Alberto Santos Portillo

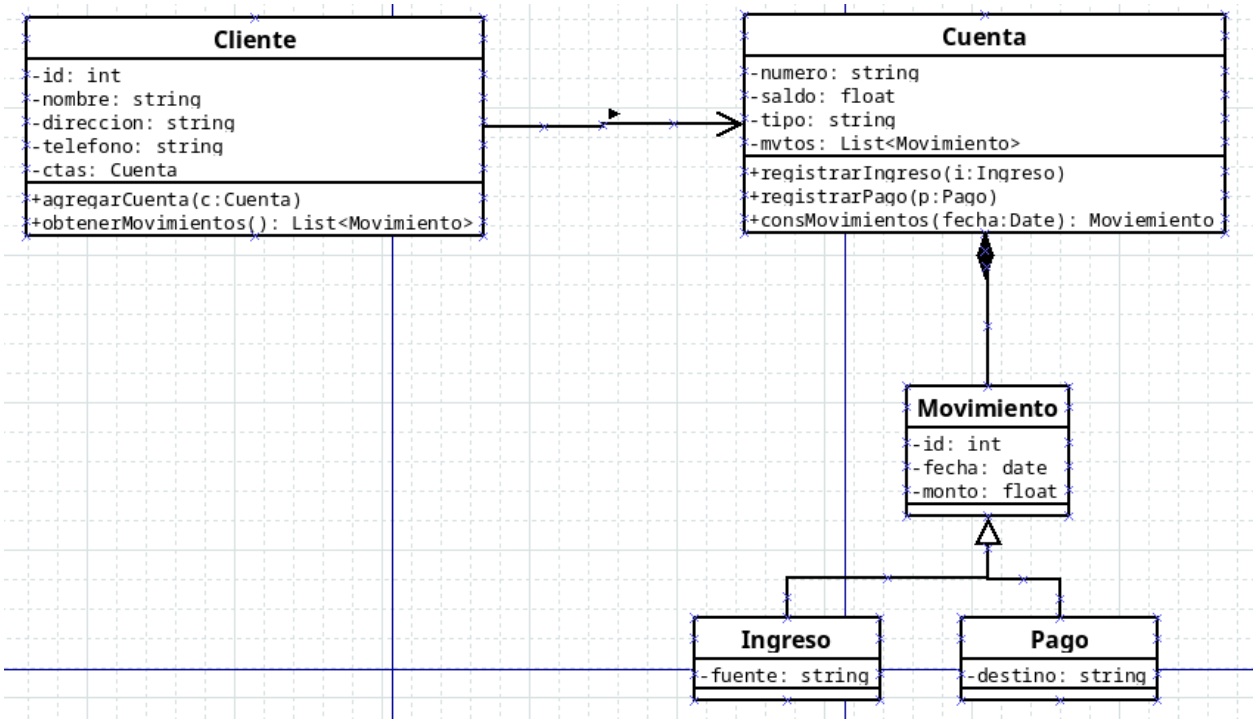
1ºDAM

Comunicación

Nos han pedido una sucursal de un banco que diseñemos un programa para ellos y poder guardar la economía que tenga un cliente, los movimientos que haga, los pagos que el/la cliente hace y los ingresos que recibe los clientes.

Diseño

diagrama de clase:



Clases:

- **Cliente**: Representa a una persona que tiene una o varias cuentas bancarias
- **Cuenta**: Representa una cuenta de manera individual
- **Movimiento**: Es la base que representa la base (Ingreso o pago) que hace un cliente
- **Ingreso**: Aquí hace una operación en la que entra dinero a la cuenta

- **Pago**: Se muestra una operación en la que sale el dinero de una cuenta

Diagrama de Secuencia:

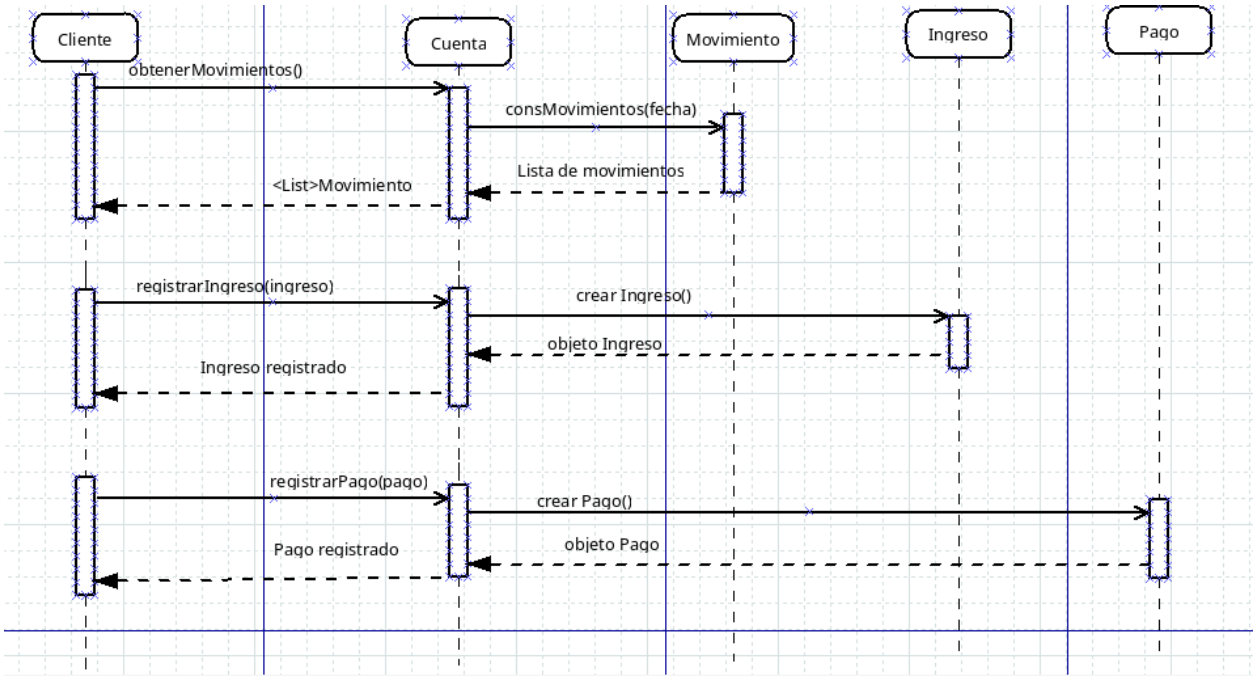
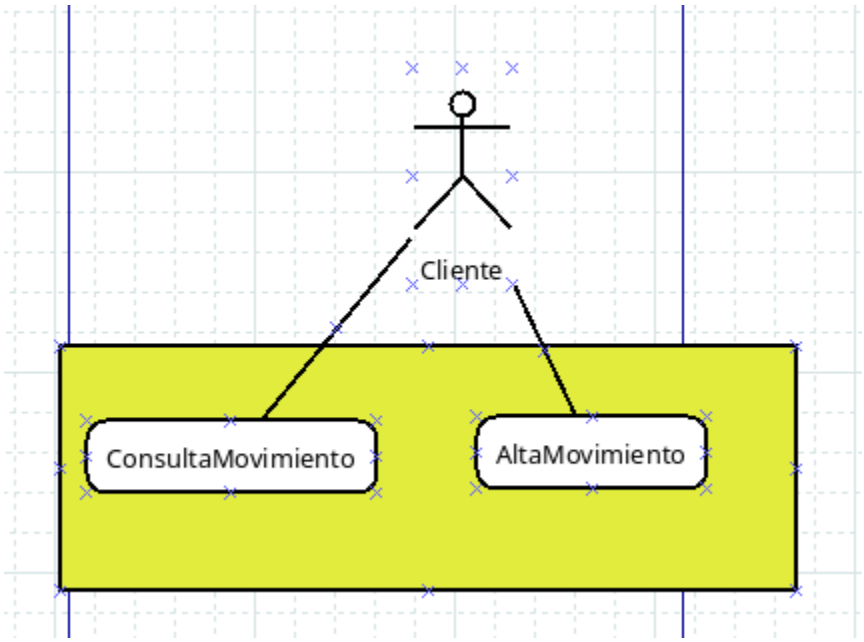


Diagrama de caso de uso:



Caso de Uso: ConsultaMovimiento

- **Actor Primario:**
  Cliente

- **Finalidad del Caso de Uso:**
  Permitir al cliente consultar todos los movimientos realizados en sus cuentas (ingresos, pagos, etc.).

- **Precondiciones:**
  - El cliente debe estar autenticado en el sistema.
  - El cliente debe tener al menos una cuenta con movimientos registrados.

- **Activador:**
  El cliente selecciona la opción "ConsultaMovimiento" en la aplicación.

Escenario:

1. El cliente accede al sistema y selecciona “ConsultaMovimiento”.
2. El sistema muestra la lista de cuentas asociadas al cliente.
3. El cliente elige una cuenta específica.
4. El sistema recupera y muestra los movimientos asociados, ordenados por fecha.
5. El cliente puede aplicar filtros (por tipo de movimiento, rango de fechas, etc.).
6. Finaliza la consulta y el cliente vuelve al menú principal.

Caso de Uso: AltaMovimiento

- **Actor Primario:**
  Cliente

- **Finalidad del Caso de Uso:**
  Permitir al cliente registrar un nuevo movimiento (ingreso o pago) en su cuenta bancaria.

- **Precondiciones:**
  - **El cliente debe estar autenticado.**
  - **Debe existir al menos una cuenta activa vinculada al cliente.**

- **Activador:**
  El cliente selecciona la opción “AltaMovimiento” desde el menú de la aplicación.

Escenario:

1. **El cliente accede al sistema y elige “AltaMovimiento”.**
2. **El sistema solicita los datos necesarios: tipo de movimiento (ingreso/pago), monto, fecha, concepto/destino.**

3. **El cliente completa el formulario y envía la solicitud.**

4. **El sistema valida que los datos son correctos (por ejemplo, que haya saldo suficiente si es un pago).**

5. **Si todo es válido, se registra el movimiento y se actualiza el saldo de la cuenta.**

6. **Se muestra un mensaje de confirmación al cliente.**

Codificación:

Este serian los tests del programa:

```java
package com.sucursalbanco;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import java.util.Date;
public class MovimientoTest {
    @Test
    public void testGetFechaYGetMonto() {
        Date fecha = new Date();
        Pago pago = new Pago(id:1, fecha, monto:250.0f, destino:"Supermercado");

        assertEquals(fecha, pago.getFecha());
        assertEquals(expected:250.0f, pago.getMonto(), delta:0.001f);
    }
}
```

```java
import java.util.Date;
import java.util.List;
public class ClienteTest {
    @Test
    public void testCrearClienteYAgregarCuenta() {
        Cliente cliente = new Cliente(id:1, nombre:"Ana Garcia", direccion:"Calle Falsa 123", telefono:"600000000");
        Cuenta cuenta = new Cuenta(numero:"123ABC", saldo:1000f, tipo:"Ahorro");

        cliente.agregarCuenta(cuenta);

        assertEquals(expected:"Ana Garcia", cliente.getNombre());

        assertEquals(expected:0, cliente.obtenerMovimientos().size());
        assertEquals(expected:0, cuenta.getMovimientos().size());



    }
    @Test
    public void testObtenerMovimientosDeVariasCuentas() {
        Cliente cliente = new Cliente(id:2, nombre:"Carlos Ruiz", direccion:"Av. Central 45", telefono:"699999999");
        Cuenta cuenta1 = new Cuenta(numero:"111AAA", saldo:500f, tipo:"Corriente");
        Cuenta cuenta2 = new Cuenta(numero:"222BBB", saldo:800f, tipo:"Ahorro");

        Date fecha = new Date();
        Ingreso ingreso1 = new Ingreso(id:1, fecha, monto:100f, fuente:"Trabajo");
        Pago pago1 = new Pago(id:2, fecha, monto:50f, destino:"Compra");

        cuenta1.registrarIngreso(ingreso1);
        cuenta2.registrarPago(pago1);

        cliente.agregarCuenta(cuenta1);
        cliente.agregarCuenta(cuenta2);

        List<Movimiento> movimientos = cliente.obtenerMovimientos();

        assertEquals(expected:2, movimientos.size());
        assertTrue(movimientos.contains(ingreso1));
        assertTrue(movimientos.contains(pago1));
    }
}
```

Implementación:

Este es el [link del proyecto](#) en github