



# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS



### PROGRAMACIÓN ORIENTADA A OBJETOS

PROFESOR:

Ing. Yadira Franco R

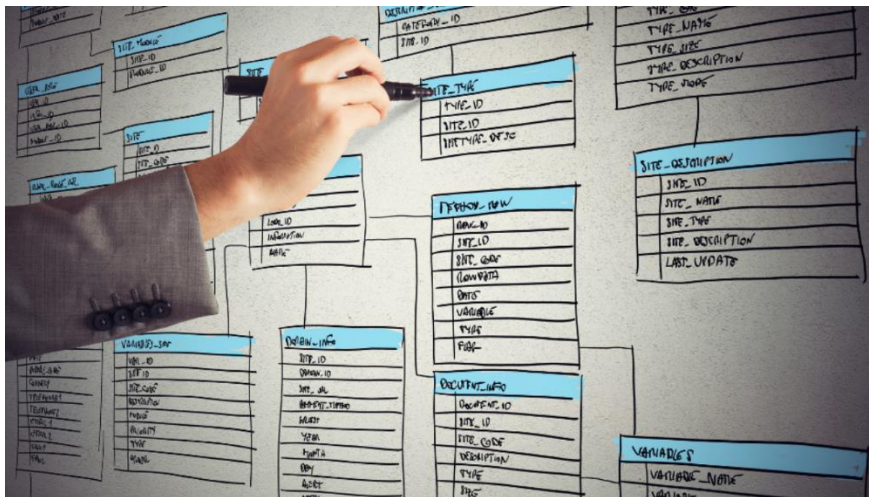
PERÍODO ACADÉMICO:

2025-A

### TAREA 6 – Grupal 2 integrantes

TÍTULO:

INVESTIGACIÓN Y PRACTICA



Estudiante

María Girón

2025-A

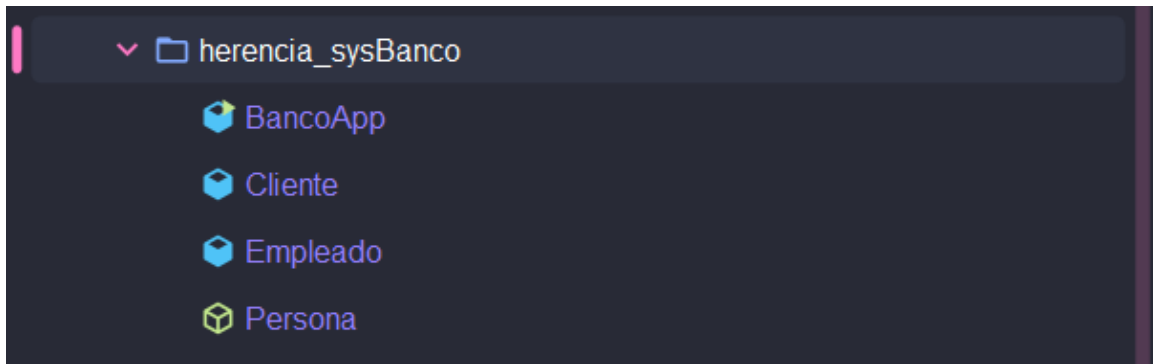
# Sistema Bancario: Uso de Clases Abstractas y Jerarquías

## 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.

```

1 package Herencia.herencia_sysBanco;
2
3 abstract class Persona { no usages 2 inheritors
4     protected String nombre;
5     protected String cedula; 1 usage
6     protected String direccion; 2 usages
7     protected String telefono; 2 usages
8
9     public Persona(String nombre, String cedula, String direccion, String telefono) { no usages
10         this.nombre = nombre;
11         this.cedula = cedula;
12         this.direccion = direccion;
13         this.telefono = telefono;
14     }
15
16     public void actualizarDatos(String direccion, String telefono) { no usages
17         this.direccion = direccion;
18         this.telefono = telefono;
19     }
20
21 public abstract void mostrarRol(); no usages 2 implementations
22 }

```

### 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.

```

package Herencia.herencia_sysBanco;

class Cliente extends Persona { no usages
    public Cliente(String nombre, String cedula, String direccion, String telefono) { no usages
        super(nombre, cedula, direccion, telefono);
    }

    public boolean ingresarAlSistema() { no usages
        return true;
    }

    public void registrarCuenta(String tipo) { no usages
        System.out.println("Cuenta " + tipo + " registrada.");
    }

    public void solicitarPrestamo(double monto) { no usages
        System.out.println("Solicitando préstamo de $" + monto);
    }

    @Override no usages
    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.

```
Cliente.java  Empleado.java  BancoApp.java
1 package Herencia.herencia_sysBanco;
2
3 class Empleado extends Persona { no usages
4     private String usuario; 2 usages
5     private String clave; 2 usages
6
7     public Empleado(String nombre, String cedula, String direccion, String telefono, String usuario, String clave) {
8         super(nombre, cedula, direccion, telefono);
9         this.usuario = usuario;
10        this.clave = clave;
11    }
12
13    public boolean autenticarEmpleado(String usuario, String clave) { no usages
14        return this.usuario.equals(usuario) && this.clave.equals(clave);
15    }
16
17    @ public void crearCuentaParaCliente(Cliente cliente, String tipo) { no usages
18        cliente.registrarCuenta(tipo);
19    }
20
21    @Override no usages
22    public void mostrarRol() {
23        System.out.println("Soy un empleado bancario.");
24    }
25 }
```

#### 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.

```
package Herencia.herencia_sysBanco;
import java.util.Scanner;

public class BancoApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Bienvenido al sistema bancario");
        System.out.println("Seleccione una opción:");
        System.out.println("1. Registrarse como Cliente");
        System.out.println("2. Ingresar como Cliente");
        System.out.println("3. Ingresar como Empleado");

        int opcion = scanner.nextInt();
        scanner.nextLine();
    }
}
```

```

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](https://github.com/MariaGiron-code/DEBERES_POO.git)

## **Bibliografía**

<https://www.youtube.com/watch?v=b0NHh8RNWK4&t=1466s&pp=vgUmVXNvIGRIENsYXNlcyBBYnN0cmFjdGFzIHkgSmVyYXJxdcOtYXM%3D>

[https://www.youtube.com/watch?v=q11f-Yr\\_pC4&t=214s&pp=vgUmVXNvIGRIENsYXNlcyBBYnN0cmFjdGFzIHkgSmVyYXJxdcOtYXM%3D](https://www.youtube.com/watch?v=q11f-Yr_pC4&t=214s&pp=vgUmVXNvIGRIENsYXNlcyBBYnN0cmFjdGFzIHkgSmVyYXJxdcOtYXM%3D)