LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Choose a Module

Choose Coursework Percentage **Individual Coursework**

2023 Spring

Student Name: Gajendra Prasad Chaudhary

London Met ID: 22068143

College ID: NP01CP4A220494

Assignment Due Date: Wednesday, May 10, 2023

Assignment Submission Date: Wednesday, May 10, 2023

Word Count: 242

Project File Links:

YouTube Link:	Keep Unlisted YouTube URL of your Project Here
Google Drive Link:	Keep Google Drive URL of your Project Here with Anyone in Organization can View Option Enabled

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

1. INTRODUCTION

Java is a popular high-level, object-oriented programming language that was initially developed by James Gosling and released by Sun Microsystems in May 1995. It is now owned by Oracle Corporation. Java is a general-purpose language that can be used to develop a wide range of software applications. One of the key features of Java is its ability to produce software that can run on multiple platforms, making it a very versatile language. Java is known for its simplicity, security, and user-friendliness, and is widely used by developers to create applications for various operating systems, including Windows, Linux, and Mac OS. Java is designed to follow the **"write once, run anywhere"(WORA)** principle, meaning that compiled Java code can run on any platform that supports Java without needing to be recompiled. (Rao, 2008)

Our first coursework includes a parent class BankCard which have two child classes name are DebitCard and CreditCard along with GUI class (BankGUI). Parent and child class have their own attributes, constructors and methods in program and GUI class have its own attributes, constructors and methods which are link with BankCard and DebitCard and CreditCard classes.

We learned how to utilize the super keyword to invoke the superclass constructor as well as the accessor and mutator methods. Additionally, during our second semester, we learned about GUI. We learned about array lists, handling events, handling exceptions, try-

catch, java awt, java swing, and many other topics. All most all of them are used in this coursework.

2. Class Diagram

A class diagram is a representation of the static structure of a system that highlights the relationships between its classes and objects. The classes in the system are thoroughly described, including information on their characteristics, purposes, and matching programming languages. Through the class diagram, we can describe the variables and processes that each class in the system uses. For this project, we've provided a class diagram that illustrates the many classes in our software, their attributes and methods, and the relationships between them. With the aid of this diagram, which clarifies the system's structure, it is easier to build and implement the software successfully.

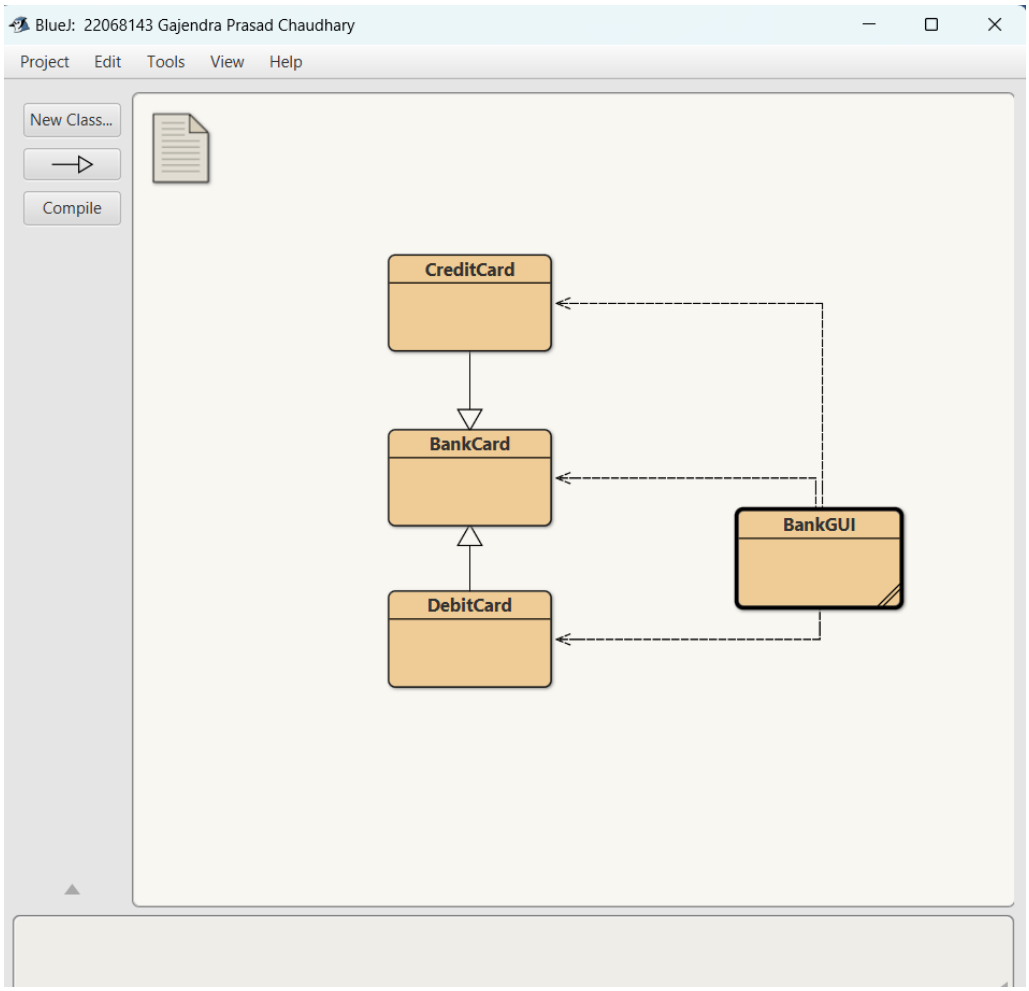


Figure 1: Class Diagram of classes in BlueJ

2.1 Bank Card (Class Diagram)

BankCard
-cardId : int -clientName : String -issuerBank : String -bankAccount : String -balanceAccount : int
+<<constructor>>BankCard(balanceAmount:int,cardId:int, bankAccount:String,issuerBank:String) +getCardId () : int +getClientName () : String +getIssuerBank () : String +getBankAccount () : String +getBalanceAmount () : int +setClientName(ClientName : String) : void +setBalanceAmount (balanceAmount : int) : void +diaplay () : void

Table 1: Class Diagram of Bank Card

2.2 Debit Card (Class Diagram)

DebitCard
-PIN : int -withdrawalAmount : int -dateOfWithdrawal : String -hasWithdrawn : boolean
+ <<constructor>>DebitCard(balanceAmount:int, cardId:int,bankAccount:String,issuerBank:String, clientName:String,PIN:int) +getPIN() : int +getWithdrawalAmount(): int +getDateOfWithdrawal(): String +getHasWithdrawn(): boolean +setWithdrawalAmount (withdrawalAmount : int) : void +withdraw (withdrawalAmount : int, dateOfWithdrawal : String, PIN : int): void +display() : void

Table 2:Class Diagram of Debit Card

2.3 Credit Card (Class Diagram)

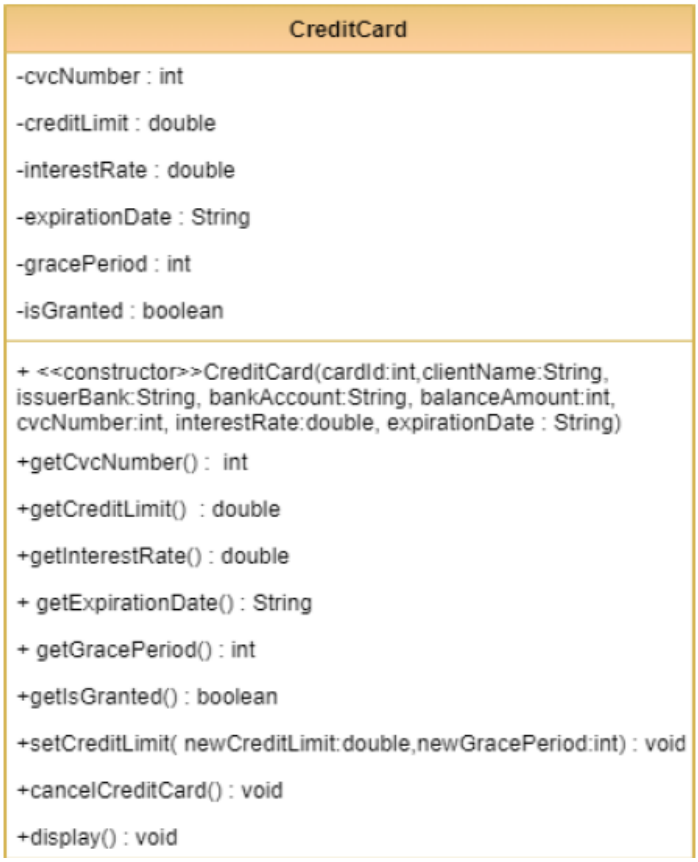


Table 3:Class Diagram of Credit Card

2.4 Bank GUI (Class Diagram)

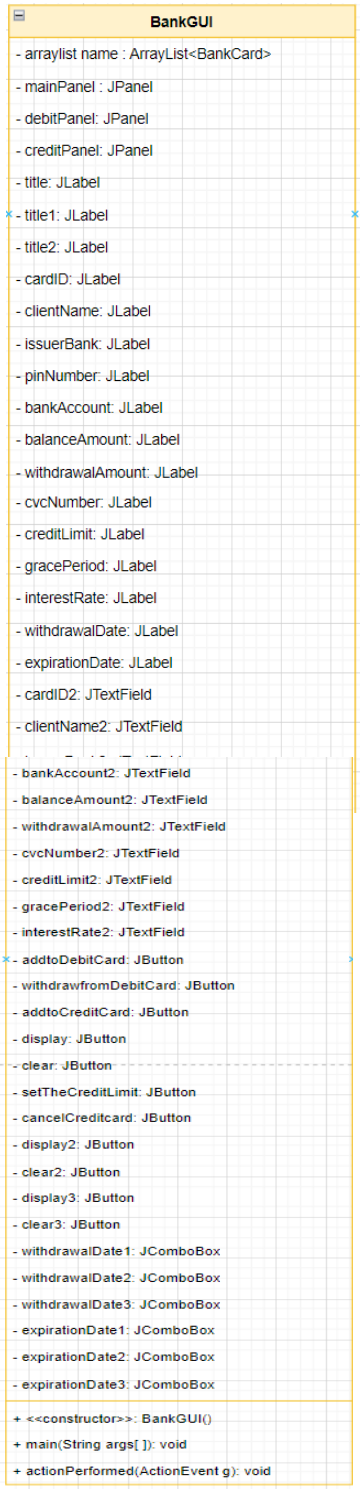
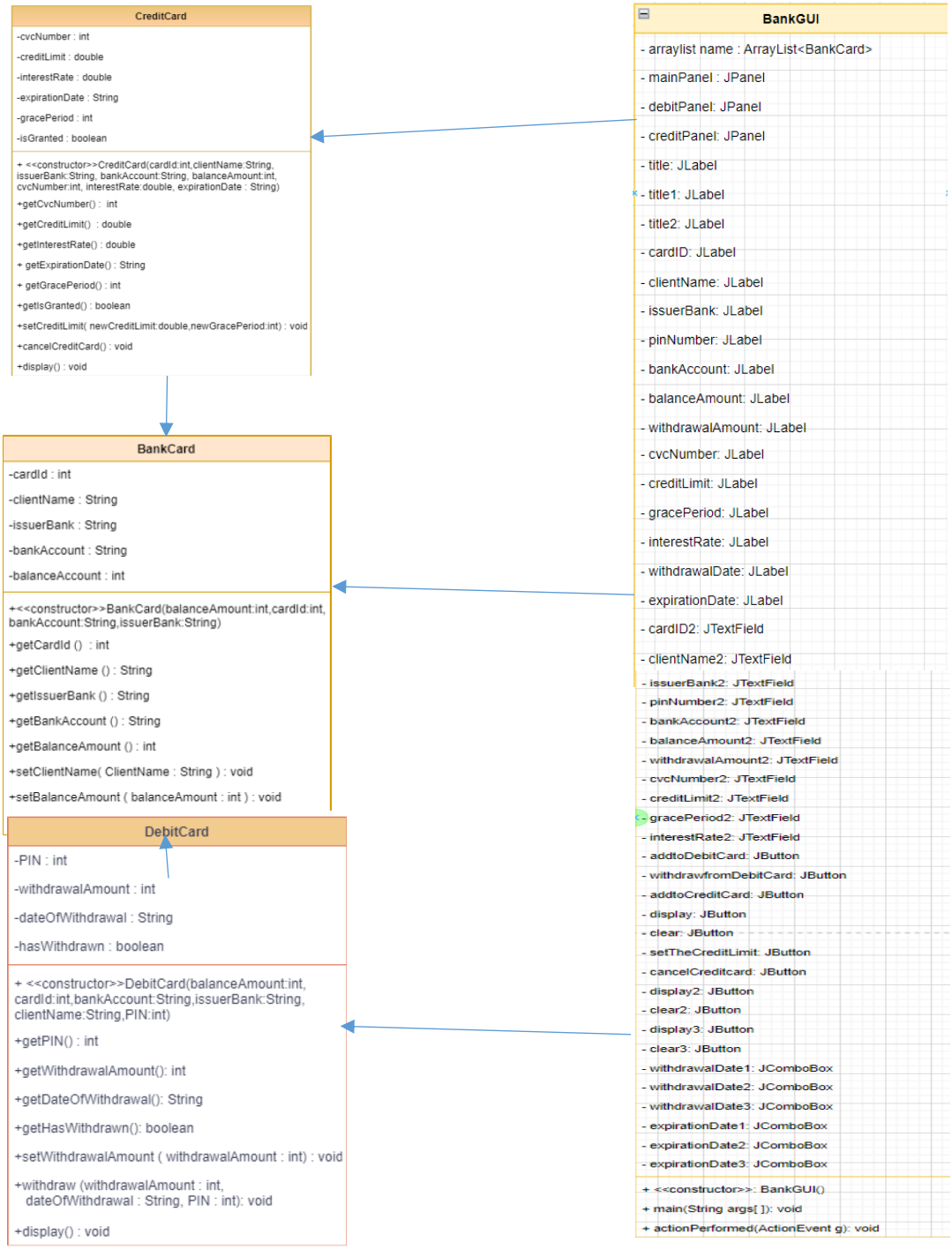


Table 4:Class Diagram of BankGUI

Inheritance Diagram

This diagram shows the interrelationship between sub-classes with the parent class and with the GUI.

Patrial view of Class Diagram.



3. Pseudocode

Pseudocode is an informal, high-level explanation of a program's logic that does not require precise syntax constraints or concerns for underlying technology. It is employed to develop a generalized or preliminary draft of a program's flow, omitting implementation-specific information. System designers can use pseudocode to make sure that programmers are aware of the specifications for a software project and are aligning their code correctly. This enables programmers to become familiar with the specifications of the application before beginning the real coding process. (Times, 2009)

3.1 BankCard

CRREATE a parent class BankCard

DO

DECLARE instance variable carId as int using private

DECLARE instance variable clientName as string using private

DECLARE instance variable issuerBank as string using private

DECLARE instance variable bankAccount as string using private

DECLARE instance variable balanceAmount as int using private

END DO

CREATE a constructor “BankCard” with parameters

- balanceAmount (int)
- cardId (int)
- bankAccount (String)
- issuerBank (String)

1 CREATE an accessor method getCardId () with return type int

DO

RETURN cardId

END DO

6

2 CREATE an accessor method getClientName () with return type string

DO

RETURN clinetName

END DO

3 CREATE an accessor method getIssuerBank () with return type string

DO

RETURN issuerBank

ENDO DO

4 CREATE an accessor method getBankAccount () with return type string

DO

RETURN bankAccount

END DO

5 CREATE an accessor method getBalanceAmount () with return type int

DO

RETURN balanceAmount

END DO

6 CREATE an mutator method setClientName () with no return type and with a input

parameter ClientName as a String data type

DO

Assign the parameter value ClientName with the private attribute

clientName of the class

END DO

7 CREATE an mutator method setBalanceAmount () with no return type and with a input

parameter balanceAmount as a int data type

7

DO

Assign the parameter value balanceAmount with the private attribute balanceAmount of the class

END DO

3.2 DebitCard

CRREATE a child class DebitCard

DO

DECLARE instance variable PIN as int using private

DECLARE instance variable withdrawalAmount as int using private

DECLARE instance variable dateOfWithdrawal as String using private

DECLARE instance variable hasWithdrawn as boolean using private

END DO

CREATE a constructor “DebitCard” with parameters

- balanceAmount (int)
- cardId (int)
- bankAccount (String)
- issuerBank (String)
- clientName (String)
- PIN (int)interestRate (double)
- expirationDate (String)

1 CREATE an accessor method getPIN() with return type int

DO

RETURN PIN

END DO

8

2 CREATE an accessor method getWithradwalAmount() with return type int

DO

RETURN withdrawalAmount

END DO

3 CREATE an accessor method getDateOfWithdrawal() with return type string

DO

RETURN dateOfWithdrawal

ENDO DO

4 CREATE an accessor method getHasWithdrawn() with return type boolean

DO

RETURN hasWithdrawn

END DO

5 CREATE an accessor method setWithdrawalAmount(). sets the withdrawal amount to the input parameter.

DO

sets the withdrawal amount to the input parameter.

END DO

6 CREATE a method "withdraw(int, String, int)" which takes withdrawal amount, date of withdrawal and PIN as input.

DO

If (PIN== PIN card)

If True

If (balance amount >= withdrawal amount)

9

if true decrease the balance amount by the withdrawal amount, set the withdrawal amount to the input value, set date of withdrawal to the input value, set hasWithdrawn to true

Print "Transaction successful.”

Else

Print "Insufficient balance."

If false

Print "Invalid PIN number."

END DO

7 CREATE a method “display()” that calls the parent class's display method

DO

prints "PIN: " + PIN

If hasWithdrawn is true

print "The Withdrawal Amount is " + withdrawalAmount and "The Date of Withdrawal is " + dateOfWithdrawal

Else

print "No withdrawal has been made yet."

END DO

3.3 CreditCard

CRREATE a child class CreditCard

DO

DECLARE instance variable cvcNumber as int using private

DECLARE instance variable creditLimit as double using private

DECLARE instance variable interestRate as double using private

DECLARE instance variable expirationDate as string using private

10

DECLARE instance variable gracePeriod as int using private

DECLARE instance variable isGranted as boolean using private

END DO

CREATE a constructor “CreditCard” with parameters

- cardId (int)
- clientName (String)
- issuerBank (String)
- bankAccount (String)
- balanceAmount (int)
- cvcNumber (int)
- interestRate (double)
- expirationDate (String)

1 CREATE an accessor method getCvcNumber () with return type int

DO

RETURN cvcNumber

END DO

2 CREATE an accessor method getCreditLimit() with return type string

DO

RETURN creditLimit

END DO

3 CREATE an accessor method getInterestRate() with return type string

DO

RETURN interestRate

ENDO DO

11

4 CREATE an accessor method getExpirationDate() with return type string

DO

RETURN expirationDate

END DO

5 CREATE an accessor method getGracePeriod() with return type int

DO

RETURN gracePeriod

END DO

6 CREATE an accessor method getIsGranted() with return type int

DO

RETURN isGranted

END DO

7 CREATE an mutator method setCreditLimit(double, int) which takes the new credit limit and grace period as input.

DO

If (balance amount <= 2.5 times the new credit limit)

set the credit limit to the input value, set grace period to the input value, and set isGranted to true, print "Credit granted. Credit limit: " + creditLimit".

Else

print "Credit cannot be issued. Credit limit exceeds 2.5 times the balance amount."

END DO

8 CREATE an mutator method cancelCreditCard() which sets cvcNumber, creditLimit, gracePeriod.

12

DO

isGranted to 0, 0.0, 0.

And

false respectively and print "Credit card has been cancelled."

END DO

9 CREATE a "display()" method that calls the parent class's display method.

DO

prints "CVC number: " + cvcNumber and "Expiration date: " + expirationDate

if credit card is granted

prints "Credit limit: " + creditLimit and "Grace Period: " + gracePeriod

ELSE

print "Grace period: " + gracePeriod

END DO

3.4 BankGUI

DECLARE the BankGUI class and implement the ActionListener interface

DO

CREATE a new JFrame object called "frame"

INITIALIZE three JPanel objects: mainPanel, debitPanel, and creditPanel.

CREATE three JLabel objects named "title", "title1", and "title2".

CREATE private JLabel variables with the names cardID, clientName, issuerBank, pinNumber, bankAccount, balanceAmount, withdrawalAmount, cvcNumber, creditLimit, gracePeriod, interestRate, withdrawalDate, and expirationDate.

CREATE private text fields named cardID2, clientName2, issuerBank2, pinNumber2, bankAccount2, balanceAmount2, withdrawalAmount2, cvcNumber2, creditLimit2, gracePeriod2, and interestRate2.

CREATE JButton object and assign them to the following variable names:

- addtoDebitCard
- withdrawfromDebitCard
- addtoCreditCard
- display
- clear
- setTheCreditLimit
- cancelCreditcard
- display2
- clear2
- display3
- clear3.

CREATE a private JComboBox variable named withdrawalDate1 that stores a drop-down list of dates.

CREATE JComboBox variables named withdrawalDate2 and withdrawalDate3 to store additional lists of dates.

CREATE JComboBox variables named expirationDate1, expirationDate2, and expirationDate3 to store drop-down lists of expiration dates

DECLARE new ArrayList called "al" to store BankCard objects

CREATE method BankGUI()

DO

CREATE a new JFrame object called "frame"

CREATE a new JPanel object called "mainPanel"

CREATE a new JPanel object called "debitPanel"

CREATE a new JPanel object called "creditPanel"

CREATE a new JFrame object with the title "BankGUI"

frame = new JFrame("BankGUI")

CREATE three new JPanel objects: mainPanel, debitPanel, and creditPanel

mainPanel = new JPanel()

debitPanel = new JPanel()

creditPanel = new JPanel()

CREATE a new label called "title" with text "Bank Card"

SET the position and size of the label using setBounds method

ADD the label to the mainPanel using the add method

SET the font of the label to Arial, bold, and size 20 using setFont method

CREATE a JButton named "display3" with the label "Display"

CREATE a JButton named "clear3" with the label "Clear"

SET the position and size of "display3" button to (920,250) and (88,34) respectively using setBounds() method.

SET the position and size of "clear3" button to (1250,250) and (88,34) respectively using setBounds() method.

ADD "display3" button to the "mainPanel" using add() method

ADD "clear3" button to the "mainPanel" using add() method

INITIALIZE JLabel objects for the following labels with their respective text:

- cardID
- clientName
- issuerBank
- bankAccount
- balanceAmount

SET the position and size of each JLabel using the setBounds method:

- cardID.setBounds(x, y, width, height);
- clientName.setBounds(x, y, width, height);
- issuerBank.setBounds(x, y, width, height);
- bankAccount.setBounds(x, y, width, height);
- balanceAmount.setBounds(x, y, width, height);

ADD the JLabel objects to the mainPanel using the add method:

- mainPanel.add(cardID);
- mainPanel.add(clientName);
- mainPanel.add(issuerBank);

- mainPanel.add(bankAccount);
- mainPanel.add(balanceAmount);

DECLARE cardID2 as new JTextField()

DECLARE clientName2as new JTextField()

DECLARE issuerBank2as new JTextField()

DECLARE bankAccount2as new JTextField()

DECLARE balanceAmount2as new JTextField()

SET the position and size of each JTextField using the setBounds method:

- cardID2.setBounds(x, y, width, height);
- clientName2.setBounds(x, y, width, height);
- issuerBank2.setBounds(x, y, width, height);
- bankAccount2.setBounds(x, y, width, height);
- balanceAmount2.setBounds(x, y, width, height);

ADD the JTextField objects to the mainPanel using the add method:

- mainPanel.add(cardID2);
- mainPanel.add(clientName2);
- mainPanel.add(issuerBank2);
- mainPanel.add(bankAccount2);
- mainPanel.add(balanceAmount2);

CREATE a JPanel named mainPanel

SET the bounds of mainPanel to (0,0,1920,380)

SET the layout of mainPanel to null

SET the visibility of mainPanel to true

SET the border of mainPanel to a white line border with a width of 6 pixels

CREATE a new JLabel object and setting its properties

title1 = new JLabel ("Debit Card")

title1.setBounds(284, 32, 120, 36)

title1.setFont(new Font("Arial", Font.BOLD, 20))

CREATE a new JLabel object and a new JTextField object and setting their properties

pinNumber = new JLabel("PIN Number:")

pinNumber2 = new JTextField()

pinNumber.setBounds(170, 80, 77, 30)

pinNumber2.setBounds(170, 105, 443, 25)

ADD the JLabel and JTextField objects to the debitPanel

debitPanel.add(title1)

debitPanel.add(pinNumber)

debitPanel.add(pinNumber2)

CREATE a label called "withdrawalAmount" with the text "Withdrawal Amount"

CREATE a text field called "withdrawalAmount2" with an empty string

SET the position of "withdrawalAmount" to (170, 140) on the debitPanel

SET the size of "withdrawalAmount" to (121, 20)

SET the position of "withdrawalAmount2" to (170, 160) on the debitPanel

SET the size of "withdrawalAmount2" to (443, 25)

ADD "withdrawalAmount" to the debitPanel

ADD "withdrawalAmount2" to the debitPanel

CREATE a new JLabel called withdrawalDate with the text "Withdrawal Date"

SET the position of withdrawalDate using setBounds method to (170,190,103,41)

ADD withdrawalDate to the debitPanel using the add method

CREATE three JComboBoxes, withdrawalDate1, withdrawalDate2, and withdrawalDate3, using the day, month, and year arrays, respectively.

SET the bounds of the JComboBoxes to (270,198,40,20), (315,198,55,20), and (376,198,65,20) respectively.

ADD the JComboBoxes withdrawalDate1,withdrawalDate2, withdrawalDate3 to the debitPanel.

CREATE a JButton called addtoDebitCard with text "Add a Debit Card"

CREATE a JButton called withdrawfromDebitCard with text "Withdraw from Debit Card"

CREATE a JButton called display with text "Display"

CREATE a JButton called clear with text "Clear"

SET the bounds of addtoDebitCard to (170,244,150,28)

SET the bounds of withdrawfromDebitCard to (430,241,190,28)

SET the bounds of display to (170,305,103,34)

SET the bounds of clear to (430,305,88,34)

ADD addtoDebitCard, withdrawfromDebitCard, display and clear buttons to the debitPanel

SET the bounds of debitPanel to (0, 380, 720, 700)

SET the layout of debitPanel to null

SET the visibility of debitPanel to true

SET the border of debitPanel to a white line border with a thickness of 5

CREATE a JLabel for the title "Credit Card"

title2 = new JLabel ("Credit Card");

SET the bounds of the title label

title2.setBounds(328, 26, 124, 36);

ADD the title label to the creditPanel

creditPanel.add(title2);

SET the font of the title label to Arial, bold and size 20

title2.setFont(new Font("Arial", Font.BOLD, 20));

DEFINE four JLabel variables with the following names:

- cvcNumber
- creditLimit
- gracePeriod
- interestRate

SET the text for each JLabel variable as follows:

- cvcNumber: "CVC Number"
- creditLimit: "Credit Limit"
- gracePeriod: "Grace Period"
- interestRate: "Interest Rate"

SET the bounds for each JLabel variable using the setBounds() method:

- cvcNumber: (169,70,87,20)
- creditLimit: (169,128,82,20)
- gracePeriod: (169,187,82,20)
- interestRate: (169,245,92,20)

ADD each JLabel variable to the creditPanel using the add() method.

DECLARE three arrays of string for day, month and year

String day2[] = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20" "21", "22", "23", "24" ,"25", "26", "27", "28" ,"29", "30","31"};

String month2[] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEPT", "OCT", "NOV", "DEC"};

String year2[] = {"2023", "2024", "2025", "2026", "2027", "2028", "2029", "2030"};

CREATE three combo boxes and assign each with one of the arrays

expirationDate1 = new JComboBox(day2);

expirationDate2 = new JComboBox(month2);

expirationDate3 = new JComboBox(year2);

SET the bounds (position and size) of the three combo boxes

expirationDate1.setBounds(275,297,49,23);

expirationDate2.setBounds(332,297,62,23);

expirationDate3.setBounds(402,297,65,23);

CREATE addtoCreditCard button

CREATE setTheCreditLimit button

CREATE cancelCreditcard button

CREATE display2 button

CREATE clear2 button

SET bounds for addtoCreditCard button

SET bounds for setTheCreditLimit button

SET bounds for cancelCreditcard button

SET bounds for display2 button

SET bounds for clear2 button

ADD add setTheCreditLimit button to creditPanel

ADD cancelCreditcard button to creditPanel

ADD display2 button to creditPanel

ADD clear2 button to creditPanel

INITIALIZE creditPanel as a new JPanel

SET the bounds of creditPanel to (720, 380, 1200, 700)

SET the layout of creditPanel to null

SET creditPanel to be visible

SET the border of creditPanel to be a white line border with a thickness of 5

Add ActionListener to display

Add ActionListener to display

Add ActionListener to display

Add ActionListener to clear

Add ActionListener to clear

Add ActionListener to clear

ADD ActionListener to withdrawfromDebitCard

ADD ActionListener to addtoDebitCard

ADD ActionListener to setTheCreditLimit

ADD ActionListener to cancelCreditcard

ADD ActionListener to addtoCreditCard

ADD ActionListener to cancelCreditcard

ADD the "mainPanel" to the "frame"

ADD the "debitPanel" to the "frame"

ADD the "creditPanel" to the "frame"

SET the layout of the "frame" to null

SET the "frame" visibility to true

SET the size of the "frame" to 1920 pixels wide and 900 pixelstall

SET the default close operation of the "frame" to exit on close

END DO

CREATE method actionPerformed(ActionEvent g) as no return type

DO

IF the array list al is empty:

create a new DebitCard object with the input values

add the new DebitCard object to the al array list

display a success message

ELSE

set a flag variable a to false

for each object obj in the al array list:

if obj is an instance of DebitCard and obj's cardId is equal to the input cardid:

create a new DebitCard object with the input values

add the new DebitCard object to the al array list

display an error message indicating that the card id has already been added

SET the flag variable a to true

break out of the for loop

if the flag variable a is still false:

create a new DebitCard object with the input values

add the new DebitCard object to the al array list

display a success message

except NumberFormatException:

display an error message indicating that the card id has an error

END IF

END DO

DO

IF (withdraw from debit card button is clicked)

try (retrieve input values for card ID, PIN number, withdrawal amount, and date of withdrawal

set a boolean variable to false

if (the list of bank cards is empty)

display a message saying there are no debit cards available

ELSE

for (each bank card in the list)

IF (the bank card is a debit card)

retrieve the card ID and PIN number of the debit card object

IF (the input card ID and PIN number match the debit card object)

withdraw the input amount from the debit card object using the withdrawal date and PIN number

display a message saying the withdrawal was successful

set the boolean variable to true

END IF

END IF

ELSE

display a message saying the input card ID and PIN number are invalid

IF (the boolean variable is false)

display a message saying the debit card could not be found

catch (a number format exception)

display a message saying the input withdrawal amount is not valid

END IF

END DO

DO

IF the event source is addtoCreditCard:

try:

get the balance amount from the balanceAmount2 field

get the card id from the cardID2 field

get the bank account number from the bankAccount2 field

get the issuer bank name from the issuerBank2 field

get the client name from the clientName2 field

get the CVC number from the cvcNumber2 field

get the interest rate from the interestRate2 field

get the expiration date from the expirationDate1, expirationDate2, and expirationDate3 fields

concatenate the expiration date strings into a single string

IF the list of bank cards is empty:

create a new CreditCard object with the entered details

add the credit card to the list of bank cards

display a success message

ELSE:

set a flag variable to false

for each bank card object in the list of bank cards:

if the object is an instance of CreditCard:

if the object's card id matches the entered card id:

display an error message and set the flag variable to true

break out of the loop

IF the flag variable is still false:

create a new CreditCard object with the entered details

add the credit card to the list of bank cards

display a success message

catch a NumberFormatException:

display an error message

END IF

END DO

DO

IF (event source is setTheCreditLimit button)

try

// **Get** the card ID, credit limit and grace period values from the input fields

cardId = parse cardID2.getText() to int

creditLimit = parse creditLimit2.getText() to double

gracePeriod = parse gracePeriod2.getText() to int

setlimit = false

// Check if the bank card array list is empty or not

IF (al is empty)

display error message "ERROR: Invalid card found."

ELSE

// Loop through the bank card array list and check if the card is a credit card

for each obj in al

if obj is an instance of CreditCard and obj's cardId equals to cardId

// **If** the card is a credit card, set the credit limit and grace period

set the credit limit and grace period of the creditCard object

display success message "The credit limit was successfully set up."

set setlimit to true

break from the loop

// **If** setlimit is still false, it means the card ID doesn't belong to a credit card or isn't real

if setlimit is false

display error message "ERROR: The card ID doesn't belong to a credit card or isn't real."

catch (NumberFormatException e)

// Catch the NumberFormatException and display an error message

display error message "ERROR: Invalid input. Please enter valid numbers."

END IF

END DO

DO

IF cancelCreditcard button is clicked:

try:

get the cardId from the cardID2 text field

set cancelcard to false

if the Objects list is empty:

display an error message that no bank card is available

else:

for each object in the Objects list:

if the object is an instance of CreditCard and its cardId matches the entered cardId:

cast the object to CreditCard type

call the cancelCreditCard method on the CreditCard object

display a success message

set cancelcard to true

break out of the loop

if cancelcard is still false:

display an **error** message that the entered cardId does not match any credit card

catch any NumberFormatException:

display an error message that the entered value is invalid

END IF

END DO

DO

IF user clicks on display button:

try:

get the card ID entered by the user from cardID2 text field

initialize a boolean variable D as false

for each card in the ArrayList al:

if the card is an instance of DebitCard and its card ID matches the one entered by the user:

cast the card as DebitCard

call the display method of the debit card

set D to true

break out of the loop

if D is still false:

display an error message

catch a NumberFormatException:

display an error message

END IF

END DO

DO

If the event source is "display2", then:

Try to get the card ID from the "cardID2" text field.

Initialize a boolean variable "cardFound" to false.

Iterate through the list of bank cards "al":

If the current bank card is an instance of CreditCard and has a matching card ID, then:

Cast the bank card to a CreditCard and store it in a variable "creditcard".

Call the "display" method of the credit card to display its details.

Set "cardFound" to true.

Break out of the loop.

If "cardFound" is still false, then display an error message saying that the card with the requested ID was not found.

If an exception of type NumberFormatException is caught, then display an error message saying that the input card ID was invalid.

END DO

```
    DO
            if (g.getSource() == display3)
try

    // get the card ID from the text field
    int cardId = Integer.parseInt(cardID2.getText());
    boolean D = false;

    // search for the credit card in the list of bank cards
    for (BankCard each: al)
       if (each instanceof CreditCard && each.getCardId() == cardId)
          CreditCard creditcard = (CreditCard) each;
          creditcard.display(); // display the credit card details
          D = true;
          break;

    // display an error message if the credit card was not found
    if (!D)
       JOptionPane.showMessageDialog(frame, "ERROR: Card with the requested ID not found.");
catch (NumberFormatException e)
```

```
        // display an error message if the card ID entered was not a number
        JOptionPane.showMessageDialog(frame, "ERROR: Invalid input. Please enter a valid card ID.");
END IF
END DO
        DO
                if (g.getSource() == clear3)
    // Clear the text fields
    cardID2.setText("");

    clientName2.setText("");
    issuerBank2.setText("");
    bankAccount2.setText("");
    balanceAmount2.setText("");

    // Display a message indicating the form has been cleared
    JOptionPane.showMessageDialog(frame, "Your form has been cleared");
  END IF
END DO

        DO
                IF (g.getSource() == clear)
    // Clear the text fields
    pinNumber2.setText("");
    withdrawalAmount2.setText("");

    // Display a message indicating the debit card form has been cleared
    JOptionPane.showMessageDialog(frame, "Your debit card form has been cleared");
```

```
    END IF
END DO
        DO
            IF (g.getSource() == clear2)
    // Clear the text fields
    cvcNumber2.setText("");
    creditLimit2.setText("");
    interestRate2.setText("");
    gracePeriod2.setText("");
    // Display a message indicating the credit card form has been cleared
    JOptionPane.showMessageDialog(frame, "Your credit card form has been cleared");
  END DO
END IF
CREATE method main with no return type
  DO
      CREATE object of type BankGUI
  END DO
```

4. Method Description

In Java, a method is a set of instructions that can be executed by calling the method name. Methods can be public, private, or protected and can take parameters and return values, but both are optional. In this project, there are three classes, each with their own set of methods. The child classes utilize the methods of the parent class, and the GUI class has its own unique methods as well.

To be more specific, the BankCard class has methods for setting and getting card ID, card holder name, and balance. The DebitCard and CreditCard classes have methods for setting and getting credit limit, withdrawal limit, interest rate, and reward points. Additionally, the CreditCard class has a display() method that displays the credit card details.

The BankGUI class has methods for initializing the GUI components, displaying messages using JOptionPane, and handling button events using actionPerformed() method. It also has methods for adding bank cards to the ArrayList, displaying bank card details, and clearing text fields. Overall, the methods used in this project serve specific purposes for each class and contribute to the functionality of the program as a whole.

4.1 BankCard

The "BankCard" Java class is the parent class which represents a bank card. The classcontains a number of instance variables, such as "cardId", "clientName", "issuerBank","bankAccount", and "balanceAmount", that store data about the card, including its unique identifier, the cardholder's name, the bank that issued it and the card's current balance.

The instance variables such as balanceAmount, cardId, bankAccount, issuerBank are initialized with the values handed in as arguments by the constructor method BankCard() of the class. The "clientName" variable is also set to an empty string by the constructor.

The class contains a number of accessor methods that enable values of instancevariables to be accessible from outside the class, including getCardId(), getClientName(),getIssuerBank(), getBankAccount() and getBalanceAmount().

The class has also mutator methods that enable data to be changed outside of the class,like setClientName(String ClientName) and setBalanceAmount(int balanceAmount).

The class has contains a "display()" method that, if clientName is not empty, displays the card information. Otherwise, it displays " oops! Client name has not been assigned."

4.2 DebitCard

The Debit Card class is child class which extends the Bank Card class. The Debit Cardclass has instance variables such as "PIN," "withdrawalAmount," "dateOfWithdrawal,"and "hasWithdrawn," that store data about the debit card, including the PIN used for withdrawals amount, the amount of cash withdrawn, the date of the withdrawal, and whether or not a withdrawal has been made.

The class has a constructor method DebitCard() that uses the "super" keyword to invoke the constructor of the superclass "BankCard" and initializes the instance variables with the inputs handed in. Additionally, it sets the "hasWithdrawn" variable to "false" by default and sets the "clientName" variable using the setter method from the superclass.

The class contains a number of accessor methods that enable values of instance variables to be retrieved from outside the class, such as "getPIN(),""getWithdrawalAmount()," "getDateOfWithdrawal()," and "getHasWithdrawn()."

Additionally, it contains mutator methods “setWithdrawalAmount(int withdrawalAmount)",that enable values of instance variables to be changed from outside the class.

The class has a method called "withdraw()" which enables the use of the debit card to withdraw cash from an ATM. This method verifies if the entered PIN code matches the card's PIN and if the card's balance is adequate to cover the withdrawal amount. If both conditions are met, the withdrawal is executed, the withdrawal amount, date of withdrawal and hasWithdrawn status are updated. A message indicating that the transaction is successful will be displayed. If either the PIN or the balance is incorrect, an error message will be displayed.

The class has a method "display()" which displays the information of the card includingthe withdrawal information if it has been made, otherwise it will state "No withdrawal hasbeen made yet."

4.3 CreditCard

The "CreditCard" class is a subclass of the "BankCard" class and it has several private variables such as 'cvcNumber', 'creditLimit', 'interestRate', 'expirationDate','gracePeriod',and 'isGranted'.

The CreditCard class has a constructor method that sets the initial values for the object'sattributes such as cardId, clientName, issuerBank, bankAccount, balanceAmount,cvcNumber, interestRate, and expirationDate upon creation of the object.

The class includes methods for accessing its private attributes, like getCvcNumber(), getCreditLimit(), and others, which are used to retrieve the values of those attributes.

The class has a method setCreditLimit() that allows setting a new credit limit, it checks if the limit exceeds 2.5 times the balance amount, if it is within limits it updates the creditlimit and grace period and shows a message 'Credit granted', else shows 'Credit cannot be issued'.

The class has a 'cancelCreditCard()' method that can cancel the credit card by resetting all the attributes and displaying a message that the card has been cancelled.

The class has a "display()" method that, if the credit has been approved, displays the carddetails and credit information; otherwise, it displays "Grace period:" along with the current grace period.

4.4 BankGUI

Here is the the methods which are used in BankGUI are given below:

- actionPerformed(ActionEvent g):

In this program, the user generates a semantic event by pressing a button. The program includes multiple buttons, each with a specific purpose, such as adding or withdrawing funds from a debit or credit card, setting credit limits, canceling credit cards, and displaying card information.

- addtoDebitCard:

This code block appears to be an event handler for an "add to Debit Card" button click. When the button is clicked, the code retrieves the input values from several text fields and attempts to parse them into appropriate data types. It then checks if the ArrayList "al" is empty. If it is, the code creates a new DebitCard object and adds it to the ArrayList, then displays a success message using a JOptionPane dialog box.

If "al" is not empty, the code iterates over its contents and checks if any DebitCard objects with the same card ID as the input already exist. If it finds a match, the code displays an error message indicating that the card ID has already been added. If no match is found, the code creates a new DebitCard object and adds it to the ArrayList, then displays a success message using a JOptionPane dialog box.

If any of the input values cannot be parsed into the appropriate data types, the code catches a NumberFormatException and displays an error message indicating that there is an error with the card ID.

Overall, this code appears to handle the process of adding new DebitCard objects to an ArrayList and ensuring that each card ID is unique.

- withdrawfromDebitCard:

This code block appears to be an event handler for a "withdraw from Debit Card" button click. When the button is clicked, the code retrieves the input values from several text fields and attempts to parse them into appropriate data types. It then checks if the ArrayList "al" is empty. If it is, the code displays a message indicating that no DebitCards are available.

If "al" is not empty, the code iterates over its contents and checks if any DebitCard objects have a matching card ID and PIN number to the input values. If it finds a match, the code calls the DebitCard object's "withdraw" method with the withdrawal amount and date of withdrawal as parameters, and displays a success message using a JOptionPane dialog box.

If no match is found, the code displays an error message indicating that the card ID or PIN number is invalid.

If any of the input values cannot be parsed into the appropriate data types, the code catches a NumberFormatException and displays an error message indicating that the withdrawal amount is invalid.

Overall, this code appears to handle the process of withdrawing money from a DebitCard object by matching the card ID and PIN number entered by the user with those stored in the ArrayList, and calling the appropriate method on the DebitCard object if a match is found.

- addtoCreditCard:

This code defines the behavior of an "add to credit card" button in a graphical user interface. When the button is clicked, the code tries to retrieve the input values from several text fields, including the card balance, card ID, bank account number, issuer bank, client name, CVC number, interest rate, and expiration date. It then checks if there are any credit cards in a list of bank cards called "al". If the list is empty, it creates a new CreditCard object using the retrieved values and adds it to the list, displaying a success message. If the list is not empty, it checks if any

CreditCard object in the list has the same card ID as the one retrieved from the text field. If there is a match, it displays an error message indicating that the card ID has been added twice. Otherwise, it creates a new CreditCard object and adds it to the list, displaying a success message. If an exception occurs during the parsing of the input values, it displays an error message.

- setTheCreditLimit:

This code defines a method that is triggered when the "Set Credit Limit" button is pressed. The method first retrieves the card ID, credit limit, and grace period values entered by the user, and then checks if the array list of bank cards (al) is empty. If the list is empty, the method displays an error message. Otherwise, the method iterates through the list of bank cards, checks if the card ID belongs to a credit card, and sets the credit limit and grace period for that card if it does. If the card ID doesn't belong to a credit card or isn't real, the method displays an error message. If the user enters invalid input, the method also displays an error message. The method does not return anything.

- cancelCreditcard:

This code is an implementation of an event listener that listens for actions on the "cancelCreditcard" button. When the button is clicked, the code retrieves the input values from several text fields, parses them into the appropriate data types (integer and boolean), and searches through a list of bank card objects to find a credit card with a matching card ID. If a credit card is found, the "cancelCreditCard" method is called on the credit card object to cancel the card. A message is displayed to the user indicating whether the cancellation was successful or not. If there are no bank cards available, an error message is displayed. If there is no credit card with the specified card ID, an error message is displayed.

- display:

It is a method for showing a new Window frame that is contained within the main Window or another Window. In this assignment, I made a new Window utilizing the JFrame to display the program's output.

- Clear:

It is just a technique for emptying the program's TextField of its contents. When it is pressed in my software, all of the TextFields are cleared

- main(String args[ ]):

This is a public static method that can be called by the Java Virtual Machine (JVM) without requiring the class to be represented. The program starts executing from this method. In the given assignment, the main method creates an instance of the ProductGUI class.

5. Testing (Inspection

5.1 **Test 1 – To Compile and running the code in Command Prompt.**

Test No :	1
Objective:	To compile and Run the program BankGUI.java in Command Prompt.
Action:	Compile the program BankGUI.java using the Javac command Javac BankGUI.java Run the class file using Command java Java BankGUI
Expected Result:	The program would be compiled and running in command prompt.
Actual Result:	The program is compiled and run in command prompt.
Conclusion:	The test is successful.

Table 5:To Run the program BankGUI.java in command prompt

Output results:

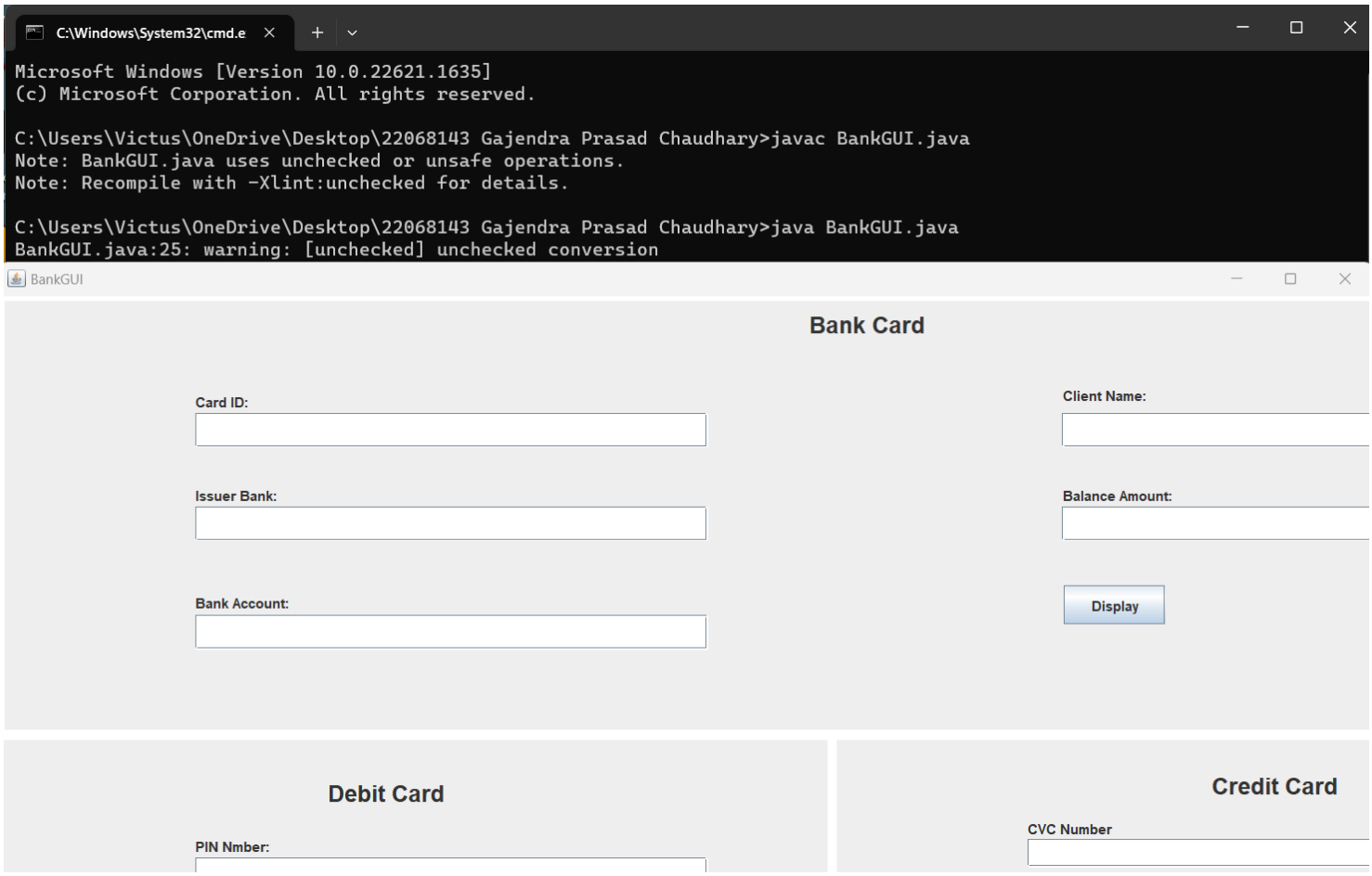


Figure 3Screenshot of BankGUI running from command prompt

5.2 Test-2 (Adding objects of DebitCard and CreditCard, withdrawing amount from debit card,setting the credit limit and removing the credit card)

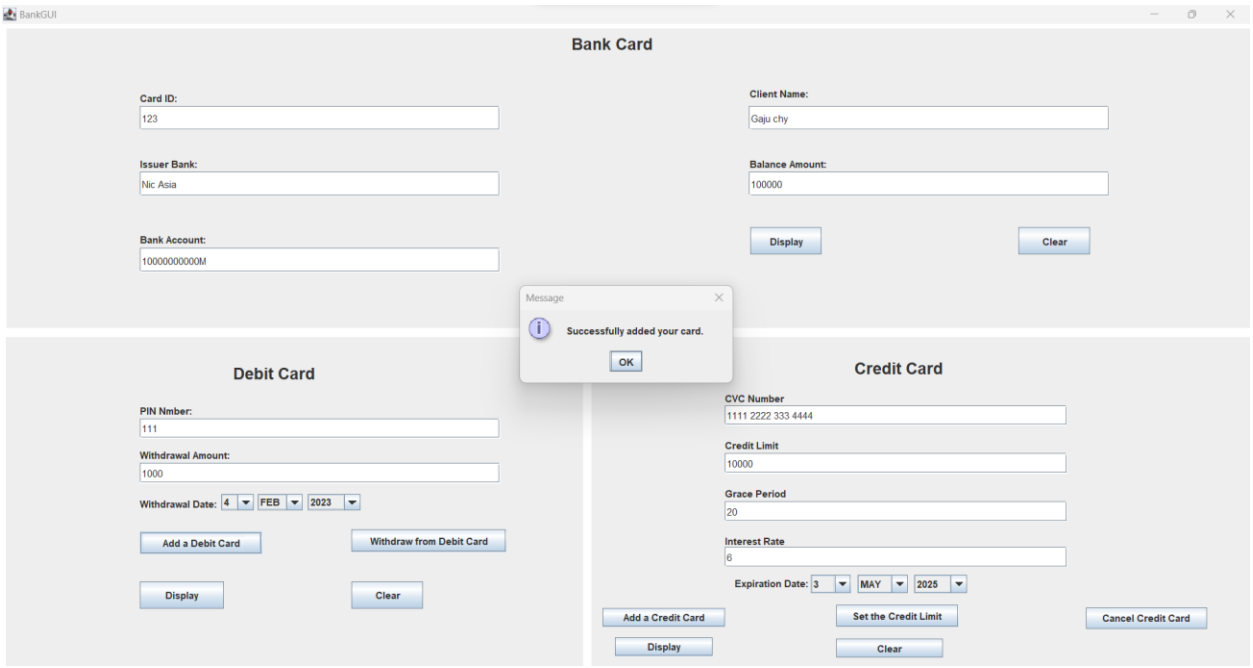


Figure 4 : Adding all both debit card and credit card

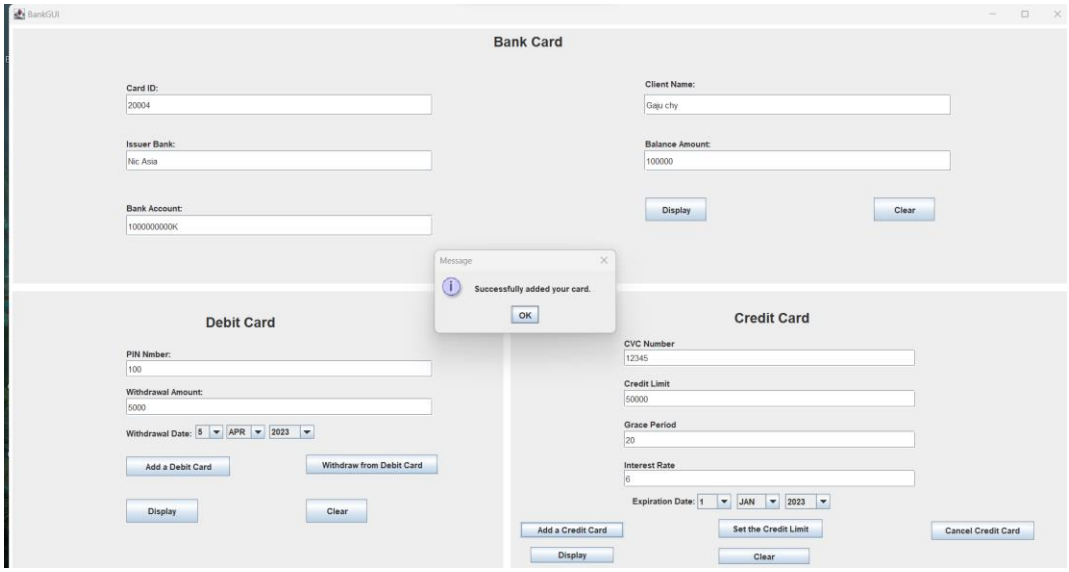


Figure 5 Adding Credit Objects

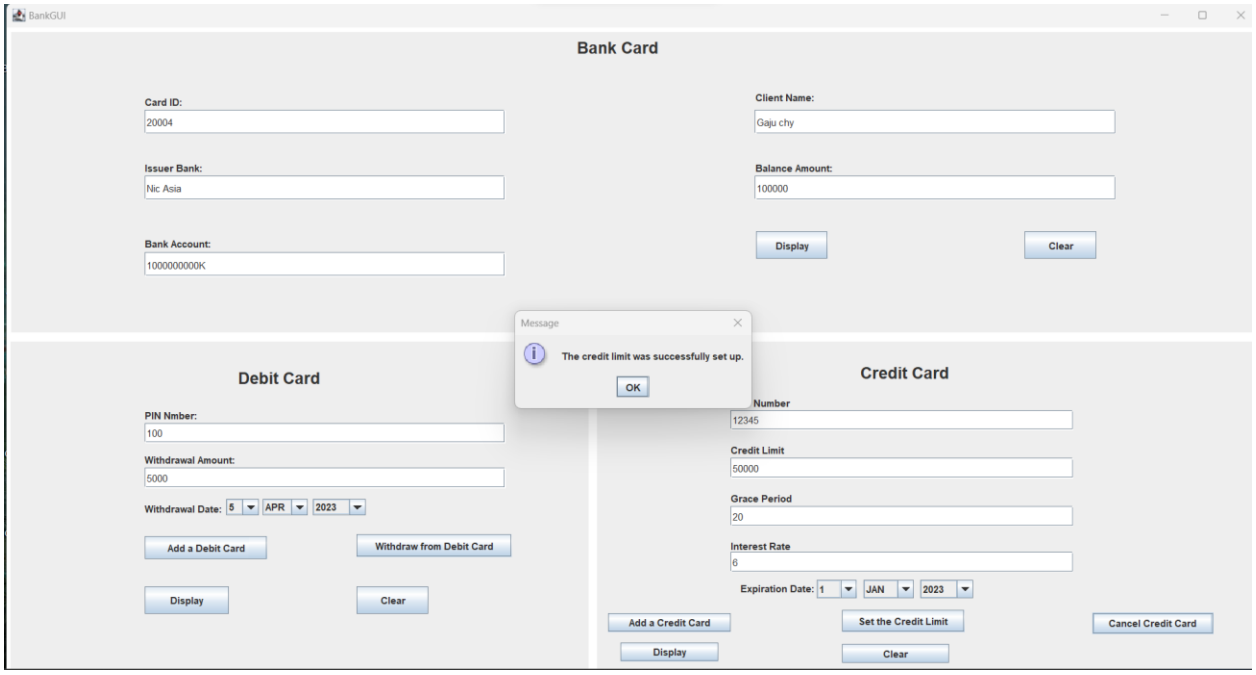


Figure 6:Screenshot of cancel credit card

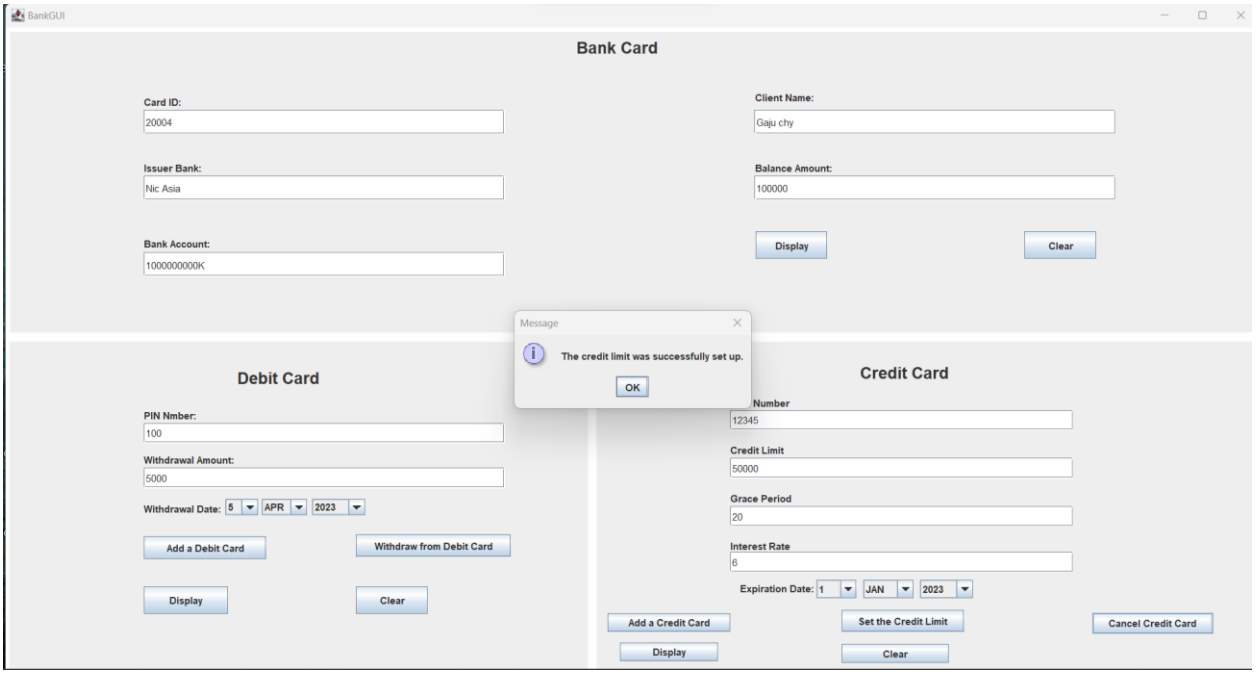


Figure 7: Screenshot of setting the credit limit

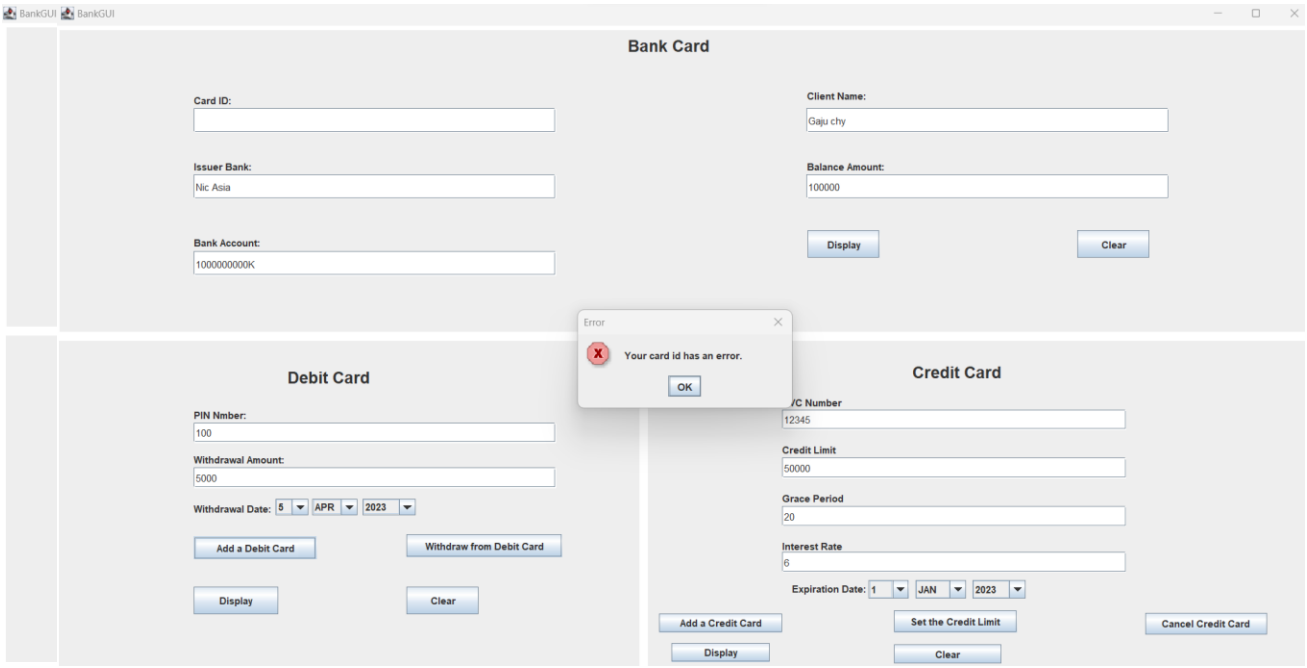


Figure 8: Screenshot of Withdraw

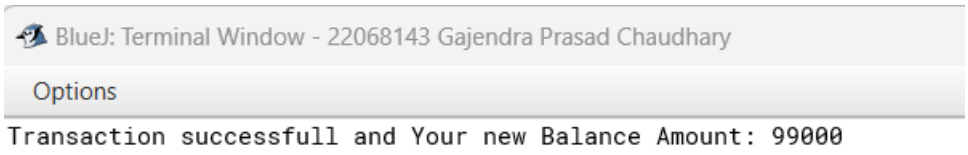


Figure 9:Display of Withdraw Amount

5.3 Test-3(Testing Appropriate Dialog boxes when unsuitable values entered)

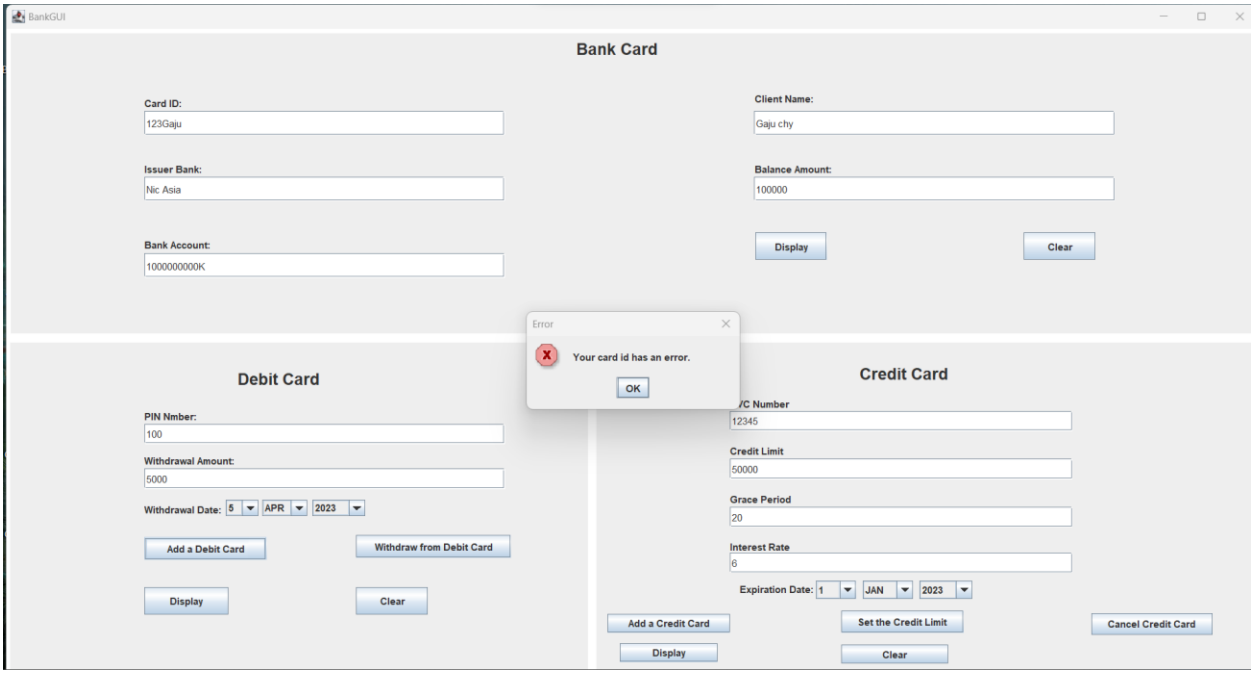


Figure 10: Screen shot of Error in card ID

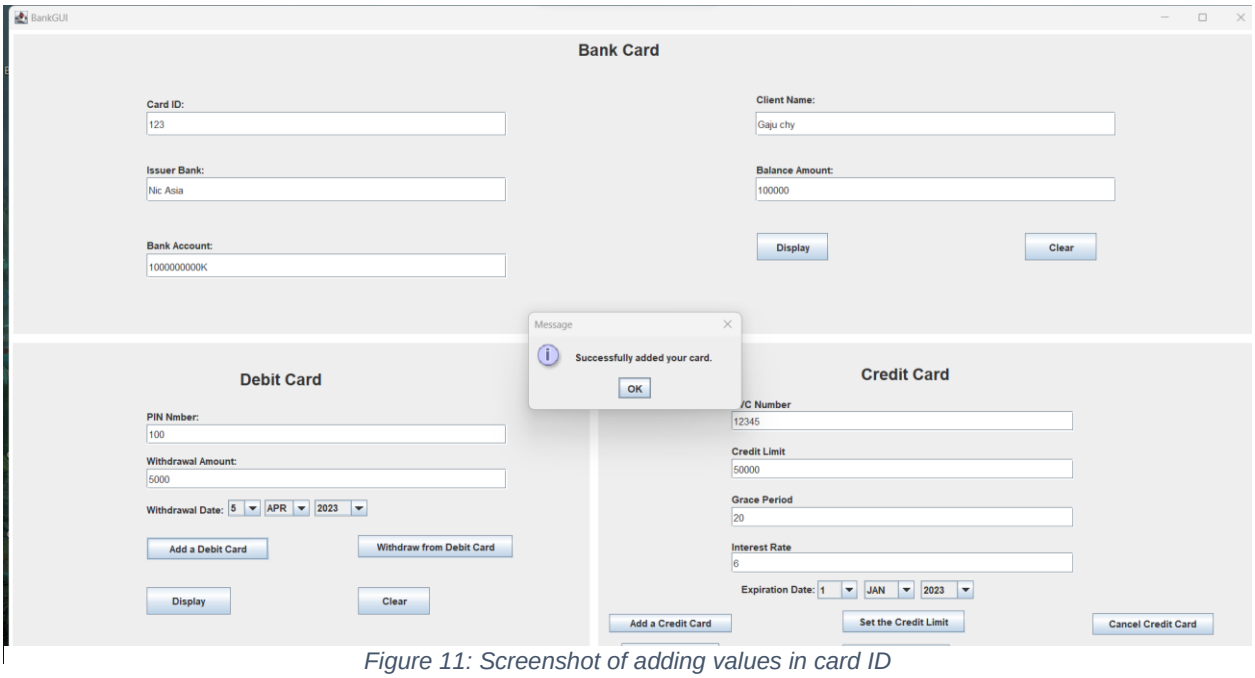


Figure 11: Screenshot of adding values in card ID

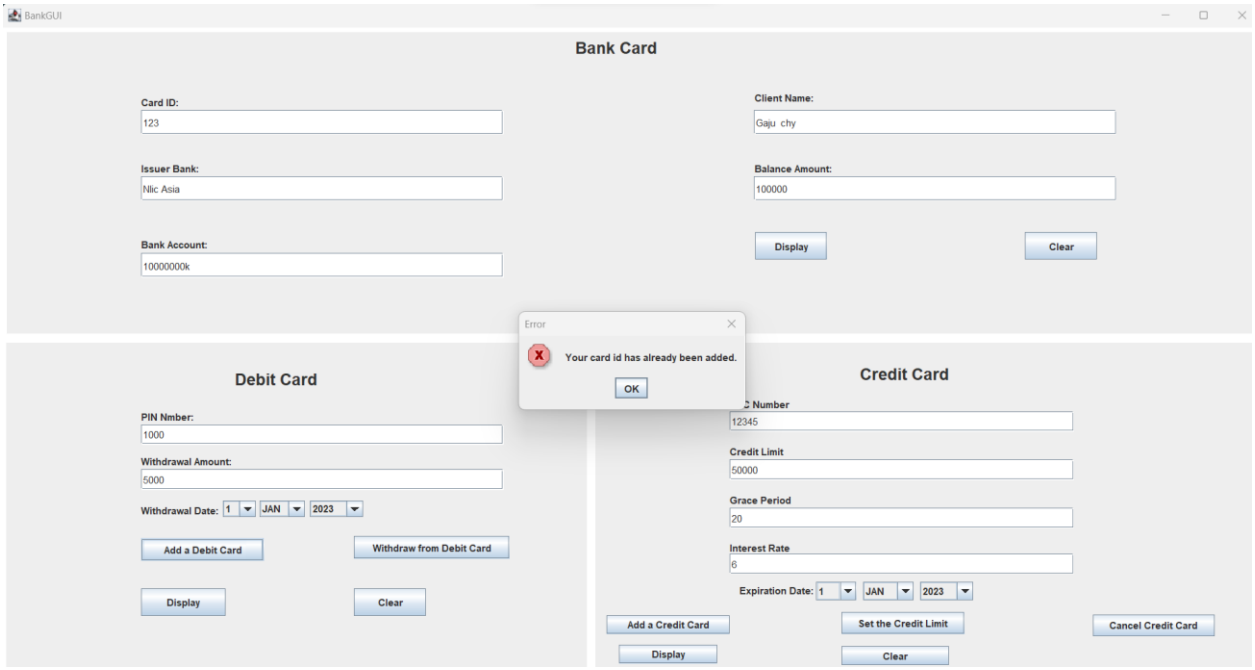


Figure 12:Screenshot of adding similar values

6. Error detection and Correction

7. Conclusion

Through this coursework, we gained a solid understanding of programming basics and project development concepts. We created three classes - BankCard (parent class), DebitCard and CreditCard (subclasses), and a GUI class called BankGUI. Each class had its own constructors and methods that were assigned with various attributes with different data types. We learned about the importance of using accessor and mutator methods, and how to call the superclass constructor in our subclasses.

Furthermore, in our second semester, we expanded our knowledge by delving into the graphical user interface (GUI) of Java. We explored topics such as array lists, abstract methods, GUI AWT and Swing libraries, dialogue boxes, exception handling, and event handling. Overall, this coursework has provided us with a solid foundation in programming and software development, preparing us for more advanced topics in the field

I had a lot of difficulties working on this coursework even though it was my first project of this kind. Due to the occurrence of several errors, writing the program was arduous, and designing the class diagram was also a difficult effort. I found pseudo code to be extremely difficult, and testing (inspection) was a brand-new subject that took a lot of reading to properly understand.

This coursework has helped me to develop my skills in time management, and has given me valuable experience in developing professional projects in the future, if needed. Moreover, I have learned to work with standard software tools like BlueJ and gained a deep understanding of Java programming language, which is an invaluable asset in my future endeavors. Overall, I learned a lot from this experience, and I will surely use it in my future work as a programmer.

8. References

© Copyright2016 by Dreamtech Press,19-A,Ansari Road,Daryanganj,New Delhi-110002. https://pdfcoffee.com/core-java-byrnageswara-rao-pdf-free.html

9. Appendix

```
import javax.swing.*;

import java.awt.event.*;

import java.awt.Color;

import java.awt.Font;

import java.util.ArrayList;


public class BankGUI  implements ActionListener{

   private JFrame frame;


   private JPanel mainPanel, debitPanel, creditPanel;
```

```
    private JLabel title, title1, title2;


    private JLabel cardID, clientName, issuerBank, pinNumber,
bankAccount,
    balanceAmount, withdrawalAmount, cvcNumber, creditLimit,
gracePeriod, interestRate, withdrawalDate, expirationDate;


    private JTextField cardID2, clientName2,  issuerBank2, pinNumber2,
bankAccount2,
    balanceAmount2, withdrawalAmount2, cvcNumber2, creditLimit2,
gracePeriod2,interestRate2;


    private JButton addtoDebitCard, withdrawfromDebitCard,
addtoCreditCard,
    display, clear, setTheCreditLimit, cancelCreditcard, display2,
clear2,display3,clear3;


    private JComboBox withdrawalDate1, withdrawalDate2,
withdrawalDate3, expirationDate1, expirationDate2, expirationDate3;


    ArrayList<BankCard> al = new ArrayList();
    public BankGUI (){
        frame = new JFrame("BankGUI");


        mainPanel   = new JPanel();
        debitPanel  = new JPanel();
        creditPanel = new JPanel();


        //---mainPanel code starts from here----
```

```
title = new JLabel ("Bank Card");
title.setBounds    (700,15,120,20);
mainPanel.add      (title);
title.setFont      (new Font("Arial",Font.BOLD,20));

display3 = new JButton ("Display");
clear3   = new JButton ("Clear");

display3.setBounds(920,250,88,34);
clear3.setBounds(1250,250,88,34);

mainPanel.add(display3);
mainPanel.add(clear3);

cardID          = new JLabel   ("Card ID:");
clientName      = new JLabel   ("Client Name:");
issuerBank      = new JLabel   ("Issuer Bank:");
bankAccount     = new JLabel   ("Bank Account:");
balanceAmount   = new JLabel   ("Balance Amount:");

cardID2           = new JTextField();
clientName2       = new JTextField();
issuerBank2       = new JTextField();
bankAccount2      = new JTextField();
balanceAmount2    = new JTextField();

cardID.setBounds        (170, 81,  48,20);
```

```
clientName.setBounds      (919, 76,  81,20);
issuerBank.setBounds      (170, 162, 74,20);
bankAccount.setBounds     (170, 245, 86,41);
balanceAmount.setBounds   (919, 152,102,41);

cardID2.setBounds         (170,101,443,30);
clientName2.setBounds     (919,101,443,30);
issuerBank2.setBounds     (170,182,443,30);
bankAccount2.setBounds    (170,276,443,30);
balanceAmount2.setBounds  (919,182,443,30);

mainPanel.add(cardID);
mainPanel.add(clientName);
mainPanel.add(issuerBank);
mainPanel.add(bankAccount);
mainPanel.add(balanceAmount);

mainPanel.add(cardID2);
mainPanel.add(clientName2);
mainPanel.add(issuerBank2);
mainPanel.add(bankAccount2);
mainPanel.add(balanceAmount2);

mainPanel.setBounds(0,0,1920,380);
mainPanel.setLayout(null);
mainPanel.setVisible(true);

mainPanel.setBorder(BorderFactory.createLineBorder(Color.WHITE,6));
```

```
//---- mainPanel code ends here -----

//-------------------------------------------------------------

//---- Debit Card code starts from here ----

title1 = new JLabel ("Debit Card");
title1.setBounds    (284,32,120,36);
debitPanel.add      (title1);
title1.setFont      (new Font("Arial",Font.BOLD,20));

pinNumber           = new JLabel    ("PIN Nmber:");
pinNumber2          = new JTextField();

pinNumber.setBounds       (170,80,77,30);
pinNumber2.setBounds      (170,105,443,25);

debitPanel.add(pinNumber);
debitPanel.add(pinNumber2);

withdrawalAmount  = new JLabel     ("Withdrawal Amount:");
withdrawalAmount2 = new JTextField ("");

withdrawalAmount.setBounds         (170,140,121,20);
withdrawalAmount2.setBounds        (170,160,443,25);

withdrawalDate = new JLabel        ("Withdrawal Date:");
withdrawalDate.setBounds           (170,190,103,41);
```

```
String day  [] = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12",
"13", "14", "15", "16", "17", "18", "19", "20",
        "21", "22", "23", "24" ,"25", "26", "27", "28" ,"29", "30","31"};
String month[] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN",
        "JUL", "AUG", "SEPT", "OCT", "NOV", "DEC"};
String year [] = {"2023", "2024", "2025", "2026", "2027", "2028",
"2029", "2030"};

withdrawalDate1   = new JComboBox (day);
withdrawalDate2   = new JComboBox (month);
withdrawalDate3   = new JComboBox (year);

withdrawalDate1.setBounds(270,198,40,20);
withdrawalDate2.setBounds(315,198,55,20);
withdrawalDate3.setBounds(376,198,65,20);

addtoDebitCard       = new JButton ("Add a Debit Card");
withdrawfromDebitCard = new JButton ("Withdraw from Debit Card");
display              = new JButton ("Display");
clear                = new JButton ("Clear");

addtoDebitCard.setBounds      (170,244,150,28);
withdrawfromDebitCard.setBounds(430,241,190,28);
display.setBounds             (170,305,103,34);
clear.setBounds               (430,305,88,34);

debitPanel.add(withdrawalAmount);
```

```
        debitPanel.add(withdrawalAmount2);

        debitPanel.add(withdrawalDate1);
        debitPanel.add(withdrawalDate2);
        debitPanel.add(withdrawalDate3);

        debitPanel.add(withdrawalDate);

        debitPanel.add(addtoDebitCard);
        debitPanel.add(withdrawfromDebitCard);
        debitPanel.add(display);
        debitPanel.add(clear);

        debitPanel.setBounds(0, 380, 720, 700);
        debitPanel.setLayout(null);
        debitPanel.setVisible(true);

debitPanel.setBorder(BorderFactory.createLineBorder(Color.WHITE,5));

        //-------Credit Card code starts from here ----
        title2 = new JLabel ("Credit Card");
        title2.setBounds     (328,26,124,36);
        creditPanel.add     (title2);
        title2.setFont(new Font("Arial",Font.BOLD,20));

        cvcNumber    = new JLabel ("CVC Number");
        creditLimit  = new JLabel ("Credit Limit");
        gracePeriod  = new JLabel ("Grace Period");
```

```
interestRate = new JLabel ("Interest Rate");

cvcNumber2          = new JTextField();
creditLimit2        = new JTextField();
gracePeriod2        = new JTextField();
interestRate2       = new JTextField();

expirationDate = new JLabel ("Expiration Date:");
expirationDate.setBounds    (181,297,100,20);
creditPanel.add             (expirationDate);

String day2  [] = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14", "15", "16", "17", "18", "19", "20",
        "21", "22", "23", "24" ,"25", "26", "27", "28" ,"29", "30","31"};

String month2[] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN",
        "JUL", "AUG", "SEPT", "OCT", "NOV", "DEC"};

String year2 [] = {"2023", "2024", "2025", "2026", "2027", "2028",
"2029", "2030"};

expirationDate1  = new JComboBox  (day2);
expirationDate2  = new JComboBox  (month2);
expirationDate3  = new JComboBox  (year2);

addtoCreditCard  = new JButton    ("Add a Credit Card");
setTheCreditLimit= new JButton    ("Set the Credit Limit");
cancelCreditcard = new JButton    ("Cancel Credit Card");
```

```
display2         = new JButton    ("Display");
clear2           = new JButton    ("Clear");

addtoCreditCard.setBounds       (19,338,150,23);
setTheCreditLimit.setBounds     (306,334,150,27);
cancelCreditcard.setBounds      (613,337,149,27);
display2.setBounds              (34,374,120,23);
clear2.setBounds                (306,376,131,23);

expirationDate1.setBounds (275,297,49,23);
expirationDate2.setBounds (332,297,62,23);
expirationDate3.setBounds (402,297,65,23);

cvcNumber.setBounds     (169,70,87,20);
creditLimit.setBounds   (169,128,82,20);
gracePeriod.setBounds   (169,187,82,20);
interestRate.setBounds  (169,245,92,20);

cvcNumber2.setBounds    (169,89, 421,25);
creditLimit2.setBounds  (169,148,421,25);
gracePeriod2.setBounds  (169,207,421,25);
interestRate2.setBounds (169,263,421,25);

creditPanel.add(cvcNumber);
creditPanel.add(creditLimit);
creditPanel.add(gracePeriod);
creditPanel.add(interestRate);
```

```
        creditPanel.add(cvcNumber2);
        creditPanel.add(creditLimit2);
        creditPanel.add(gracePeriod2);
        creditPanel.add(interestRate2);

        creditPanel.add(expirationDate1);
        creditPanel.add(expirationDate2);
        creditPanel.add(expirationDate3);

        creditPanel.add(addtoCreditCard);
        creditPanel.add(setTheCreditLimit);
        creditPanel.add(cancelCreditcard);
        creditPanel.add(display2);
        creditPanel.add(clear2);

        creditPanel.setBounds(720,380,1200,700);
        creditPanel.setLayout(null);
        creditPanel.setVisible(true);

creditPanel.setBorder(BorderFactory.createLineBorder(Color.WHITE,5));

        //------------------------------------------------------------------------

        display.addActionListener(this);
        display2.addActionListener(this);
        display3.addActionListener(this);

        clear.addActionListener(this);
```

```
        clear2.addActionListener(this);
        clear3.addActionListener(this);

        withdrawfromDebitCard.addActionListener(this);
        addtoDebitCard.addActionListener(this);

        setTheCreditLimit.addActionListener(this);
        cancelCreditcard.addActionListener(this);
        addtoCreditCard.addActionListener(this);
        cancelCreditcard.addActionListener(this);

        //-------------------------------------
        frame.add(mainPanel);
        frame.add(debitPanel);
        frame.add(creditPanel);
        frame.setLayout(null);
        frame.setVisible(true);
        frame.setSize(1920,900);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent g){
        if (g.getSource()== addtoDebitCard){
            try {
                // Get the values from the text fields
                int    balanceAmount =
Integer.parseInt(balanceAmount2.getText());
```

```
            int    cardid       = Integer.parseInt(cardID2.getText());
            String bankAccount   = String.valueOf(bankAccount2.getText());
            String issuerBank    = String.valueOf(issuerBank2.getText());
            String clientName    = String.valueOf(clientName2.getText());
            int    pinNumber     = Integer.parseInt(pinNumber2.getText());


            if (al.isEmpty()) {
                DebitCard debitCard = new DebitCard(balanceAmount,
cardid, bankAccount,issuerBank, clientName, pinNumber);
                al.add(debitCard);
                JOptionPane.showMessageDialog(frame, "Successfully
added your card.");
            } else {
                boolean a = false;
                for (BankCard obj : al) {
                    if (obj instanceof DebitCard && obj.getCardId() == cardid ) {
                        {
                            DebitCard debitCard = new
DebitCard(balanceAmount, cardid, bankAccount, issuerBank, clientName,
pinNumber);
                            al.add(debitCard);
                            JOptionPane.showMessageDialog(frame, "Your card
id has already been added.", "Error", JOptionPane.ERROR_MESSAGE);
                            a = true;
                            break;
                        }
                    }
                }
                if (!a) {
```

```
                    JOptionPane.showMessageDialog(frame, "Successfully
added your card.");
                }
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog(frame, "Your card id has an
error.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }


    else if (g.getSource() == withdrawfromDebitCard) {
        try {
            int cardId              = Integer.parseInt(cardID2.getText());
            int pinNumber           = Integer.parseInt(pinNumber2.getText());
            int withdrawalAmount    =
Integer.parseInt(withdrawalAmount2.getText());
            String dateOfWithdrawal = withdrawalDate.getText();


            boolean c = false;


            if (al.isEmpty()) {
                JOptionPane.showMessageDialog(frame, "No DebitCard
available");
            } else {
                for (BankCard obj : al) {
                    if(obj instanceof DebitCard){
                        DebitCard debitObj = (DebitCard)obj;
```

```
                    if(debitObj.getCardId() == cardId && debitObj.getPIN()
== pinNumber){
                        debitObj.withdraw(withdrawalAmount,
dateOfWithdrawal, pinNumber );
                        JOptionPane.showMessageDialog(frame, "Withdrawal
Successful");
                    }
                    else{
                        JOptionPane.showMessageDialog(frame, "Invalid pin
number or card id.");
                    }
                }



            }
        }


    }
    catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(frame, "Please enter a valid
withdrawal amount");
    }
}


//credit card
else if (g.getSource() == addtoCreditCard) {
    try {
        int balanceAmount =
Integer.parseInt(balanceAmount2.getText());
        int cardid = Integer.parseInt(cardID2.getText());
```

```
            String bankAccount = String.valueOf(bankAccount2.getText());
            String issuerBank = String.valueOf(issuerBank2.getText());
            String clientName = String.valueOf(clientName2.getText());
            int cvcNumber = Integer.parseInt(cvcNumber2.getText());
            double intrestrate =
Double.parseDouble(interestRate2.getText());


            String expiration1 =
(expirationDate1.getSelectedItem()).toString();
            String expiration2 =
(expirationDate2.getSelectedItem()).toString();
            String expiration3 =
(expirationDate3.getSelectedItem()).toString();
            String expiration = expiration1+expiration2+expiration3;
            if (al.isEmpty()) {
                CreditCard creditCard = new CreditCard(cardid, clientName,
issuerBank, bankAccount, balanceAmount, cvcNumber, intrestrate,
expiration);
                al.add(creditCard);
                JOptionPane.showMessageDialog(frame, "Successfully
added your card.");
            } else {
                boolean b = false;
                for (BankCard obj : al) {
                    if (obj instanceof CreditCard) {
                        if (obj.getCardId() == cardid) {
                            JOptionPane.showMessageDialog(frame, "Your card
id has been added twice.", "Error", JOptionPane.ERROR_MESSAGE);
                            b = true;
                            break;
```

```
                    }
                }
            }
            if (b == false) {
                CreditCard creditCard = new CreditCard(cardid,
clientName, issuerBank, bankAccount, balanceAmount, cvcNumber,
intrestrate, expiration);
                al.add(creditCard);
                JOptionPane.showMessageDialog(frame, "Your credit card
has been added successfully");
            }
        }
    } catch (NumberFormatException exception2) {
        JOptionPane.showMessageDialog(frame, "ERROR OCCUR.");
    }
}
else if(g.getSource() == setTheCreditLimit){

    System.out.println("Test");
    try{
        int cardId = Integer.parseInt(cardID2.getText());
        double creditLimit = Double.parseDouble(creditLimit2.getText());
        int gracePeriod = Integer.parseInt(gracePeriod2.getText());
        boolean setlimit = false ;

        if(al.isEmpty()){
            JOptionPane.showMessageDialog(frame,"ERROR: Invalid
card found.");
        }else{
```

```
                for(BankCard obj:al){
                    if(obj instanceof CreditCard && obj.getCardId() == cardId){
                        CreditCard creditCard = (CreditCard) obj;
                        creditCard.setCreditLimit(creditLimit, gracePeriod);
                        JOptionPane.showMessageDialog(frame,"The credit
limit was successfully set up.");
                        setlimit = true;
                        break;
                    }
                }
                if(!setlimit){

                    JOptionPane.showMessageDialog(frame,"ERROR: The
card ID doesn't belong to a credit card or isn't real.");

                }
            }
        }catch(NumberFormatException s){
            JOptionPane.showMessageDialog(frame,"ERROR: Invalid
input. Please enter valid numbers.");

        }

    }
    else if(g.getSource() == cancelCreditcard){
        try{
            int cardId = Integer.parseInt(cardID2.getText());
            boolean cancelcard = false;
```

```
            if (al.isEmpty())
            {
                // Display an error message if the Objects list is empty
                JOptionPane.showMessageDialog(frame,"ERROR: No such
bank card is avaliable.");
            }else{
                for(BankCard obj:al)
                {
                    if(obj instanceof CreditCard && obj.getCardId() == cardId ){


                        CreditCard creditCard = (CreditCard)obj;
                        creditCard.cancelCreditCard();
                        JOptionPane.showMessageDialog(frame,"The credit
limit was successfully set up.");
                        cancelcard = true;
                        break;


                    }
                }
                if(cancelcard == false)
                {
                    JOptionPane.showMessageDialog(frame,"The Card ID has
not been match  the credit
card","ERROR",JOptionPane.ERROR_MESSAGE);
                }
            }


        }catch(NumberFormatException creditcancel)
```

```
            {
                JOptionPane.showMessageDialog(frame, "ERROR.Assign
value must not avaliable OR Text field should not be empty.");


            }
        }
        else if(g.getSource() == display)
        {
            try
            {


                int cardId = Integer.parseInt(cardID2.getText());
                boolean D = false;
                for (BankCard each:al)
                {
                    {
                        if (each instanceof DebitCard && each.getCardId() ==
cardId)
                        {
                            DebitCard debitcard = (DebitCard) each;


                            debitcard.display();
                            D = true;
                            break;
                        }
                    }
                }
                if (D == false)
```

```
            {
                JOptionPane.showMessageDialog(frame,"ERROR.Assign
value must not avaliable OR There shouldn't be a blank text field.");
            }
        }


        catch(NumberFormatException e)
        {
            JOptionPane.showMessageDialog(frame,"ERROR.Assign value
must not avaliable OR There shouldn't be a blank text field.");
        }
    }


    else if(g.getSource() == display2)
    {
        try
        {


            int cardId = Integer.parseInt(cardID2.getText());
            boolean cardFound = false;


            for (BankCard each:al)
            {
                if (each instanceof CreditCard && each.getCardId() ==
cardId)
                {
                    CreditCard creditcard = (CreditCard) each;
                    creditcard.display();
                    cardFound = true;
```

```
                    break;
                }
            }

            if (!cardFound)
            {
                JOptionPane.showMessageDialog(frame,"ERROR: Not found: Card with the requested ID.");
            }
        }

        catch(NumberFormatException e)
        {
            JOptionPane.showMessageDialog(frame,"ERROR: Invalid input. Please! Enter a genuine card ID here. ID.");
        }
    }

    else if(g.getSource() == display3)
    {
        try
        {

            int cardId = Integer.parseInt(cardID2.getText());
            boolean D = false;
            for (BankCard each:al)
            {
                {
```

```
                    if (each instanceof CreditCard && each.getCardId() ==
cardId)
                    {
                        CreditCard creditcard = (CreditCard) each;


                        creditcard.display();
                        D = true;
                        break;
                    }
                }
            }
            if (D == false)
            {
                JOptionPane.showMessageDialog(frame,"ERROR.Assign
value must not avaliable OR There shouldn't be a blank text field.");
            }
        }


        catch(NumberFormatException e)
        {
            JOptionPane.showMessageDialog(frame,"ERROR.Assign value
must not avaliable OR There shouldn't be a blank text field.");
        }
    }


    else if(g.getSource()== clear3)
    {
        cardID2.setText("");
        clientName2.setText("");
```

```
            issuerBank2.setText("");
            bankAccount2.setText("");
            balanceAmount2.setText("");

            JOptionPane.showMessageDialog(frame,"Your form has been
cleared");
        }
        else if(g.getSource()== clear)
        {

            pinNumber2.setText("");
            withdrawalAmount2.setText("");

            JOptionPane.showMessageDialog(frame,"Your debitcard form has
been cleared");
        }
        else if(g.getSource()== clear2)
        {

            cvcNumber2.setText("");
            creditLimit2.setText("");
            interestRate2.setText("");
            gracePeriod2.setText("");

            JOptionPane.showMessageDialog(frame,"Your credit card form
has been cleared");
        }
    }
```

```
    public static void main(String args []){
        new BankGUI ();
    }
}
```