



Universidad De Talca

Facultad De Ingeniería

Departamento De Ciencias De La Computación

Informe Tarea 2 Programación Avanzada

Tarea Simulación Bancaria

Fecha: 05/07/2021

Autores: Valentina Yáñez

Antonia Fuenzalida

Diego Fernández

e-mail: vyañez20@alumnos.otalca.cl

afuenzalida20@alumnos.otalca.cl

dfernandez19@alumnos.otalca.cl

Introducción

En este informe damos a conocer nuestra solución a la simulación bancaria, consiste en una simulación que permite a los clientes saber el rendimiento de sus productos, además permite contemplar diferentes escenarios y situaciones que el cliente podría presentar. Primero abordamos el programa en un UML, este nos ayudó al momento de crear nuestra simulación y así visualizarla, luego lo pasamos a un código en Java, en el cual implementamos la modelación orientada a objetos, con relaciones de herencia, asociación y persistencia de objetos, aplicando así todo lo aprendido en la segunda unidad de Programación Avanzada.

Análisis del problema

Este programa es una simulación bancaria que permite a los clientes saber los movimientos de su cuenta, su rendimiento, además contempla diferentes situaciones a las que el cliente podría presentarse. Esta simulación guarda el nombre y la cédula del cliente con la cual se puede identificar la cuenta de él. Consta de tres productos financieros. Su cuenta de ahorro en donde el cliente recibe un interés mensual, una cuenta de ahorro que no recibe interés y un certificado de depósito a término, en esta cuenta el cliente no puede consignar ni retirar dinero.

En este problema el profesor nos proporcionó un simulador que consta de trece funcionalidades que se vieron en la unidad uno, ahora nuestro trabajo es implementar nuevas funcionalidades a la simulación en el cual debemos externalizar los objetos, utilizando como metodología lo aprendido en la unidad uno y dos, como la serialización, el encapsulamiento, el polimorfismo, la herencia, clases abstractas, interfaces, entre otras metodologías. Todo esto aplicado en un UML para luego poder pasarlo a código java.

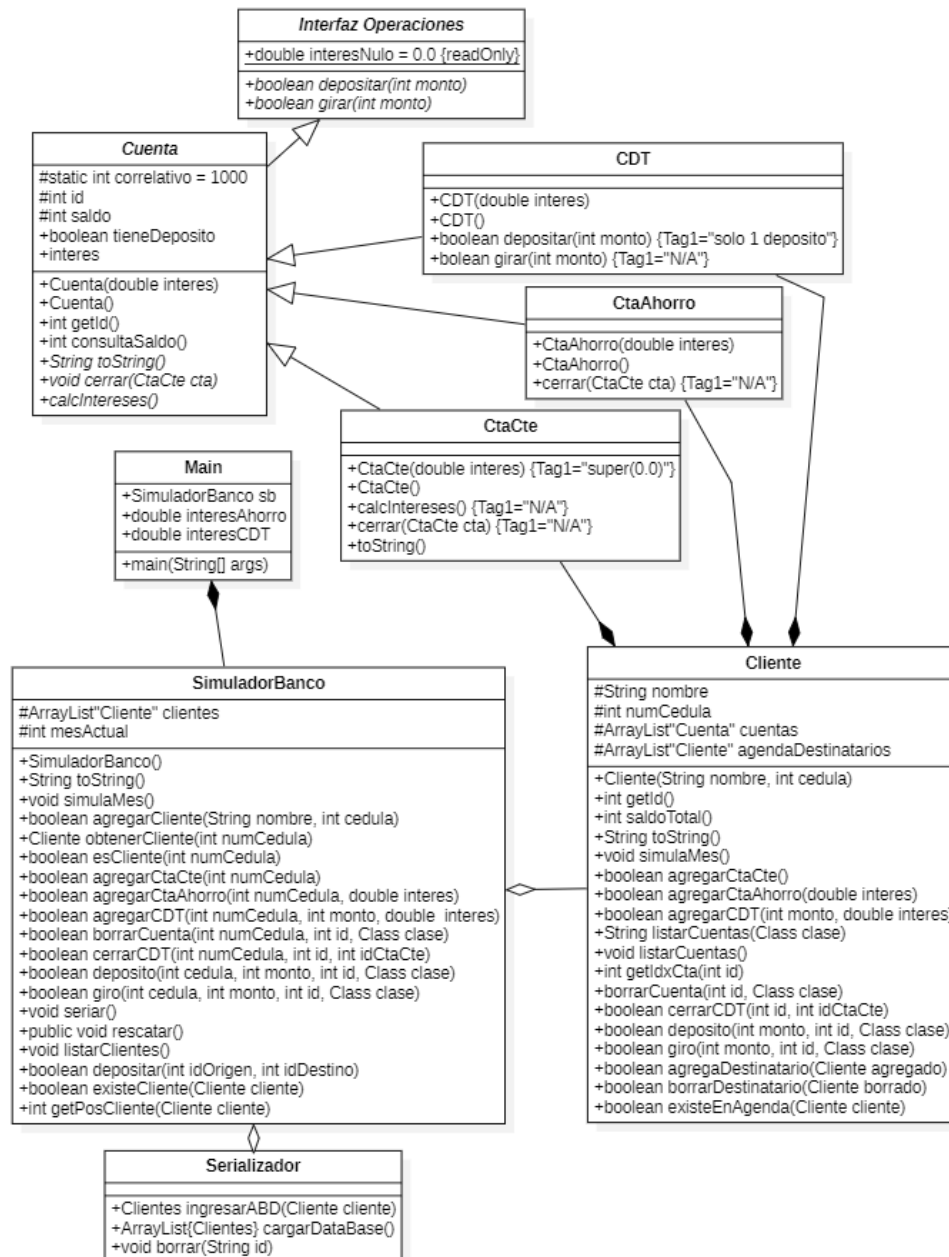
Solución del problema

- **Descripción de la solución**

En esta sección se da a conocer nuestra solución aplicada en UML.

Este UML solo fue completado con las funcionalidades que se añadieron a esta simulación bancaria, ya que el profesor nos proporcionó un UML que contenía las funcionalidades principales que él implementó.

El método serializador se creó para manipular los datos de la memoria física del computador y se agregaron las nuevas funcionalidades, además de un atributo en cliente que contiene la agenda de destinatarios.



● Implementación:

Este simulador cuenta con veinte funcionalidades, trece fueron aplicadas por nuestro profesor con anterioridad y los siete restantes las aplicamos nosotros. Ahora se dará a conocer de forma más específica cada una de estas nuevas funcionalidades.

Funcionalidad 14 consiste en eliminar un cliente según su cédula, esto se puede realizar solo si el saldo de sus cuentas es 0, al momento de borrar al cliente se elimina de la base de datos.

```
case 14: //borrar un cliente por cédula.
    System.out.print("Indica la cédula: ");
    cedula = in.nextInt();
    clieAux = sb.obtenerCliente(cedula);
    if(sb.existeCliente(clieAux) && clieAux.saldoTotal()==0){
        sb.seriar();
        Serializador s = new Serializador();
        sb.clientes.remove(clieAux);
        s.borrar(Integer.toString(clieAux.getId()));
        System.out.println("se eliminó el cliente de cédula "+ clieAux.getId());
    }else{
        System.err.println("El saldo del cliente no es 0 por lo que no se puede eliminar");
    }
    sb.seriar();
    break;
```

Funcionalidad de Mantener una Agenda de Destinatarios. Se hizo de una manera diferente ya que mantener una agenda no es un método, es más un atributo, entonces se agregó a los clientes el atributo de tener cada uno una agenda.

```
public class Cliente implements Serializable{
    protected String nombre;
    protected int numCedula;
    protected ArrayList<Cuenta> cuentas;
    protected ArrayList<Cliente> agendaDestinatarios; //nuevo atributo
```

Funcionalidad 15 Agrega un destinatario a la agenda por cédula, primero se verifica si este invalida las cuentas anteriormente serializadas, luego verifica que los clientes existan para luego agregar a clieAux2 a agendaDestinatarios de clieAux el cual se encuentra en un array clientes dentro del simulador bancario.

```
System.out.print("Indica la cédula del dueño de la agenda: ");
clieAux= sb.obtenerCliente(in.nextInt()); //dueño de la agenda destino
System.out.print("Indica la cédula del destinatario a agregar: ");
clieAux2 = sb.obtenerCliente(in.nextInt()); //cliente a agregar a la agenda destino

if(sb.existeCliente(clieAux) && sb.existeCliente(clieAux2)){
    sb.clientes.get(sb.getPosCliente(clieAux)).agregaDestinatario(clieAux2);
}
break;
```

Funcionalidad 16 borramos un destinatario de la Agenda por cédula, primero se solicita la cédula del dueño de la cuenta, luego se pide el ID de destinatario a eliminar de la agenda. Entonces si los clientes existen se borra al cliente de la agenda, en caso de no existir no se borra nada.

```
case 16: //borra un destinatario de la agenda por cédula.
    System.out.print("Indica la cédula del dueño de la agenda: ");
    clieAux= sb.obtenerCliente(in.nextInt()); //dueño de la agenda destino
    System.out.print("Indica la cédula del destinatario a eliminar: ");
    clieAux2 = sb.obtenerCliente(in.nextInt()); //cliente a agregar a la agenda des

    if(sb.existeCliente(clieAux) && sb.existeCliente(clieAux2)){
        sb.clientes.get(sb.getPosCliente(clieAux)).borrarDestinatario(clieAux2);
    }
    sb.seriar();
    break;
```

Funcionalidad 17 transfiere un monto a un destinatario, introduce al dueño de la lista destinatarios, para introducir el Rut del destinatario. Luego se verifica si el ID es una cuenta que se encuentra en el arrayDestinatarios de la cuenta del cliente emisor para quitarle la misma cantidad de dinero a la cuenta corriente del cliente emisor.

```
public boolean depositar(int idOrigen, int idDestino) {
    Scanner in = new Scanner(System.in);
    Cliente cOrigen = obtenerCliente(idOrigen);
    Cliente cDestino = obtenerCliente(idDestino);
    if(cOrigen == null)return false;
    if(cDestino==null)return false;
    for (Cliente cliente : cOrigen.getAgenda()) {
        if(cliente.getId() == idDestino){
            System.out.println("Eliga cuenta del destinatario: \n");
            for (Cuenta cuenta : cliente.cuentas)System.out.println(cuenta.getId());
            System.out.print("nmro cta: ");
            int nmroCtaDestino = in.nextInt();
            System.out.println("Eliga cuenta origen: \n");
            for(Cuenta cuenta : obtenerCliente(idOrigen).cuentas)System.out.println(cuenta);
            System.out.print("nmro cta: ");
            int nmroCtaOrigen = in.nextInt();
            for (Cuenta cuentaD : cliente.cuentas) {
                for(Cuenta cuenta0 : obtenerCliente(idOrigen).cuentas){
                    if(cuentaD.getId()==nmroCtaDestino && cuenta0.getId()==nmroCtaOrigen){
                        System.out.print("Cantidad a depositar: ");
                        int dep = in.nextInt();
                        if(!cuentaD.depositar(dep))return false;
                        if(!cuenta0.girar(dep))return false;
                        return true;
                    }
                }
            }
        }
    }
    return false;
}
```

Estas dos últimas funcionalidades tienen relación con la serialización o externalizar el simulador, pero primero en Cliente y Cuenta se tuvo que implementar Serializable, para así asegurarnos de que el objeto va a ser serializable. Ahora de forma detallada se explicarán estas últimas dos funcionalidades. Aquí llamamos a la librería java.nio que nos sirvió para recorrer la carpeta con los clientes y rescatar el nombre de cada uno de los archivos, el cual sería su id.

Funcionalidad 18 Guarda el estado del sistema, restaura a los clientes actuales junto a sus cuentas y agendas de destinatarios. Aquí se serializa a un solo cliente dentro de ingresarABD, pero acá en serial, que se encuentra en SimuladorBanco, se llama a ingresar a ingresarABD para cada cliente que hay en la lista de clientes de la memoria del programa.

```
public class Serializador implements Serializable{
    //clase para serializar el objeto cliente
    public Cliente ingresarABD(Cliente cliente) throws IOException{
        FileOutputStream file = new FileOutputStream("DataBase/"+cliente.getId());
        ObjectOutputStream output = new ObjectOutputStream(file);
        if(output != null){
            output.writeObject(cliente);

            output.close();
        }
        file.close();
        return cliente;
    }
}
```

Funcionalidad 19 restaura el estado del sistema. El método cargarDataBase del serializador recorre la carpeta DataBase, abre cada archivo y los agrega a una lista de clientes, al finalizar el método se retorna la lista de los clientes, si la carpeta está vacía retorna null.

```
public ArrayList<Cliente> cargarDataBase() throws IOException{
    FileInputStream file;
    ObjectInputStream input;
    ArrayList<Cliente> clientes = new ArrayList<Cliente>();
    Cliente cliente = null;

    try (DirectoryStream<Path> stream = Files.newDirectoryStream(Paths.get("DataBase/"))){
        for (Path path : stream) {
            file = new FileInputStream("DataBase/"+path.getFileName().toString());
            input = new ObjectInputStream(file);

            cliente = (Cliente)input.readObject();
            clientes.add(cliente);
        }
    } catch (Exception e) {
        System.err.println(e);
    }

    return clientes;
}
```

- **Modo de uso:**

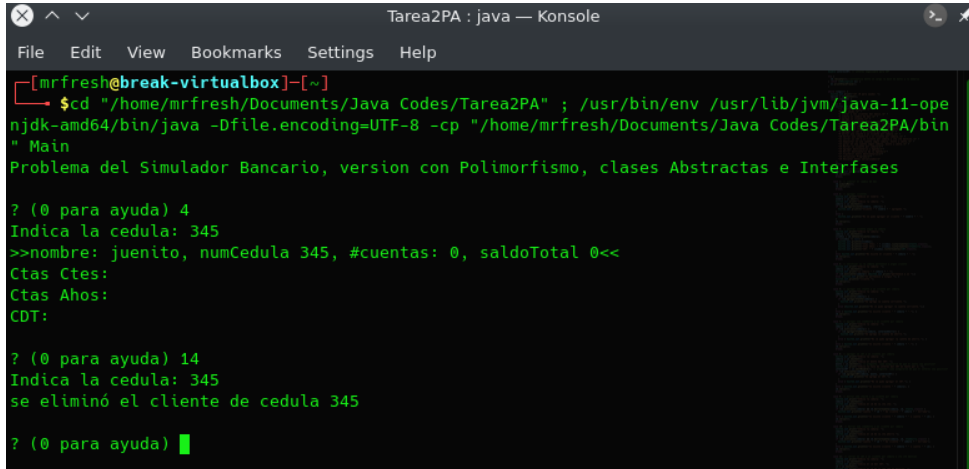
Nuestro programa fue escrito en el lenguaje Java, su JDK es javaSe-10 (eclipse java versión) y su IDE de desarrollo fue eclipse, además se utilizó el editor Visual Studio Code.

Para poder utilizarlo se ejecuta el código con vscode en la carpeta .vscode hay perfiles de ejecución que se pueden utilizar, el de launch external terminal, lo ejecuta en una terminal a parte y launch app lo ejecuta en la terminal de visual, existe una opción alternativa a visual estudio se llama “vscodium”, de todas maneras es posible ejecutarlo por símbolos del sistema mediante los símbolos del sistema, utilizando los comando de compilación del lenguaje, en la carpeta “src” se contiene el path de estructuras y la carpeta “bin” contiene los archivos compilados. Las entradas del programa son por teclado, al momento de ejecutar el programa se cargará la “base de datos” de la carpeta “DataBase”, este programa utiliza serialización a los clientes del arraylist de clientes de simuladorbanco, en la carpeta “DataBase” se encuentran los clientes ordenados por id, cada vez que se realice una operación se cargará el cambio en la base de datos para así evitar perdida de información.

Pruebas

A continuación, se darán a conocer nuestras pruebas realizadas al momento de compilar el código.

En nuestra primera prueba utilizamos la funcionalidad 14, en la cual se elimina a un cliente.



```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help

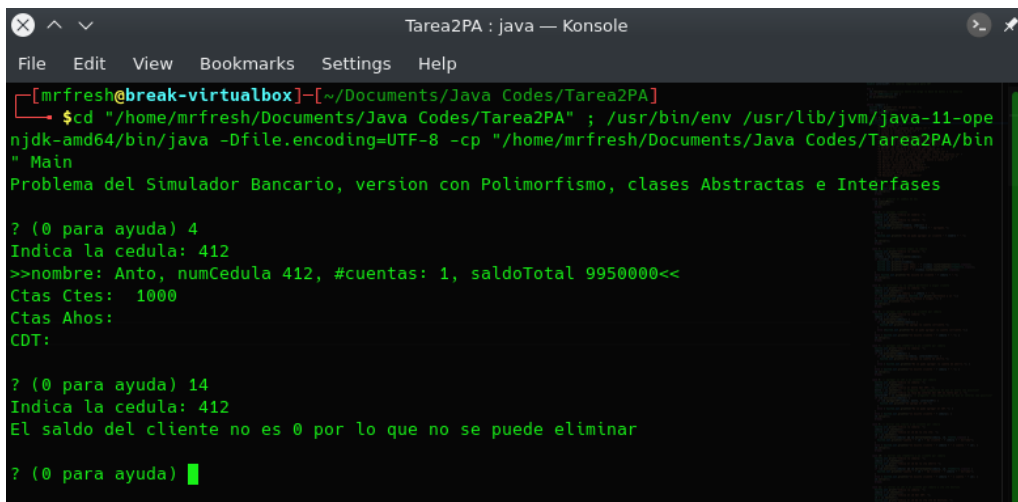
[mrfresh@break-virtualbox]~$ cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" "Main"
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfases

? (0 para ayuda) 4
Indica la cedula: 345
>>nombre: juenito, numCedula 345, #cuentas: 0, saldoTotal 0<<
Ctas Ctes:
Ctas Ahos:
CDT:

? (0 para ayuda) 14
Indica la cedula: 345
se eliminó el cliente de cedula 345

? (0 para ayuda) █
```

En este caso como el saldo del cliente es superior a cero no lo elimina.



```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help

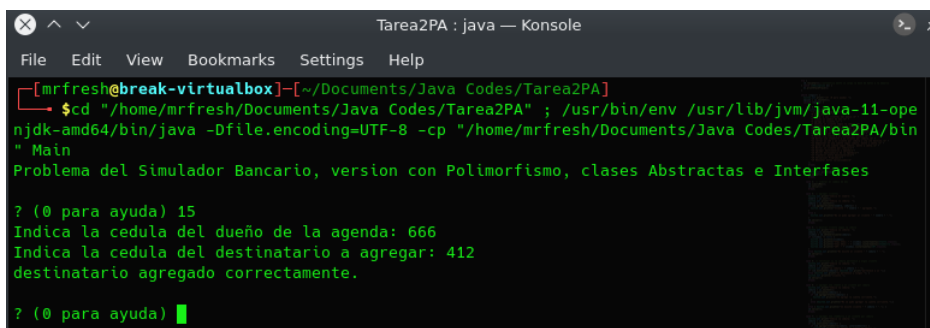
[mrfresh@break-virtualbox]~/Documents/Java Codes/Tarea2PA$ cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" "Main"
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfases

? (0 para ayuda) 4
Indica la cedula: 412
>>nombre: Anto, numCedula 412, #cuentas: 1, saldoTotal 9950000<<
Ctas Ctes: 1000
Ctas Ahos:
CDT:

? (0 para ayuda) 14
Indica la cedula: 412
El saldo del cliente no es 0 por lo que no se puede eliminar

? (0 para ayuda) █
```

La funcionalidad 15 agrega un destinatario a la agenda.



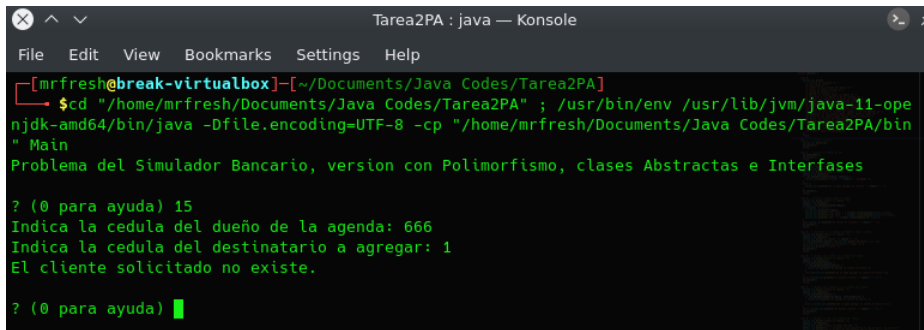
```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help

[mrfresh@break-virtualbox]~/Documents/Java Codes/Tarea2PA$ cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" "Main"
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfases

? (0 para ayuda) 15
Indica la cedula del dueño de la agenda: 666
Indica la cedula del destinatario a agregar: 412
destinatario agregado correctamente.

? (0 para ayuda) █
```


El destinatario no existe.



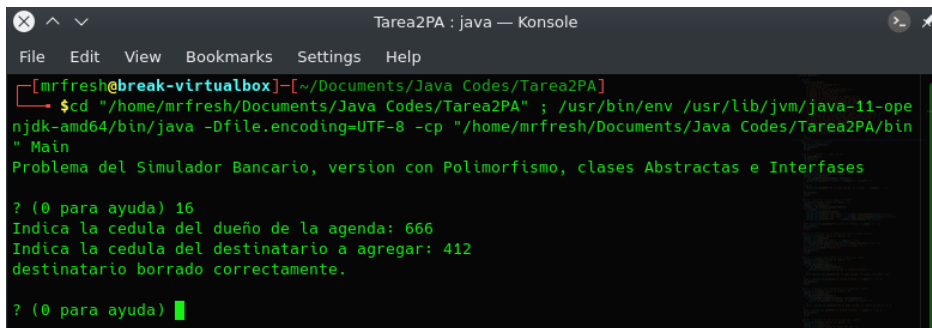
```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help

[mrfresh@break-virtualbox]~/Documents/Java Codes/Tarea2PA
$ cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" Main
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfases

? (0 para ayuda) 15
Indica la cedula del dueño de la agenda: 666
Indica la cedula del destinatario a agregar: 1
El cliente solicitado no existe.

? (0 para ayuda) █
```

En la funcionalidad 16 borra un destinatario de la agenda.



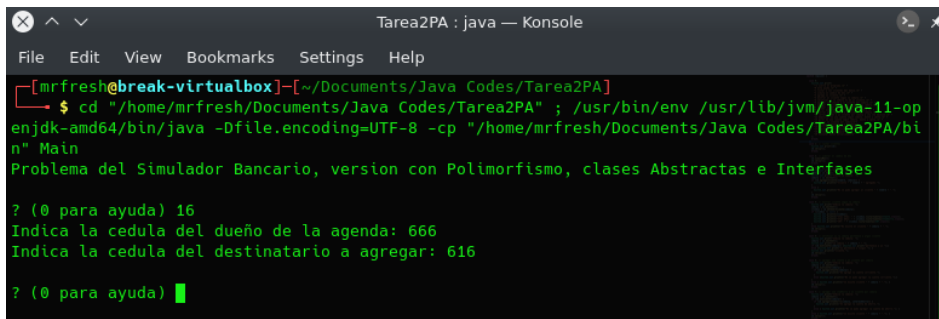
```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help

[mrfresh@break-virtualbox]~/Documents/Java Codes/Tarea2PA
$ cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" Main
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfases

? (0 para ayuda) 16
Indica la cedula del dueño de la agenda: 666
Indica la cedula del destinatario a agregar: 412
destinatario borrado correctamente.

? (0 para ayuda) █
```

En este caso el programa intenta borrar un contacto de una agenda en la cual no está, por lo que no se muestra un mensaje.



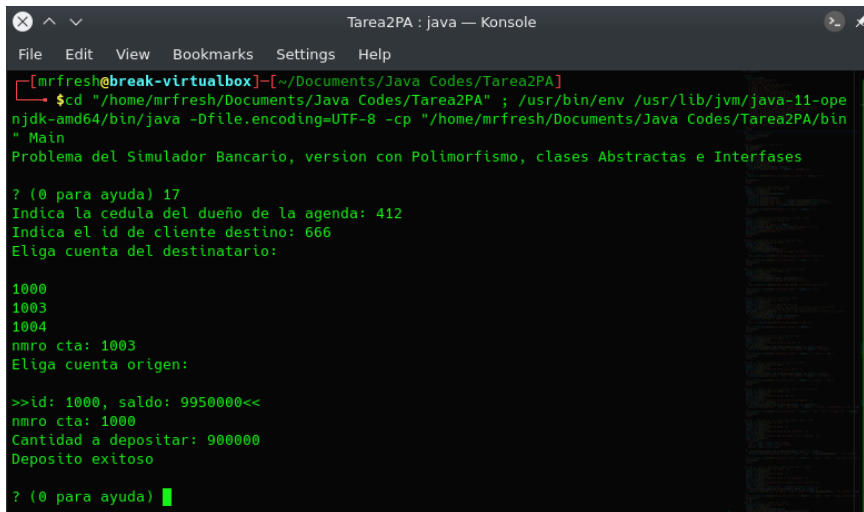
```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help

[mrfresh@break-virtualbox]~/Documents/Java Codes/Tarea2PA
$ cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" Main
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfases

? (0 para ayuda) 16
Indica la cedula del dueño de la agenda: 666
Indica la cedula del destinatario a agregar: 616
destinatario borrado correctamente.

? (0 para ayuda) █
```

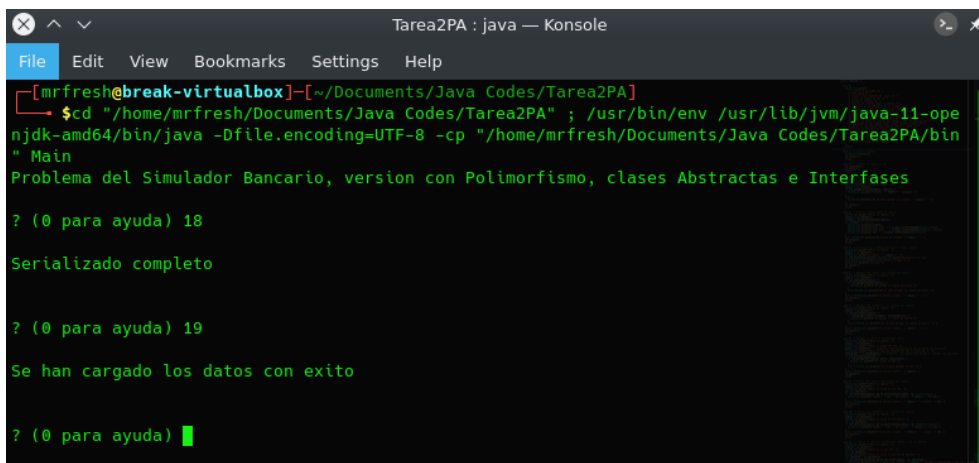
La funcionalidad 17 transfiere un monto de dinero a otro destinatario.



```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help
[mrfresh@break-virtualbox]~/Documents/Java Codes/Tarea2PA
$cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" Main
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfaces

? (0 para ayuda) 17
Indica la cedula del dueño de la agenda: 412
Indica el id de cliente destino: 666
Eliga cuenta del destinatario:
1000
1003
1004
nmro cta: 1003
Eliga cuenta origen:
>>id: 1000, saldo: 9950000<<
nmro cta: 1000
Cantidad a depositar: 900000
Deposito exitoso
? (0 para ayuda)
```

La funcionalidad 18 carga los datos de la lista de clientes a la carpeta DataBase y la 19 recupera los datos de la carpeta DataBase y los manda al banco.



```
Tarea2PA : java — Konsole
File Edit View Bookmarks Settings Help
[mrfresh@break-virtualbox]~/Documents/Java Codes/Tarea2PA
$cd "/home/mrfresh/Documents/Java Codes/Tarea2PA" ; /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 -cp "/home/mrfresh/Documents/Java Codes/Tarea2PA/bin" Main
Problema del Simulador Bancario, version con Polimorfismo, clases Abstractas e Interfaces

? (0 para ayuda) 18
Serializado completo

? (0 para ayuda) 19
Se han cargado los datos con exito

? (0 para ayuda)
```