Software Development 2
Technological University Dublin
Assignment 2

Krystian Adam Kunowski

Student number : B00123405

TABLE OF CONTENTS :

INSTRUCTION (HOW TO RUN CREDITUNION APPLICATION)

Please to compile and run in correct order:

1. CreateBankFile.java
2. Record.java
3. CreditUnion.java

Introduction

This program been created for a purpose to manages user accounts using an appropriate graphical user interface. It is allowing user to create an a Bank Account.

creates account by :

- Allowing a user to enter a account number
- Allowing a user to enter a first name
- Allowing a user to enter a second name
- Allowing a user to enter a balance
- Allowing a user to enter a overdraft limit

When data entry is complete, we can created account by clicking " **Create Account** " then display account by clicking " **View Account** ".We have menu with the following options :

1. **Lodgement**
2. **Withdrawal**
3. **Delete Account**
4. **New Overdraft**
5. **Info**
6. **Exit**

For each above functions was implemented an appropriate **method**.

GRAPHICAL USER INTERFACE

CreditUnion GUI :

1. Allow a user to open a new Credit Union account

2. Allow a user to display a Credit Union account

3. Allow an account holder to lodge money

4. Allow an account holder to withdrawal money

5. Allow an account holder do close/delete account in Credit Union

6. Allow an account holder to request a new overdraft limit

7. Information about Project Credit Union

8. Exit program

CREATEBANKFILE CODE

```
        /*
        Krystian Kunowski
        Student ID : B00123405
        Technological University Dublin
        Assignment 2
        Software Development 2
       */

import java.io.*;
//import java input/output stream

public class CreateBankFile
//creating a class called CreateBankFile
 {
  private Record blank;
  private RandomAccessFile file;
//initialize variable

  public CreateBankFile()
//creating a method called CreateBankFile
   {
    blank = new Record();

  try
```

```
  {

   file = new RandomAccessFile( "bank.dat", "rw" );

   for (int i=0; i<100; i++)
   blank.write( file );

  }
//creating a RandomAccessFile "bank.dat" and
//writing 100 empty records into file

 catch(IOException e)

 {

  System.err.println("File not opened properly\n" + e.toString() );

  System.exit( 1 );

 }
//creating a catch IOException to display and track any errors
}


public static void main( String [] args )
//creating a main method
{
        CreateBankFile employees = new CreateBankFile();
}
```

```
}
```

RECORD CODE

```
/*
        Krystian Kunowski
        Student ID : B00123405
        Technological University Dublin
        Assignment 2
        Software Development 2
*/

import java.io.*;
//import java input/output stream

public class Record
//creating a class called Record
{
private int accountID;
private String firstNameID;
private String lastNameID;
private double balance;
private double limitOverDraft;
//initialize variable

public void read( RandomAccessFile file ) throws IOException
//creating a method called read ( reading records from file )
{
```

```
accountID = file.readInt();

char first[] = new char[15];

for ( int i=0; i < first.length; i++ )
        first[ i ] = file.readChar();

lastNameID = new String (first);

char last[] = new char[15];

for (int i =0; i<last.length; i++)
        last[i] = file.readChar();

firstNameID = new String (last);

balance = file.readDouble();
limitOverDraft = file.readDouble();
}
public void write( RandomAccessFile file) throws IOException
//creating a method called write ( writing records into file )
{
StringBuffer buf;

file.writeInt( accountID );

if (lastNameID != null)
        buf = new StringBuffer( lastNameID);
```

```
else
buf = new StringBuffer( 15 );

buf.setLength( 15 );

file.writeChars( buf.toString() );

if ( firstNameID != null )
buf = new StringBuffer( firstNameID );
else
buf = new StringBuffer( 15);

buf.setLength( 15 );

file.writeChars( buf.toString() );

file.writeDouble( balance);
if (lastNameID != null)

file.writeDouble( limitOverDraft );



}

public void setAccountID( int a ) { accountID = a; }

public int getAccounts() { return accountID;}

public void setFirstName( String f) {lastNameID = f;}
```

```
public String getFirstName() { return lastNameID; }

public void setLastName ( String l) { firstNameID = l; }

public String getLastName() { return firstNameID; }

public void setBalancePay( double b) {balance=b;}

public double getBalance() {return balance;}

public void setlimitOverDraft( double c) {limitOverDraft = c;}

public double getlimitOverDraft() {return limitOverDraft;}

public static int size() { return 80;}
//creating a methods to write/read records from/into fail
}
```

CREDITUNION CODE

```
/*

        Krystian Kunowski

        Student ID : B00123405

        Technological University Dublin

        Assignment 2

        Software Development 2


*/


import java.io.*;

import java.text.DecimalFormat;

import java.awt.*;

import javax.swing.*;

import java.awt.event.*;

//imports


public class CreditUnion extends JFrame implements ActionListener

//creating a class called CreditUnion

{

  //create GUI componements

  private ImageIcon labelBank;

  private JLabel labelWelcome;

  private JTextField accountNumberField, firstNameField, lastNameField, balanceField,
overDraftLimitField;

  private JButton enter,next,lodgement,withdrawal,delete,changeOd,info,

//send record to file

 done; //exit program

  private RandomAccessFile output, input;     //file for input & output
```

```
private Record data;

//contructor -- initalises the Frame

public CreditUnion()
 {
  super("CreditUnion");

  try
       {
       //set up files for read & write
       input = new RandomAccessFile( "bank.dat", "rw" );
       output = new RandomAccessFile( "bank.dat", "rw" );
       }
catch ( IOException e )
       {
       System.err.println(e.toString() );
       System.exit(1);
       }
  //open file
  data = new Record();

 //access record

 //set out layout
 setSize( 450, 300 );
 setLayout( new GridLayout(11,2) );

 //add components to GUI
```

```
add(new JLabel(new ImageIcon("images/bank.png")));
add ( new JLabel(" "));


        add( new JLabel("  Account number") );
        accountNumberField = new JTextField();
        add( accountNumberField );

        add( new JLabel("  First Name") );
        firstNameField = new JTextField();
        add( firstNameField );

        add( new JLabel("  Last Name") );
        lastNameField = new JTextField();
        add( lastNameField );

        add( new JLabel("  Balance") );
        balanceField = new JTextField();
        add( balanceField );

        add( new JLabel("  Over Draft Limit") );
        overDraftLimitField = new JTextField();
        add( overDraftLimitField );

        enter = new JButton ("Create Account");
        enter.addActionListener(this);
        add (enter);
```

```
    next = new JButton ("View Account");
    next.addActionListener(this);
    add (next);

    add( new JLabel("") );
    add( new JLabel("") );

    lodgement = new JButton ("Lodgement");
    lodgement.addActionListener(this);
    add (lodgement);

withdrawal = new JButton ("Withdrawal");
withdrawal.addActionListener(this);
add (withdrawal);

delete = new JButton ("Delete Account");
delete.addActionListener(this);
add (delete);

changeOd = new JButton ("New Overdraft");
changeOd.addActionListener(this);
add (changeOd);

info = new JButton ("Info");
info.addActionListener(this);
add (info);

done = new JButton ("Exit");
```

```
done.addActionListener(this);

add (done);


            setVisible( true );

}


//create method for adding records to file

public void openAccount()

{

 int accountNumber = 0;

 Double pay;

 Double limit;

//initialize variable


 if ( ! accountNumberField.getText().equals( "" ) );

 {


 try

  {

   accountNumber = Integer.parseInt( accountNumberField.getText() );

   pay = Double.parseDouble( balanceField.getText() );

   limit = Double.parseDouble( overDraftLimitField.getText() );

   System.out.println("In second try block");


    if (accountNumber < 1  || accountNumber > 100)  //validate account number is in range

     {

 JOptionPane.showMessageDialog(this, "Account number must be between 1 & 100",
"Warning",

JOptionPane.WARNING_MESSAGE);
```

```
}

else if (accountNumber > 0 && accountNumber <= 100 ) {

            //read file to check if account ID number already exists.

            output.seek((long) (accountNumber - 1) * Record.size());

            data.read(output);

           if (data.getAccounts() == accountNumber)

//if account number exists,display dialog box to user

             {

             JOptionPane.showMessageDialog(this,"Account number already exists! Please try a different Account number", "Warning",

JOptionPane.WARNING_MESSAGE);

 accountNumberField.setText("");

// clear Account ID textfield

      }

              else   //once conditions are met, data is written to file.

        {

         data.setAccountID( accountNumber );

         data.setFirstName( firstNameField.getText() );

         data.setLastName( lastNameField.getText() );

         pay = new Double( balanceField.getText() );

         data.setBalancePay( pay.doubleValue() );

         limit = new Double (overDraftLimitField.getText());

         data.setlimitOverDraft( limit.doubleValue() );

         output.seek( (long) ( accountNumber-1 ) * Record.size() );

         data.write( output );
```

```
JOptionPane.showMessageDialog(this, "Account Created");
        }
   }
         //clear textfields
         accountNumberField.setText("");
         firstNameField.setText("");
         lastNameField.setText("");
         balanceField.setText("");
         overDraftLimitField.setText("");

 }//end try statement
        catch (NumberFormatException nfe )
        {
         System.err.println("You must enter an integer account number");
        }
        catch (IOException io)
        {
        System.err.println("error during write to file\n" + io.toString() );
      //creating a catch IOException to display and track any errors
       }


 }//end initial if statement
} //end openAccount method


public void actionPerformed (ActionEvent e)  //add actionperformed events
{
```

```
System.out.println("button"+e.getSource());

 if (e.getSource() == enter)
 {
             openAccount();
 }

if (e.getSource() == next)
 {
  readRecord();
 }
if (e.getSource() == done)
{
             closeFile2();
}
if (e.getSource() == delete)
{

             closeAccount();
}
if (e.getSource() == lodgement)
{
            makeLodgement();
}
if (e.getSource() == withdrawal)
{
            makeWithdrawal();
}
```

```
if (e.getSource()== changeOd)

{

        requestOverdraft();

}

if (e.getSource()== info)

{

        JOptionPane.showMessageDialog(this, "CreditUnion created by Krystian Kunowski
student at TU Dublin, student no. B00123405");

}


}


  public static void main(String [] args )

  //create main method

  {

   new CreditUnion();


  }


  //READ RECORD METHOD

  public void readRecord()

   {

    DecimalFormat twoDigits = new DecimalFormat( "0.00" );

    // set decimal format

     try

     {

      do {

                  data.read(input);

           }
```

```
      while (data.getAccounts() == 0);//once conditions are met, data is read from file.

   accountNumberField.setText(String.valueOf( data.getAccounts() ) );
   firstNameField.setText( data.getFirstName() );
   lastNameField.setText( data.getLastName() );
   balanceField.setText( String.valueOf(
   twoDigits.format( data.getBalance() ) ) );
   overDraftLimitField.setText( String.valueOf(
         twoDigits.format( data.getlimitOverDraft() ) ) );

    }
      catch (EOFException eof )
    {
    JOptionPane.showMessageDialog(this, "No Accounts to view");
    closeFile();    }
    catch (IOException e )
    {
    System.err.println("Error during read from file\n " + e.toString() );
    System.exit( 1 );
   //creating a catch IOException to display and track any errors
    }
  }

//method to close file ( not closing Application )
private void closeFile()
{
      try
      {
```

```
                input.close();
            }
            catch( IOException e)
            {
                System.err.println( "Error closing file \n" + e.toString() );
            }
    }

    private void closeFile2()
    //method to close file ( closing Application )
    {
            try
            {
                input.close();
                System.exit( 0 );
            }
            catch( IOException e)
            {
                System.err.println( "Error closing file \n" + e.toString() );
            }
    }


  public void closeAccount()
  //create method to delete account
  {
            try
            {
                int accountNumber10 = Integer.parseInt( accountNumberField.getText() );
```

```
try {
        output.seek((long) (accountNumber10 - 1) * Record.size());
        } catch (IOException e1) {
        e1.printStackTrace();
        }


          data.setAccountID(0);
          data.setFirstName(null);
          data.setLastName(null);
          data.setBalancePay(0);
          data.setlimitOverDraft(0);
    // setting all records to 0/null

                data.write( output );
                input.seek((long) (accountNumber10 - 1) * Record.size());
                // writing all the records into file


          JOptionPane.showMessageDialog(this, "Account has been deleted");


     accountNumberField.setText("");
     firstNameField.setText("");
     lastNameField.setText("");
     balanceField.setText("");
     overDraftLimitField.setText("");
     // clearing textfields
```

```
        }
        catch (EOFException eof )
  {
JOptionPane.showMessageDialog(this, "Please to click 'View Account' button to choose an account to delete");
  }
  catch (IOException e )
  {
  System.err.println("Error during read from file\n " + e.toString() );
 //creating a catch IOException to display and track any errors
  }
 }

 private void requestOverdraft()
 //create method to set new overdraft limit
 {
        DecimalFormat twoDigits = new DecimalFormat( "0.00" );
       // set decimal format
        int accountNumber5 = 0;
        double newOdInput;

newOdInput = Double.parseDouble(JOptionPane.showInputDialog(null," How much do you want to set for Overdraft Limit  ? "));
       //initialize variable

        if ( newOdInput > 0  );
              Double field8;
              Double field9 = newOdInput;
              //initialize variable in if statement
              {
```

```
accountNumber5 = Integer.parseInt( accountNumberField.getText() );
try {
        output.seek((long) (accountNumber5 - 1) * Record.size());
        } catch (IOException e1) {
        e1.printStackTrace();
        //creating a catch IOException to display and track any errors
        }

        overDraftLimitField.setText( String.valueOf(
         twoDigits.format( data.getBalance() ) ) );

     field8 = Double.parseDouble( overDraftLimitField.getText() );
     field8 = new Double( overDraftLimitField.getText() );
     field8 = field9;
     data.setlimitOverDraft(  field8.doubleValue() );
     // creating new record

                try {
                        data.write( output );
                        input.seek((long) (accountNumber5 - 1) * Record.size());
                        // writing new record into file
                        readRecord();
                        // displaying new record from file

} catch (IOException e) {
System.err.println( "Error while creating Overdraft Limit  \n" + e.toString() );
 //creating a catch IOException to display and track any errors
}
```

```
 }


}
 private void makeLodgement()
 //create method to make a Lodgement
 {
          DecimalFormat twoDigits = new DecimalFormat( "0.00" );
         // set decimal format
          int accountNumber3 = 0;
          double logementInput;



 logementInput = Double.parseDouble(JOptionPane.showInputDialog(null," How much do
you want to pay into bank account? "));
         //initialize variable


          if ( logementInput > 0  );
                   Double field3;
                   Double field5 = logementInput;
                   //initialize variable in if statement
                   {


         accountNumber3 = Integer.parseInt( accountNumberField.getText() );
         try {
                   output.seek((long) (accountNumber3 - 1) * Record.size());
         } catch (IOException e1) {
         e1.printStackTrace();
          //creating a catch IOException to display and track any errors
         }
```

```
        balanceField.setText( String.valueOf(
            twoDigits.format( data.getBalance() ) ) );

        field3 = Double.parseDouble( balanceField.getText() );
        field3 = new Double( balanceField.getText() );
        field3 = field3 + field5;
        data.setBalancePay(  field3.doubleValue() );
        // creating new record

    try {
          data.write( output );
          input.seek((long) (accountNumber3 - 1) * Record.size());
          // writing new record into file
          readRecord();
          // displaying new record from file

    } catch (IOException e) {
    System.err.println( "Error while making Lodgement  \n" + e.toString() );
    //creating a catch IOException to display and track any errors
    }
 }

}
 private void makeWithdrawal()
 //create method to make a Withdrawal
 {
     DecimalFormat twoDigits = new DecimalFormat( "0.00" );
          // set decimal format
```

```
int accountNumber4 = 0;

double withdrawalInput;

double limitQ;


withdrawalInput = Double.parseDouble(JOptionPane.showInputDialog(null," How much do you want to withdrawal from bank account? "));

//initialize variable


if ( withdrawalInput > 0);

{


Double field4;

Double field6 = withdrawalInput;

//initialize variable in if statement



accountNumber4 = Integer.parseInt( accountNumberField.getText() );

try {

output.seek((long) (accountNumber4 - 1) * Record.size());

} catch (IOException e1) {


e1.printStackTrace();

//creating a catch IOException to display and track any errors

}


    balanceField.setText( String.valueOf(

      twoDigits.format( data.getBalance() ) ) );
```

```
limitQ = Double.parseDouble( overDraftLimitField.getText() );

field4 = Double.parseDouble( balanceField.getText() );


if ((field4 - field6) > (0-limitQ) ){

    field4 = new Double( balanceField.getText() );

        field4 = field4 - field6;

        data.setBalancePay(  field4.doubleValue() );

        // creating new record


    try {

    data.write( output );

    input.seek((long) (accountNumber4 - 1) * Record.size());

    // writing new record into file

    readRecord();

    // displaying new record from file


} catch (IOException e) {

//creating a catch IOException to display and track any errors

}


  }else {

JOptionPane.showMessageDialog(this, "Not enough funds to make this transaction",

 "Warning",JOptionPane.WARNING_MESSAGE);

    }


}// closing if statement


}// closing method

}// closing calss CreditUnion
```

TEST

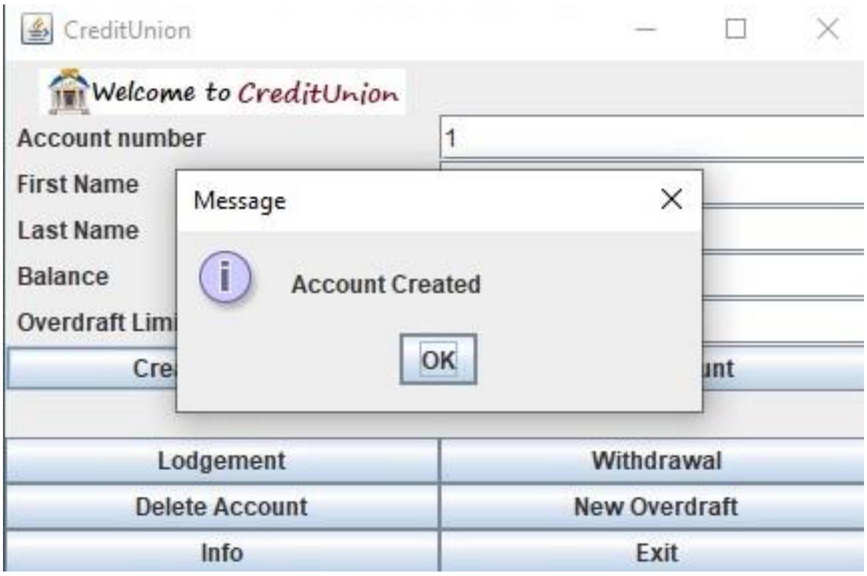
1.Program prompts user to enter an account number, first name, last name, balance and overdraft limit.

(fig1)

2. If user entered incorrect account number program will display appropriate information.

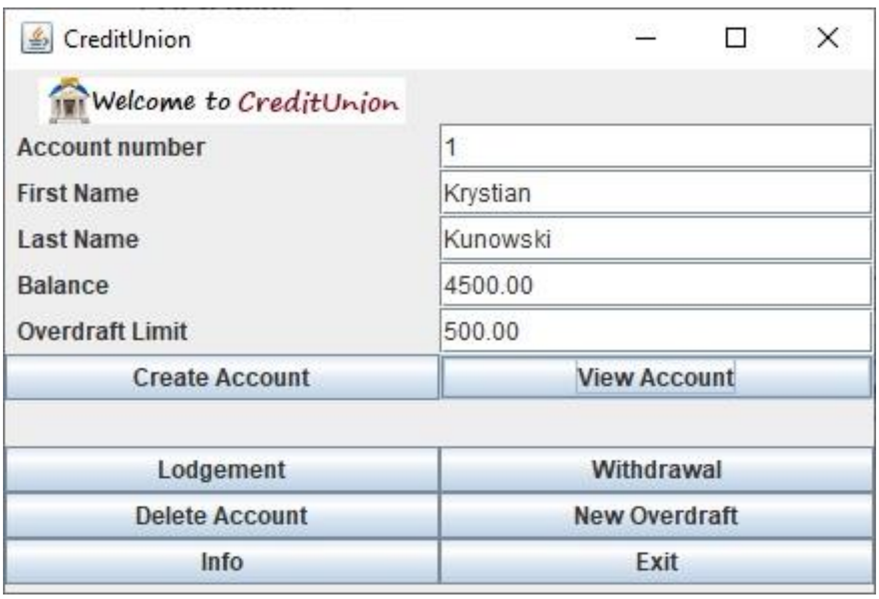
(fig1.1)

3.After user entered all information and hits button “Create Account” program will create and store records in bank.dat file.



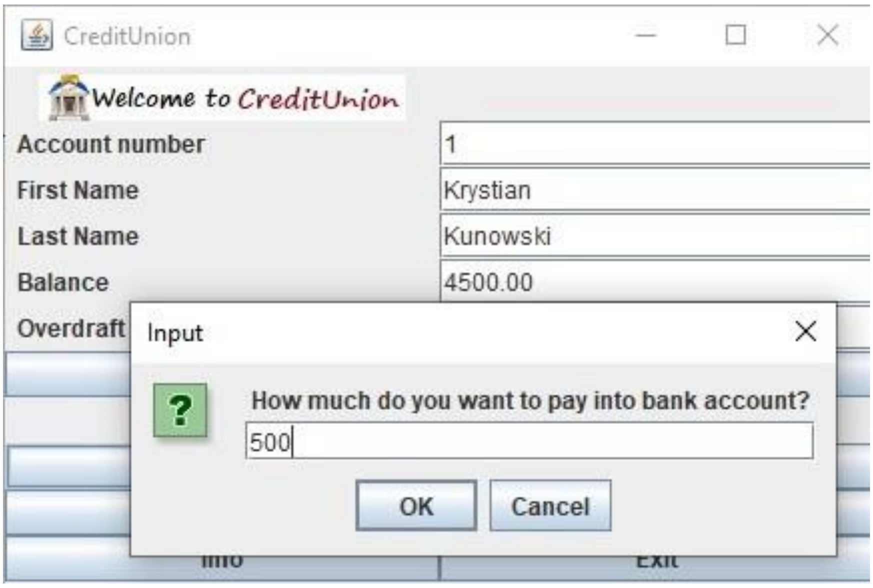
(fig2)

4.After user hits button “View Account” program will display records from bank.dat file.



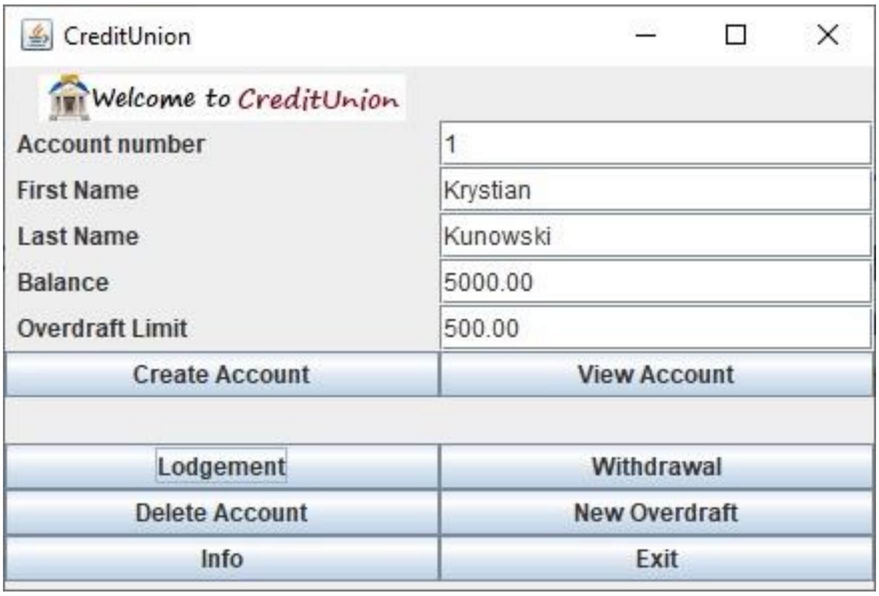
(fig3)

5. After user hits button “Lodgement” program will prompts user to enter a sum of money into your bank account.



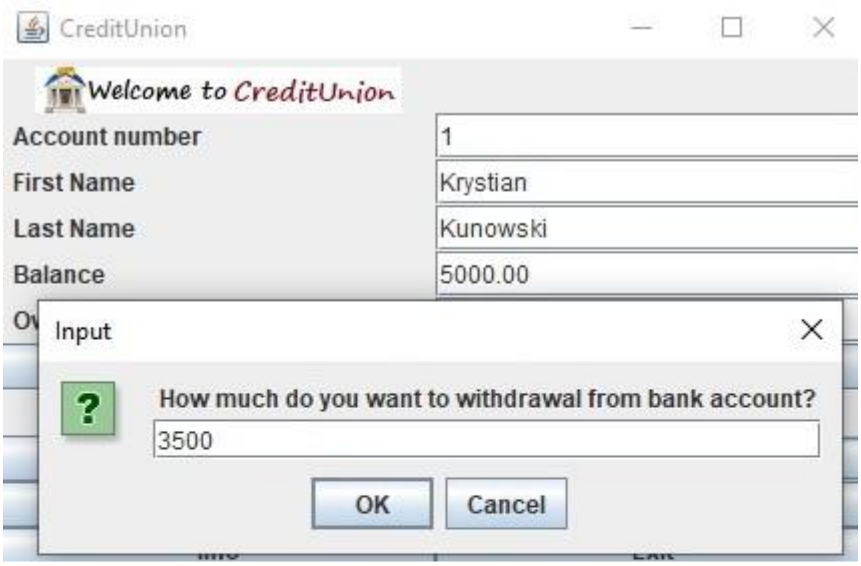
(fig4)

6. After user entered all information and hits button “OK” program will display new balance.



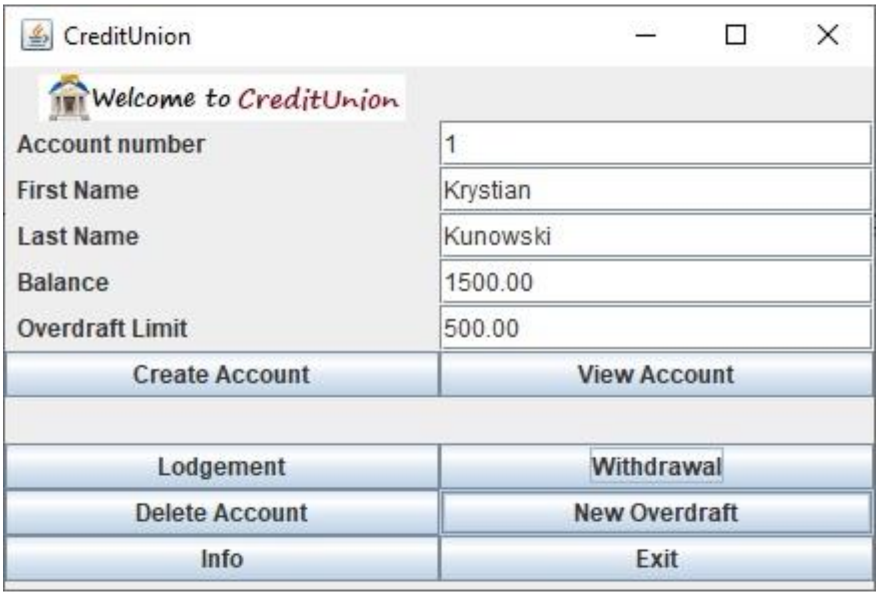
(fig4.1)

7. After user hits button "Withdrawal" program will prompts user to enter a sum of money to takes from bank account.



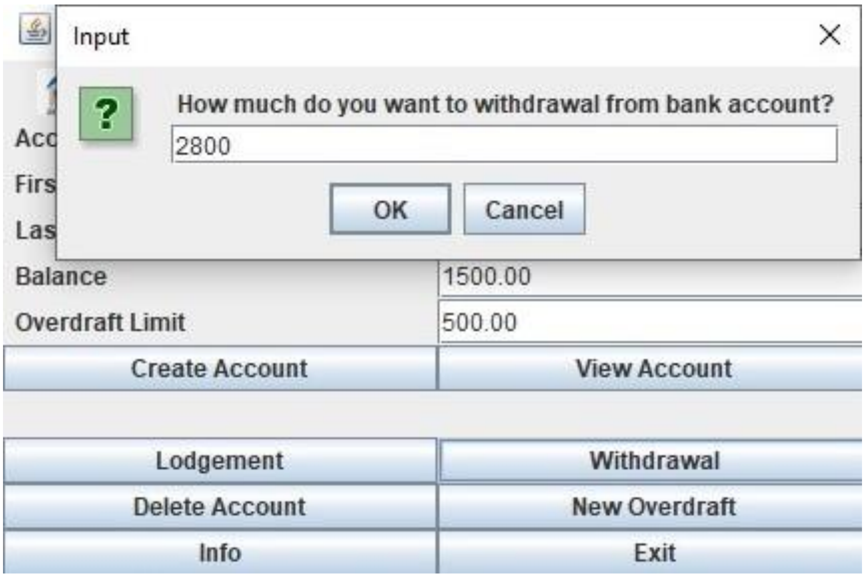
(fig5)

8.After user entered all information and hits button "OK" program will display new balance.

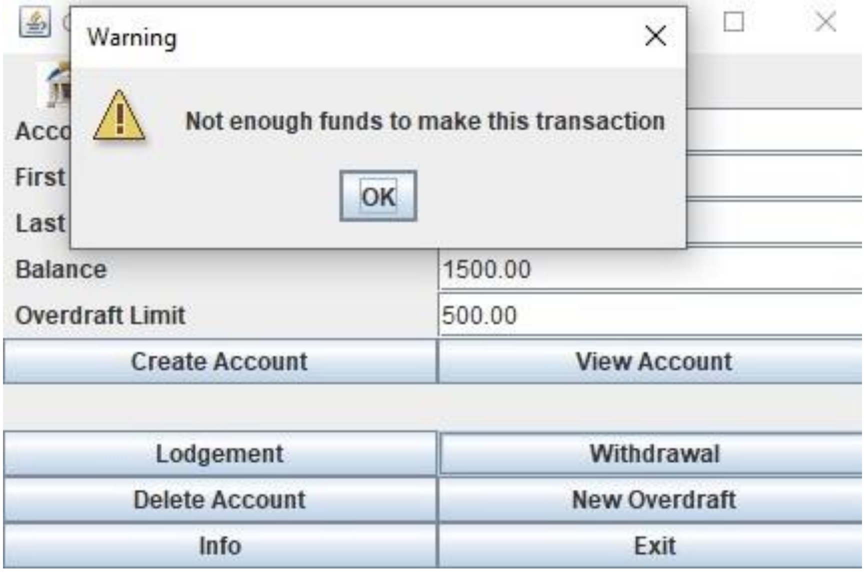


(fig5.1)

9.After user entered higher sum of money to withdrawal than overdraft limit and hits button "OK" program will show a warning massage.

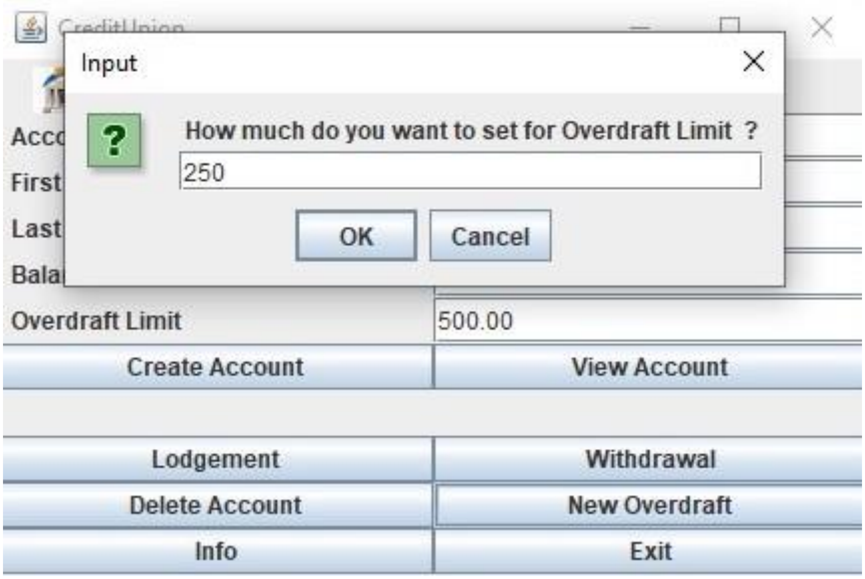


(fig5.2)



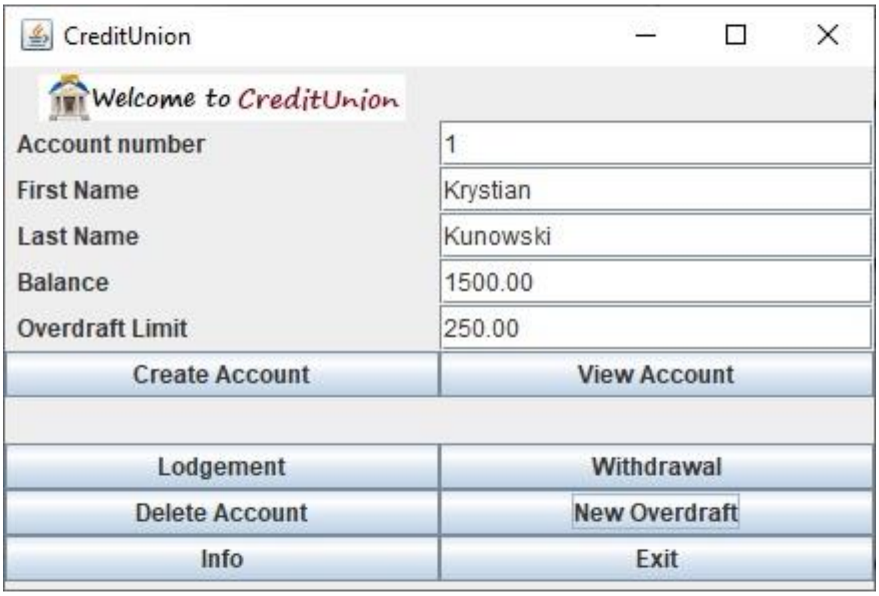
(fig5.3)

10. After user hits button “New Overdraft” program will prompts user to enter a new Overdraft limit.



(fig6)

11.After user entered all information and hits button “OK” program will display a new Overdraft limit.

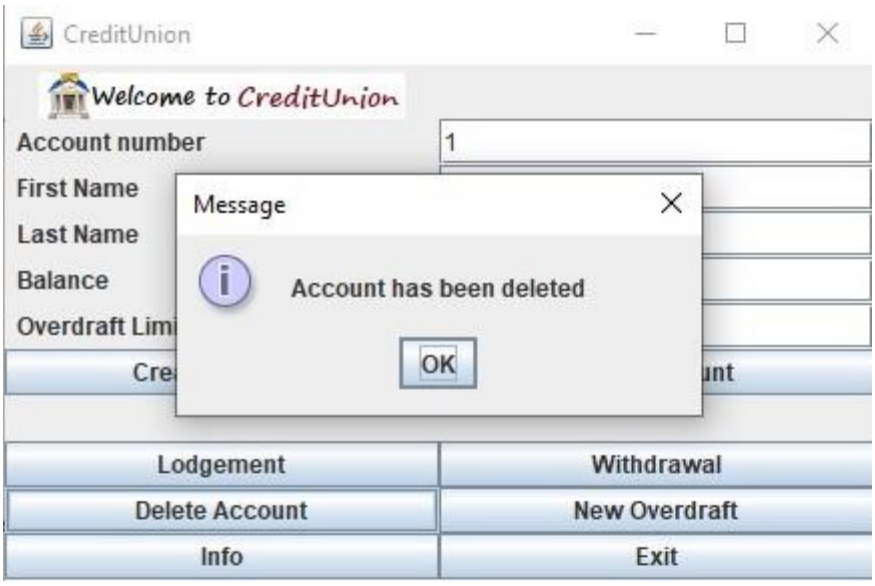


(fig6.1)

12. After user hits button “Info” program will shows information about author.

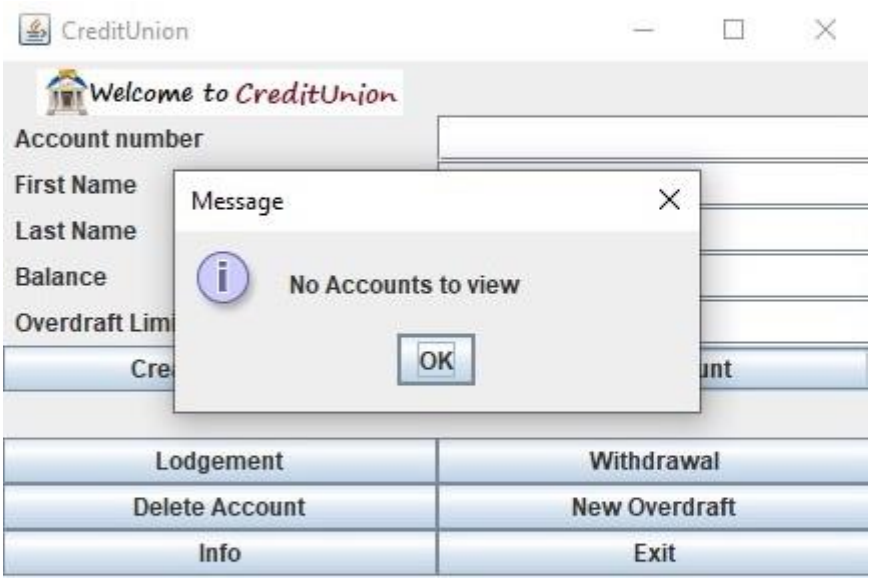
(fig7)

13. After user hits button “Delete Account” program will close/delete an account.



(fig8)

After user hits button “View Account” program shows a message :



(fig8.1)

14. After user hits button “Exit” program will close.