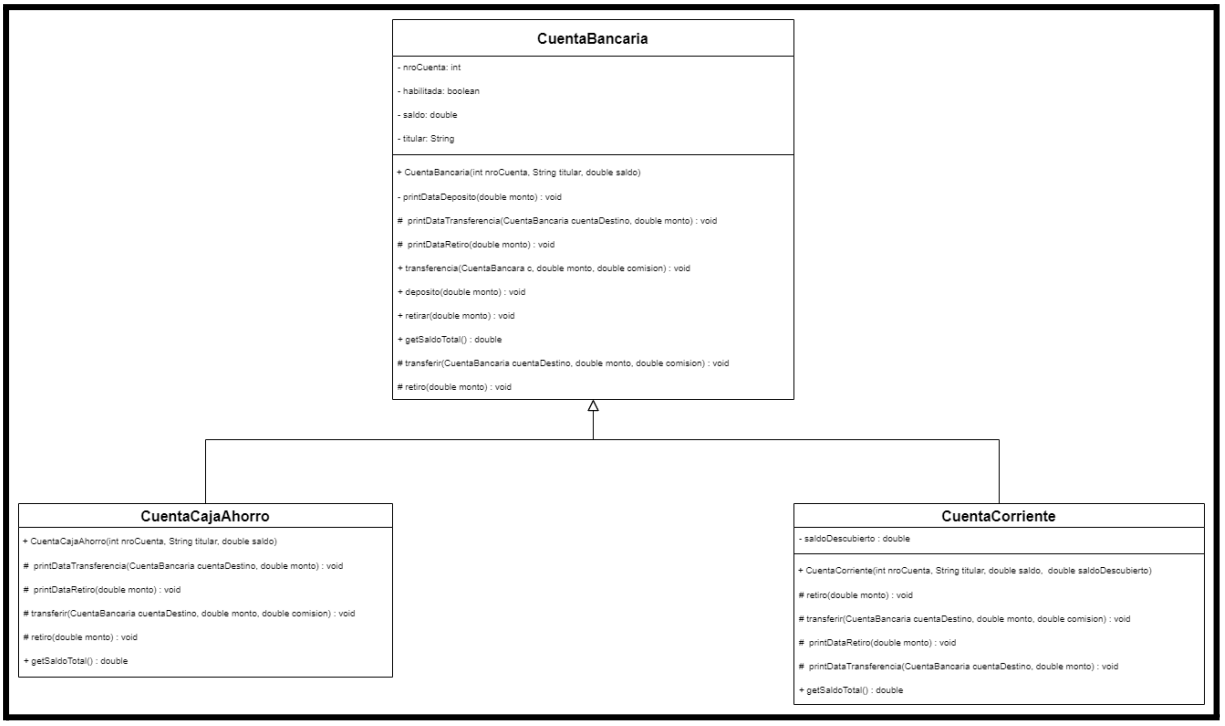
Ejercicio 1

Para esta parte el enunciado dice que ambas cuentas comparten ciertos atributos como nombre del titular, saldo, identificador de cuenta y si se encuentra habilitada, luego nos comenta sobre la cuenta corriente que además del saldo va a tener un atributo saldo descubierto.
Al decirnos que comparten atributos se decidió realizar una abstracción y encapsular el comportamiento parecido entre ambas en una clase abstracta dejando a sus clases hijas que implementen lo que se haría distinto dependiendo el tipo de cuenta.
Por lo que el diagrama de clases sería el siguiente:



Ejercicio 2

Ejercicio 2 a

Para realizar este punto se decidió hacer un método abstracto en la clase padre llamado retiro el cual lo implementaran sus hijas de forma diferente.

Pseudocódigo para caja ahorro:

```
void Retiro(monto)
    if cuenta.habilitada and monto <= saldo
        realizó el Retiro
    else
        imprimo el error
```

Pseudocódigo para cuenta corriente:

```
void Retiro(monto)
    if cuenta.habilitada
        if monto <= saldo
            realizó el Retiro
        else
            if monto <= saldo + saldoDescubierto
                realizó el Retiro
            else
                imprimo el error
    else
        imprimo el error
```

Si bien el objetivo de ambas es poder retirar dinero, ambas lo realizan diferente ya que en cuenta corriente si no contamos con el saldo para poder llevar a cabo el retiro, tendremos la opción de utilizar el saldo descubierto.

Ejercicio 2 b

El método transferencia se encargará de hacer los chequeos correspondientes para ambas clases y luego delega a cada clase hija como se va a hacer la transferencia.

```
void Transferencia(cuentaDestino,monto)
    if cuenta.habilitada
         if cuentaOrigen.tipo == cuentaDestio.tipo
            Transferir(monto,cuentaDestino)
        else
            Transferir(monto + comision, cuentaDestino)
    else
        imprimo el error
```

Para la clase caja de ahorro realizará el retiro si dispone del saldo para realizarlo mientras que en la clase cuenta corriente realizará el retiro si dispone del saldo pero si no llega a disponer preguntará si llega con el saldo descubierto a realizar la operación.

Ejercicio 2 c

En este ejercicio bastará con poder hacer un llamado a la clase padre preguntando si la cuenta o cuentas que están involucradas en la operación se encuentran habilitadas.
Se tomó la decisión de que por defecto en el constructor ponga como True al atributo habilitada ya que cuando se crea una cuenta sería como darla ir a hacerte una cuenta al banco en la vida real por lo que debería estar habilitada para operaciones.

Ejercicio 2 d

Este punto se puede ver implementado en todas las operaciones, sin embargo aquí no supe cómo tratar con el saldo descubierto ya que no había mucha información sobre el mismo, lo que se decidió fue si el saldo no alcanza para abastecer la operación usar el saldo descubierto y actualizar este saldo, pero aquí quizás también hubiese estado bien además de actualizar el descubierto ponerle una deuda al usuario indicandola con saldo negativo.

Ejercicio 2 e

Este punto cuando la operación se realiza con éxito llama a los print correspondientes.

Ejercicio 2 f

Para el manejo de errores se decidió imprimir por pantalla cuál era el error que estaba pasando así el usuario sabía del mismo.

Ejercicio 3

Para el ejercicio 3 como había un bonus de no utilizar ciclos, se realizó con la biblioteca Stream la cual hace un manejo más eficiente de las estructuras.
Acá se encuentra el mayor error a mi entender del trabajo y es que no acate la consigna y no hice un método que reciba por parámetro una lista y trabaje a partir de ella, lo que hice fue hacer una clase banco que tiene una lista de cuentas e implementa estos dos métodos. Sin embargo para poder hacerlo de la forma que se pide deberia cambiar solo algunas cosas:

```
public ArrayList<String> getTitularesParaPrestamos(ArrayList<CuentasBancarias> c){
        return (ArrayList<String>) c.cuentas.stream()
                .filter(e-> e.getSaldoTotal() > 10000 && e.getHabilitada() == true)
                .map(el -> el.getTitular().toUpperCase())
                .distinct()
                .collect(Collectors.toList());
    }

    public boolean algunasCuentasPuedenSerHackeada(ArrayList<CuentasBancarias> c){
        ArrayList<CuentaBancaria> l = (ArrayList<CuentaBancaria>) this.cuentas.stream()
                .filter(e-> e.getSaldoTotal() > 50000 &&
                e.getTitular().length() > 15 &&
                e.getNroCuenta()%2 == 0)
                .collect(Collectors.toList());

        if(l.size() > 0)
            return  true;
        return false;
    }
```

Ejercicio 4

Los test que se realizaron fueron los siguientes:

Depósito:
- Un usuario quiere depositar plata pero la cuenta no esta habilitada.
- Un usuario quiere depositar plata pero por un bug del sistema el monto que quiere depositar es negativo.
- Un usuario quiere depositar plata.

Retiro (CajaAhorro):
- Un usuario quiere retirar plata pero la cuenta no se encuentra habilitada.
- Un usuario retira una parte de su saldo.
- Un usuario retira todo el saldo.
- Un usuario retira mucho más que el saldo que tiene disponible.

Retiro (Cuenta Corriente):
- Un usuario quiere retirar plata pero la cuenta no se encuentra habilitada.
- Un usuario retira saldo pero no todo.
- Un usuario retira todo el saldo y la mitad del descubierto.
- Un usuario retira todo el saldo y saldo descubierto.
- Un usuario retira mucho más que el saldo total que tiene.

Transferencia:
- Usuario origen tiene la cuenta deshabilitada e intenta transferir.
- Usuario destino tiene la cuenta e intenta recibir dinero.
- Transferencia entre cajas de ahorro.
- Transferencia entre cuentas del mismo tipo pero no dispone de saldo para realizarla.
- Transferencia de caja ahorro a cuenta corriente pero no tiene saldo para pagar la comisión.
- Transferencia de caja ahorro a cuenta corriente pero no tiene saldo.
- Transferencia de caja ahorro a cuenta corriente.
- Transferencia entre cuentas corrientes.
- Transferencia entre cuentas corrientes pero usando el saldo descubierto.
- Transferencia entre cuentas corrientes pero no tiene saldo.
- Transferencia de cuenta corriente a caja ahorro pero no tiene saldo.
- Transferencia de cuenta corriente a caja ahorro pero no tiene saldo para pagar la comisión.
- Transferencia de cuenta corriente a caja ahorro.

Préstamos:
- Ninguna cuenta tiene saldo mayor a 10000.
- Ninguna cuenta se encuentra habilitada.
- Todas las cuentas cumplen las condiciones pero todas son iguales.
- Un banco da préstamos y alguna cuenta cumple con las condiciones.

Vulnerabilidad:
- Ninguna cuenta tiene saldo mayor a 50000.
- Ninguna cuenta es par.
- Ninguna cuenta tiene titular mayor a 15 letras.
- Un banco quiere ver vulnerabilidad y un usuario es vulnerable.