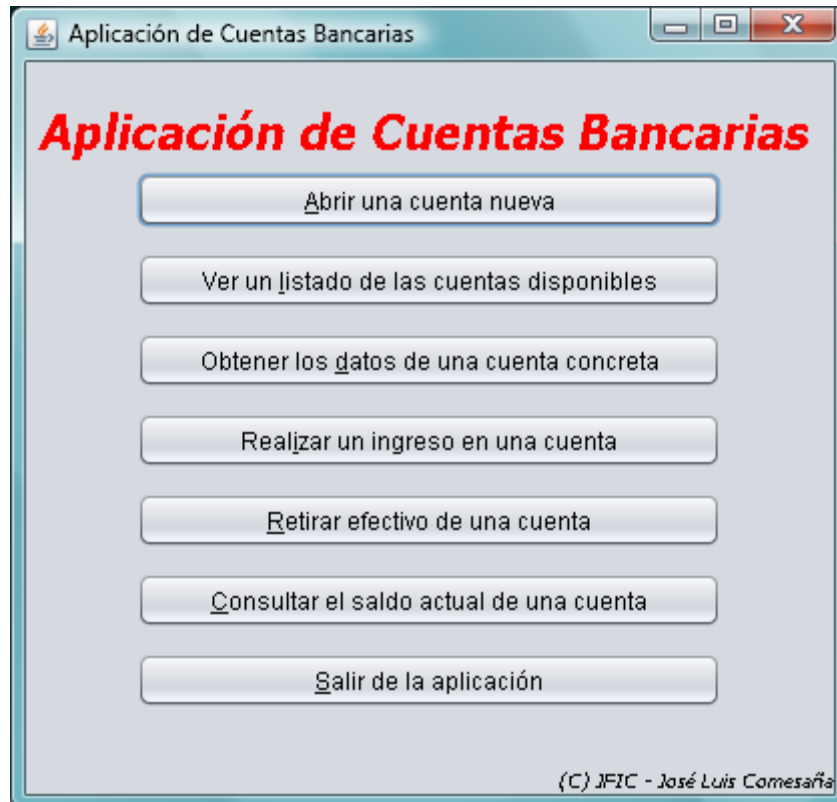


Para la realización de esta tarea he utilizado todos los conceptos aprendidos en la unidad, y que iré detallando.

El apartado principal o menú consta de las siete opciones solicitadas, de las cuales las dos primeras son las más complejas.

AplicacionCuentaBancaria.java será con la que se inicie el programa mostrando la siguiente pantalla:



Esta clase no tiene mayor problema que el de presentar varios botones con un ActionListener que nos envíe a los distintos apartados cuando pulsemos sobre su botón. Éste sería el único JFrame de la aplicación, ya que todas las demás ventanas serán de tipo JDialog, para que cuando cerremos una ventana, volvamos a este apartado principal. Heredará de JFrame.

Le he puesto un mnemonic para poder utilizar atajos de teclado con las distintas opciones.

Si pulsamos sobre **Abrir una cuenta nueva**, iremos a CuentaNueva.java que es una clase que hereda de JDialog, y que nos valdrá para introducir los datos de cualquier cuenta (sólo mostrará los campos propios de cada cuenta cuando se seleccione en el ComboBox habilitado para ello).

Al principio pide los datos comunes, y he



puesto un calendario que se activa con la pulsación sobre el botón a la derecha de Fecha de Nacimiento para que sea más fácil seleccionar una fecha correcta, aunque si no es así, avisará con el error correspondiente. El calendario es una clase que me descargué de internet y que adapté para colocarlo sobre un JDialog y le añadí el botón de aceptar para conseguir capturar la fecha seleccionada, por eso en el proyecto CuentasBancarias existen dos paquetes que no son míos (org.freixas.jcalendar y org.freixas.jcalendar.images) y que importo al principio del documento con su correspondiente cláusula import.

Si seleccionamos Cuenta de Ahorro como tipo de cuenta, nos aparece el campo de intereses, si seleccionamos Cuenta Corriente Personal nos aparecerá el campo Comisión y un JTable para incluir los datos que queramos como Entidades Autorizadas. Si seleccionamos Cuenta Corriente de Empresa aparecerá Intereses descubierto, Máximo descubierto y comisión por descubierto, así como otro JTable para las entidades autorizadas en este tipo de cuenta.

En este apartado se controla que no existan campos vacíos en la parte común a los tres tipos de cuenta, que en los campos numéricos no se introduzcan valores de texto ni negativos, que la fecha sea correcta, que los dígitos de control sean correctos y que el código de cuenta no esté dado de alta con anterioridad.

Cuando rellenamos toda la información y pulsamos sobre el botón de Aceptar es cuando se realiza toda la comprobación de los datos y en caso de existir algún error se nos mostrará una ventana emergente donde nos indicará qué información hemos de corregir. En caso que todo sea correcto se hará la grabación en el

ArrayList correspondiente al tipo de cuenta que hayamos seleccionado en el ComboBox.

Si pulsamos en Cancelar no grabará ninguna información, haciendo desaparecer la ventana y volviendo de nuevo al menú principal

The image displays three sequential screenshots of a Java Swing dialog box titled "Abrir una cuenta nueva". The dialog box contains the following fields and components:

- Nombre:** Text field with "José Luis".
- Apellidos:** Text field with "Comesaña Cabeza".
- Saldo:** Text field with "1234".
- Fecha de Nacimiento:** Text field with "28-4-1965" and a calendar icon to its right.
- Tipo de cuenta:** A dropdown menu showing "Cuenta de Ahorro" in the first screenshot, "Cuenta Corriente Pers..." in the second, and "Cuenta Corriente de E..." in the third.
- Número de la Cuenta de Cliente:** A text field with a masked value "8888 / 9999 / 10 / 0000000000".
- Intereses:** A text field with "12" (only visible in the first screenshot).
- Comisión:** A text field with "10" (only visible in the second screenshot).
- Entidades Autorizadas:** A JTable with two columns: "Entidad" and "Cantid...". It contains two rows: "omega" with value "999" and "epsilon" with value "888".
- Intereses descubierto:** A text field with "15" (only visible in the third screenshot).
- Máximo descubierto:** A text field with "1000" (only visible in the third screenshot).
- Comisión por descubierto:** A text field with "25" (only visible in the third screenshot).
- Buttons:** "Cancelar" and "Aceptar" buttons at the bottom.

En **Ver un listado de las cuentas disponibles** decidí poner cuatro botones en la parte superior para seleccionar si deseamos ver un listado de los clientes de *Cuentas de Ahorro*, *Cuentas Corrientes Personales*, *Cuentas Corrientes de Empresa* o *Todos los clientes*.



En caso de no existir datos de una cuenta concreta e intentar mostrarlo, se nos aparecerá una ventana avisándonos de ello. Si existen datos de alguna cuenta podremos mostrarlo desde el apartado de todas las cuentas, pero si no existen datos en ninguna de ellas, en este punto también nos aparecerá la ventana avisándonos de ello.

Si pulsamos sobre salir desaparecerá esta ventana y volveremos al menú principal

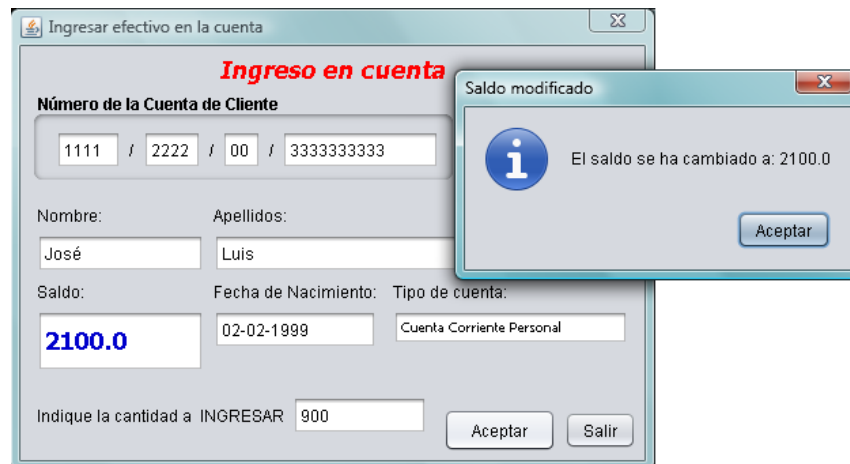
En **Obtener los datos de una cuenta concreta** nos aparece una ventana con sólo la posibilidad de teclear el número de la cuenta para que al pulsar sobre Aceptar nos muestre los datos propios de dicha cuenta o, en caso de no existir ninguna con dicha numeración, nos avise con el correspondiente mensaje.

Aquí se comprueba que los datos tecleados sean dígitos numéricos válidos, que se hayan tecleado los dígitos de control correctos. También se inhabilitan todos los campos una vez que se está mostrando la información del cliente para no poder modificarlos.

Si nos está mostrando la información de una cuenta y pulsamos de nuevo sobre Aceptar, desaparecerá la información y volverá a salirnos únicamente el código de la cuenta habilitando de nuevo la introducción de datos para volver a buscar a otro cliente. Cuando pulsamos Salir desaparece la ventana y volvemos al menú principal.

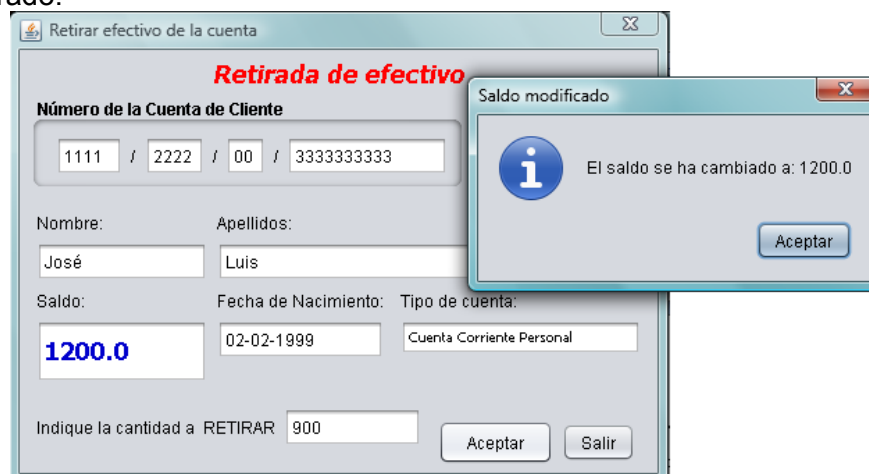
En la opción de **Realizar un ingreso en una cuenta** nos vuelve a aparecer únicamente los campos del número de cuenta para poder buscar el cliente, en el que se controla que no sean valores negativos, texto, que los dígitos de control sea correctos y que el código de cuenta exista dado de alta.

Cuando le indicamos un valor de cuenta válido nos muestra la información deshabilitando todos los campos para impedir su cambio, y en la parte inferior nos saldrá una caja de texto donde deberemos indicar la cantidad a ingresar (se comprueba que sea numérico y positivo).



Cuando hemos tecleado un valor y pulsamos sobre Grabar, nos aparece una ventana informándonos del nuevo saldo al mismo tiempo que se cambia el valor de la caja de texto que contiene el Saldo. Cuando pulsamos Aceptar en la ventana emergente desaparece toda la información y volvemos a tener la posibilidad de teclear otro número de cuenta, de nuevo habilitado.

En el apartado de **Retirar Efectivo de una cuenta** se procede de la misma forma que en la opción anterior, es decir, tecleamos el número de cuenta completo y nos aparece la información del cliente al que pertenece deshabilitando todos los campos o una ventana indicándonos que no se ha encontrado.



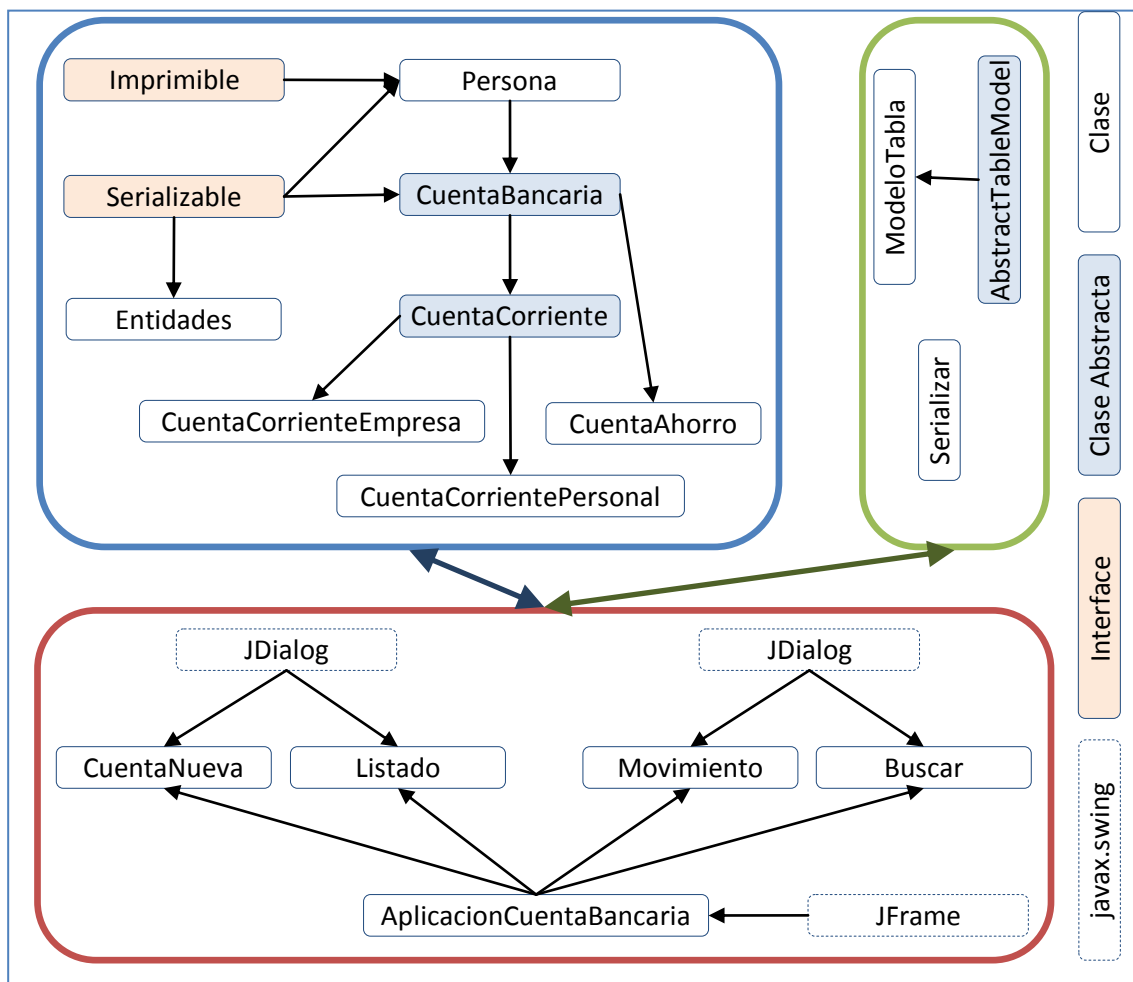
En la parte inferior nos solicita la cantidad a retirar y comprobamos que sea numérica, positiva y menor que el saldo mas el máximo de descubierto (en caso de ser una cuenta corriente de empresa).

Si hemos tecleado una cantidad correcta, se descuenta y nos muestra una ventana donde nos informa cómo se queda el saldo actual de la cuenta, al mismo tiempo que cambia el valor en el campo de texto de saldo.

En la opción de **Consultar el saldo actual de una cuenta** nos aparece una ventana igual que la de las dos opciones anteriores pero sin la línea inferior.

Con la última opción nos limitamos a salir de la aplicación mediante `System.exit(0)`.

Este sería un esquema de la aplicación resultante:



En esta tarea se ha utilizado gran parte de lo aprendido en la unidad, por ejemplo:

Composición:

```
import org.freixas.jcalendar.JCalendar;

...
public class CuentaNueva extends javax.swing.JDialog {
    ...
    // Botón del calendario
    private void calendarioActionPerformed(java.awt.event.ActionEvent evt) {
        JCalendar jcal=new JCalendar();
        SimpleDateFormat formatoFecha = new SimpleDateFormat("dd-MM-yyyy");
        // Si tenemos alguna fecha tecleada, la usamos para asignarla al calendario
        if(fecNac.getText()!=null){
            try{
                Date fechaBoton=formatoFecha.parse(fecNac.getText());
                jcal.setDate(fechaBoton);
            }catch(Exception e){}
        }
        // Dibujamos el calendario en una ventana emergente, igual que cualquier aviso
        JOptionPane.showMessageDialog(this, jcal,"Seleccione la fecha",JOptionPane.PLAIN_MESSAGE);
        // Cuando seleccionemos una fecha en el calendario la ponemos en fecNac
        // en el formato DD-MM-AAAA
        fecNac.setText(
            String.valueOf(jcal.getCalendar().get(5))+"-"+
            String.valueOf(jcal.getCalendar().get(2)+1)+"-"+
            String.valueOf(jcal.getCalendar().get(1)));
    }
    ...
}
```

Implementación de interfaces:

```
public abstract class Persona implements Serializable,Imprimible {...}
public abstract class CuentaBancaria extends Persona implements Serializable{...}
public class CuentaAhorro extends CuentaBancaria implements Serializable{...}
```

Herencia:

```
public class AplicacionCuentaBancaria extends javax.swing.JFrame {...}
public class CuentaNueva extends javax.swing.JDialog {...}
public class Listados extends javax.swing.JDialog {...}
public class Movimiento extends javax.swing.JDialog {...}
public class Buscar extends javax.swing.JDialog {...}
public abstract class CuentaBancaria extends Persona implements Serializable{...}
public class CuentaAhorro extends CuentaBancaria implements Serializable{...}
public abstract class CuentaCorriente extends CuentaBancaria{...}
public class CuentaCorrienteEmpresa extends CuentaCorriente {...}
public class CuentaCorrientePersonal extends CuentaCorriente {...}
public class ModeloTabla extends AbstractTableModel {...}
```

Redefinición de métodos heredados

```
public class Buscar extends javax.swing.JDialog {
    ...
    public static void main(String args[]) {
        ...
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                Buscar dialog = new Buscar(new javax.swing.JFrame(), true);
                dialog.addWindowListener(new java.awt.event.WindowAdapter() {

                    @Override
                    public void windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    }
                });
            }
        });
    }
}
```

```
        });
        dialog.setVisible(true);
    }
    });
}
...
}

public abstract class CuentaBancaria extends Persona implements Serializable{
...
    @Override
    public String getNombre(){
        return nombre;
    }

    @Override
    public String getApellidos(){
        return apellidos;
    }

    @Override
    public GregorianCalendar getFechaNac(){
        return fechaNac;
    }

    @Override
    public void setNombre(String nombre){
        this.nombre=nombre;
    }

    @Override
    public void setApellidos(String apellidos){
        this.apellidos=apellidos;
    }

    @Override
    public void setFechaNac(GregorianCalendar fechaNac){
        this.fechaNac=fechaNac;
    }
    ...
}
```

Clases abstractas:

```
public abstract class Persona implements Serializable, Imprimible {...}
public abstract class CuentaBancaria extends Persona implements Serializable{...}
public abstract class CuentaCorriente extends CuentaBancaria{...}
```

Interfaces:

```
public interface Imprimible {
    ArrayList ContenidoArrayList();
    Hashtable ContenidoHashtable();
}
```

Espero con esta tarea haber cumplido todas las premisas requeridas para la realización del ejercicio. Además, he generado el javadoc y el ejecutable.