (opcion) {
            case 1:
                System.out.println("Ingrese su nombre:");
                String nombre = scanner.nextLine();
                System.out.println("Ingrese su cédula:");
                String cedula = scanner.nextLine();
                System.out.println("Ingrese su dirección:");
                String direccion = scanner.nextLine();
                System.out.println("Ingrese su teléfono:");
                String telefono = scanner.nextLine();

                Cliente cliente = new Cliente(nombre, cedula, direccion, telefono);
                cliente.mostrarRol();
                System.out.println("Cliente registrado con éxito.");
                break;
```

```
                case 2:
                    System.out.println("Funcionalidad de acceso de cliente en desarrollo.");
                    break;

                case 3:
                    System.out.println("Funcionalidad de acceso de empleado en desarrollo.");
                    break;

                default:
                    System.out.println("Opción no válida.");
            }

            scanner.close();
        }

    }
```

Conclusión

Este sistema bancario demuestra cómo las clases abstractas pueden estructurar un programa en Java, promoviendo la reutilización de código y un diseño eficiente. La separación de roles mediante subclases facilita la gestión de clientes y empleados, asegurando que cada uno cumpla sus funciones específicas en el sistema.

GitHub

https://github.com/MariaGiron-code/DEBERES_POO.git

Bibliografía

https://www.youtube.com/watch?v=b0NHh8RNWK4&t=1466s&pp=ygUmVXNvIGRlIENsYXNlcyBBYnN0cmFjdGFzIHkgSmVyYXJxdcOtYXM%3D

https://www.youtube.com/watch?v=q11f-Yr_pC4&t=214s&pp=ygUmVXNvIGRlIENsYXNlcyBBYnN0cmFjdGFzIHkgSmVyYXJxdcOtYXM%3D