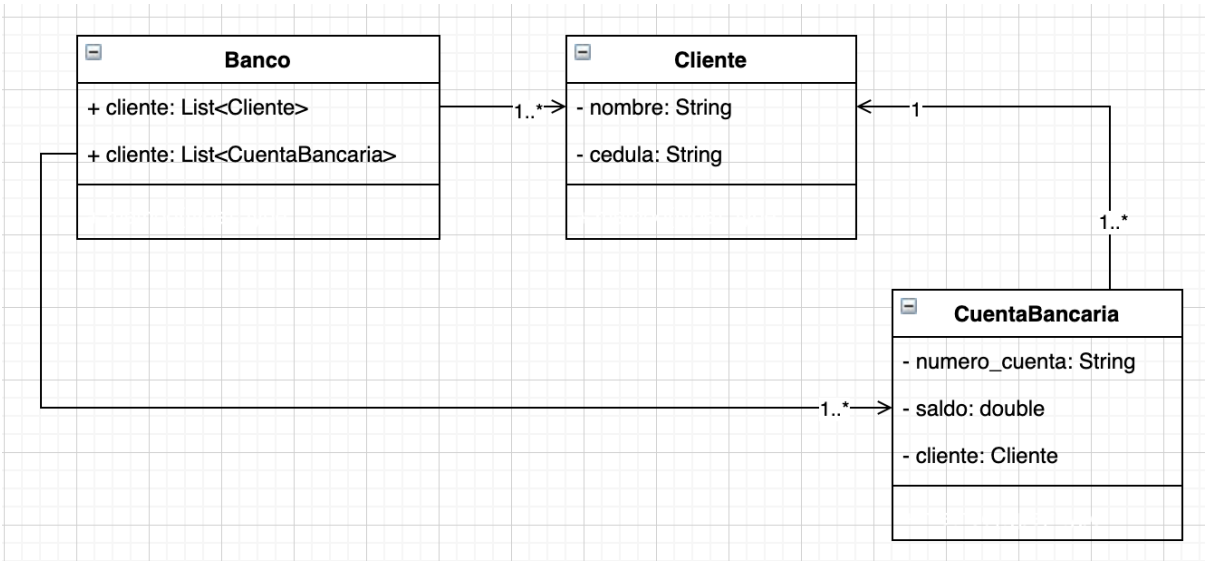
Escuela Superior Politécnica del Litoral
Programación Orientada a Objetos
Taller Manejo de Excepciones y Archivos
2020 – 1T

Se le ha pedido a usted que ayude a desarrollar un sistema para administrar las cuentas bancarias de un banco. A continuación, se presenta el diagrama de clases del sistema



La información de los clientes se guarda en el archivo de texto **src/recursos/clientes.txt**. El archivo tiene el siguiente formato:

cedula, nombre

La información de las cuentas bancarias se guarda en el archivo de texto **src/recursos/cuentas.txt** y tiene el siguiente formato:

numerocuenta, saldocuenta, cedulacliente

Se le pide que realice lo siguiente:

PARTE 1

1. En el paquete data cree una clase llamada **CuentasData.java** tal que:
    a. Tenga una variable estática ARCHIVOCUENTAS con la ruta relativa al archivo de texto donde se encuentra la información de las cuentas bancarias
2. Defina un método estático **obtenerCuentas** que retorne un ArrayList de objetos de tipo CuentaBancaria.
    a. En caso de producirse una excepción de tipo FileNotFoundException cree y lance una excepción de tipo **ProblemasConArchivoException** con el mensaje “Archivo no existe”

   b. En caso de producirse una excepción de tipo IOException cree y lance una excepción de tipo **ProblemasConArchivoException** con el mensaje "de la excepción IOException"
3. Defina un método estático **agregarCuenta** que reciba un objeto de tipo Cuenta y crea un nuevo registro en el archivo cuentas.
   a. En caso de producirse una excepción de tipo IOException cree y lance una excepción de tipo **ProblemasConArchivoException** con el mensaje "de la excepción IOException"
4. Defina un método estático **actualizarCuentas** que reciba una lista de tipo CuentaBancaria y sobreescriba el archivo cuentas bancarias con la información recibida como parametro.
   a. En caso de producirse una excepción de tipo IOException cree y lance una excepción de tipo **ProblemasConArchivoException** con el mensaje "de la excepción IOException"
5. Lee en el archivo test/PruebaCuentas.java
   - llame al método **obtenerCuentas** y muestre las cuentas bancarias devueltas en pantalla.
   - Cree una nueva cuenta y agréguela a la lista de cuentas
   - Cambie el saldo de una cuenta y actualice el archivo

¿**ProblemasConArchivoException** es una excepción de tipo checked o unchecked?

PARTE 2

6. En el paquete modelo en la clase CuentaBancaria realice lo siguiente
   - Implemente el método **depositarDinero** que recibe la cantidad de dinero a depositar en la cuenta.
     - El monto a depositar deber ser mayor a cero. En caso de no serlo lance una excepción de tipo **IllegalArgumentException** con el mensaje ¨El monto a depositar debe ser mayor a cero¨

```
public void depositarDinero(double cantidad){
    if(cantidad<=0)//LA CANTIDAD A RETIRAR DEBE SER MAYOR A CERO
        throw new IllegalArgumentException
    ("El monto a depositar debe ser mayor a cero");

    this.monto += cantidad;
}
```

     - ¿Por qué Java no nos obliga a **capturar o declarar** la excepción de tipo **IllegalArgumentException**?

PARTE 3

7. En el paquete modelo defina una excepción llamada NoHayFondosExcepcion que herede de Exception. Esta debe tener un constructor vacío y un constructor que reciba un mensaje.
   ¿**NoHayFondosExcepcion** es una excepción de tipo checked o unchecked?

8. En la clase **CuentaBancaria** realice lo siguiente

- Implemente el método **retirarDinero** para que reciba la cantidad de dinero a retirar de la cuenta. Si no hay suficiente dinero en la cuenta el método lanza una excepción de tipo **NoHayFondosExcepcion**
    - El monto a depositar deber ser mayor a cero. En caso de no serlo lance una excepción de tipo **IllegalArgumentException** con el mensaje ¨El monto a depositar debe ser mayor a cero¨

PARTE 4

9. Clase **Banco**:
    a. El constructor inicializa la lista de clientes y cuentas. Este método no maneja la excepción ProblemasConArchivoException sino que declara la excepción en la firma.

```
/**
 * Metodo que carga la informacion de clientes y cuentas en el sistema
 * @throws ProblemasConArchivoException cuando hay problemas al leer
 * la informacion
 * de los archivos de clientes o cuentas
 */
public Banco() throws ProblemasConArchivoException{
    //leer los datos de los clientes del banco y cargarlo en la lista clientes
    clientes = ClientesData.leerClientes();
    //leer los datos de las cuentas del banco y cagarlos en la lista cuentas
    cuentas = CuentasData.obtenerCuentasBancaria();
}
```

    b. Implemente el método **crearCuenta.** Este método crea una nueva cuenta y la agrega a la lista de cuentas del banco y al archivo cuentas.txt

```
/**
 * Crea una nueva cuenta bancaria para un cliente dada
 * @param numero
 * @param saldo
 */
public void crearCuenta(Cliente c, String numero, double saldo)
        throws ProblemasConArchivoException{

    //cree un nuevo objeto de tipo cuenta
    CuentaBancaria cta = new CuentaBancaria(numero, saldo, c);

    //agregar el objeto al arreglo de cuentas
    cuentas.add(cta);

    //escribir la cuenta bancaria en el archivo llamando al
    //metodo agregarCuenta de CuentasData
    CuentasData.agregarCuenta(cta);
}
```

10. Clase **BancoUI**: Esta clase representa la vista de la aplicación, con la que el usuario interactúa.

a. En el constructor de la clase creamos una nueva instancia de Banco. Si se produce una excepción de tipo ProblemasConArchivoException terminamos la ejecución del programa ya que no se podría continuar.

```
public BancoUI(){
    try{
        sc = new Scanner(System.in);
        b = new Banco();
    }catch(ProblemasConArchivoException ex){
        //no podemos continuar sino podemos leer la informacion
        //de los clientes, terminamos la ejecucion del programa
        System.out.println("Problemas cargando informacion");
        System.out.println("Archivo: "+ex.getNombre_archivo());
        System.out.println(ex.getMessage());
        System.exit(0);//no nos podemos recuperar - detenemos
        //la ejecucion del programa
    }
}
```

b. Complete el método **crearCuentaBancaria** para que para que pida el número de cedula del cliente al que se le va a agregar la cuenta. Si el cliente no existe salga del método. Si el cliente exista pide el número de cuenta y el saldo inicial de la misma. Su programa no debe caerse en caso que el cliente ingrese un valor distinto a numérico.

```
public void crearCuentaBancaria(){
    System.out.println("Ingrese cedula del cliente ");
    //obtener el cliente a partir de la cedula, si el cliente no existe
    //imprima un mensaje y salga del metodo
    String cedula = sc.nextLine();
    Cliente c= b.buscarCliente(cedula);
    if(c==null){
        System.out.println("Cliente no existe");
        return;
    }
    //pida el numero de cuenta
    System.out.println("Ingrese numero de cuenta");
    String numero = sc.nextLine();
    //pida el monto inicial
    //manejamos las excepciones de tipo NumberFormatException
    boolean continuar = true;
    double saldo = 0;
    do{
        try{
            System.out.println("Ingrese monto inicial de la cuenta");
            saldo = Double.parseDouble(sc.nextLine());
            continuar = false;
        }catch(NumberFormatException ex){
            System.out.println("Monto inicial invalido");
        }
    }while(continuar);
    try {
        //agregue la cuentabancaria
        b.crearCuenta(c, numero, saldo);
        System.out.println("Cuenta creada");
    } catch (ProblemasConArchivoException ex) {
        System.out.println("No se puede agregar la cuenta");
        System.out.println(ex.getMessage());
    }
}
```

PARTE 5

11. Clase **Banco**

a. Implemente el método BuscarCuenta

```
/**
 * Retorna la informacion de una cuenta dada su numero
 * si la cuenta no existe retorna numer
 * @param numerocuenta
 * @return el cuenta con el numero dado
 */
public CuentaBancaria buscarCuenta(String numerocuenta){
    //retorna la cuenta con el numero dado
    return null;
}
```

b. Implemente el método depositarDinero según las instrucciones dadas en la clase

```
/**
 * Realiza el deposito del monto pasado como parametro a la cuenta
 * Maneje las excepciones que se pueden producir mostrando el mensaje
 * de la excepcion producida en pantalla
 * @param numeroCuenta
 * @param monto a depositar
 */
public void depositarDinero(CuentaBancaria cta, double monto){

    //llame al metodo depositarDinero de cta

    //actualice el archivo cuentasBancarias llamando al metodo
    //actualizarCuentas de CuentasData
    //declare las excepciones de tipo checked que se pueden lanzar en la
    //firma del metodo

}
```

c. Implemente el método retirarDinero según las instrucciones dadas en la clase

```
/**
 * Realiza el retiro del monto pasado como parametro a la cuenta
 * Maneje las excepciones que se pueden producir mostrando el mensaje
 * de la excepcion producida en pantalla
 * @param cta
 * @param monto a retirar
 */
public void retirarDinero(CuentaBancaria cta, double monto){

    //llame al metodo retirarDinero de cta
    //declare las excepciones de tipo checked que se pueden lanzar en la
    //firma del metodo


    //actualice el archivo cuentasBancarias llamando al metodo
    //actualizarCuentas de CuentasData
    //declare las excepciones de tipo checked que se pueden lanzar en la
    //firma del metodo
}
```

12. Clase **BancoUI**

a. Implemente el método depositarDinero

```java
public void depositarDinero(){
    //pida el numero de cuenta del que quiere realiza la transaccion
    System.out.println("Ingrese numero de cuenta");
    //pida el numeo de cuenta, si la cuenta no existe
    //imprima un mensaje y salga del metodo


    //pida el monto a depositar
    System.out.println("Ingrese el monto a depositar");


    //llame al metodo depositarDinero de Banco
    //mostrar el nuevo saldo de la cuenta
    //si se produce una excepcion de tipo IlegarlArgumentException
    //vuelva a pedir el valor
    //si se produce una excepcion de tipo ProblemasConArchivoException,
    //muestre el mensaje de la excepcion y salga del metodo
}
```

b. Implemente el método retirarDinero

```java
public void retirarDinero(){
    System.out.println("Ingrese numero de cuenta");
    ///pida el numeo de cuenta, si la cuenta no existe
    //imprima un mensaje y salga del metodo


    //pedir el monto a retirar
    double monto = sc.nextDouble();


    //llame al metodo retirarDinero de Banco
    //mostrar el nuevo saldo de la cuenta
    //si se produce una excepcion de tipo IlegarlArgumentException
    //vuelva a pedir el valor
    //si se produce una excepcion de tipo NoHayFondosExcepcion, muestre el
    //el mensaje de la excepcion y salga del metodo
    //si se produce una excepcion de tipo ProblemasConArchivoException,
    //muestre el mensaje de la excepcion y salga del metodo
}
```