LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Spring

Student Name: MIRAJ DEEP BHANDARI

London Met ID: 22067814

College ID: NP01CP4A220197

Group: L1C8

Assignment Due Date: Friday, May 12, 2023

Assignment Submission Date: Saturday, May 6, 2023

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

Table of Figures

Table of Tables

1. INTRODUCTION

1.1 ABOUT THE COURSEWORK

Java is a general-purpose, class-based, object-oriented programming language that was developed by Sun Microsystems (now owned by Oracle Corporation) in the mid-1990s. Java is known for its "write once, run anywhere" principle, which means that Java code can run on any platform that has a Java Virtual Machine (JVM) installed, regardless of the underlying hardware and operating system (Hartman, 2023).

The primary aim of this coursework is to create a user-friendly Graphical User Interface (GUI) for BankCards. The project expands on the previously developed classes, including BankCard, DebitCard, and CreditCard. In the prior coursework, object values were assigned in the BlueJ environment. However, in this project, the emphasis is on developing a GUI that enables users to input various details such as CardID, Bank Account, Pin Number, Balance Amount etc and perform various operations on BankCards.

The GUI is developed using the BankGui Class, which contains the essential code for creating the GUI. The BankGui Class initializes the user interface components, including text fields, labels, and buttons, and handles user input and output. The user interface is designed to be user-friendly and easy to navigate, enabling users to perform their desired operations on their BankCards.

This coursework enables us to create a GUI using Java Swing and AWT components. The BankGui Class manages all user actions via an action listener, making it the active class for user interactions. In addition to the GUI, an ArrayList of BankCard Class is

utilized to store DebitCard and CreditCard objects, which can be used for various operations through the concept of downcasting. The implementation of downcasting is also included. Overall, this coursework covers the creation of the GUI, implementation of an ArrayList to store objects of the DebitCard and CreditCard classes, and the concept of downcasting.

In conclusion, this coursework is an essential project for the development of a user-friendly GUI for BankCards. The project covers various aspects, including the creation of the GUI using Java Swing and AWT components, the implementation of an ArrayList for storing DebitCard and CreditCard Class objects, and the concept of downcasting. The project will enable users to perform operations on their BankCards more efficiently and enhance their overall user experience.

1.2 TOOLS USED

1.2.1 BLUEJ

BlueJ is a free Java development platform created by Monash University for teaching object-oriented programming. It requires JDK 1.3 or later and has a user-friendly interface with tools specific to education, as well as standard development tools. It offers convenient features like sample programs and easy debugging and can run on multiple platforms. It also allows for interaction with objects (harleenk_99, 2022).

I used BlueJ as my main development platform for my coursework and found it to be extremely helpful in identifying and resolving errors in my code. It also provided a wide range of features and options for Java programming, including support for object-oriented programming principles and easy implementation of these principles. Overall, BlueJ proved to be an effective tool for my programming needs

1.2.2 Draw.IO

Draw.io is a user-friendly and versatile diagram software with a range of features for creating professional diagrams and charts. It has a drag-and-drop interface and an automatic layout function for arranging elements. It also offers a variety of shapes and visual elements for creating unique diagrams that can be used in different fields like software development, data visualization, project management and more (Hope, 2020).

Draw.io was a key tool in my coursework, allowing me to easily create class diagrams for BankCard, DebitCard, CreditCard and BankGui Class. It has a user-friendly drag-and-drop interface, and offers pre-built shapes which streamlined the process of creating diagrams. Overall, it was crucial for creating class diagrams efficiently and effectively.

1.2.3 MS WORD

Microsoft Word, also known as Winword, MS Word, or simply Word, is a widely-used word processing application developed by Microsoft and part of the Microsoft Office Suite. It is a versatile tool designed to streamline document creation, editing, and management and widely used by professionals and students for various tasks such as reports and presentations (Hope, 2021).

I used Microsoft Word, also known as MS Word or Word, to create report documents in my coursework. It is a widely-used and user-friendly word processor with features such as image, shape, icon, graph and chart insertions. It proved to be an essential tool for my coursework by allowing me to create professional-looking report documents efficiently.

1.2.4 FIGMA

Figma is a web-based collaborative design tool that allows multiple designers to work together on a single project in real-time. It is primarily used for creating user interface designs but also supports prototyping, wireframing, and graphic design. Figma's cloud-based platform allows designers to access their designs from any device with an internet connection, making it easy to work remotely or with a team across different locations. Its collaborative design capabilities, coupled with its design tools and prototyping features, make it a popular choice for design teams.

One of the standout features of Figma is its drag-and-drop functionality, which allows me to easily move components around on the canvas and place them exactly where i want them. In addition to its drag-and-drop feature, Figma also provides a variety of tools and options for designing the different parts of your GUI. For instance, i can adjust the size and positions of my components, as well as change their color and style to match my overall design aesthetic.

2. CLASS DIAGRAM

2.1 CLASS DIAGRAM OF BankCard

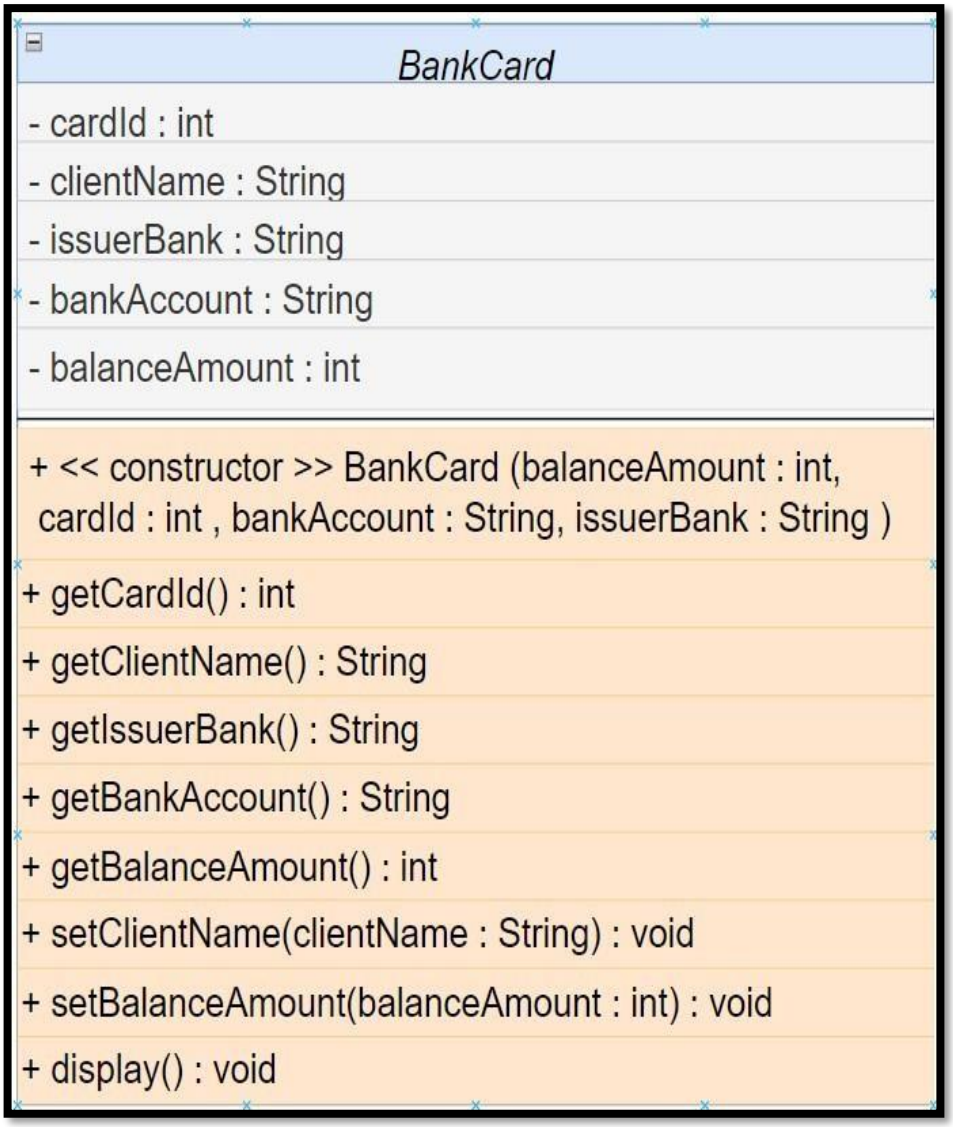

Figure 1: Class Diagram Of BankCard Class

2.2 CLASS DIAGRAM OF DebitCard

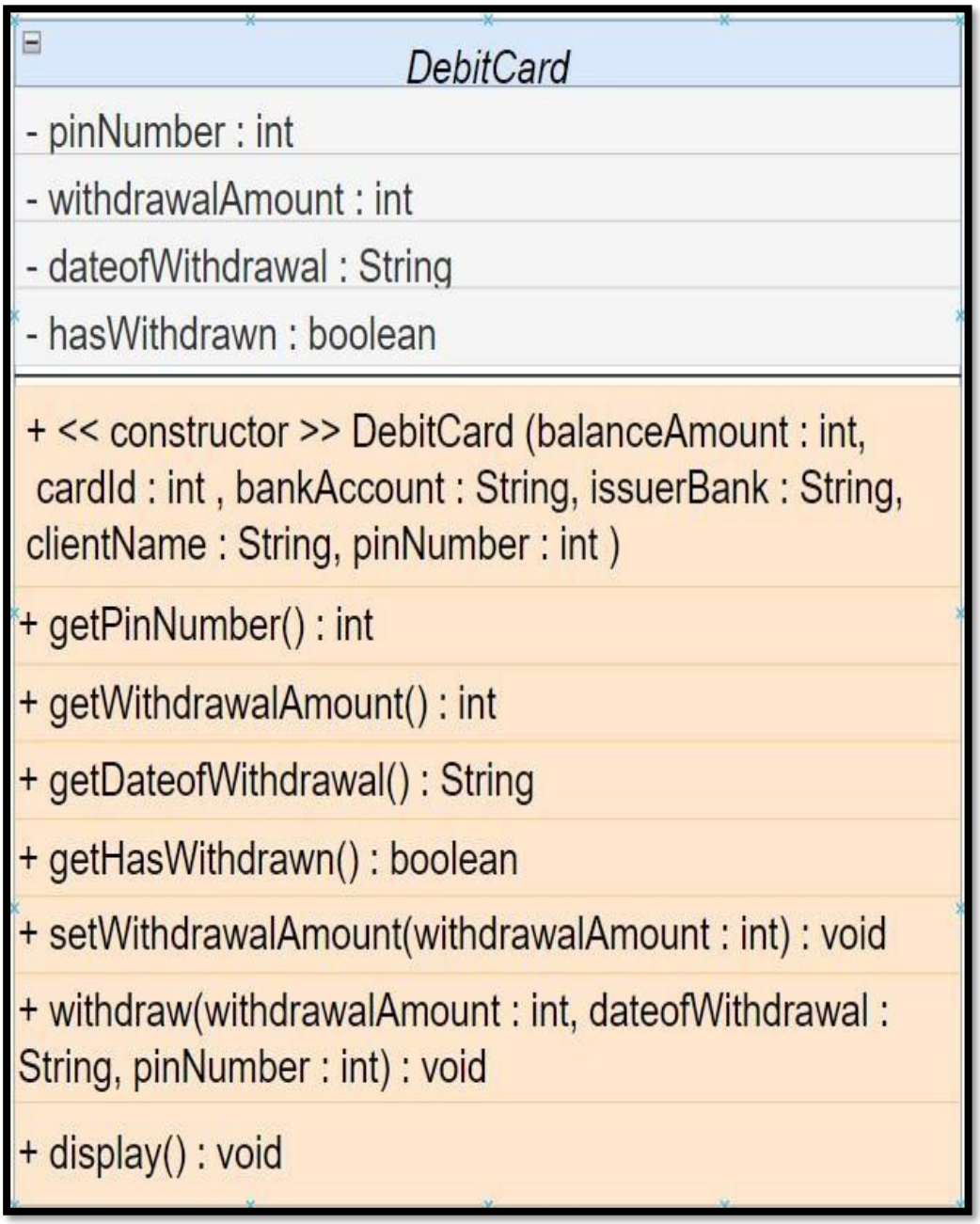

Figure 2: Class Diagram Of DebitCard Class

2.3 CLASS DIAGRAM OF CreditCard

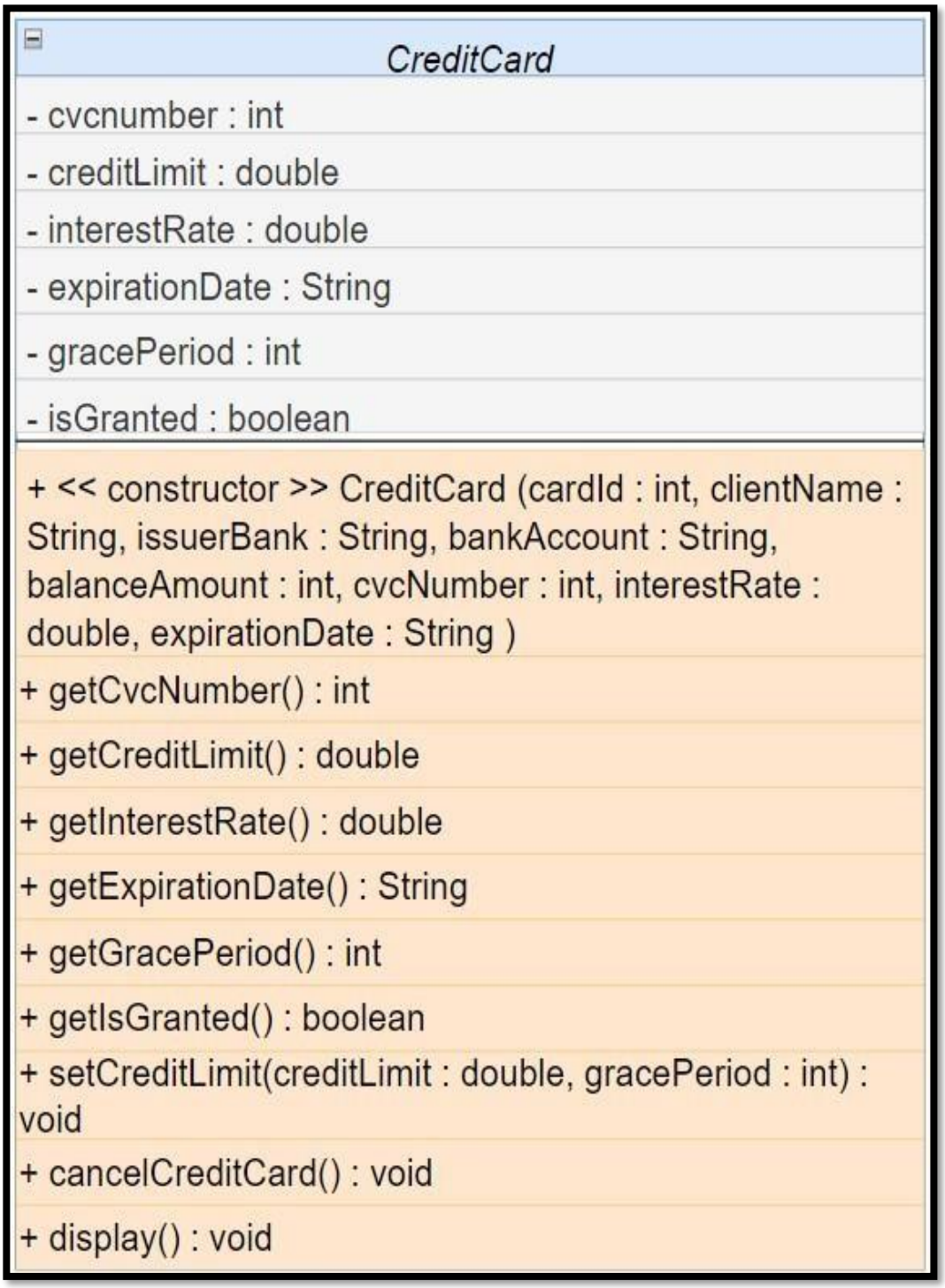

Figure 3:Class Diagram Of CreditCard Class

2.4 CLASS DIAGRAM OF BankGui

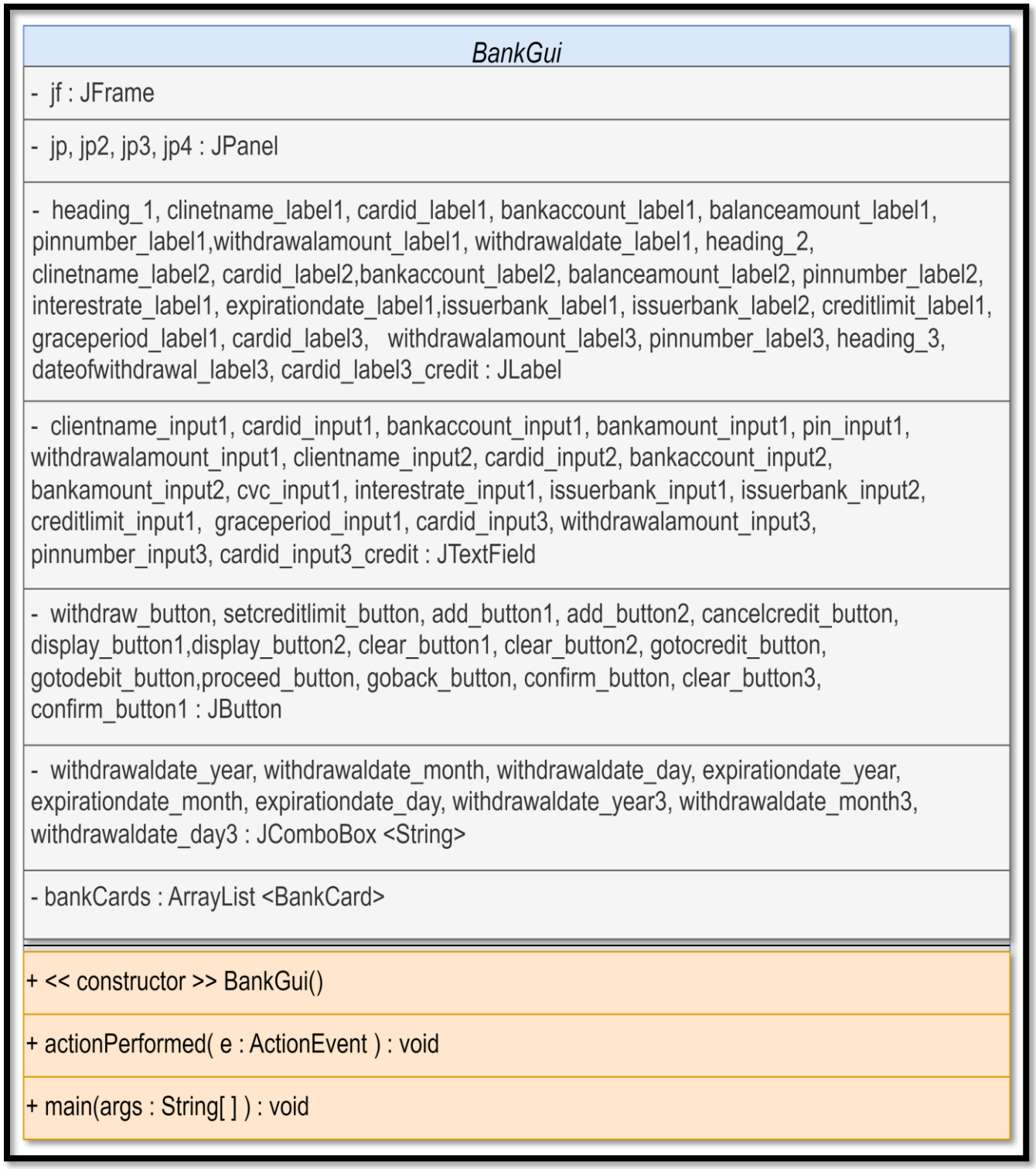

Figure 4: Class Diagram Of BankGui Class

2.5 CLASS DIAGRAM OF BankCard, DebitCard ,CreditCard and BankGui

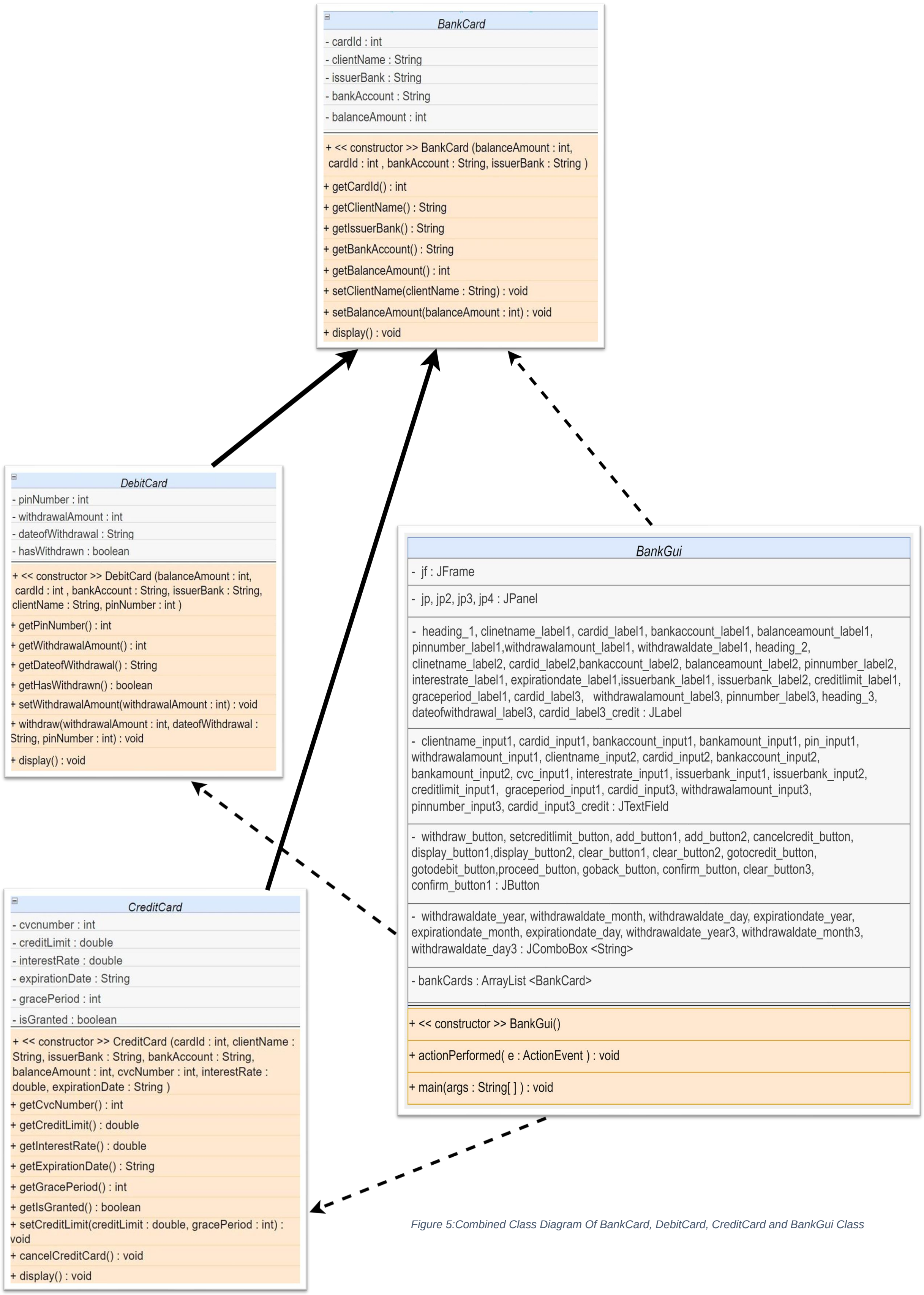

Figure 5:Combined Class Diagram Of BankCard, DebitCard, CreditCard and BankGui Class

3. PSEUDOCODE OF CLASS BankGui

IMPORT The Required Libraries

CREATE a class BankGui that implements ActionListener

DECLARE a JFrame component jf

DECLARE JPanel components jp, jp2, jp3, jp4

DECLARE JLabel components JLabel heading_1, clinetname_label1, cardid_label1, bankaccount_label1, balanceamount_label1, pinnumber_label1, withdrawalamount_label1, withdrawaldate_label1, heading_2, clinetname_label2, cardid_label2, bankaccount_label2, balanceamount_label2,pinnumber_label2,interestrate_label1,expirationdate_label1,issuerbank_label1,issuerbank_label2, creditlimit_label1,graceperiod_label1,cardid_label3,withdrawalamount_label3, pinnumber_label3, heading_3, dateofwithdrawal_label3, cardid_label3_credit

DECLARE JTextField components JTextField clientname_input1, cardid_input1, bankaccount_input1, bankamount_input1, pin_input1,withdrawalamount_input1, clientname_input2, cardid_input2, bankaccount_input2, bankamount_input2,cvc_input1, interestrate_input1, issuerbank_input1, issuerbank_input2,creditlimit_input1,graceperiod_input1,cardid_input3,withdrawalamount_input3, pinnumber_input3, cardid_input3_credit

DECLARE JComboBox components withdrawaldate_year, withdrawaldate_month, withdrawaldate_day, expirationdate_year, expirationdate_monthexpirationdate_day, withdrawaldate_year3, withdrawaldate_month3, withdrawaldate_day3

DECLARE JButton components withdraw_button, setcreditlimit_button, add_button1, add_button2, cancelcredit_button, display_button1, display_button2, clear_button1, clear_button2, gotocredit_button, gotodebit_button, proceed_button, goback_button, confirm_button, clear_button3, confirm_button1

CREATE an ArrayList of BankCard named bankCards

CREATE a constructor for the BankGui class

INITIALIZE JFrame and **SET** its Size and Position

INITIALIZE all Components of JPanel and **SET** their Size , Position and BackgroundColor

INITIALIZE all Components of JLabel and **SET** their Size, Position and Font

INITIALIZE all Components of JTextField and **SET** their Size, Position and Font

INITIALIZE all Components of JComboBox and **SET** their Size, Position and Font

INITIALIZE all Components of JButton **SET** their Size, Position, Font, BackColor and ForegroundColor

ADD JPanel jp, jp2, jp3 to JFrame Jf

ADD all Required Components of JLabel , JTextField ,Jbutton and JcomboBox to JPanel jp, jp2, JP3

ADD JPanel jp4 to JP2

REGISTER all Components of JButton to ActionListener

DISABLE the proceed_button and setcreditlimit_button

SET setFocusable to False to all Components of JButton

SET Layout null to JFrame and all Components of JPanel

SET Visible false for jf, jp2, jp3

SET Resizable false for jf

SET DefaultCloseOperation of JFrame as EXIT_ON_CLOSE

DO

CALL actionPerformed Method with parameter ActionEvent a

IF the source of the action is equal to gotocredit_button:

HIDE the Jpanel jp

SHOW the Jpanel jp2

CHANGE the position and size of the JFrame

SET the size of the JPanel jp2

END IF

ELSE IF the source of the action is equal to gotodebit_button:

SHOW the JPanel jp

HIDE the JPanel jp2

CHANGE the position and size of the Jframe

END IF

IF the source of the action is equal to withdraw_button:

HIDE the JPanel jp

SHOW the JPanel jp3

END IF

ELSE IF the source of the action is equal to goback_button:

SHOW the JPanel jp

HIDE the JPanel jp3

END IF

IF the source of the action is equal to add_button1:

GET the text from clientname_input1 and store it in clientnameInput

GET the text from Issuerbank_input1 and store it in issuerBankInput

GET the text from bankaccount_input1 and store it in bankAccount

GET the text from cardid_input1 and store it in cardIdInput

GET the text from pin_input1 and store it in pinInput

GET the text from bankamount_input1 and store it in balanceAmountInput

IF clientnameInput is empty OR issuerBankInput is empty OR bankAccount is empty OR cardIdInput is empty OR pinInput is empty OR balanceAmountInput is empty:

SET the minimum size of the OptionPane to 350x120

SET the font of the OptionPane message to Arial, bold, size 15

SHOW an error message dialog with the message "OPPS! Please fill in all the fields."

RETURN from the method

END IF

DECLARE a local variable cardId as Integer

DECLARE a local variable pinnumber as Integer

DECLARE a local variable issuerBank as String

DECLARE a local variable clientname as String

DECLARE a local variable balanceAmount as integer

TRY:

CONVERT cardIdInput to an integer and store it in cardId

CATCH the error that occur during conversion:

SHOW an error message dialog with the message "Card ID should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT pinInput to an integer and store it in pinnumber

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Pin Number should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT balanceAmountInput to an integer and store it in balanceAmount

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Balance Amount should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT clientnameInput to a double and store it in value

SHOW an error message dialog with the message "OPPS! Client name cannot be a number."

RETURN from the method

CATCH the error that occurs during conversion:

SET clientname to clientnameInput

END TRY

TRY:

CONVERT issuerBankInput to a double and store it in value

SHOW an error message dialog with the message "OPPS! Issuer Bank cannot be a number."

RETURN from the method

CATCH the error that occurs during conversion:

SET issuerBank to issuerBankInput

END TRY

DECLARE a local variable cardIdExists as Boolean and set it to false

FOR each card in bankCards:

IF card is an instance of DebitCard:

IF card's cardId is equal to local variable cardId:

SET cardIdExists to true

BREAK out of the loop

END IF

END IF

END FOR

IF cardIdExists is **false**:

CREATE a new DebitCard object with balanceAmount, cardId, bankAccount, issuerBank, clientname and pinnumber

ADD the new DebitCard object to bankCards

SHOW a message dialog with the message "DEBIT CARD ADDED SUCCESSFULLY!" and details about the card

END IF

ELSE:

SHOW an error message dialog with the message "Card ID already exists! Please enter a different Card ID."

END IF

IF the source of the action is equal to add_button2:

GET the text from clientname_input2 and store it in clientnameInput

GET the text from issuerbank_input2 and store it in issuerBankInput

GET the text from bankaccount_input2 and store it in bankAccount

GET the text from cardid_input2 and store it in cardIdInput

GET the text from bankamount_input2 and store it in balanceAmountInput

GET the text from cvc_input1 and store it in CVCnumberInput

GET the text from interestrate_input1 and store it in interestrateInput

GET the selected item from expirationdate_year and store it in year

GET the selected item from expirationdate_month and store it in month

GET the selected item from expirationdate_day and store it in day

CONCATENATE year, month, and day with "-" as a separator and store it in expirationDate

IF clientnameInput is empty OR issuerBankInput is empty OR bankAccount is empty OR cardIdInput is empty OR CVCnumberInput is empty OR balanceAmountInput is empty OR interestrateInput is empty OR year equals "Year" OR month equals "Month" OR day equals "Day":

SHOW an error message dialog with the message "OPPS! Please fill in all the fields."

RETURN from the method

END IF

DECLARE a local variable cardId as Integer

DECLARE a local variable balanceAmount as Integer

DECLARE a local variable cvcnumber as Integer

DECLARE a local variable interestrate as Double

DECLARE a local variable clientname as String

DECLARE a local variable issuerBank as String

TRY:

CONVERT balanceAmountInput to an integer and store it in balanceAmount

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Balance Amount should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT cardIdInput to an integer and store it in cardId

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Card ID should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT CVCnumberInput to an integer and store it in cvcnumber

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "CVC number should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT interestrateInput to a double and store it in interestrate

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Interest Rate should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT clientnameInput to a double and store it in value

SHOW an error message dialog with the message "OPPS! Client name cannot be a number."

RETURN from the method

CATCH the error that occurs during conversion:

SET clientname to clientnameInput

END TRY

TRY:

CONVERT issuerBankInput to a double and store it in value

SHOW an error message dialog with the message "OPPS! Issuer Bank name cannot be a number."

RETURN from the method

CATCH the error that occurs during conversion:

SET issuerBank to issuerBankInput

END TRY

DECLARE a local variable cardIdExists as Boolean and set it to false

FOR each card in bankCards:

IF card is an instance of CreditCard:

IF card's cardId is equal to local variable cardId:

SET cardIdExists to true

BREAK out of the loop

END IF

END IF

END FOR

IF cardIdExists is **false**:

CREATE a new CreditCard object with cardId, clientname, issuerBank, bankAccount, balanceAmount, cvcnumber, interestrate and expirationDate

ADD the new CreditCard object to bankCards

SHOW a message dialog with the message "CREDIT CARD ADDED SUCCESSFULLY!" and details about the card

END IF

ELSE:

SHOW an error message dialog with the message "Card ID already exists! Please enter a different Card ID."

ELSE IF

IF the source of the action is equal to clear_button1:

SET the text of clientname_input1 to an empty string

SET the text of issuerbank_input1 to an empty string

SET the text of bankaccount_input1 to an empty string

SET the text of cardid_input1 to an empty string

SET the text of pin_input1 to an empty string

SET the text of bankamount_input1 to an empty string

END IF

IF the source of the action is equal to clear_button3:

SET the text of withdrawalamount_input3 to an empty string

SET the text of cardid_input3 to an empty string

SET the text of pinnumber_input3 to an empty string

SET the selected index of withdrawaldate_year3 to 0

SET the selected index of withdrawaldate_month3 to 0

SET the selected index of withdrawaldate_day3 to 0

END IF

IF the source of the action is equal to clear_button2:

SET the text of clientname_input2 to an empty string

SET the text of issuerbank_input2 to an empty string

SET the text of bankaccount_input2 to an empty string

SET the text of cardid_input2 to an empty string

SET the text of pin_input1 to an empty string

SET the text of bankamount_input2 to an empty string

SET the text of cvc_input1 to an empty string

SET the text of interestrate_input1 to an empty string

SET the text of graceperiod_input1 to an empty string

SET the text of creditlimit_input1 to an empty string

SET the text of cardid_input3_credit to an empty string

SET the selected index of expirationdate_year to 0

SET the selected index of expirationdate_month to 0

SET the selected index of expirationdate_day to 0

END IF

IF the source of the action is equal to confirm_button:

GET the text from cardid_input3 and store it in cardIdInput

GET the text from withdrawalamount_input3 and store it in withdrawalamountInput

GET the text from pinnumber_input3 and store it in pinnumberInput

GET the selected item from withdrawaldate_year3 and store it in year

GET the selected item from withdrawaldate_month3 and store it in month

GET the selected item from withdrawaldate_day3 and store it in day

CONCATENATE year, month, and day with "-" as a separator and store it in withdrawaldate

IF cardIdInput is empty OR withdrawalamountInput is empty OR pinnumberInput is empty OR year equals "Year" OR month equals "Month" OR day equals "Day":

SHOW an error message dialog with the message "OPPS! Please fill in all the fields."

RETURN from the method

END IF

DECLARE a local variable cardId as Integer

DECLARE a local variable withdrawalamount as Integer

DECLARE a local variable pinnumber as Integer

TRY:

CONVERT cardIdInput to an integer and store it in cardId

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Card ID should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT withdrawalamountInput to an integer and store it in withdrawalamount

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Withdrawal Amount should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT pinnumberInput to an integer and store it in pinnumber

CATCH the error that occurs during conversion:

SET the minimum size of the OptionPane to 400x130

SET the font of the OptionPane message to Arial, bold, size 15

SHOW an error message dialog with the message "Pin Number should be a valid number."

RETURN from the method

END TRY

DECLARE a local variable cardIdExists as Boolean and set it to false

FOR each card in bankCards:

IF card is an instance of DebitCard:

IF card's cardId is equal to local variable cardId:

SHOW a message dialog with the message "CONFIRM" and details about the withdrawal

ENABLE proceed_button

SET cardIdExists to true

BREAK out of the loop

END IF

END IF

END FOR

IF cardIdExists is false:

SHOW a message dialog with the message "Invalid Card ID! Please Enter Your Correct Card Id"

END IF

END IF

IF the source of the action is equal to proceed_button:

CONVERT cardid_input3 to an integer and store it in cardId

CONVERT withdrawalamount_input3 to an integer and store it in withdrawalamount

GET the selected item from withdrawaldate_year3 and store it in year

GET the selected item from withdrawaldate_month3 and store it in month

GET the selected item from withdrawaldate_day3 and store it in day

CONCATENATE year, month, and day with "-" as a separator and store it in withdrawaldate

CONVERT pinnumber_input3 to an integer and store it in pinnumber

FOR each card in bankCards:

IF card is an instance of DebitCard:

IF card's cardId is equal to local variable cardId:

CAST card to a DebitCard object and **CALL** withdraw method on card with withdrawalamount, withdrawaldate, and pinnumber as arguments

SHOW a message dialog with the message "Withdraw Process Is Successfully Done!"

DISABLE proceed_button

BREAK out of the loop

END IF

END IF

END FOR

END IF

IF the source of the action is equal to confirm_button1:

GET the text from cardid_input3_credit and store it in cardIdInput

GET the text from graceperiod_input1 and store it in graceperiodInput

GET the text from creditlimit_input1 and store it in creditlimitInput

IF cardIdInput is empty OR graceperiodInput is empty OR creditlimitInput is empty:

SHOW an error message dialog with the message "OPPS! Please fill in all the fields To Set Credit Limit"

RETURN from the method

END IF

DECLARE a local variable cardId as Integer

DECLARE a local variable graceperiod as Integer

DECLARE a local variable creditlimit as Double

TRY:

CONVERT cardIdInput to an integer and store it in cardId

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Card ID should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT graceperiodInput to an integer and store it in graceperiod

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Grace Period should be a valid number."

RETURN from the method

END TRY

TRY:

CONVERT creditlimitInput to a double and store it in creditlimit

CATCH the error that occurs during conversion:

SHOW an error message dialog with the message "Credit Limit should be a valid number."

RETURN from the method

END TRY

DECLARE a local variable cardIdExists as Boolean and set it to false

FOR each card in bankCards:

```
        IF card is an instance of CreditCard:
            IF card's cardId is equal to local variable cardId:
                SHOW a message dialog with the message "CONFIRM" and details about the credit limit and grace period
                ENABLE setcreditlimit_button
                SET cardIdExists to true
                BREAK out of the loop
            END IF
        END IF
    END FOR

    IF cardIdExists is false:
        SHOW a message dialog with the message "Invalid Card ID! Please Enter Your Correct Card Id To Set Credit Limit."
    END IF
END IF

IF the source of the action is the cancelcredit_button:
    READ cardIdInput from cardid_input3_credit
    IF cardIdInput is empty:
        DISPLAY error message "OPPS! Please Enter The Card Id field."
        RETURN
    END IF

  DECLARE a local variable cardId as Integer

  TRY:
        CONVERT cardIdInput to an integer and store it in cardId
  CATCH NumberFormatException:
```

DISPLAY error message "Card ID should be a valid number."

RETURN

END TRY

DECLARE a local variable cardIdExists as Boolean and set it to false

FOR each card in bankCards:

IF card is an instance of CreditCard:

IF card's cardId is equal to cardId:

CAST card to CreditCard object and call its cancelCreditCard method

DISPLAY a success message with values

SET cardIdExists to **true**

BREAK out of the loop

END IF

END IF

END FOR

IF cardIdExists is false:

DISPLAY an error message with the text "Invalid Card ID! Please Enter Your Correct Card Id"

END IF

END IF

IF the source of the action is the setcreditlimit_button:

CONVERT the text from cardid_input3_credit to an integer and store it in cardId

CONVERT the text from creditlimit_input1 to a double and store it in creditlimit

CONVERT the text from graceperiod_input1 to an integer and store it in graceperiod

DECLARE a local variable cardIdExists as Boolean and set it to false

FOR each card in bankCards:

IF card is an instance of CreditCard:

IF card's cardId is equal to cardId:

CAST card to CreditCard object and call its setCreditLimit method with creditlimit and graceperiod as arguments

SET cardIdExists to true

DISABLE the setcreditlimit_button

DISPLAY a success message with the text "Credit Limit Is Set Successfully!"

BREAK out of the loop

END IF

END IF

END FOR

END IF

IF the source of the action is the display_button1:

FOR each card in bankCards:

IF card is an instance of DebitCard

PRINT "CARD TYPE: DEBIT CARD"

CAST card to DebitCard object and call its display method

ENDIF

ENDFOR

ENDIF

IF the source of the event is the display_button2:

FOR each card in bankCards:

IF card is an instance of CreditCard:

```
                PRINT "CARD TYPE: CREDIT CARD"
                CAST card to CreditCard object and call its display method
            ENDIF
        ENDFOR
    ENDIF
END DO

DECLARE a main method with an array of Strings as its argument:
    CREATE a new instance of BankGui
```

4. METHOD DESCRIPTION OF ALL THE BUTTONS

4.1) METHOD DESCRIPTION OF BUTTONS OF DEBIT CARD PANEL

4.1.1) Add Debit Card

When the user clicks on the **"Add Debit Card"** button, it triggers the "Action Performed" method. Within this method, an instance of the DebitCard class is created and added to an ArrayList of BankCard.

4.1.2) Clear

When the user clicks on the **"Clear"** button, it triggers the "Action Performed" method. This method resets all the input fields in the Debit Card panel to empty values.

4.1.3) Display

When the user clicks on the **"Display"** button, it triggers the "Action Performed" method. Within this method, the display() method of the DebitCard class is called using the card object. This displays the information of the Debit Card.

4.1.4) WithDraw

When the user clicks on the "**Withdraw"** button, it triggers an "Action Performed" method which redirects the user to withdraw panel

4.1.5) Go To Credit Card

When the user clicks on the "**Go To Credit Card"** button, it triggers an "Action Performed" method which redirects the user to Credit Card panel

4.2) METHOD DESCRIPTION OF BUTTONS OF CREDIT CARD PANEL

4.2.1) Add Credit Card

When the user clicks on + **"Add Credit Card"** button, it triggers the "Action Performed" method. Within this method, an instance of the CreditCard class is created and added to an ArrayList of BankCard.

4.2.2) Clear

When the user clicks on the **"Clear"** button, it triggers the "Action Performed" method. This method resets all the input fields in the Credit Card panel to empty values.

4.2.3) Display

When the user clicks on the **"Display"** button, it triggers the "Action Performed" method. Within this method, the display() method of the CreditCard class is called using the card object. This displays the information of the Credit card.

4.2.4) Confirm

When the user clicks on the **“Confirm”** button, it triggers the “Action Performed” method. This method checks if the input given by the user to set the Credit Limit is valid or not. If the input is valid, the “Set Credit Limit” button is enabled

4.2.5) Set Credit Limit

When the user clicks on the **“Set Credit Limit”** button, it triggers the “Action Performed” method. This method calls the setCreditLimit method on the CreditCard object, passing in the creditlimit and graceperiod values as arguments. This sets the credit limit and grace period for the credit card.

4.2.6) Cancel Credit Card

When the user clicks on the **“Cancel Credit Card”** button, it triggers the “Action Performed” method. This method calls the cancelCreditCard() method on the CreditCard object. This sets values of the credit limit, grace period and CVC number to zero.

4.2.7) Go To Debit Card

When the user clicks on the “**Go To Debit Card”** button, it triggers an “Action Performed” method which redirects the user to Debit Card panel

4.3) METHOD DESCRIPTION OF BUTTONS OF WITHDRAW PANEL

4.3.1) Confirm

When the user clicks on the **“Confirm”** button, it triggers the “Action Performed” method. This method checks if the input given by the user to withdraw is valid or not. If the input is valid, the **“PROCEED”** button is enabled.

4.3.2) Clear

When the user clicks on the **“Clear”** button, it triggers the “Action Performed” method. This method resets all the input fields in the WithDraw panel to empty values

4.3.3) PROCEED

When the **"PROCEED"** button is clicked, it triggers the "Action Performed" method which then calls the "withdraw" method on a DebitCard object with the withdrawal amount, date, and PIN number as arguments. As a result, specified amount of money will be withdrawn from Debit Card.

4.3.4) Go Back

When the user clicks on the **“Go Back”** button, it triggers an “Action Performed” method which redirects the user to Debit Card panel

5. Testing(Inspection)

5.1 Test 1 – To Compile and Run Program using Command Prompt

Test NO.	1
Objective:	To Compile and Run Program using Command Prompt.
Action:	➡ The Command Prompt was opened and navigated to the directory containing the BankGui class and other required classes. ➡ The version of Java installed on the system was checked by entering java --version in the Command Prompt. ➡ The BankGui class was compiled by entering javac BankGui.java in the Command Prompt. ➡ The BankGui class was executed by entering java BankGui in the Command Prompt.
Expected Result:	The Command Prompt would navigate to the directory with the BankGui class. The installed Java version would be displayed. The BankGui class would compile without errors. The BankGui program would run and display its interface.
Actual Result:	The Command Prompt successfully navigated to the directory with the BankGui class. The installed Java version was displayed as expected. The BankGui class was compiled without any errors. The BankGui program ran and its interface was displayed correctly.
Conclusion:	The test is successful.

Table 1: To Compile and Run Program using Command Prompt

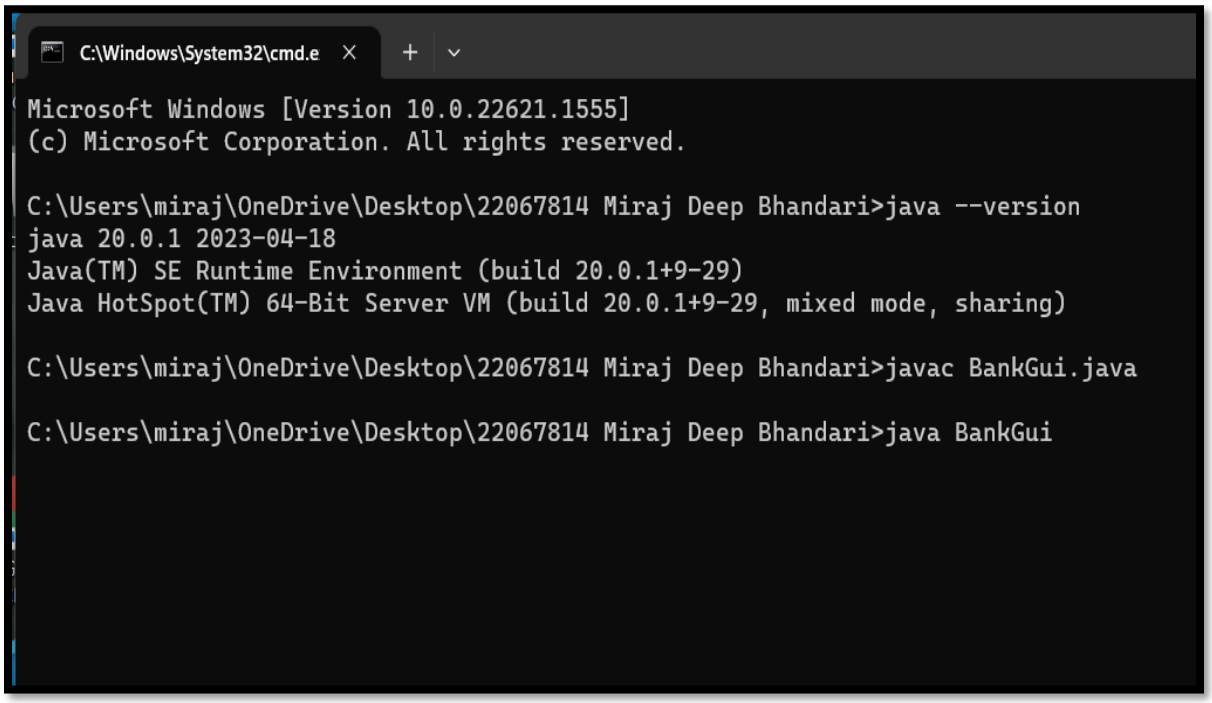

Figure 6: Screenshot of opening command promt and giving values to compile and run program

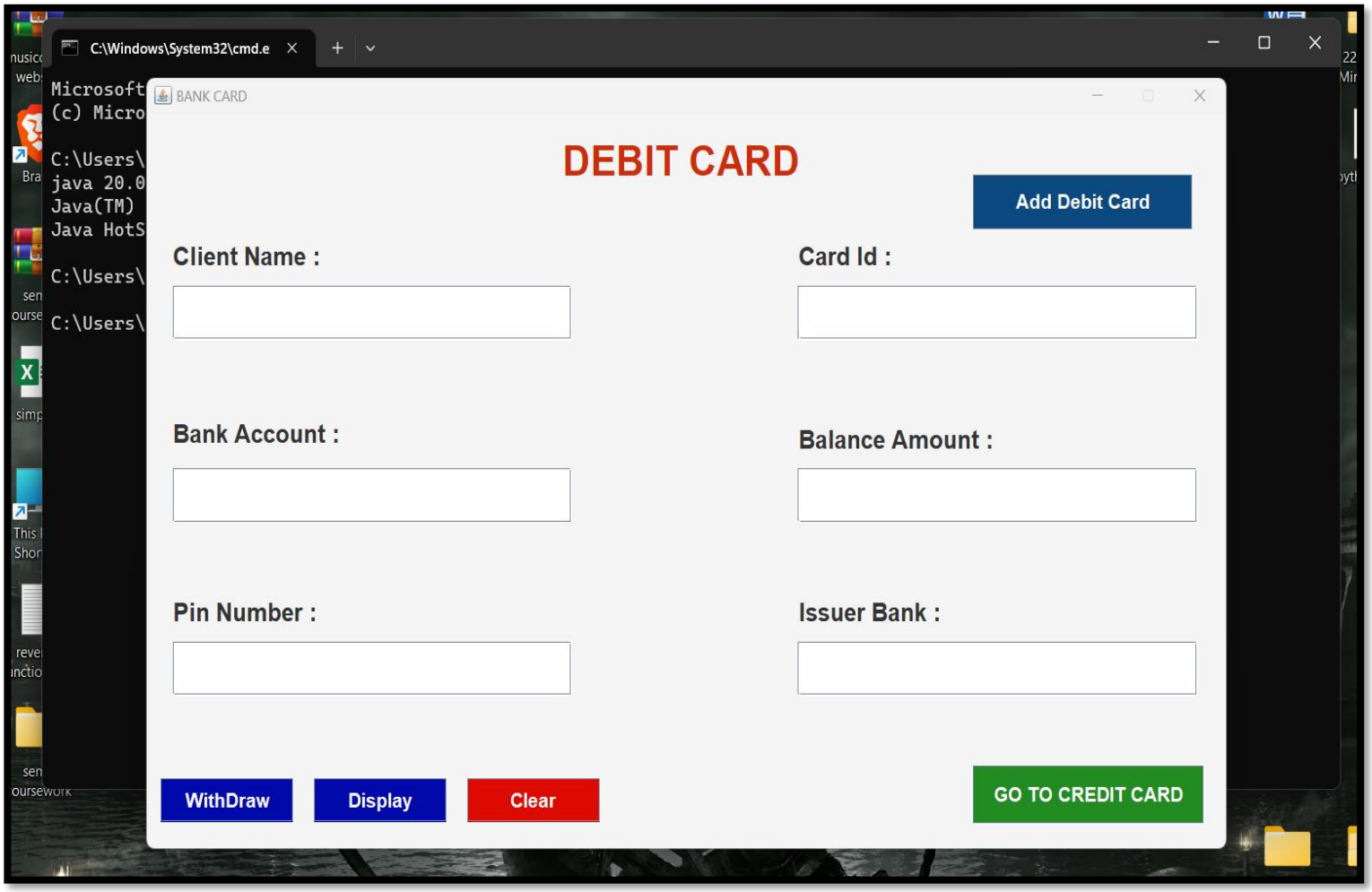

Figure 7: Screenshot of successful run of BankGui Class after compilation from command prompt

5.2 Test 2 – To Show Evidences of Functionality of Different Debit Card and Credit Card Buttons

5.2.1 - To Show Evidence of Working of Add DebitCard

Test NO.	2.1
Objective:	To Show Evidence of Working of Add DebitCard.
Action:	➡ The BankGui Class was compiled and executed. ➡ The input fields for Client Name, Bank Account, Pin Number, Card Id, Balance Amount, and Issuer Bank were filled with the values “miraj bhandari”, “0mir123”, “1234”, “4321”, “10000”, and “everest” respectively. ➡ The “Add Debit Card” button was clicked. ➡ The “Add Debit Card” button was clicked again with the same value for Card Id but different values for the other input fields.
Expected Result:	When the “Add Debit Card” button is clicked for the first time, the debit card will be added successfully. If the button would be clicked again with the same Card Id, an error message would be displayed.

Actual Result:	The “Add Debit Card” button was clicked and the debit card was added successfully. When the button was clicked again with the same Card Id, an error message was displayed.
Conclusion:	The test is successful.

Table 2:To Show Evidence of Working of Add DebitCard

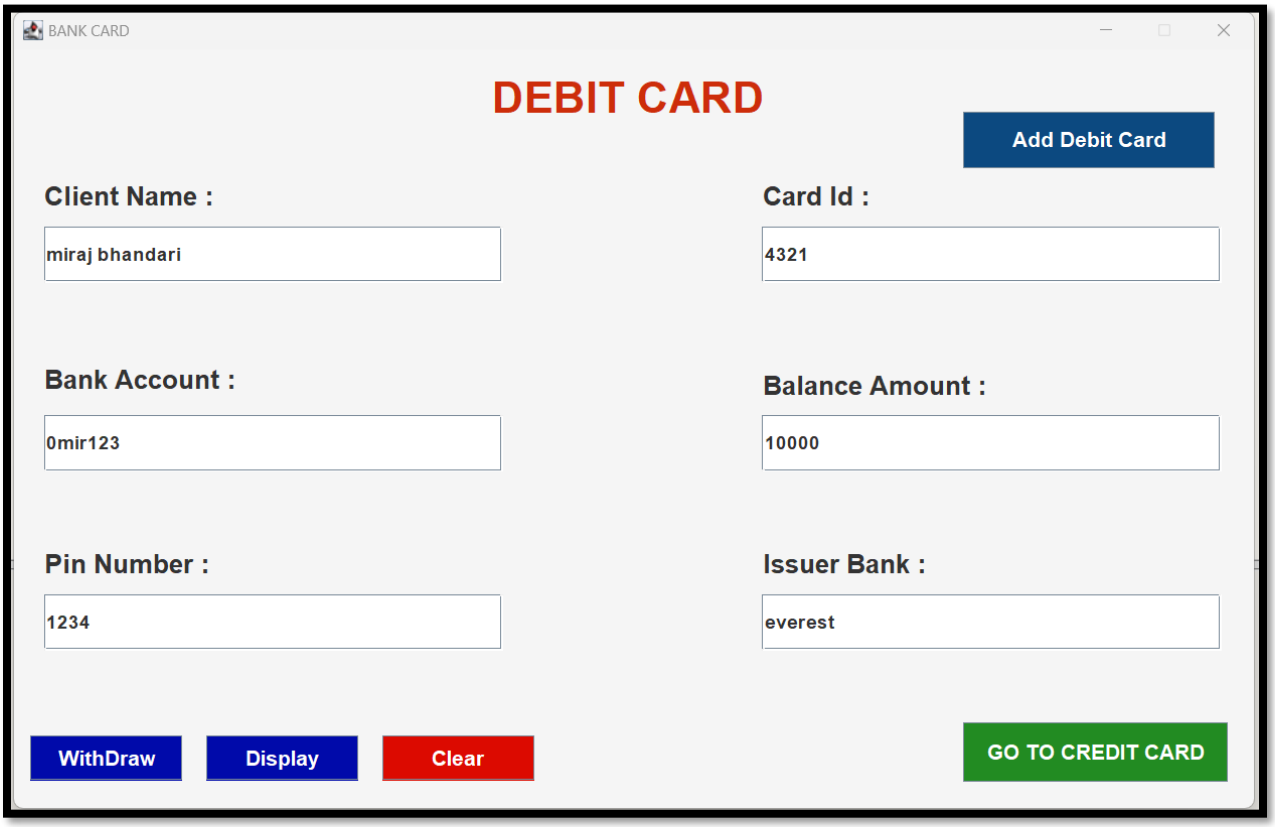

Figure 8: Screenshot of filling values in Input field before clicking Add Debit Card

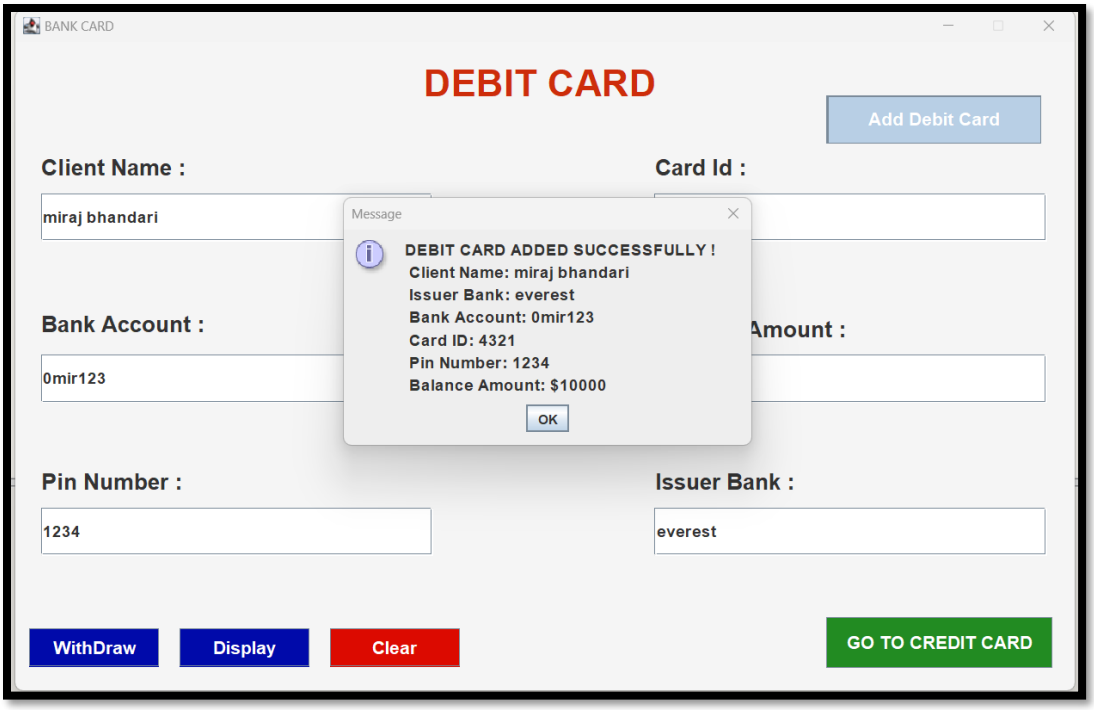

Figure 9:Screenshot of Adding Debit Card after filling Input fields

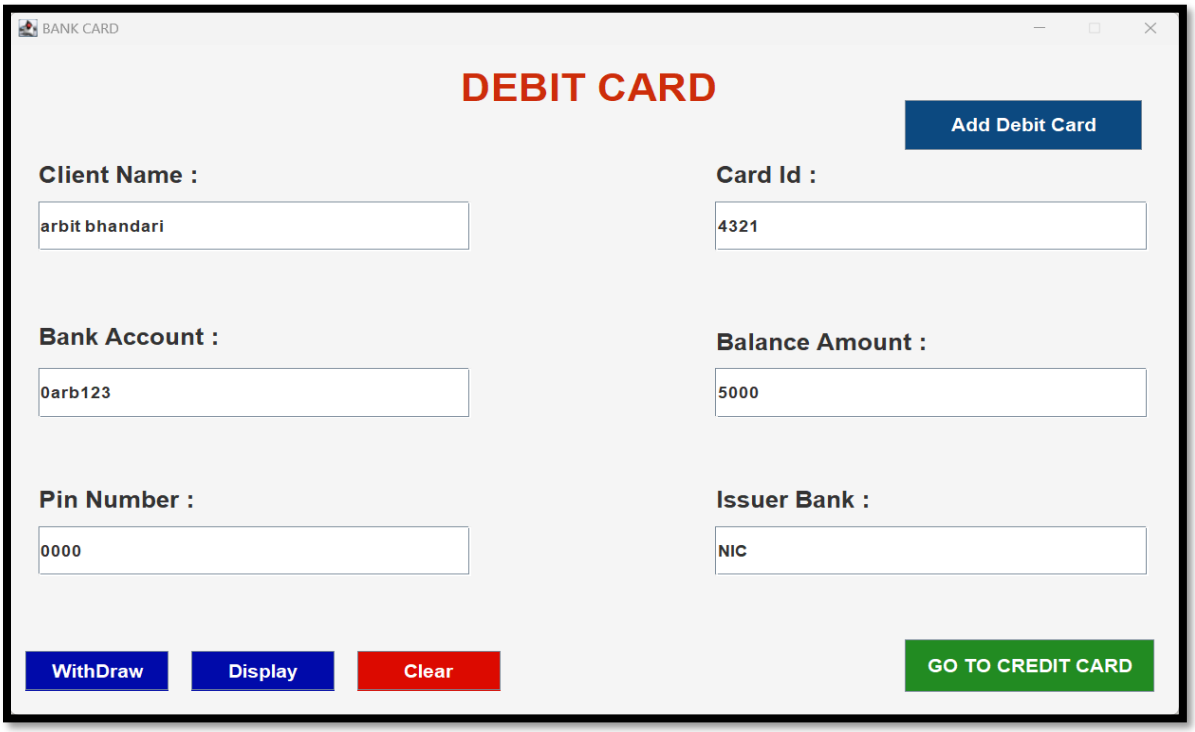

Figure 10:Screenshot of giving already Exist value for card id and Different values for other Input fields Before clicking Add Debit Card Button

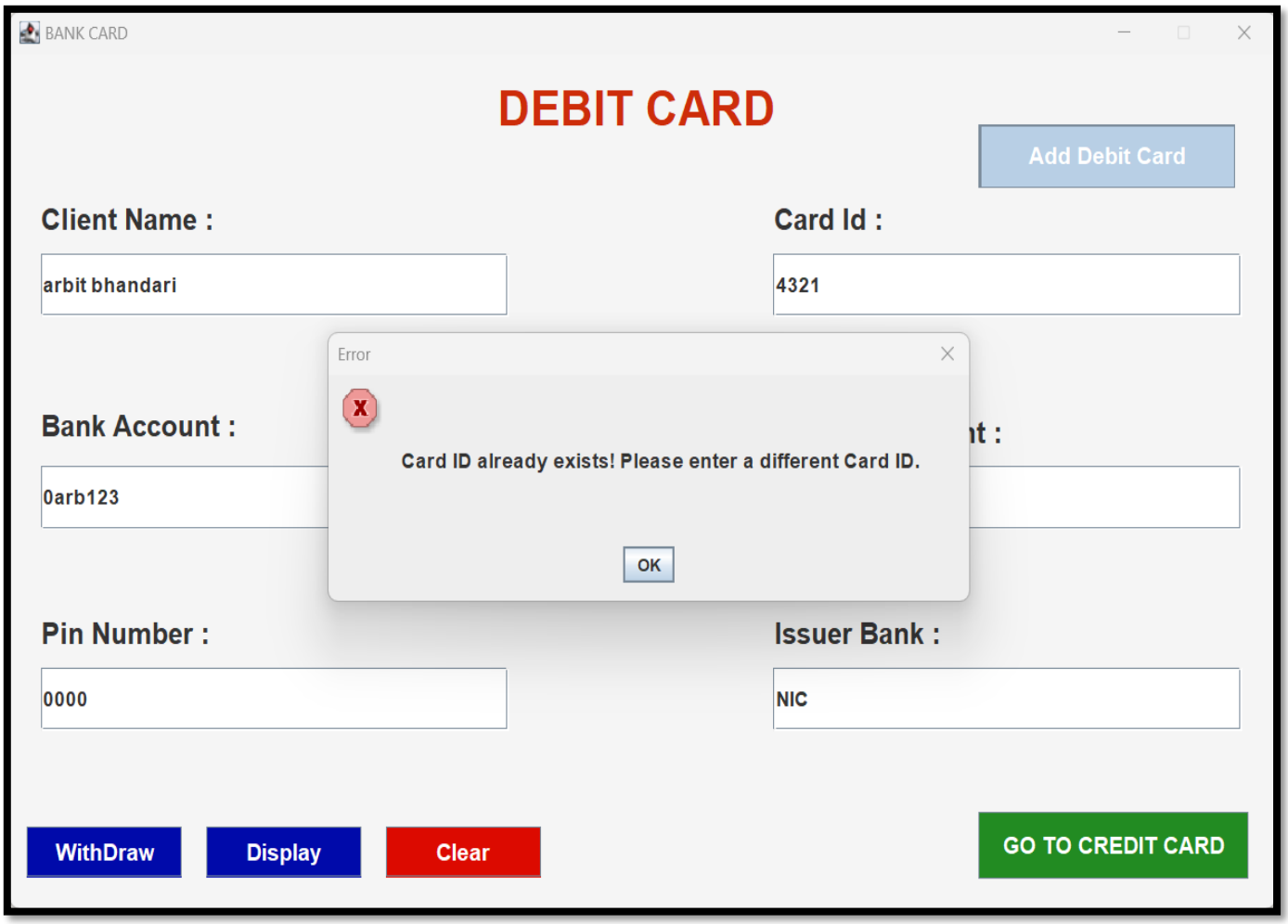

Figure 11: Screenshot after Clicking Add Debit Card button when already exist card id is given

5.2.2 - To Show Evidence of Working of Add CreditCard

Test NO.	2.2
Objective:	To Show Evidence of Working of Add Credit Card.
Action:	➡ The BankGui Class was compiled and executed and Go to Credit Card Button is clicked. ➡ The input fields for Client Name, Bank Account, CVC Number, Expiration Date, Card Id, Balance Amount, Interest Rate and Issuer Bank were filled with the values “rashmi bhandari”, “0ras123”, “123”, “2025-01-01”, “4444”, “10000”, ”5” and “everesr” respectively. ➡ The “Add Credit Card” button was clicked. ➡ The “Add Credit Card” button was clicked again with the same value for Card Id but different values for the other input fields.
Expected Result:	When the “Add Credit Card” button is clicked for the first time, the Credit card will be added successfully. If the button would be clicked again with the same Card Id, an error message would be displayed.

Actual Result:	The “Add Credit Card” button was clicked and the Credit card was added successfully. When the button was clicked again with the same Card Id, an error message was displayed.
Conclusion:	The test is successful.

Table 3:To Show Evidence of Working of Add CreditCard

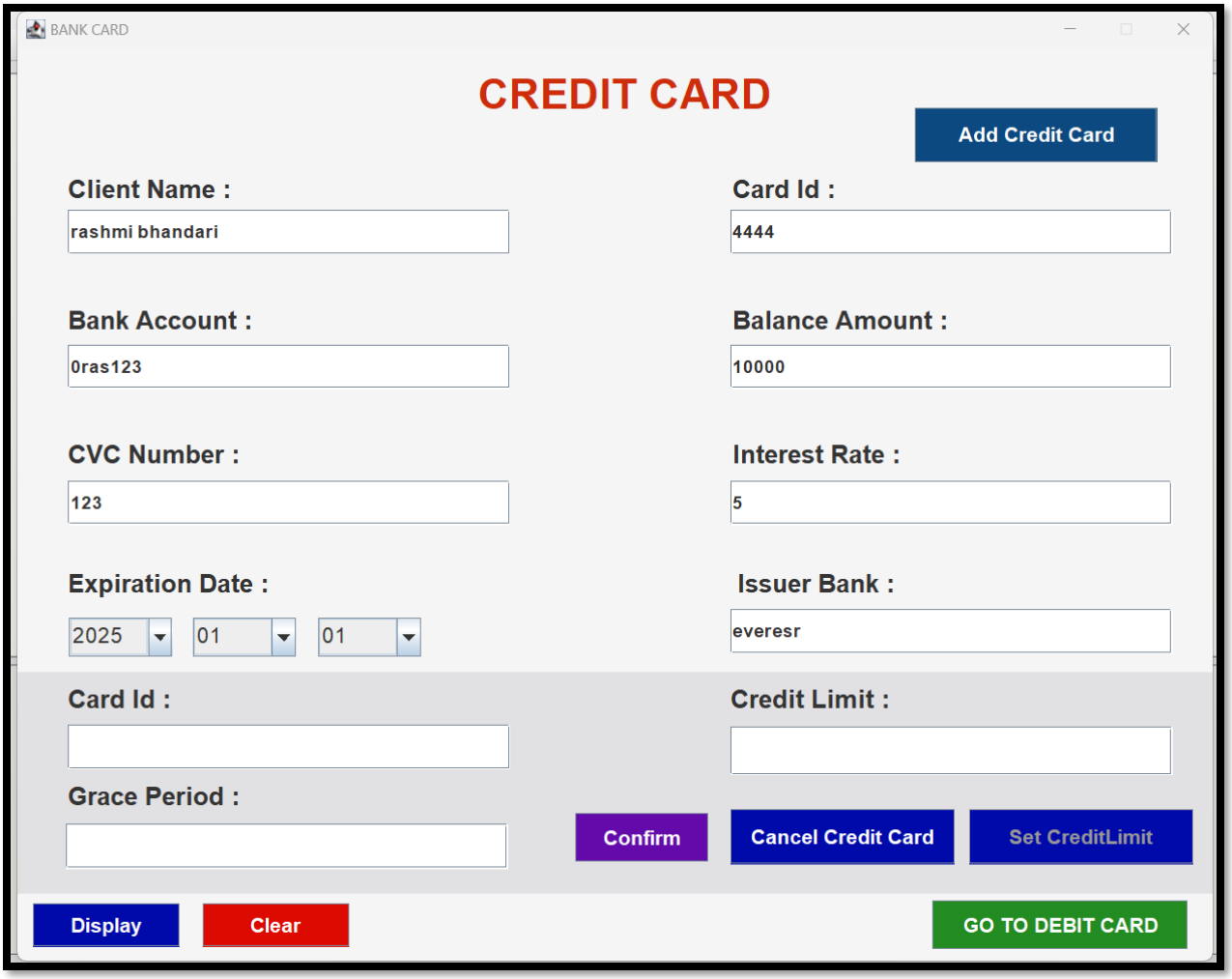

Figure 12:Screenshot of filling values in Input field before clicking Add Credit Card

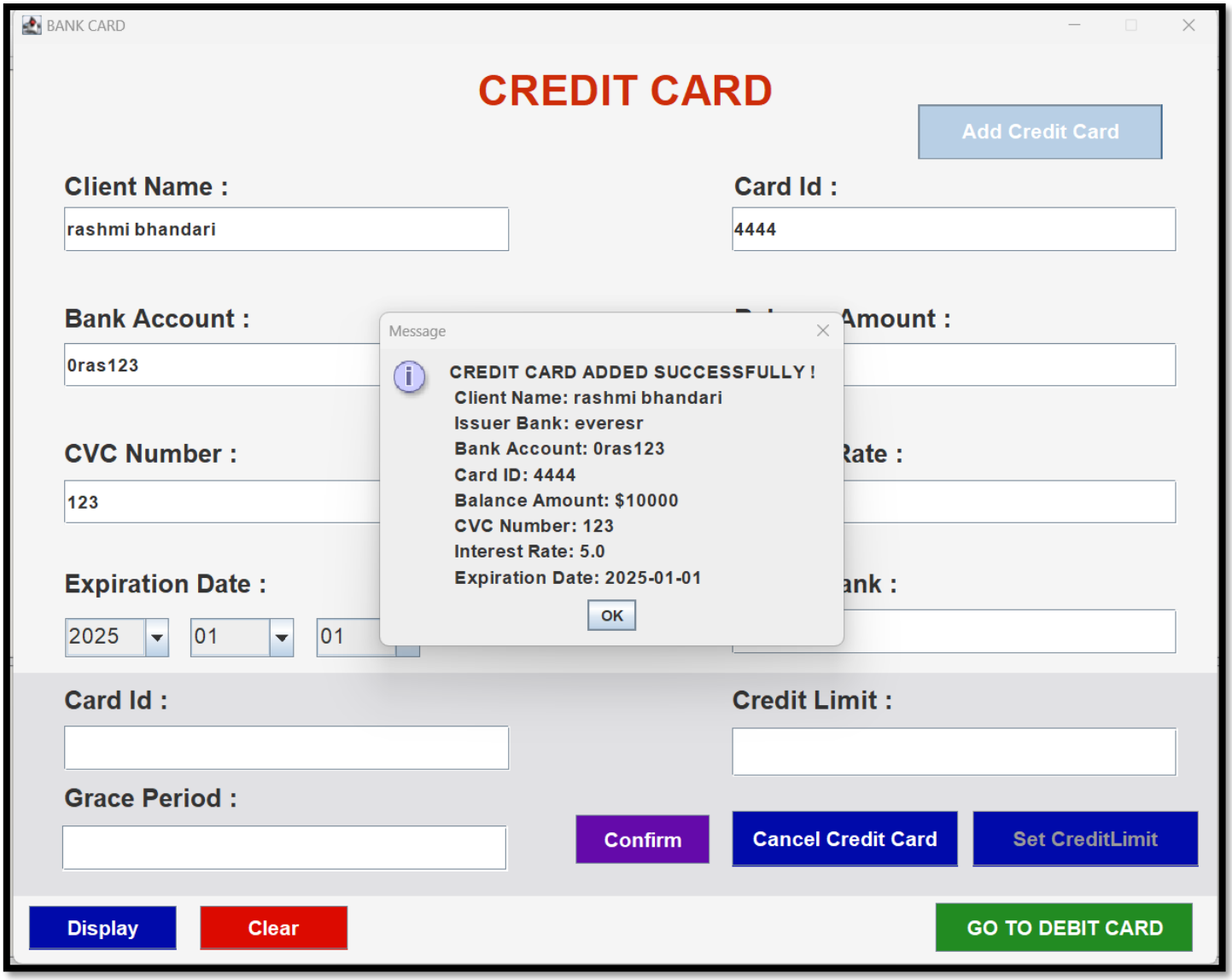

Figure 13:Screenshot of Adding Credit Card after filling Input fields

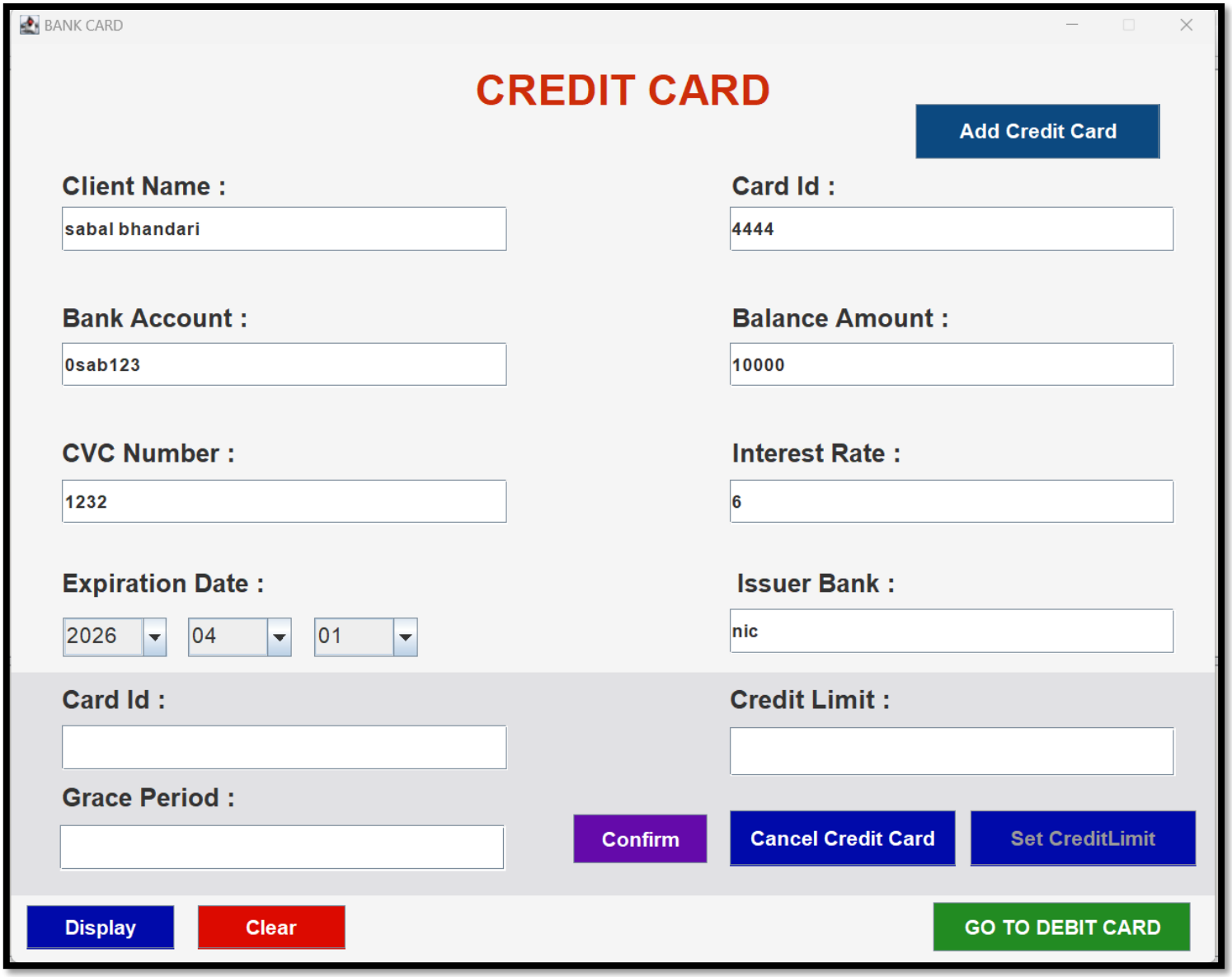

Figure 14:Screenshot of giving already Exist value for card id and Different values for other Input fields Before clicking Add Credit Card Button

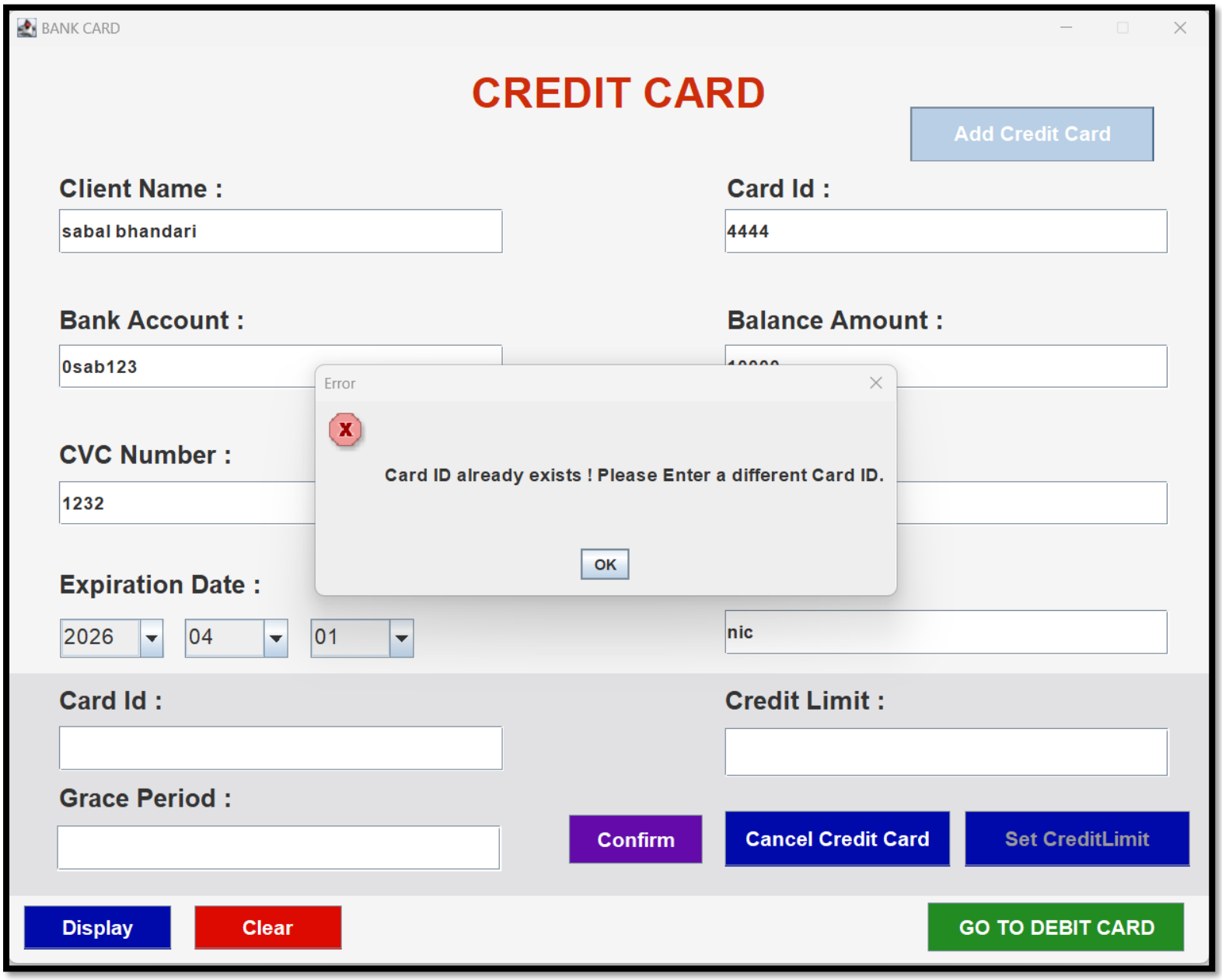

Figure 15:Screenshot after Clicking Add Credit Card button when already exist card id is given

5.2.3 - To Show Evidence of Withdraw amount from Debit card

Test NO.	2.3
Objective:	To Show Evidence of Withdraw amount from Debit card.
Action:	➡ The With Draw Button was clicked in Debit Card. ➡ The input fields for Card Id, Date Of Withdrawal, Withdrawal Amount, and Pin Number were filled with the values “4321”, “2023-04-05”, “2000”, and “1234” respectively. ➡ The “CONFIRM” button was clicked after filling all Input fields. ➡ The “PROCEED” button was clicked to withdraw the withdrawal amount.
Expected Result:	withdrawal amount of 2000 will be successfully withdrawn from the account associated with the Card Id “4321”.
Actual Result:	withdrawal amount of 2000 was successfully withdrawn from the account associated with the Card Id “4321”.
Conclusion:	The test is successful.

Table 4: To Show Evidence of Withdraw amount from Debit card

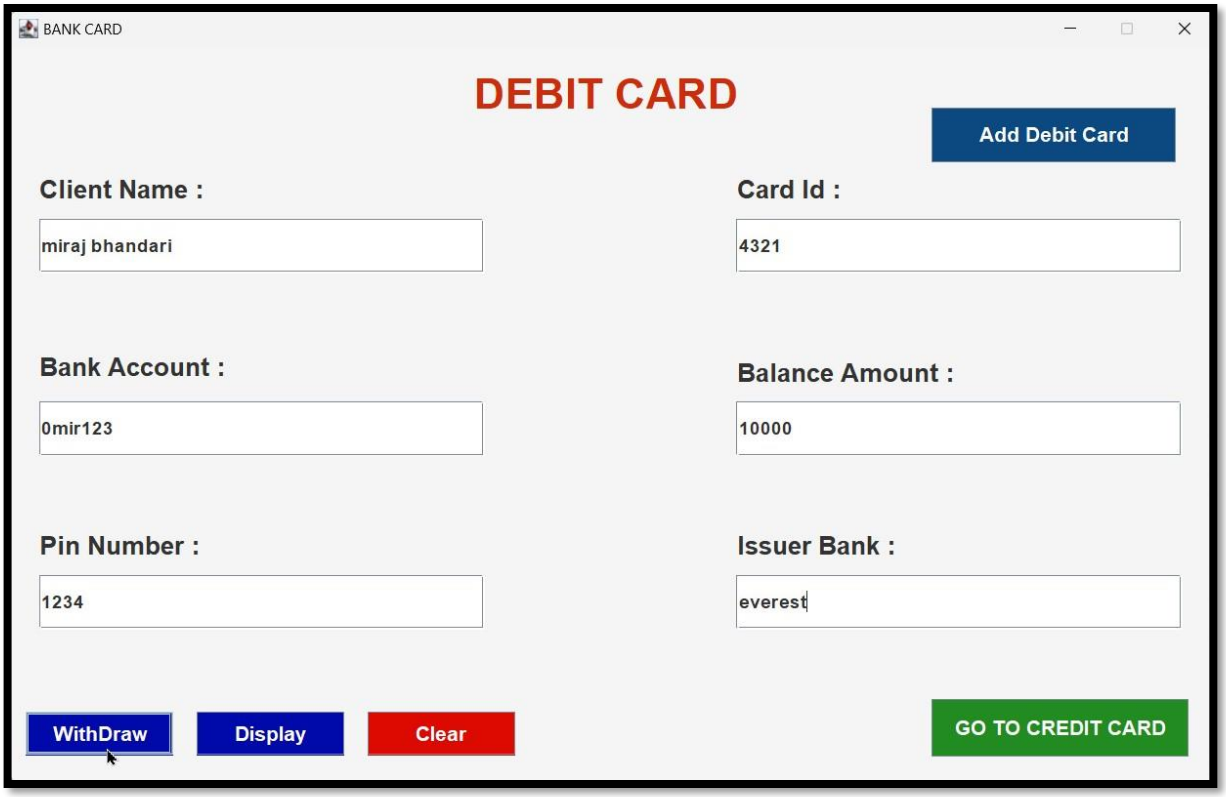

Figure 16:Screenshot of clicking WithDraw button in Debit Card

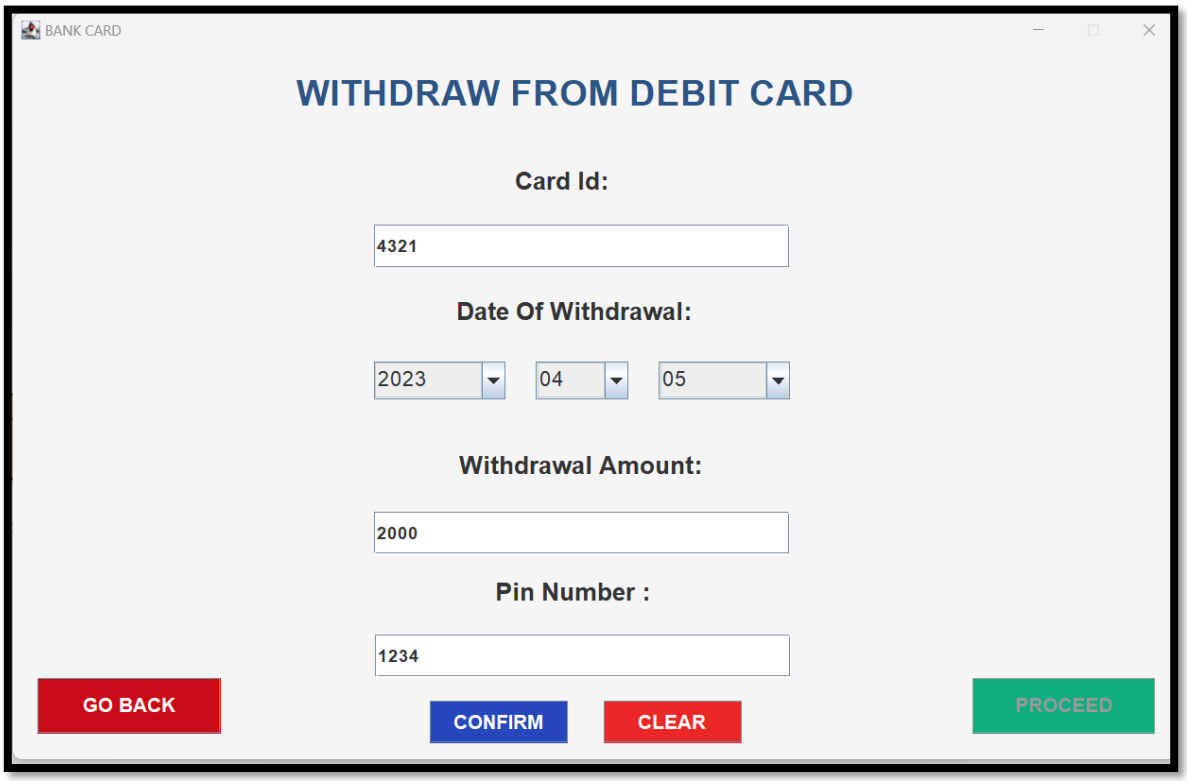

Figure 17:ScreenShot of filling details to WithDraw from Debit Card

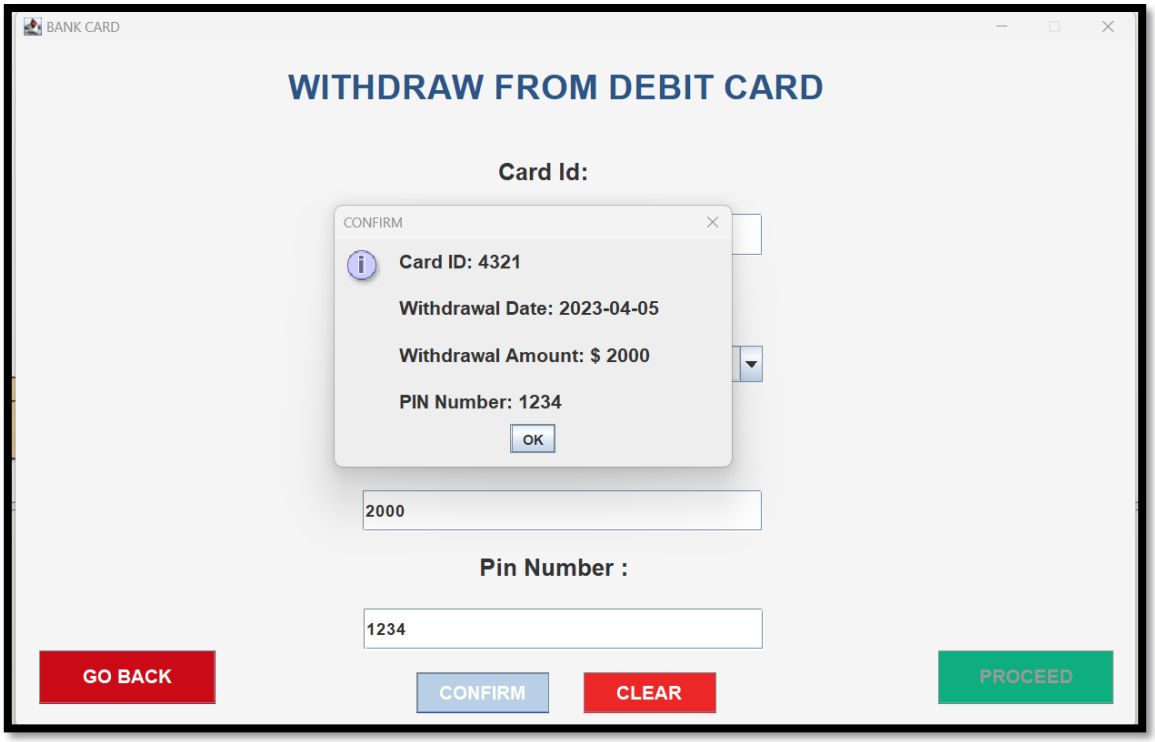

Figure 18:Screenshot after Clicking Confirm Button after filling Details in Input fileds to withdraw

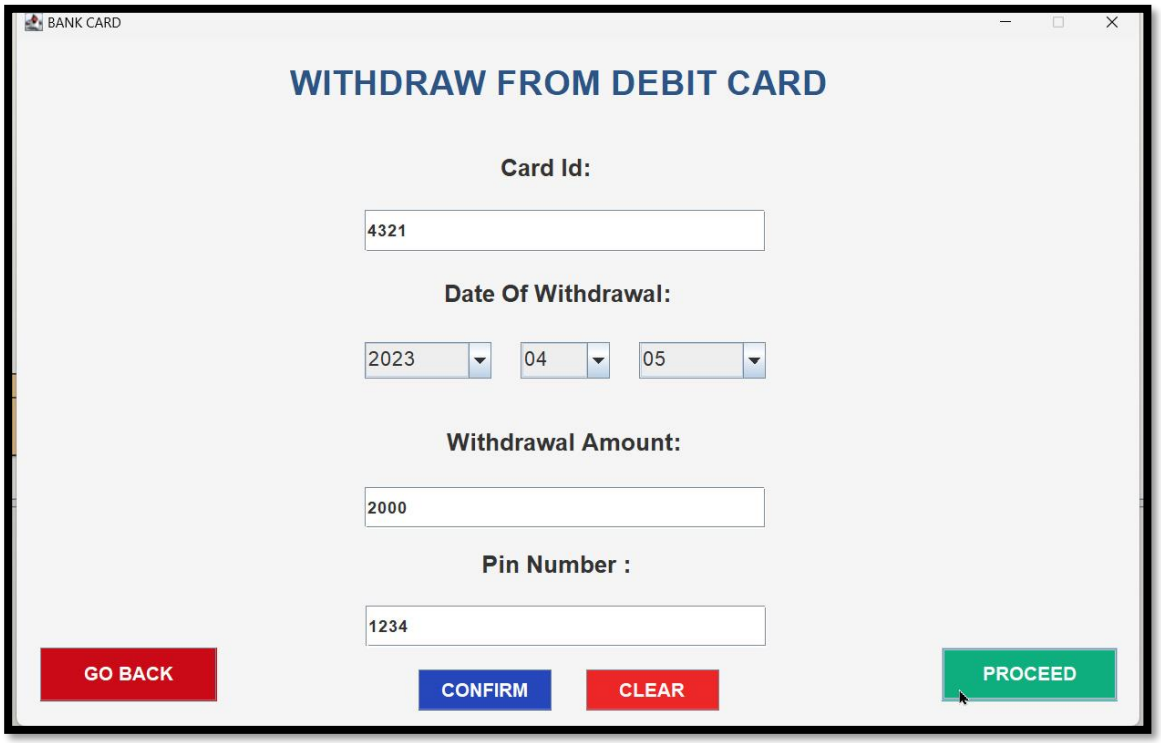

Figure 19: Screenshot of clicking PROCEED Button

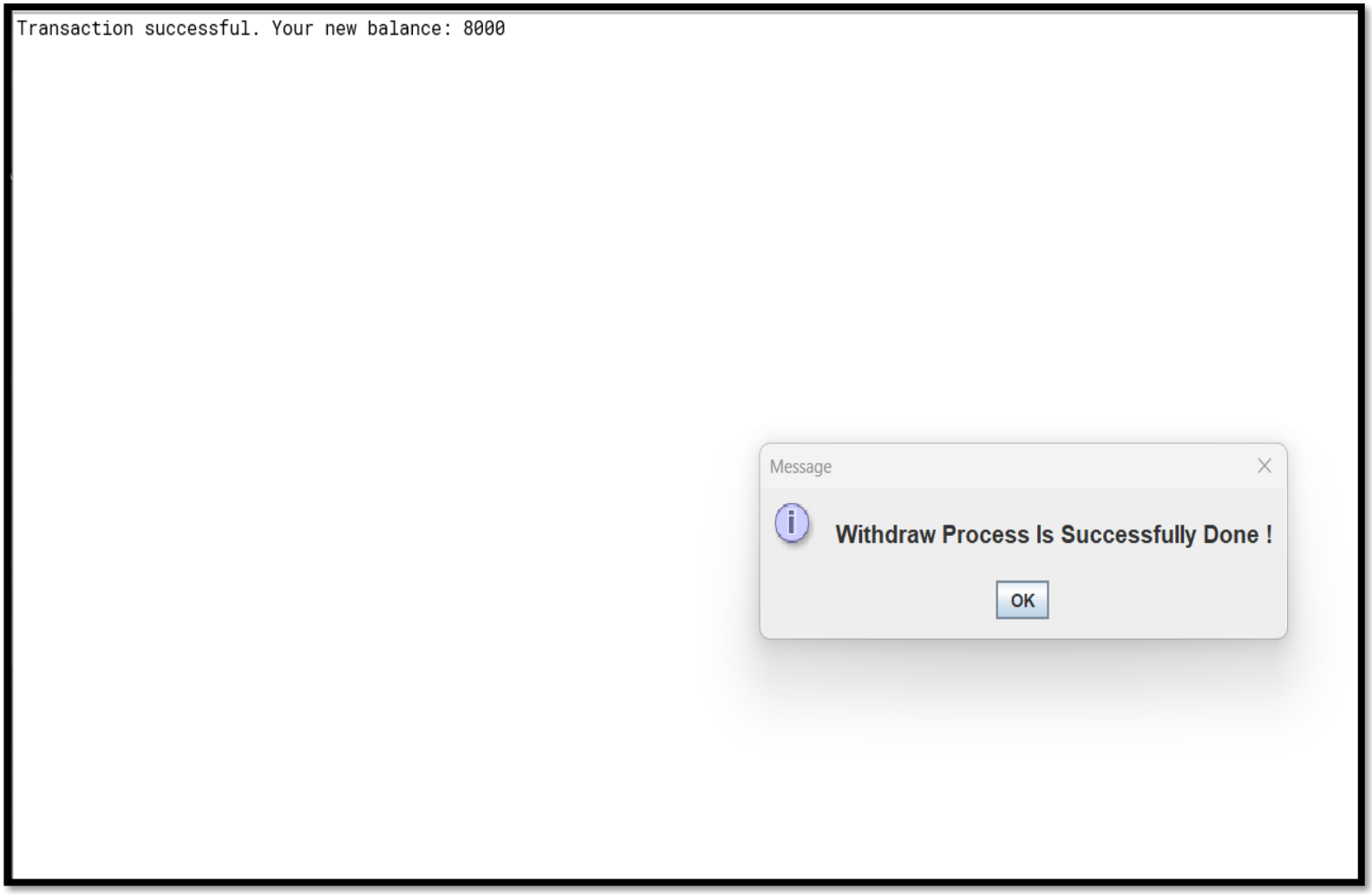

Figure 20: Screenshot of Successful withdraw from Debit Card

5.2.4 - To Show Evidence of Set the credit limit

Test NO.	2.4
Objective:	To Show Evidence of Set the credit limit
Action:	➡ The input fields for Card Id, Credit Limit and Grace Period were filled with the values “4444”, “5000”, “25” respectively in Credit Card. ➡ The “CONFIRM” Button was clicked. ➡ The “Set Credit Limit” button was clicked after clicking “CONFIRM” Button
Expected Result:	The credit limit for the card with Id “4444” would be successfully set to “5000” with a grace period of “25” days.
Actual Result:	The credit limit for the card with Id “4444” was successfully set to “5000” with a grace period of “25” days.
Conclusion:	The test is successful.

Table 5: To Show Evidence of Set the credit limit

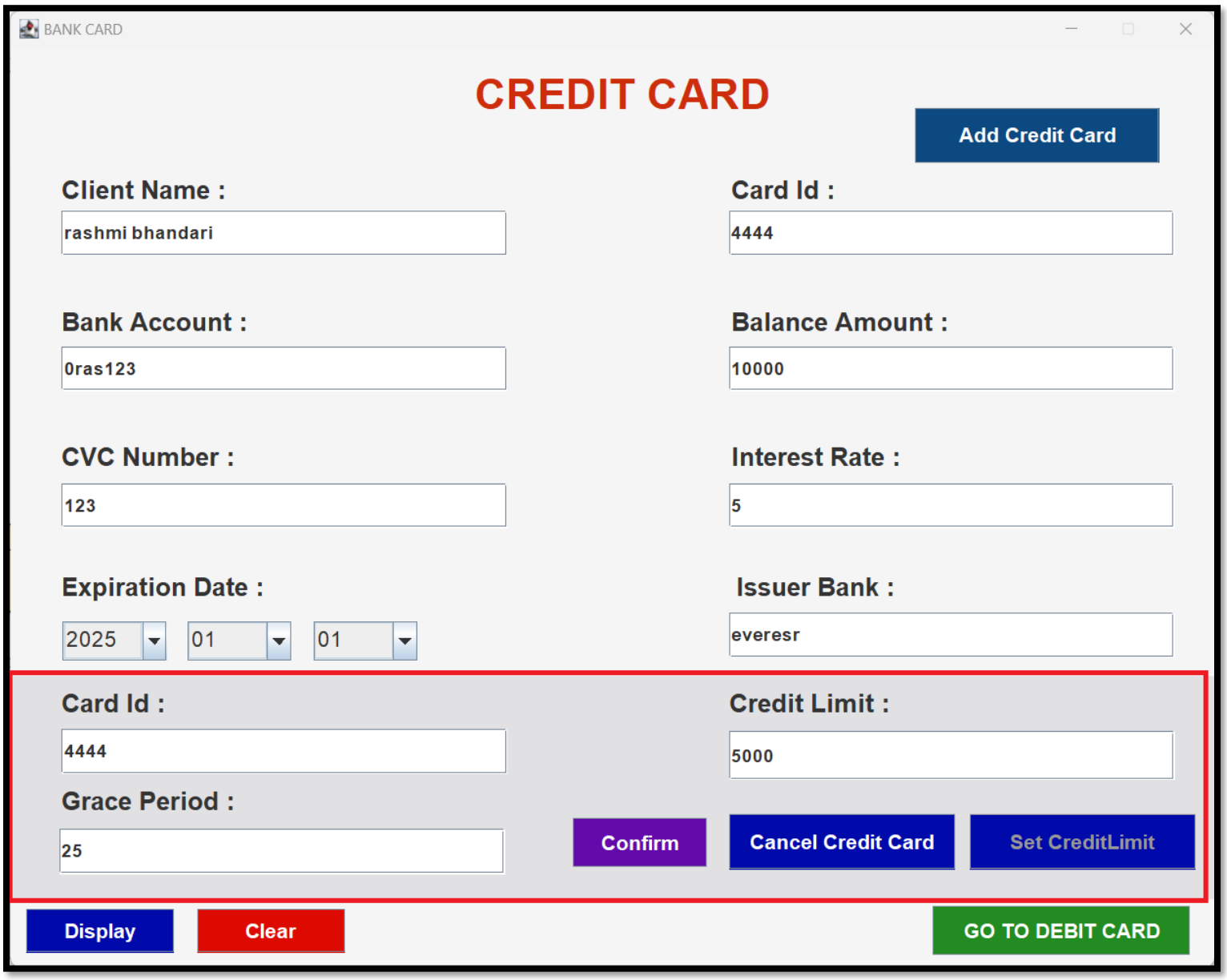

Figure 21: ScreenShot of filling details to Set Credit Limit in Credit Card

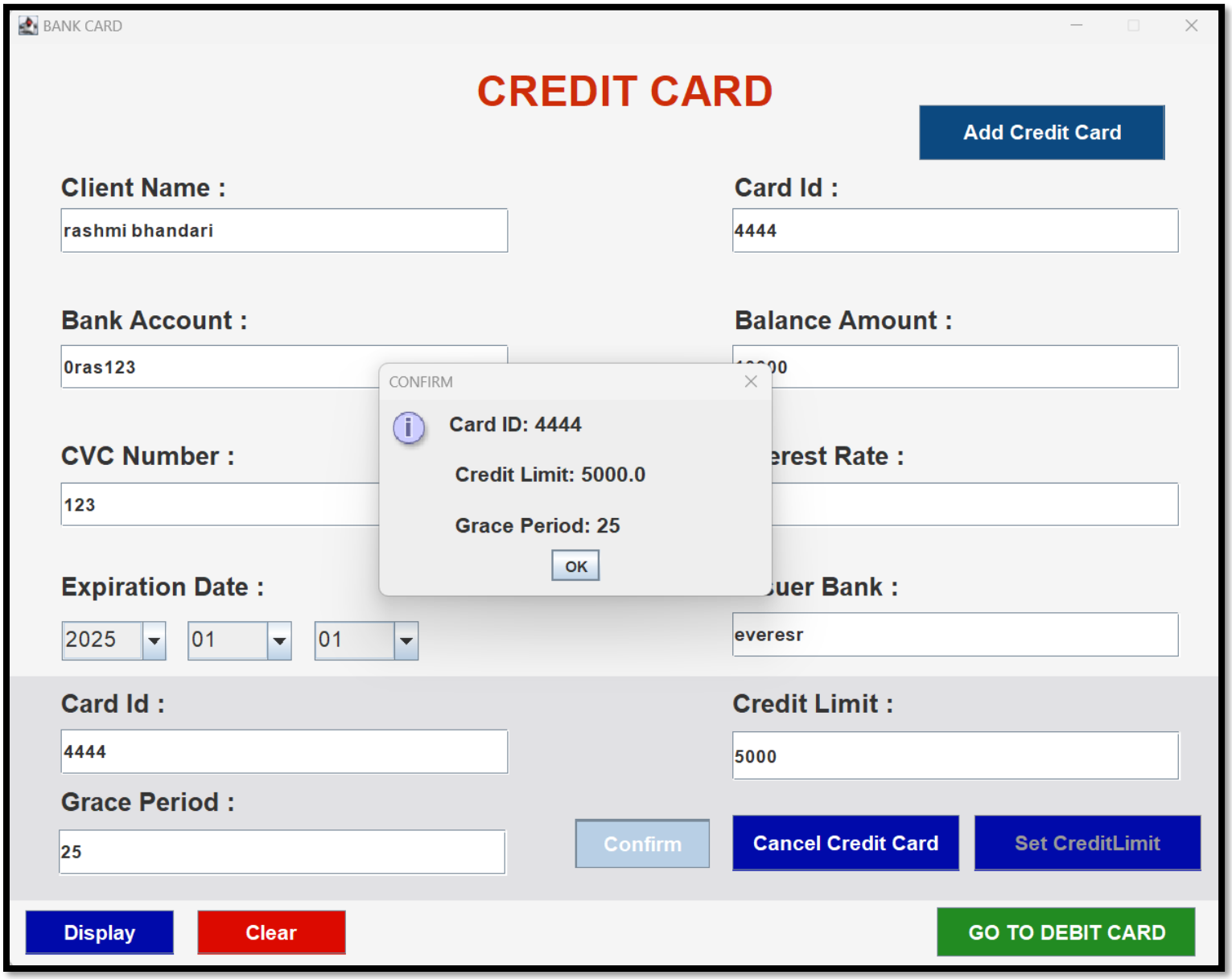

Figure 22: Screenshot of Clicking Confirm Button after filling Details in Input fileds to Set Credit Limit

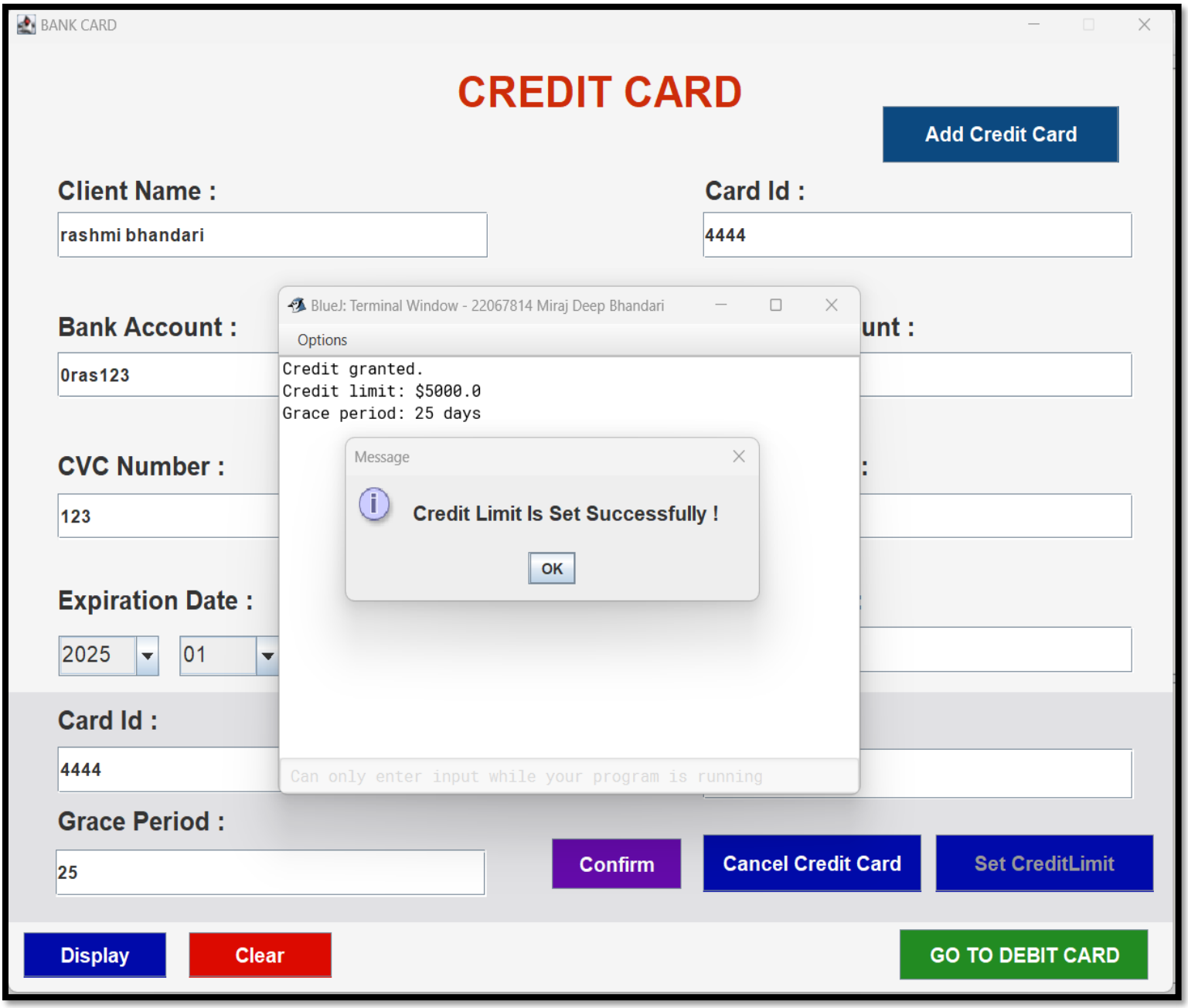

Figure 23:Screenshot of Successful Set Of Credit Limit from Credit Card after clicking Set CreditLimit button

5.2.5 - To Show Evidence of Remove the credit card

Test NO.	2.5
Objective:	To Show Evidence of Remove the credit card.
Action:	➡ The input fields for Card Id was filled with the value “4444 in Credit Card. ➡ The “Cancel Credit Card” button was clicked.
Expected Result:	The CVC Number, Grace Period and credit limit for the card with Id “4444” would be successfully set to “0” .
Actual Result:	The CVC Number, Grace Period and credit limit for the card with Id “4444” was successfully set to “0” .
Conclusion:	The test is successful.

Table 6: To Show Evidence of Remove the credit card

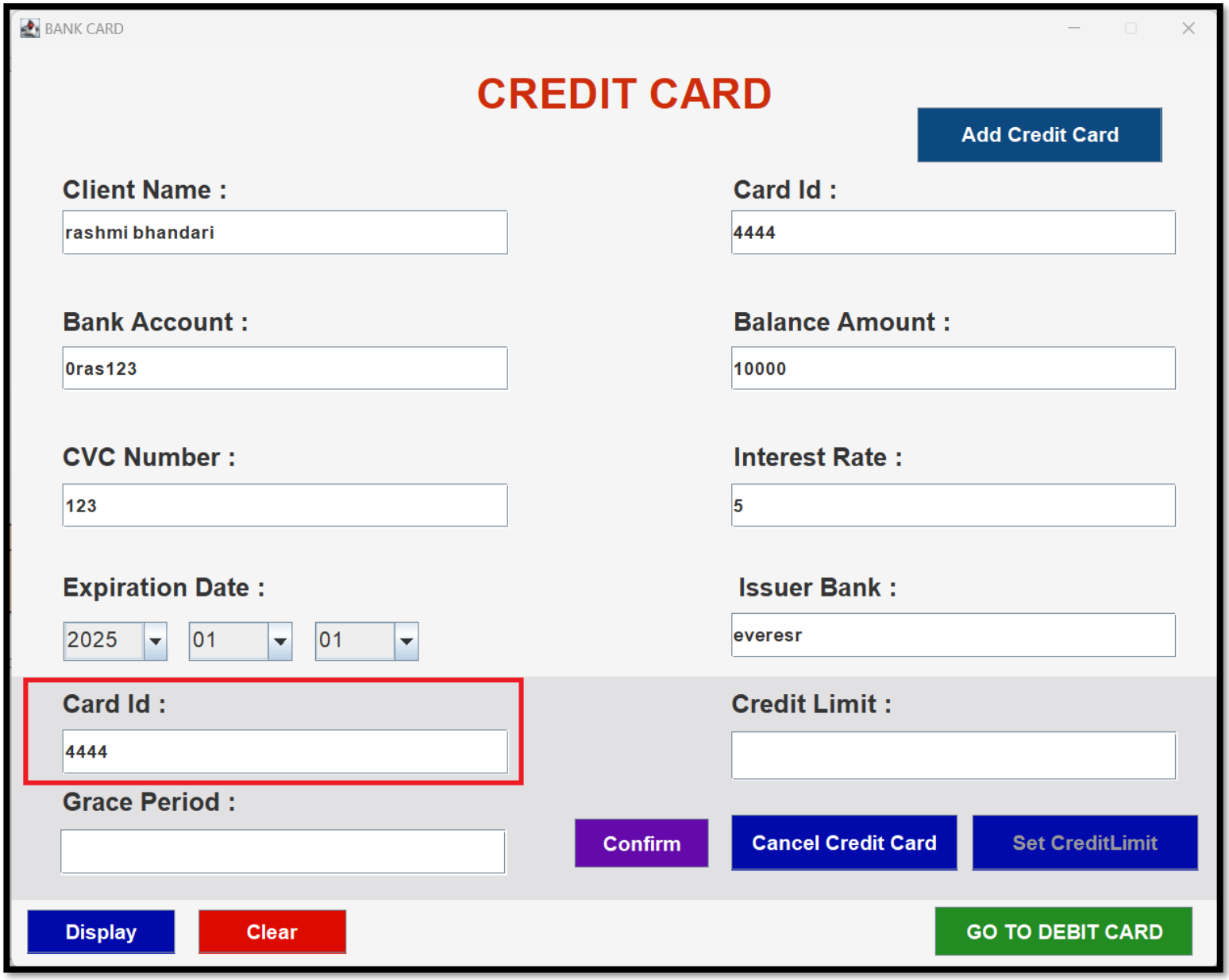

Figure 24: Screenshot of Giving Card Id to remove the Credit Card

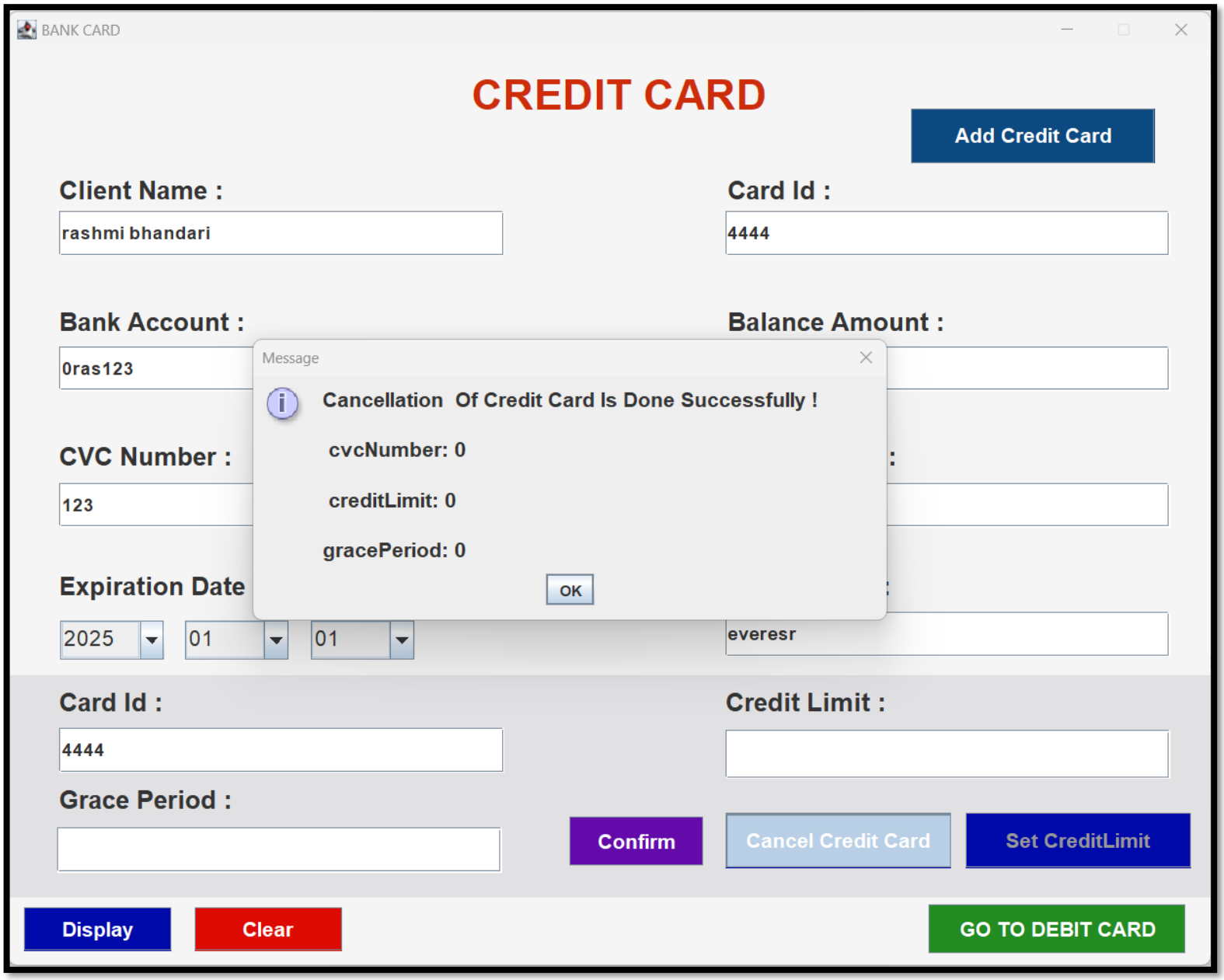

Figure 25: Screenshot of Successful Remove of Credit Card

5.3 Test 3 – To Test that appropriate dialog boxes appear when unsuitable values are entered for the Card ID

5.3.1 - To Test that appropriate dialog boxes appear when Alphabetic values are Entered for the Card ID

Test NO.	3.1
Objective:	To Test that appropriate dialog boxes appear when Alphabetic values are Entered for the Card ID.
Action:	➡ The input fields for Card Id was filled with the value “abcd in Debit Card and Credit Card. ➡ The “Add to Debit Card and Add to Credit Card ” button was clicked. ➡The input fields for Card Id was filled with the value “abcd during withdraw, setting Credit Limit and Cancellation of Credit Card.
Expected Result:	A error messege would be pop up with the message “ Card Id should be valid number”.
Actual Result:	A error messege was pop up with the message “ Card Id should be valid number”.
Conclusion:	The test is successful.

Table 7: To Test that appropriate dialog boxes appear when Alphabetic values are Entered for the Card ID.

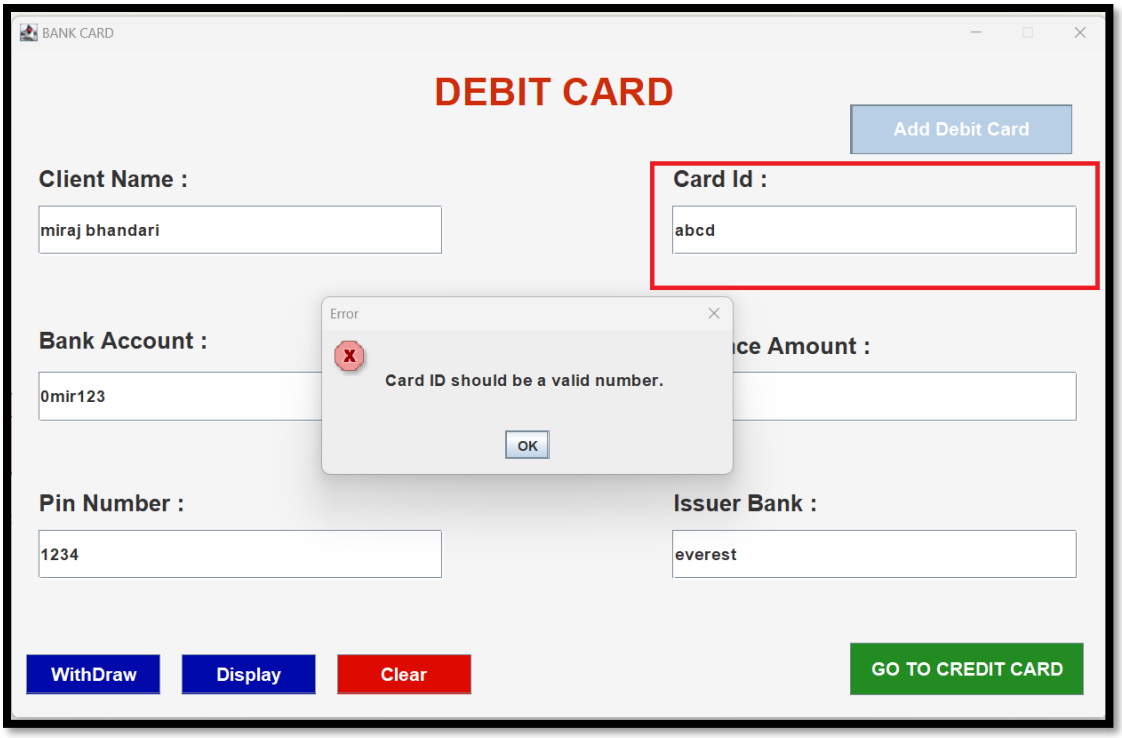

Figure 26:Screenshot of Popup of Error Message when Alphabetic Value is entered into Card Id while adding Debit Card

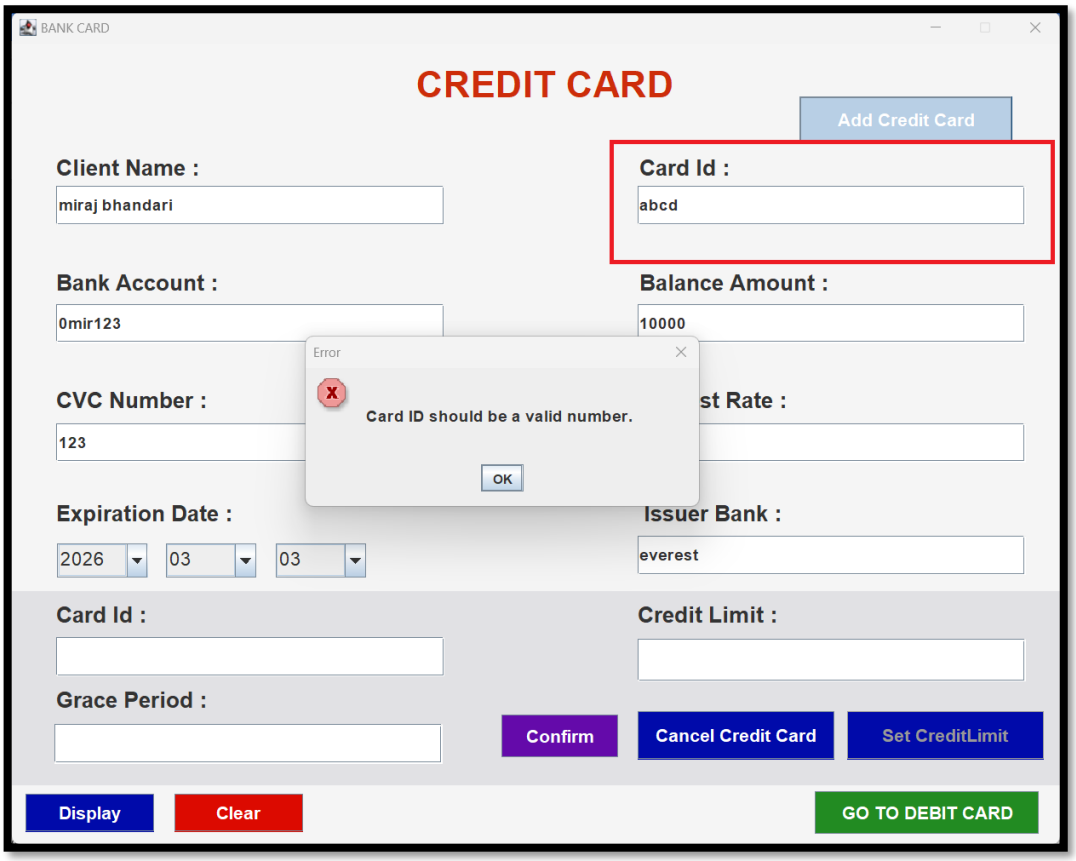

Figure 27: Screenshot of Popup of Error Message when Alphabetic Value is entered into Card Id while adding Credit Card

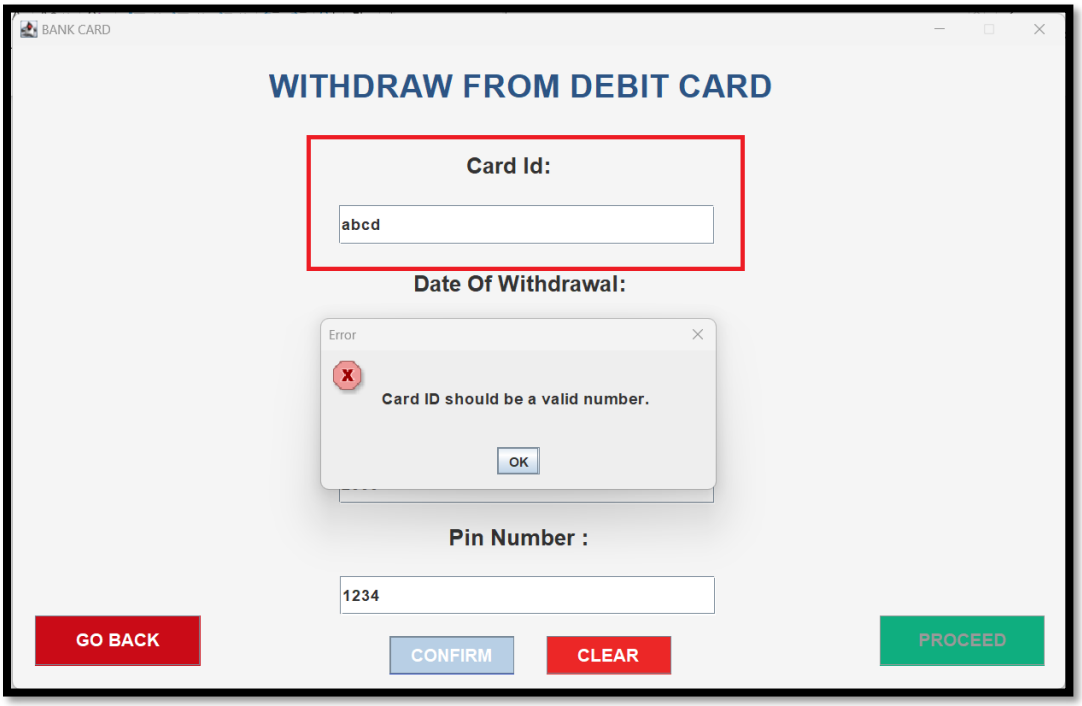

Figure 28: Screenshot of Popup of Error Message when Alphabetic Value is entered into Card Id during withdraw

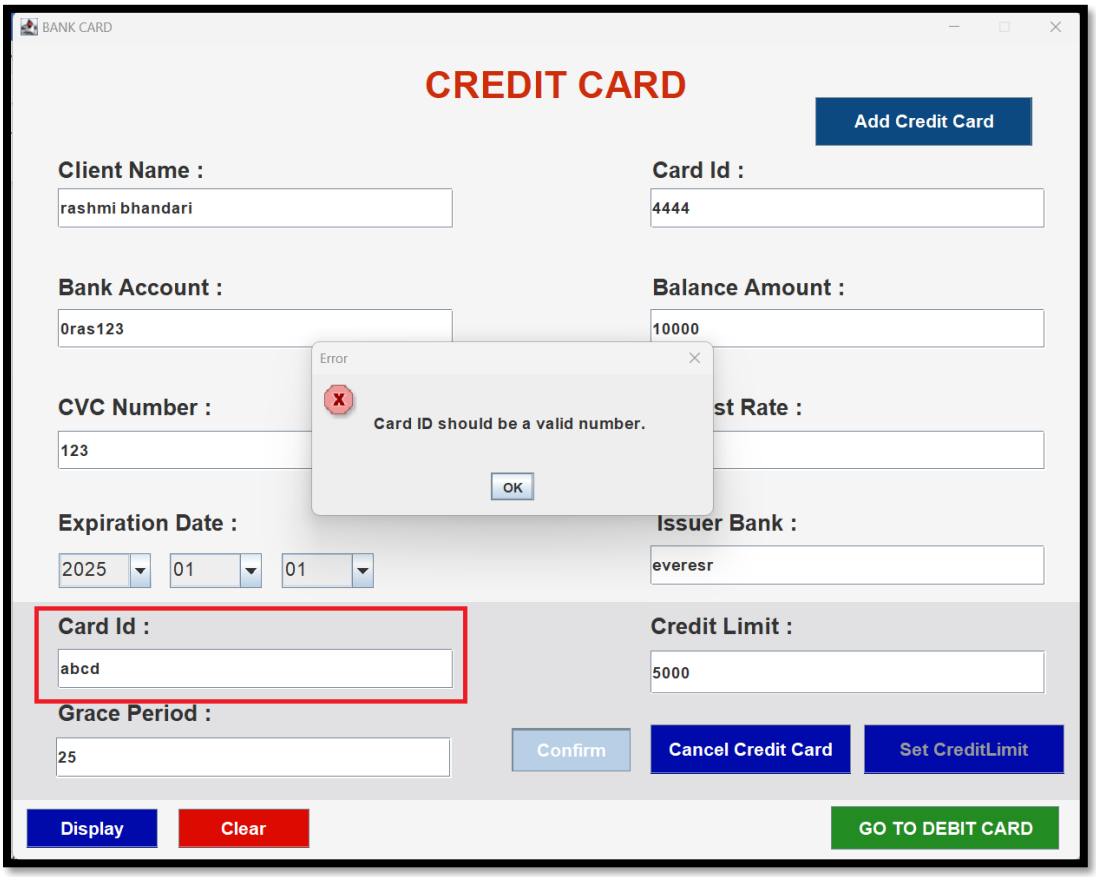

Figure 29:Screenshot of Popup of Error Message when Alphabetic Value is entered into Card Id during Setting Credit Limit and Cancelling Credit Card

5.3.2 - To Test that appropriate dialog box appear when Invalid values are Entered for the Card ID during withdraw

Test NO.	3.2
Objective:	To Test that appropriate dialog box appear when Invalid values are Entered for the Card ID during withdraw.
Action:	➡ The input fields for Client Name, Bank Account, Pin Number, Card Id, Balance Amount, and Issuer Bank were filled with the values “miraj bhandari”, “0mir123”, “1234”, “4321”, “10000”, and “eversest” respectively. ➡ The “Add Debit Card” button was clicked. ➡The input fields for Card Id, Date Of Withdrawal, Withdrawal Amount, and Pin Number was filled with the values “43211”, “2023-04-05”, “2000”, and “1234” respectively. ➡ The “CONFIRM” button was clicked after filling all Input fileds with invalid Card Id
Expected Result:	A error messege would be pop up with the message “Invalid Card Id ! Please Enter Your Correct Card Id”.

Actual Result:	A error messege was pop up with the message "Invalid Card Id ! Please Enter Your Correct Card Id".
Conclusion:	The test is successful.

Table 8: To Test that appropriate dialog box appear when Invalid values are Entered for the Card ID during withdraw

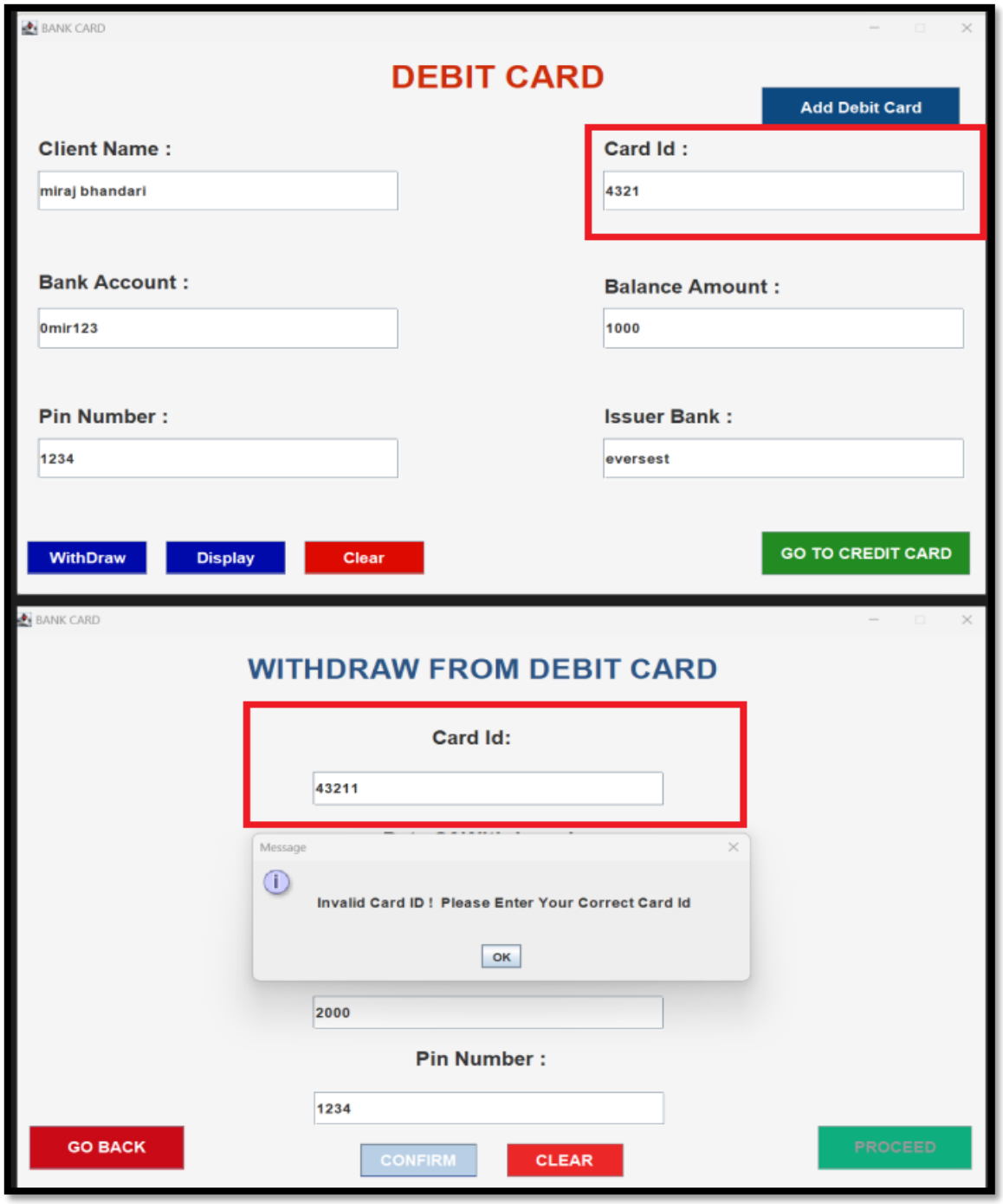

Figure 30: Screenshot of popup of Error Message when Invalid value is entered into Card Id during withdraw

5.3.3 - To Test that appropriate dialog box appear when Invalid values are Entered for the Card ID during Setting Credit Limit and Cancelling Credit Card

Test NO.	3.3
Objective:	To Show Evidence of Working of Add Credit Card.
Action:	➡ The input fields for Client Name, Bank Account, CVC Number, Expiration Date, Card Id, Balance Amount, Interest Rate and Issuer Bank were filled with the values “rashmi bhandari”, “0ras123”, “123”, “2025-01-01”, “4444”, “10000”, ”5” and “everesr” respectively. ➡ The “Add Credit Card” button was clicked. The input fields for Card Id, Credit Limit and Grace Period were filled with the values “44445”, “5000”, “25” respectively in Credit Card. ➡ The “CONFIRM” button was clicked after filling all Input fileds with invalid Card Id ➡ The “Cancel Credit Card” button was clicked.

Expected Result:	A error messege would be pop up with the message “Invalid Card Id ! Please Enter Your Correct Card Id”.
Actual Result:	A error messege was pop up with the message “Invalid Card Id ! Please Enter Your Correct Card Id”.
Conclusion:	The test is successful.

Table 9:To Test that appropriate dialog box appear when Invalid values are Entered for the Card ID during Setting Credit Limit and Cancelling Credit Card

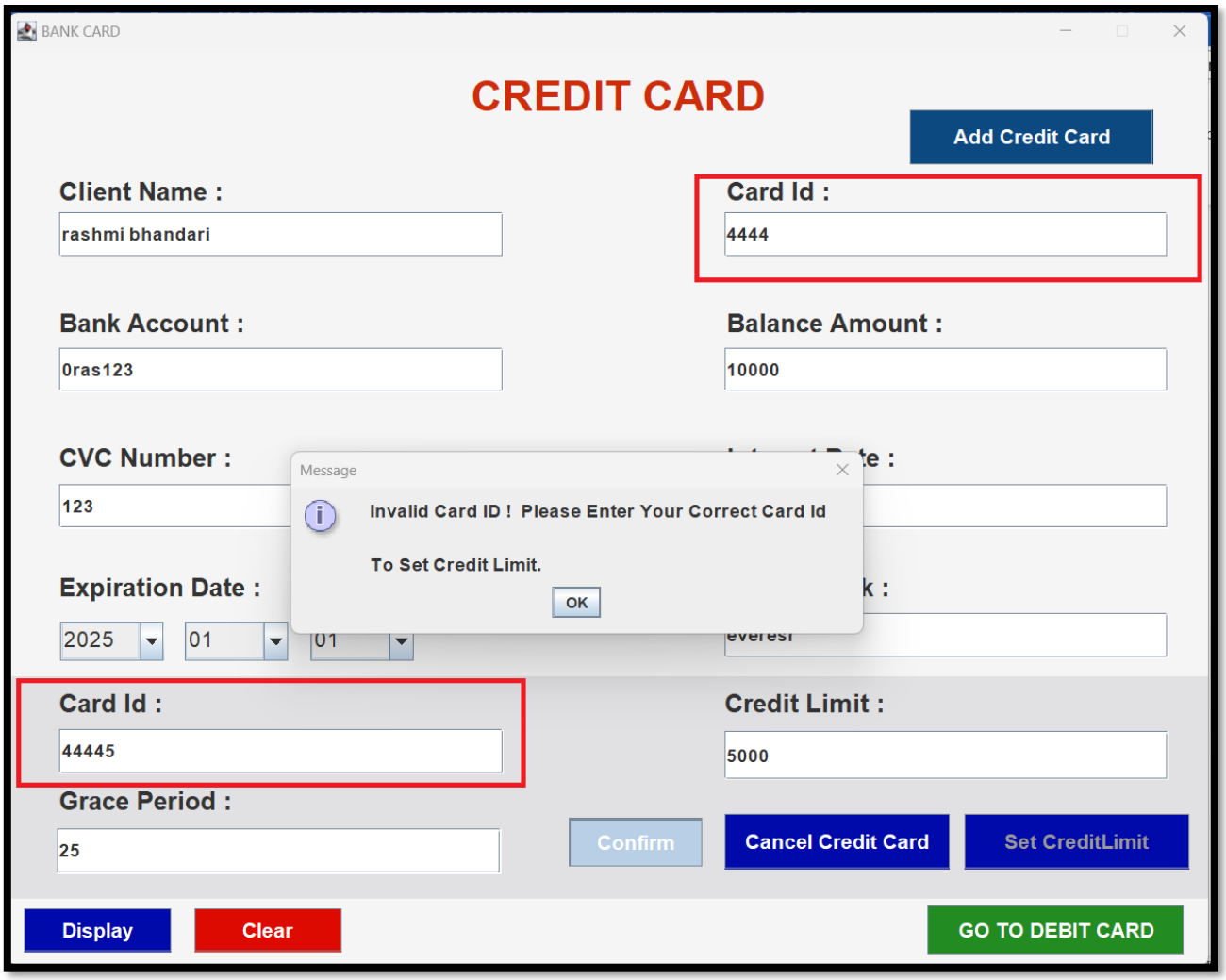

Figure 31: Screenshot of popup of Error Message when Invalid value is entered into Card Id during Setting Credit Limit

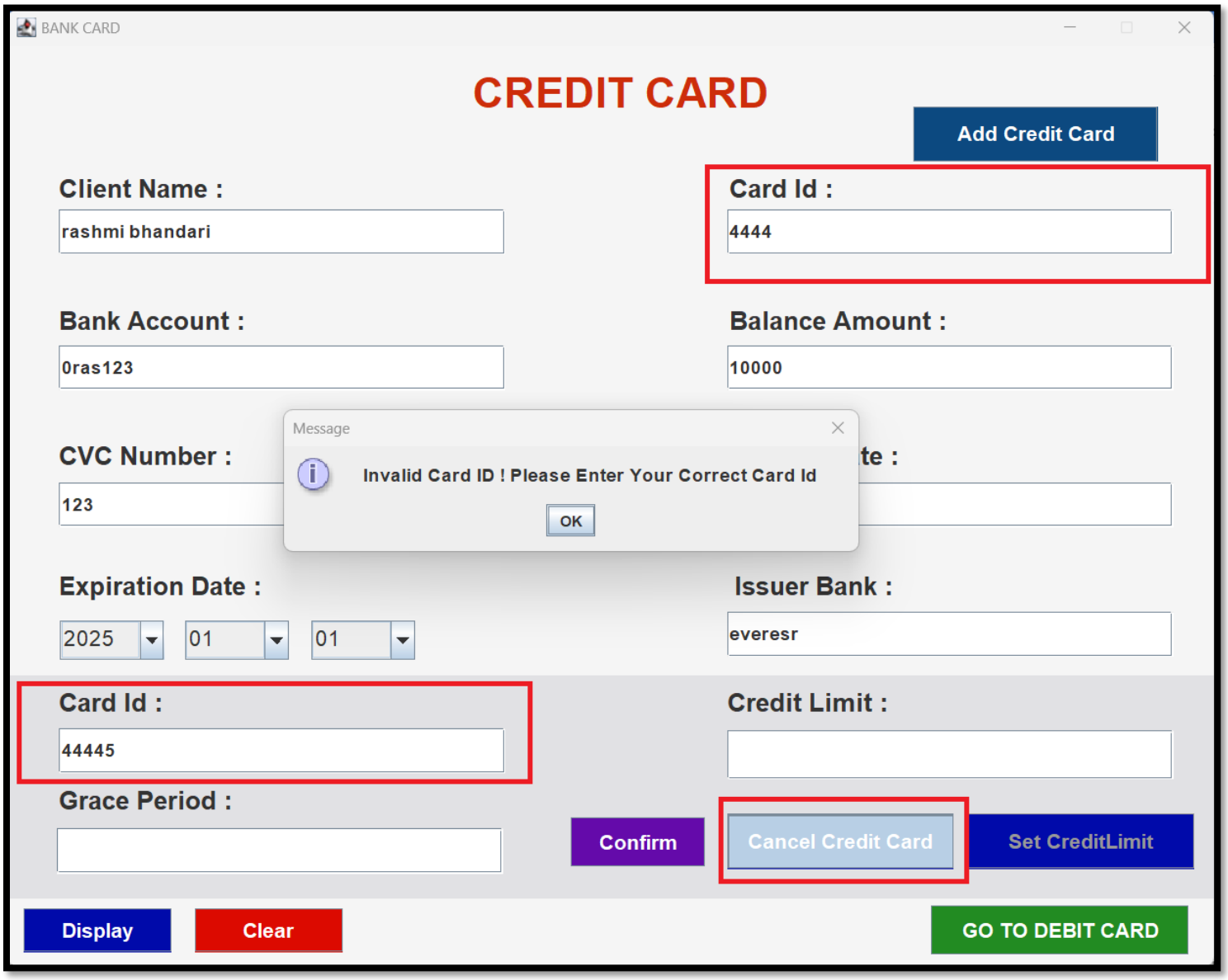

Figure 32: Screenshot of popup of Error Message when Invalid value is entered into Card Id during Cancellation of Credit Card

6. ERROR DETECTION AND CORRECTION

In Java programming, an error means a serious problem that prevents the Java Virtual Machine (JVM) from running a program or application. There are three main types of errors in Java:

1)Syntax errors

2)Logical errors

3)Semantic errors

Syntax errors - Syntax errors occur when there is an error in the program's code that violates the syntax rules of the programming language. Examples of syntax errors in Java include missing semicolons, incorrect use of braces, or misspelled keywords. These errors are usually caught by the compiler during the compilation process.

Logical errors - Logical errors occur when the program runs without any syntax errors, but the output is not what was expected or desired. These errors are often caused by mistakes in the program's logic or reasoning, such as using the wrong variable, using the wrong formula, or using an incorrect algorithm. Logical errors are typically more difficult to detect and fix than syntax errors.

Semantic errors - Semantic errors occur when the program runs without any syntax or logical errors, but the output is still not correct or expected. These errors are caused by issues with the meaning or interpretation of the program's code. Examples of semantic errors in Java include using a variable before it is initialized, accessing an array index that is out of bounds, or passing the wrong type of argument to a method. These errors are usually more subtle than syntax or logical errors and can be difficult to detect and fix.

Working with Java in the BlueJ development environment, I faced syntax, logical, and semantic errors. Though it was challenging, I researched and collaborated with peers, instructors, and the technical community to better understand the errors and develop effective debugging strategies. Solving the errors was satisfying, and it helped me become a better developer.

I will share my experience of facing and resolving errors in my Java development work using the BlueJ environment in the following section :-

6.1 FIRST ERROR AND ITS SOLUTION

<u>ERROR TYPE: SYNTAX ERROR</u>

```
31
32  public BankGui() {
33      jf = new JFrame("BANK CARD");
34      jf.setBounds(300, 100, 1000, 640);
35
36      jp = new JPanel();
37      jp.setSize(1000, 640);
38      jp.setBackground(new Color(245, 245, 245));
39
40      // for panel 2
41      jp2 = new JPanel();
42      jp2.setSize(1000, 640);
43      jp2.setBackground(new Color(245, 245, 245));
44
45      // for withdraw panel
46      jp3 = new JPanel();
47      jp3.setSize(1000, 640);
48      jp3.setBackground(new Color(245, 245, 245));
49
50      jp4 = new JPanel();
51      jp4.setBounds(0, 515, 1000, 182);
52      jp4.setBackground(new Color(225, 225, 228));
53
54      heading_1 = new JLabel(
55              "<html><span style='font-size: 27px; font-weight: bold; text-shadow: 2px 2px 2px #000; color:#CD2D0A;'> DEBIT CARD</span></html>");
56      heading_1.setBounds(380, 1, 400, 70);
57
58      // for panel 2 credit card
Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.
```

Figure 33: First Error (Type:Syntax Error)

I introduced a syntax error in my code as shown in above Screenshot by adding an extra closing parenthesis **)** in the jp.setBackground(new Color(245, 245, 245))) statement inside the BankGui constructor. The setBackground() method accepts a Color object as its parameter, and the new Color(245, 245, 245) statement creates a valid Color object. However, the extra closing parenthesis is not necessary and causes a syntax error.

SOLUTION OF THE ERROR

```
31
32      public BankGui() {
33          jf = new JFrame("BANK CARD");
34          jf.setBounds(300, 100, 1000, 640);
35
36          jp = new JPanel();
37          jp.setSize(1000, 640);
38          jp.setBackground(new Color(245, 245, 245));
39
40          // for panel 2
41          jp2 = new JPanel();
42          jp2.setSize(1000, 640);
43          jp2.setBackground(new Color(245, 245, 245));
44
45          // for withdraw panel
46          jp3 = new JPanel();
47          jp3.setSize(1000, 640);
48          jp3.setBackground(new Color(245, 245, 245));
49
50          jp4 = new JPanel();
51          jp4.setBounds(0, 515, 1000, 182);
52          jp4.setBackground(new Color(225, 225, 228));
53
54          heading_1 = new JLabel(
55                  "<html><span style='font-size: 27px; font-weight: bold; text-shadow: 2px 2px 2px #000; color:#CD2D0A;'> DEBIT CARD</span></html>");
56          heading_1.setBounds(280, 1, 400, 70);
```

Class compiled - no syntax errors

Figure 34: Solution of First Error (Type: Syntax Error)

I solved this syntax error by simply removing the extra closing parenthesis from the jp.setBackground(new Color(245, 245, 245))) statement inside the BankGui constructor. With the extra parenthesis removed, the code compiled successfully and the program ran as intended.

6.2 SECOND ERROR AND ITS SOLUTION

ERROR TYPE: LOGICAL ERROR

```
531 if (a.getSource() == add_button1) {
532     // String clientname = clientname_input1.getText();
533     String clientnameInput = clientname_input1.getText();
534     String issuerBankInput = issuerbank_input1.getText();
535     String bankAccount = bankaccount_input1.getText();
536     String cardIdInput = cardid_input1.getText();
537     String pinInput = pin_input1.getText();
538     String balanceAmountInput = bankamount_input1.getText();
539
540     if (clientnameInput.isEmpty() && issuerBankInput.isEmpty() && bankAccount.isEmpty() && cardIdInput.isEmpty()
541             && pinInput.isEmpty() && balanceAmountInput.isEmpty()) {
542         // Show an error message indicating that some fields are empty
543         UIManager.put("OptionPane.minimumSize", new Dimension(350, 120));
544         UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD, 15));
545
546         JOptionPane.showMessageDialog(null, "OPPS!  Please fill in all the fields.", "Error",
547                 JOptionPane.ERROR_MESSAGE);
548         return; // Return from the method as further processing is not possible
549     }
550
551     int cardId, pinnumber, balanceAmount;
552     String clientname, issuerBank;
553
Class compiled - no syntax errors
```

Figure 35:Second Error (Type:Logical Error)

I introduced a logical error in my code by using the && (logical AND) operator when checking if any of the input fields are empty. The code which is shown above the Screenshot would only show an error message if **all** the input fields are empty, instead of showing the error message if **any** of the input fields are empty so this is the first logical error which I made during my coursework.

SOLUTION OF THE ERROR

```
531     if (a.getSource() == add_button1) {
532         // String clientname = clientname_input1.getText();
533         String clientnameInput = clientname_input1.getText();
534         String issuerBankInput = issuerbank_input1.getText();
535         String bankAccount = bankaccount_input1.getText();
536         String cardIdInput = cardid_input1.getText();
537         String pinInput = pin_input1.getText();
538         String balanceAmountInput = bankamount_input1.getText();
539
540         if (clientnameInput.isEmpty() || issuerBankInput.isEmpty() || bankAccount.isEmpty() || cardIdInput.isEmpty()
541                 || pinInput.isEmpty() || balanceAmountInput.isEmpty()) {
542             // Show an error message indicating that some fields are empty
543             UIManager.put("OptionPane.minimumSize", new Dimension(350, 120));
544             UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD, 15));
545
546             JOptionPane.showMessageDialog(null, "OPPS!  Please fill in all the fields.", "Error",
547                     JOptionPane.ERROR_MESSAGE);
548             return; // Return from the method as further processing is not possible
549         }
550
551         int cardId, pinnumber, balanceAmount;
552         String clientname, issuerBank;
553
Class compiled - no syntax errors
```

Figure 36:Solution of Second Error (Type: Logical Error)

I solved the logical error by making a minor alteration to the code. Specifically, I replaced the && operator with the || operator to display the error message when any of the input fields were empty, not just when all of them were empty. This change ensured that the code functioned as intended and showed the correct error message.

6.3 THIRD ERROR AND ITS SOLUTION

ERROR TYPE: SEMANTIC ERROR

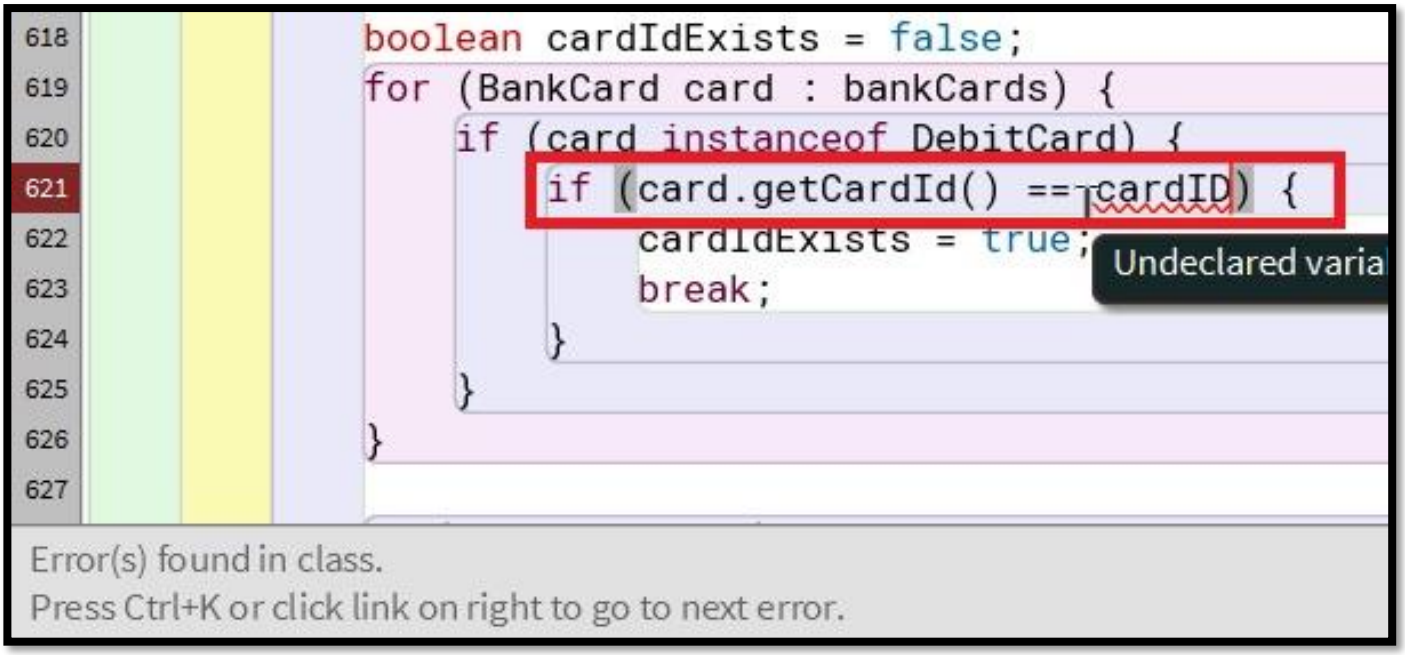

Figure 37:Third Error (Type:Semantic Error)

While working on my Java code in BlueJ, I encountered a semantic error. This error was caused by using an incorrect variable name for the card ID. In the if-statement within the for-loop, the variable name used was "**cardID**" instead of the correct variable name, which was "**cardId**". Due to this mistake, an **undeclared variable** "CardID" was used for the operation, leading to the error.

SOLUTION OF THE ERROR

```
boolean cardIdExists = false;
for (BankCard card : bankCards) {
    if (card instanceof DebitCard) {
        if (card.getCardId() == cardId) {
            cardIdExists = true;
            break;
        }
    }
}
```

Class compiled - no syntax errors

Figure 38:Solution of Third Error (Type: Semantic Error)

I fixed this semantic error by changing the variable name from **"CardID"** to **"cardId"** (with a lowercase "d") in the if-statement inside the for-loop. This corrected the case-sensitivity issue and ensured that the correct **declared variable** name was used for the card ID. After making this change, the code compiled and executed without any further issues.

7. CONCLUSION

7.1 THINGS I LEARNED IN THIS COURSEWORK

Throughout the CourseWork, I gained a comprehensive understanding of the graphical user interface (GUI) and its importance in software development. I learned about the functionality of GUI and how it enables users to interact with programs. Additionally, I learned how to design GUI frameworks using **Figma tools**, which allowed me to create visually appealing and better designs.

After designing the GUI framework, I converted the design into code using **Java's AWT** and **Swing libraries**. I learned about the various features of these libraries and how to utilize them to create dynamic and functional interfaces. I was able to implement user input prompts and **exception handling** to ensure that users input valid data. Furthermore, I utilized the concept of **ArrayLists** to store objects of DebitCard and CreditCard Class and perform various operations within my program. This allowed me to efficiently manage data and streamline the program's functionality.I also learned how to implement the **DownCasting** concepts in Inheritance.

In addition to coding, I learned how to create **class diagrams** using **draw.io**. These diagrams helped me to visually represent the structure of my program and enabled me to better understand its functionality. Moreover, I learned how to prepare reports using **MS Word**, which enabled me to effectively communicate the purpose and outcomes of my program. Additionally, I learned how to use **Photoshop** to select the ideal color combinations for my GUI. This allowed me to create visually appealing and aesthetically pleasing interfaces that enhanced the overall user experience.

Overall, this course provided me with a broad range of skills and knowledge necessary for designing and developing effective and efficient graphical user interfaces. I gained an in-depth understanding of GUI frameworks, programming concepts, and design principles. These skills will enable me to create user-friendly, visually appealing, and intuitive interfaces in the future.

7.2 CHALLENGES AND ITS OVERCOME WHILE DOING THIS COURSEWORK

While working on my first graphical user interface (GUI) project using AWT and Swing libraries, I encountered several challenges and difficulties. As a beginner, I made many syntax errors while coding the GUI, which required me to debug my code frequently. Although I was able to identify and fix these errors, it was time-consuming, and I felt discouraged.

However, the biggest obstacle I faced was setting the correct size and positioning of the components in the GUI. BlueJ, the IDE I was using, did not have a drag-and-drop feature to help me select the size and position of the components. As a result, I had to rely on the setBounds method to manually position each component, which was a daunting task for me. The components appeared differently than what I expected, and I spent a considerable amount of time adjusting the size and position until I finally got them right.

To seek guidance, I approached my teacher, who recommended using Figma to design the GUI components. With Figma, I was able to visualize the components and their sizes before implementing them in BlueJ. It simplified the design process, and I was able to create a visually appealing and functional GUI in less time.

Although the GUI part was relatively easier, I found the logical part more challenging. Specifically, understanding the action performed method and the flow of the button's working was a significant hurdle. I had to read documentation, search for examples online, and discuss with my tutor to comprehend the working of the button and the event handling. After several attempts and hours of research, I finally understood the action performed method and how it worked.

Another problem that troubled me was object casting. When I tried to downcast and perform operations on objects, I received exceptions. I struggled to overcome this issue and spent many hours searching for a solution. With the help of my tutor, I learned to check the object using the instanceof keyword before performing a downcast. It allowed me to perform the operation without exceptions and helped me understand the concept of object casting better.

Overall, despite the challenges and difficulties, working on the GUI project using AWT and Swing libraries was a valuable learning experience. It taught me the importance of perseverance, patience, and seeking guidance when facing challenges. I gained valuable coding skills and improved my problem-solving abilities, which I will apply in my future projects.

8. References

harleenk_99, 2022. *geeksforgeek.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-of-bluej/
[Accessed 6 5 2023].

Hartman, J., 2023. *GURU99.* [Online]
Available at: https://www.guru99.com/java-platform.html
[Accessed 6 5 2023].

Hope, C., 2020. *computerhope.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 6 5 2023].

Hope, C., 2021. *computerhope.* [Online]
Available at: https://www.computerhope.com/jargon/m/microsoft-word.htm
[Accessed 6 5 2023].

9. APPENDIX

Code of BankGui Class

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.ArrayList;

public class BankGui implements ActionListener {
    private JFrame jf;

    private JPanel jp, jp2, jp3, jp4;

    private JLabel heading_1, clinetname_label1, cardid_label1, bankaccount_label1,
balanceamount_label1, pinnumber_label1,
            withdrawalamount_label1, withdrawaldate_label1, heading_2,
clinetname_label2, cardid_label2,
            bankaccount_label2, balanceamount_label2, pinnumber_label2,
interestrate_label1, expirationdate_label1,
            issuerbank_label1, issuerbank_label2, creditlimit_label1, graceperiod_label1,
cardid_label3,
            withdrawalamount_label3, pinnumber_label3, heading_3,
dateofwithdrawal_label3, cardid_label3_credit;

    private JTextField clientname_input1, cardid_input1, bankaccount_input1,
bankamount_input1, pin_input1,
            withdrawalamount_input1, clientname_input2, cardid_input2,
bankaccount_input2, bankamount_input2,
            cvc_input1, interestrate_input1, issuerbank_input1, issuerbank_input2,
creditlimit_input1,
```

```
        graceperiod_input1, cardid_input3, withdrawalamount_input3,
pinnumber_input3, cardid_input3_credit;


    private JComboBox <String> withdrawaldate_year, withdrawaldate_month,
withdrawaldate_day, expirationdate_year, expirationdate_month,

        expirationdate_day, withdrawaldate_year3, withdrawaldate_month3,
withdrawaldate_day3;


    private JButton withdraw_button, setcreditlimit_button, add_button1, add_button2,
cancelcredit_button, display_button1,

        display_button2, clear_button1, clear_button2, gotocredit_button,
gotodebit_button, proceed_button,

        goback_button, confirm_button, clear_button3, confirm_button1;


    private ArrayList<BankCard> bankCards = new ArrayList<BankCard>();



    public BankGui() {

        jf = new JFrame("BANK CARD");

        jf.setBounds(300, 100, 1000, 640);


        jp = new JPanel();

        jp.setSize(1000, 640);

        jp.setBackground(new Color(245, 245, 245));


        // for panel 2

        jp2 = new JPanel();

        jp2.setSize(1000, 640);

        jp2.setBackground(new Color(245, 245, 245));
```

```
// for withdraw panel
jp3 = new JPanel();
jp3.setSize(1000, 640);
jp3.setBackground(new Color(245, 245, 245));

jp4 = new JPanel();
jp4.setBounds(0, 515, 1000, 182);
jp4.setBackground(new Color(225, 225, 228));

heading_1 = new JLabel(
        "<html><span style='font-size: 27px; font-weight: bold; text-shadow: 2px 2px
2px #000; color:#CD2D0A;'> DEBIT CARD</span></html>");
heading_1.setBounds(380, 1, 400, 70);

// for panel 2 credit card
heading_2 = new JLabel(
        "<html><span style='font-size: 27px; font-weight: bold; text-shadow: 2px 2px
2px #000; color:#CD2D0A;'> CREDIT CARD</span></html>");
heading_2.setBounds(380, 1, 400, 70);

clinetname_label1 = new JLabel("Client Name :");
clinetname_label1.setBounds(24, 105, 200, 24);
clinetname_label1.setFont(new Font("Arial", Font.BOLD, 21));

clinetname_label2 = new JLabel("Client Name :");
clinetname_label2.setBounds(42, 105, 200, 24);
clinetname_label2.setFont(new Font("Arial", Font.BOLD, 21));

cardid_label1 = new JLabel("Card Id :");
```

```
cardid_label1.setBounds(595, 105, 200, 24);
cardid_label1.setFont(new Font("Arial", Font.BOLD, 21));

cardid_label2 = new JLabel("Card Id :");
cardid_label2.setBounds(590, 105, 200, 24);
cardid_label2.setFont(new Font("Arial", Font.BOLD, 21));

bankaccount_label1 = new JLabel("Bank Account :");
bankaccount_label1.setBounds(24, 251, 200, 24);
bankaccount_label1.setFont(new Font("Arial", Font.BOLD, 21));

bankaccount_label2 = new JLabel("Bank Account :");
bankaccount_label2.setBounds(42, 213, 200, 24);
bankaccount_label2.setFont(new Font("Arial", Font.BOLD, 21));

balanceamount_label1 = new JLabel("Balance Amount :");
balanceamount_label1.setBounds(595, 256, 200, 24);
balanceamount_label1.setFont(new Font("Arial", Font.BOLD, 21));

balanceamount_label2 = new JLabel("Balance Amount :");
balanceamount_label2.setBounds(590, 213, 200, 24);
balanceamount_label2.setFont(new Font("Arial", Font.BOLD, 21));

pinnumber_label1 = new JLabel("Pin Number :");
pinnumber_label1.setBounds(24, 397, 200, 24);
pinnumber_label1.setFont(new Font("Arial", Font.BOLD, 21));

pinnumber_label2 = new JLabel("CVC Number :");
```

```
pinnumber_label2.setBounds(42, 324, 200, 24);
pinnumber_label2.setFont(new Font("Arial", Font.BOLD, 21));

interestrate_label1 = new JLabel("Interest Rate :");
interestrate_label1.setBounds(590, 324, 200, 24);
interestrate_label1.setFont(new Font("Arial", Font.BOLD, 21));

expirationdate_label1 = new JLabel("Expiration Date :");
expirationdate_label1.setBounds(42, 430, 200, 24);
expirationdate_label1.setFont(new Font("Arial", Font.BOLD, 21));

issuerbank_label1 = new JLabel("Issuer Bank :");
issuerbank_label1.setBounds(595, 397, 130, 24);
issuerbank_label1.setFont(new Font("Arial", Font.BOLD, 21));

issuerbank_label2 = new JLabel("Issuer Bank :");
issuerbank_label2.setBounds(594, 430, 130, 24);
issuerbank_label2.setFont(new Font("Arial", Font.BOLD, 21));

creditlimit_label1 = new JLabel("Credit Limit :");
creditlimit_label1.setBounds(588, 525, 150, 24);
creditlimit_label1.setFont(new Font("Arial", Font.BOLD, 21));

graceperiod_label1 = new JLabel("Grace Period :");
graceperiod_label1.setBounds(42, 605, 150, 24);
graceperiod_label1.setFont(new Font("Arial", Font.BOLD, 21));

cardid_label3_credit = new JLabel("Card Id :");
```

```
cardid_label3_credit.setBounds(42, 525, 200, 24);
cardid_label3_credit.setFont(new Font("Arial", Font.BOLD, 21));

// withdraw form debit label

heading_3 = new JLabel(
        "<html><span style='font-size: 24px; font-weight: bold; text-shadow: 2px 2px
2px #000; color:#2C5484; '> WITHDRAW FROM DEBIT CARD</span></html>");
heading_3.setBounds(240, 1, 600, 70);

cardid_label3 = new JLabel("Card Id:");
cardid_label3.setBounds(425, 99, 150, 29);
cardid_label3.setFont(new Font("Arial", Font.BOLD, 21));

withdrawalamount_label3 = new JLabel("Withdrawal Amount:");
withdrawalamount_label3.setBounds(378, 340, 209, 29);
withdrawalamount_label3.setFont(new Font("Arial", Font.BOLD, 21));

dateofwithdrawal_label3 = new JLabel("Date Of Withdrawal:");
dateofwithdrawal_label3.setBounds(375, 210, 215, 29);
dateofwithdrawal_label3.setFont(new Font("Arial", Font.BOLD, 21));

pinnumber_label3 = new JLabel("Pin Number :");
pinnumber_label3.setBounds(408, 449, 150, 24);
pinnumber_label3.setFont(new Font("Arial", Font.BOLD, 21));

clientname_input1 = new JTextField();
clientname_input1.setBounds(24, 141, 365, 45);
clientname_input1.setFont(new Font("Arial", Font.BOLD, 15));
```

```
clientname_input2 = new JTextField();
clientname_input2.setBounds(42, 134, 365, 37);
clientname_input2.setFont(new Font("Arial", Font.BOLD, 15));

cardid_input1 = new JTextField();
cardid_input1.setBounds(595, 141, 365, 45);
cardid_input1.setFont(new Font("Arial", Font.BOLD, 15));

cardid_input2 = new JTextField();
cardid_input2.setBounds(588, 134, 365, 37);
cardid_input2.setFont(new Font("Arial", Font.BOLD, 15));

bankaccount_input1 = new JTextField();
bankaccount_input1.setBounds(24, 291, 365, 45);
bankaccount_input1.setFont(new Font("Arial", Font.BOLD, 15));

bankaccount_input2 = new JTextField();
bankaccount_input2.setBounds(42, 245, 365, 37);
bankaccount_input2.setFont(new Font("Arial", Font.BOLD, 15));

bankamount_input1 = new JTextField();
bankamount_input1.setBounds(595, 291, 365, 45);
bankamount_input1.setFont(new Font("Arial", Font.BOLD, 15));

bankamount_input2 = new JTextField();
bankamount_input2.setBounds(588, 245, 365, 37);
bankamount_input2.setFont(new Font("Arial", Font.BOLD, 15));
```

```
pin_input1 = new JTextField();
pin_input1.setBounds(24, 433, 365, 45);
pin_input1.setFont(new Font("Arial", Font.BOLD, 15));

cvc_input1 = new JTextField();
cvc_input1.setBounds(42, 357, 365, 37);
cvc_input1.setFont(new Font("Arial", Font.BOLD, 15));

interestrate_input1 = new JTextField();
interestrate_input1.setBounds(588, 357, 365, 37);
interestrate_input1.setFont(new Font("Arial", Font.BOLD, 15));

issuerbank_input1 = new JTextField();
issuerbank_input1.setBounds(595, 433, 365, 45);
issuerbank_input1.setFont(new Font("Arial", Font.BOLD, 15));

issuerbank_input2 = new JTextField();
issuerbank_input2.setBounds(588, 463, 365, 37);
issuerbank_input2.setFont(new Font("Arial", Font.BOLD, 15));

creditlimit_input1 = new JTextField();
creditlimit_input1.setBounds(588, 560, 365, 40);
creditlimit_input1.setFont(new Font("Arial", Font.BOLD, 15));

graceperiod_input1 = new JTextField();
graceperiod_input1.setBounds(40, 640, 365, 37);
graceperiod_input1.setFont(new Font("Arial", Font.BOLD, 15));
```

```
cardid_input3_credit = new JTextField();
cardid_input3_credit.setBounds(42, 558, 365, 37);
cardid_input3_credit.setFont(new Font("Arial", Font.BOLD, 15));

// withdraw form debit input

cardid_input3 = new JTextField();
cardid_input3.setBounds(306, 150, 352, 37);
cardid_input3.setFont(new Font("Arial", Font.BOLD, 15));

withdrawalamount_input3 = new JTextField();
withdrawalamount_input3.setBounds(306, 393, 352, 37);
withdrawalamount_input3.setFont(new Font("Arial", Font.BOLD, 15));

pinnumber_input3 = new JTextField();
pinnumber_input3.setBounds(307, 497, 352, 37);
pinnumber_input3.setFont(new Font("Arial", Font.BOLD, 15));

// withdraw form debit combo box

String Wyear[] = { "Year", "2018", "2019", "2020", "2021", "2022", "2023" };
withdrawaldate_year3 = new JComboBox<String>(Wyear);
withdrawaldate_year3.setFont(new Font("Arial", Font.PLAIN, 18));
withdrawaldate_year3.setBounds(306, 266, 111, 32);

String Wmonth[] = { "Month", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10",
"11", "12" };
withdrawaldate_month3 = new JComboBox<String>(Wmonth);
```

```
withdrawaldate_month3.setFont(new Font("Arial", Font.PLAIN, 18));
withdrawaldate_month3.setBounds(443, 266, 78, 32);

String Wday[] = { "Day", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
"12", "13", "14",
        "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28",
"29", "30", "31" };

withdrawaldate_day3 = new JComboBox<String>(Wday);
withdrawaldate_day3.setFont(new Font("Arial", Font.PLAIN, 18));
withdrawaldate_day3.setBounds(547, 266, 111, 32);

String Eyear[] = { "Year", "2023", "2024", "2025", "2026", "2027" };
expirationdate_year = new JComboBox<String>(Eyear);
expirationdate_year.setFont(new Font("Arial", Font.PLAIN, 18));
expirationdate_year.setBounds(43, 470, 85, 32);

String Emonth[] = { "Month", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10",
"11", "12" };
expirationdate_month = new JComboBox<String>(Emonth);
expirationdate_month.setFont(new Font("Arial", Font.PLAIN, 18));
expirationdate_month.setBounds(145, 470, 85, 32);

String Eday[] = { "Day", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11",
"12", "13", "14",
        "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28",
"29", "30", "31" };
expirationdate_day = new JComboBox<String>(Eday);
expirationdate_day.setFont(new Font("Arial", Font.PLAIN, 18));
expirationdate_day.setBounds(248, 470, 85, 32);
```

```
withdraw_button = new JButton("WithDraw");
withdraw_button.setBounds(13, 545, 121, 36);
withdraw_button.setBackground(new Color(0, 10, 170));
withdraw_button.setForeground(Color.WHITE);
withdraw_button.setFont(new Font("Arial", Font.BOLD, 17));

setcreditlimit_button = new JButton("Set CreditLimit");
setcreditlimit_button.setBounds(785, 628, 185, 45);
setcreditlimit_button.setBackground(new Color(0, 10, 170));
setcreditlimit_button.setForeground(Color.WHITE);
setcreditlimit_button.setFont(new Font("Arial", Font.BOLD, 17));

add_button1 = new JButton("Add Debit Card");
add_button1.setBounds(755, 50, 200, 45);
add_button1.setBackground(new Color(12, 73, 128, 255));
add_button1.setForeground(Color.WHITE);
add_button1.setFont(new Font("Arial", Font.BOLD, 17));

add_button2 = new JButton("Add Credit Card");
add_button2.setBounds(740, 50, 200, 45);
add_button2.setBackground(new Color(12, 73, 128, 255));
add_button2.setForeground(Color.WHITE);
add_button2.setFont(new Font("Arial", Font.BOLD, 17));

cancelcredit_button = new JButton("Cancel Credit Card");
cancelcredit_button.setBounds(588, 628, 185, 45);
cancelcredit_button.setBackground(new Color(0, 10, 170));
```

```
cancelcredit_button.setForeground(Color.WHITE);
cancelcredit_button.setFont(new Font("Arial", Font.BOLD, 17));

display_button1 = new JButton("Display");
display_button1.setBounds(153, 545, 121, 36);
display_button1.setBackground(new Color(0, 10, 170));
display_button1.setForeground(Color.WHITE);
display_button1.setFont(new Font("Arial", Font.BOLD, 17));

display_button2 = new JButton("Display");
display_button2.setBounds(13, 705, 121, 36);
display_button2.setBackground(new Color(0, 10, 170));
display_button2.setForeground(Color.WHITE);
display_button2.setFont(new Font("Arial", Font.BOLD, 17));

clear_button1 = new JButton("Clear");
clear_button1.setBounds(293, 545, 121, 36);
clear_button1.setBackground(new Color(220, 10, 0));
clear_button1.setForeground(Color.WHITE);
clear_button1.setFont(new Font("Arial", Font.BOLD, 17));

clear_button2 = new JButton("Clear");
clear_button2.setBounds(153, 705, 121, 36);
clear_button2.setBackground(new Color(220, 10, 0));
clear_button2.setForeground(Color.WHITE);
clear_button2.setFont(new Font("Arial", Font.BOLD, 17));

gotocredit_button = new JButton("GO TO CREDIT CARD");
```

```
gotocredit_button.setBounds(755, 535, 210, 47);
gotocredit_button.setBackground(new Color(34, 139, 34));
gotocredit_button.setForeground(Color.WHITE);
gotocredit_button.setFont(new Font("Arial", Font.BOLD, 17));

gotodebit_button = new JButton("GO TO DEBIT CARD");
gotodebit_button.setBounds(755, 703, 210, 40);
gotodebit_button.setBackground(new Color(34, 139, 34));
gotodebit_button.setForeground(Color.WHITE);
gotodebit_button.setFont(new Font("Arial", Font.BOLD, 17));

// withdraw form debit button
proceed_button = new JButton("PROCEED");
proceed_button.setBounds(812, 534, 155, 47);
proceed_button.setBackground(new Color(15, 174, 127));
proceed_button.setForeground(Color.WHITE);
proceed_button.setFont(new Font("Arial", Font.BOLD, 17));

goback_button = new JButton("GO BACK");
goback_button.setBounds(21, 534, 155, 47);
goback_button.setBackground(new Color(202, 11, 23));
goback_button.setForeground(Color.WHITE);
goback_button.setFont(new Font("Arial", Font.BOLD, 17));

confirm_button = new JButton("CONFIRM");
confirm_button.setBounds(353, 553, 117, 36);
confirm_button.setBackground(new Color(38, 71, 187));
confirm_button.setForeground(Color.WHITE);
```

```
confirm_button.setFont(new Font("Arial", Font.BOLD, 17));

confirm_button1 = new JButton("Confirm");
confirm_button1.setBounds(460, 631, 110, 40);
confirm_button1.setBackground(new Color(100, 10, 170));
confirm_button1.setForeground(Color.WHITE);
confirm_button1.setFont(new Font("Arial", Font.BOLD, 17));

clear_button3 = new JButton("CLEAR");
clear_button3.setBounds(500, 553, 117, 36);
clear_button3.setBackground(new Color(236, 39, 39));
clear_button3.setForeground(Color.WHITE);
clear_button3.setFont(new Font("Arial", Font.BOLD, 17));

// Add
jf.add(jp);
jf.add(jp2);
jf.add(jp3);

jp.add(heading_1);
jp.add(clinetname_label1);
jp.add(cardid_label1);
jp.add(bankaccount_label1);
jp.add(balanceamount_label1);
jp.add(pinnumber_label1);
jp.add(issuerbank_label1);

jp.add(clientname_input1);
```

```
jp.add(cardid_input1);
jp.add(bankaccount_input1);
jp.add(bankamount_input1);
jp.add(pin_input1);
jp.add(issuerbank_input1);

jp.add(withdraw_button);
jp.add(display_button1);
jp.add(add_button1);
jp.add(clear_button1);
jp.add(gotocredit_button);

jp2.add(heading_2);
jp2.add(clinetname_label2);
jp2.add(cardid_label2);
jp2.add(bankaccount_label2);
jp2.add(balanceamount_label2);
jp2.add(pinnumber_label2);
jp2.add(interestrate_label1);
jp2.add(expirationdate_label1);
jp2.add(issuerbank_label2);
jp2.add(creditlimit_label1);
jp2.add(graceperiod_label1);
jp2.add(cardid_label3_credit);

jp2.add(clientname_input2);
jp2.add(cardid_input2);
jp2.add(bankaccount_input2);
```

```
jp2.add(bankamount_input2);
jp2.add(cvc_input1);
jp2.add(interestrate_input1);
jp2.add(issuerbank_input2);
jp2.add(creditlimit_input1);
jp2.add(graceperiod_input1);
jp2.add(cardid_input3_credit);
jp2.add(pinnumber_input3);

jp2.add(expirationdate_year);
jp2.add(expirationdate_month);
jp2.add(expirationdate_day);

jp2.add(setcreditlimit_button);
jp2.add(cancelcredit_button);
jp2.add(display_button2);
jp2.add(clear_button2);
jp2.add(gotodebit_button);
jp2.add(add_button2);
jp2.add(confirm_button1);

jp2.add(jp4);

// FOR JP3 DEBIIT WITHDRAW PANEL
jp3.add(heading_3);
jp3.add(cardid_label3);
jp3.add(withdrawalamount_label3);
jp3.add(pinnumber_label3);
```

```
jp3.add(dateofwithdrawal_label3);

jp3.add(cardid_input3);
jp3.add(withdrawalamount_input3);
jp3.add(pinnumber_input3);

jp3.add(withdrawaldate_year3);
jp3.add(withdrawaldate_month3);
jp3.add(withdrawaldate_day3);

jp3.add(proceed_button);
jp3.add(goback_button);
jp3.add(confirm_button);
jp3.add(clear_button3);

// register the event
gotocredit_button.addActionListener(this);
gotodebit_button.addActionListener(this);
add_button1.addActionListener(this);
add_button2.addActionListener(this);
clear_button1.addActionListener(this);
clear_button2.addActionListener(this);
withdraw_button.addActionListener(this);
goback_button.addActionListener(this);
confirm_button.addActionListener(this);
clear_button3.addActionListener(this);
proceed_button.addActionListener(this);
confirm_button1.addActionListener(this);
```

```
cancelcredit_button.addActionListener(this);
setcreditlimit_button.addActionListener(this);
display_button1.addActionListener(this);
display_button2.addActionListener(this);

// disable
proceed_button.setEnabled(false);
setcreditlimit_button.setEnabled(false);

proceed_button.setFocusable(false);
display_button2.setFocusable(false);
display_button1.setFocusable(false);
goback_button.setFocusable(false);
confirm_button.setFocusable(false);
clear_button3.setFocusable(false);
confirm_button1.setFocusable(false);
add_button2.setFocusable(false);
add_button1.setFocusable(false);
gotodebit_button.setFocusable(false);
clear_button1.setFocusable(false);
clear_button2.setFocusable(false);
display_button2.setFocusable(false);
cancelcredit_button.setFocusable(false);
setcreditlimit_button.setFocusable(false);
setcreditlimit_button.setFocusable(false);
cancelcredit_button.setFocusable(false);
display_button2.setFocusable(false);
clear_button2.setFocusable(false);
```

```
    gotodebit_button.setFocusable(false);
    add_button2.setFocusable(false);
    confirm_button1.setFocusable(false);



    jp.setLayout(null);
    jp2.setLayout(null);
    jp3.setLayout(null);
    jp4.setLayout(null);
    jp2.setVisible(false);
    jp3.setVisible(false);
    jf.setResizable(false);
    jf.setLayout(null);
    jf.setVisible(true);
    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

@Override
public void actionPerformed(ActionEvent a) {
    if (a.getSource() == gotocredit_button) {
        jp.setVisible(false);
        jp2.setVisible(true);
        jf.setBounds(300, 20, 1000, 790);
        jp2.setSize(1000, 800);
    } else if (a.getSource() == gotodebit_button) {
        jp.setVisible(true);
        jp2.setVisible(false);
```

```
        jf.setBounds(300, 100, 1000, 640);

    }

    if (a.getSource() == withdraw_button) {

        jp.setVisible(false);
        jp3.setVisible(true);

    } else if (a.getSource() == goback_button) {
        jp.setVisible(true);
        jp3.setVisible(false);
    }

    if (a.getSource() == add_button1) {
        // String clientname = clientname_input1.getText();
        String clientnameInput = clientname_input1.getText();
        String issuerBankInput = issuerbank_input1.getText();
        String bankAccount = bankaccount_input1.getText();
        String cardIdInput = cardid_input1.getText();
        String pinInput = pin_input1.getText();
        String balanceAmountInput = bankamount_input1.getText();

        if (clientnameInput.isEmpty() || issuerBankInput.isEmpty() ||
bankAccount.isEmpty() || cardIdInput.isEmpty()
                || pinInput.isEmpty() || balanceAmountInput.isEmpty()) {
            // Show an error message indicating that some fields are empty
            UIManager.put("OptionPane.minimumSize", new Dimension(350, 120));
```

```
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "OPPS!  Please fill in all the fields.",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        int cardId, pinnumber, balanceAmount;
        String clientname, issuerBank;

        try {
            cardId = Integer.parseInt(cardIdInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Card ID should be a valid number.",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            pinnumber = Integer.parseInt(pinInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that pinnumber is not a valid number
```

```
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Pin Number should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            balanceAmount = Integer.parseInt(balanceAmountInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that balanceAmount is not a valid
number
            UIManager.put("OptionPane.minimumSize", new Dimension(420, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Balance Amount should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            // Try to parse the input as a double
            double value = Double.parseDouble(clientnameInput);
            // If parsing succeeds, it means the input is a number, so show an error
message
```

```
            UIManager.put("OptionPane.minimumSize", new Dimension(400, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));
            JOptionPane.showMessageDialog(null, "OPPS !  Client name cannot be a
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        } catch (NumberFormatException e) {
            // If parsing fails, it means the input is a string
            clientname = clientnameInput;
        }

        try {
            // Try to parse the input as a double
            double value = Double.parseDouble(issuerBankInput);
            // If parsing succeeds, it means the input is a number, so show an error
message
            UIManager.put("OptionPane.minimumSize", new Dimension(450, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));
            JOptionPane.showMessageDialog(null, "OPPS !  Issuer Bank name cannot
be a number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        } catch (NumberFormatException e) {
            // If parsing fails, it means the input is a string
            issuerBank = issuerBankInput;
        }

        boolean cardIdExists = false;
```

```
        for (BankCard card : bankCards) {
            if (card instanceof DebitCard) {
                if (card.getCardId() == cardId) {
                    cardIdExists = true;
                    break;
                }
            }
        }

        if (!cardIdExists) {
            DebitCard debitCard = new DebitCard(balanceAmount, cardId, bankAccount,
issuerBank, clientname,
                    pinnumber);

            bankCards.add(debitCard);

            UIManager.put("OptionPane.minimumSize", new Dimension(380, 150));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));
            JOptionPane.showMessageDialog(null,
                    "DEBIT CARD ADDED SUCCESSFULLY ! \n Client Name: " +
clientname + "\n Issuer Bank: "
                            + issuerBank + "\n Bank Account: " + bankAccount + "\n Card ID: "
+ cardId
                            + "\n Pin Number: " + pinnumber + "\n Balance Amount: $" +
balanceAmount,
                    "Message", JOptionPane.INFORMATION_MESSAGE);

        } else {
            // Show an error message indicating that the card ID already exists
```

```
            UIManager.put("OptionPane.minimumSize", new Dimension(500, 160));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Card ID already exists! Please enter
a different Card ID.",
                    "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    // to add credit card
    if (a.getSource() == add_button2) {

        String clientnameInput = clientname_input2.getText();

        String issuerBankInput = issuerbank_input2.getText();

        String bankAccount = bankaccount_input2.getText();

        String cardIdInput = cardid_input2.getText();

        String balanceAmountInput = bankamount_input2.getText();

        String CVCnumberInput = cvc_input1.getText();

        String interestrateInput = interestrate_input1.getText();

        String year = (String) expirationdate_year.getSelectedItem();
        String month = (String) expirationdate_month.getSelectedItem();
```

```
        String day = (String) expirationdate_day.getSelectedItem();
        String expirationDate = year + "-" + month + "-" + day;

        if (clientnameInput.isEmpty() || issuerBankInput.isEmpty() ||
bankAccount.isEmpty() || cardIdInput.isEmpty()
                || CVCnumberInput.isEmpty() || balanceAmountInput.isEmpty() ||
interestrateInput.isEmpty()
                || year.equals("Year") || month.equals("Month") || day.equals("Day")) {
            // Show an error message indicating that some fields are empty
            UIManager.put("OptionPane.minimumSize", new Dimension(350, 120));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "OPPS! Please fill in all the fields.",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        int cardId, balanceAmount, cvcnumber;
        double interestrate;
        String clientname, issuerBank;

        try {

            balanceAmount = Integer.parseInt(balanceAmountInput);

        } catch (NumberFormatException e) {
            // Show an error message indicating that balanceAmount is not a valid
number
```

```
            UIManager.put("OptionPane.minimumSize", new Dimension(420, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Balance Amount should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            cardId = Integer.parseInt(cardIdInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Card ID should be a valid number.",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            cvcnumber = Integer.parseInt(CVCnumberInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
```

```
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "CVC number should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            interestrate = Double.parseDouble(interestrateInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(400, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Interest Rate should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            // Try to parse the input as a double
            double value = Double.parseDouble(clientnameInput);
            // If parsing succeeds, it means the input is a number, so show an error
message
            UIManager.put("OptionPane.minimumSize", new Dimension(400, 130));
```

```
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));
            JOptionPane.showMessageDialog(null, "OPPS !  Client name cannot be a
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        } catch (NumberFormatException e) {
            // If parsing fails, it means the input is a string
            clientname = clientnameInput;
        }

        try {
            // Try to parse the input as a double
            double value = Double.parseDouble(issuerBankInput);
            // If parsing succeeds, it means the input is a number, so show an error
message
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));
            UIManager.put("OptionPane.minimumSize", new Dimension(450, 130));
            JOptionPane.showMessageDialog(null, "OPPS !  Issuer Bank name cannot
be a number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        } catch (NumberFormatException e) {
            // If parsing fails, it means the input is a string
            issuerBank = issuerBankInput;
        }

        boolean cardIdExists = false;
        for (BankCard card : bankCards) {
```

```
            if (card instanceof CreditCard) {
                if (card.getCardId() == cardId) {
                    cardIdExists = true;
                    break;
                }
            }
        }

        if (!cardIdExists) {

            CreditCard creditCard = new CreditCard(cardId, clientname, issuerBank,
bankAccount, balanceAmount,
                    cvcnumber, interestrate, expirationDate);

            bankCards.add(creditCard);

            UIManager.put("OptionPane.minimumSize", new Dimension(380, 150));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null,
                    "CREDIT CARD ADDED SUCCESSFULLY ! \n Client Name: " +
clientname + "\n Issuer Bank: "
                            + issuerBank + "\n Bank Account: " + bankAccount + "\n Card ID: "
+ cardId
                            + "\n Balance Amount: $" + balanceAmount + "\n CVC Number: " +
cvcnumber
                            + "\n Interest Rate: " + interestrate + "\n Expiration Date: " +
expirationDate,
                    "Message", JOptionPane.INFORMATION_MESSAGE);
```

```
        }

        else {

            UIManager.put("OptionPane.minimumSize", new Dimension(395, 160));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Card ID already exists ! Please Enter
a different Card ID.",
                    "Error", JOptionPane.ERROR_MESSAGE);

        }

    }

    // to clear input values of debit panel
    if (a.getSource() == clear_button1) {

        clientname_input1.setText("");
        issuerbank_input1.setText("");
        bankaccount_input1.setText("");
        cardid_input1.setText("");
        pin_input1.setText("");
        bankamount_input1.setText("");

    }

    // to clear input values of withdraw panel
```

```
if (a.getSource() == clear_button3) {

    withdrawalamount_input3.setText("");
    cardid_input3.setText("");
    pinnumber_input3.setText("");
    withdrawaldate_year3.setSelectedIndex(0);
    withdrawaldate_month3.setSelectedIndex(0);
    withdrawaldate_day3.setSelectedIndex(0);

}

// to clear input values of credit panel
if (a.getSource() == clear_button2) {

    clientname_input2.setText("");
    issuerbank_input2.setText("");
    bankaccount_input2.setText("");
    cardid_input2.setText("");
    pin_input1.setText("");
    bankamount_input2.setText("");
    cvc_input1.setText("");
    interestrate_input1.setText("");
    graceperiod_input1.setText("");
    creditlimit_input1.setText("");
    cardid_input3_credit.setText("");
    expirationdate_year.setSelectedIndex(0);
    expirationdate_month.setSelectedIndex(0);
    expirationdate_day.setSelectedIndex(0);
```

```
        }

        // after cliking confirm button to display info
        if (a.getSource() == confirm_button) {

            String cardIdInput = cardid_input3.getText();
            String withdrawalamountInput = withdrawalamount_input3.getText();
            String pinnumberInput = pinnumber_input3.getText();

            String year = (String) withdrawaldate_year3.getSelectedItem();
            String month = (String) withdrawaldate_month3.getSelectedItem();
            String day = (String) withdrawaldate_day3.getSelectedItem();
            String withdrawaldate = year + "-" + month + "-" + day;

            if (cardIdInput.isEmpty() || withdrawalamountInput.isEmpty() ||
pinnumberInput.isEmpty()
                    || year.equals("Year") || month.equals("Month") || day.equals("Day")) {
                // Show an error message indicating that some fields are empty
                UIManager.put("OptionPane.minimumSize", new Dimension(350, 120));
                UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

                JOptionPane.showMessageDialog(null, "OPPS!  Please fill in all the fields.",
"Error",
                        JOptionPane.ERROR_MESSAGE);
                return; // Return from the method as further processing is not possible
            }
```

```
        int cardId, withdrawalamount, pinnumber;

        try {
            cardId = Integer.parseInt(cardIdInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Card ID should be a valid number.",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            withdrawalamount = Integer.parseInt(withdrawalamountInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(430, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "withdrawal Amount should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }
```

```
        try {
            pinnumber = Integer.parseInt(pinnumberInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(400, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Pin Number should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        boolean cardIdExists = false;
        for (BankCard card : bankCards) {
            if (card instanceof DebitCard) {
                if (card.getCardId() == cardId) {

                    // Display success message with values and set dialog size
                    UIManager.put("OptionPane.minimumSize", new Dimension(350, 200));
                    UIManager.put("OptionPane.messageFont", new Font("Arial",
Font.BOLD, 17));
                    JOptionPane.showMessageDialog(null,
                            "Card ID: " + cardId + "\n\nWithdrawal Date: " + withdrawaldate
                                    + "\n\nWithdrawal Amount: $ " + withdrawalamount + "\n\nPIN
Number: "
                                    + pinnumber,
                            "CONFIRM", JOptionPane.INFORMATION_MESSAGE);
                    proceed_button.setEnabled(true);
```

```
                cardIdExists = true;
                break;
            }
        }
    }

    if (!cardIdExists) {

        UIManager.put("OptionPane.minimumSize", new Dimension(500, 130));
        JOptionPane.showMessageDialog(null, "  Invalid Card ID !  Please Enter Your
Correct Card Id ",
                "Message", JOptionPane.INFORMATION_MESSAGE);

    }

}

// after clicking proceed button
if (a.getSource() == proceed_button) {

    int cardId = Integer.parseInt(cardid_input3.getText());
    int withdrawalamount = Integer.parseInt(withdrawalamount_input3.getText());

    String year = (String) withdrawaldate_year3.getSelectedItem();
    String month = (String) withdrawaldate_month3.getSelectedItem();
    String day = (String) withdrawaldate_day3.getSelectedItem();
    String withdrawaldate = year + "-" + month + "-" + day;

    int pinnumber = Integer.parseInt(pinnumber_input3.getText());
```

```
            for (BankCard card : bankCards) {
                if (card instanceof DebitCard) {
                    if (card.getCardId() == cardId) {


                        ((DebitCard) card).withdraw(withdrawalamount, withdrawaldate,
pinnumber);
                        UIManager.put("OptionPane.minimumSize", new Dimension(370, 100));
                        JOptionPane.showMessageDialog(null, "Withdraw Process Is
Successfully Done !", "Message",
                                JOptionPane.INFORMATION_MESSAGE);
                        proceed_button.setEnabled(false);
                        break;
                    }


                }
            }


        }


        if (a.getSource() == confirm_button1) {
            String cardIdInput = cardid_input3_credit.getText();
            String graceperiodInput = graceperiod_input1.getText();
            String creditlimitInput = creditlimit_input1.getText();


            if (cardIdInput.isEmpty() || graceperiodInput.isEmpty() ||
creditlimitInput.isEmpty()) {
                // Show an error message indicating that some fields are empty
                UIManager.put("OptionPane.minimumSize", new Dimension(480, 120));
```

```
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "OPPS!  Please fill in all the fields To
Set Credit Limit", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        int cardId, graceperiod;
        double creditlimit;

        try {
            cardId = Integer.parseInt(cardIdInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Card ID should be a valid number.",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            graceperiod = Integer.parseInt(graceperiodInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
```

```
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Grace Period should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        try {
            creditlimit = Double.parseDouble(creditlimitInput);

        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Credit Limit should be a valid
number.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }

        boolean cardIdExists = false;
        for (BankCard card : bankCards) {

            if (card instanceof CreditCard) {
                if (card.getCardId() == cardId) {
```

```
                    // Display success message with values and set dialog size
                    UIManager.put("OptionPane.minimumSize", new Dimension(320, 150));
                    UIManager.put("OptionPane.messageFont", new Font("Arial",
Font.BOLD, 17));
                    JOptionPane.showMessageDialog(null, "Card ID: " + cardId + "\n\n
Credit Limit: " + creditlimit
                            + "\n\n Grace Period: " + graceperiod, "CONFIRM",
JOptionPane.INFORMATION_MESSAGE);
                    setcreditlimit_button.setEnabled(true);
                    cardIdExists = true;


                    break;
                }
            }
        }

        if (!cardIdExists) {

            UIManager.put("OptionPane.minimumSize", new Dimension(470, 120));
            JOptionPane.showMessageDialog(null,
                    "  Invalid Card ID !  Please Enter Your Correct Card Id\n\n  To Set Credit
Limit. ", "Message",
                    JOptionPane.INFORMATION_MESSAGE);

        }

    }

    if (a.getSource() == cancelcredit_button) {
```

```
        String cardIdInput = cardid_input3_credit.getText();

        if (cardIdInput.isEmpty()) {
            UIManager.put("OptionPane.minimumSize", new Dimension(360, 120));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "OPPS!  Please Enter The Card Id
field.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            return;
        }

        int cardId;

        try {
            cardId = Integer.parseInt(cardIdInput);
        } catch (NumberFormatException e) {
            // Show an error message indicating that cardId is not a valid number
            UIManager.put("OptionPane.minimumSize", new Dimension(370, 130));
            UIManager.put("OptionPane.messageFont", new Font("Arial", Font.BOLD,
15));

            JOptionPane.showMessageDialog(null, "Card ID should be a valid number.",
"Error",
                    JOptionPane.ERROR_MESSAGE);
            return; // Return from the method as further processing is not possible
        }
```

```
        boolean cardIdExists = false;
        for (BankCard card : bankCards) {
            if (card instanceof CreditCard) {
                if (card.getCardId() == cardId) {

                    // Display success message with values and set dialog size
                    ((CreditCard) card).cancelCreditCard();
                    UIManager.put("OptionPane.minimumSize", new Dimension(520, 200));
                    UIManager.put("OptionPane.messageFont", new Font("Arial",
Font.BOLD, 17));
                    JOptionPane.showMessageDialog(null,
                            "Cancellation  Of Credit Card Is Done Successfully !\n\n
cvcNumber: 0\n\n creditLimit: 0\n\ngracePeriod: 0 ",
                            "Message", JOptionPane.INFORMATION_MESSAGE);
                    cardIdExists = true;

                    break;
                }
            }
        }

        if (!cardIdExists) {

            UIManager.put("OptionPane.minimumSize", new Dimension(470, 70));
            JOptionPane.showMessageDialog(null, "  Invalid Card ID ! Please Enter Your
Correct Card Id ", "Message",
                    JOptionPane.INFORMATION_MESSAGE);

        }
```

```
        }
        // set credit limit
         if (a.getSource() == setcreditlimit_button)

        {
            int cardId = Integer.parseInt(cardid_input3_credit.getText());
            double creditlimit = Double.parseDouble(creditlimit_input1.getText());
            int graceperiod = Integer.parseInt(graceperiod_input1.getText());

            boolean cardIdExists = false;
            for (BankCard card : bankCards) {
                if (card instanceof CreditCard) {
                    if (card.getCardId() == cardId) {

                        ((CreditCard) card).setCreditLimit(creditlimit, graceperiod);
                        cardIdExists = true;
                        setcreditlimit_button.setEnabled(false);
                        UIManager.put("OptionPane.minimumSize", new Dimension(350, 100));
                        JOptionPane.showMessageDialog(null, "Credit Limit Is Set Successfully
!", "Message",
                                JOptionPane.INFORMATION_MESSAGE);

                        break;
                    }
                }
            }

        }
```

```
// for display method debit card
if (a.getSource() == display_button1) {

    for (BankCard card : bankCards) {
        if (card instanceof DebitCard) {
            System.out.println();
            System.out.println();
            System.out.println("CARD TYPE: DEBIT CARD");
            ((DebitCard) card).display();
            System.out.println();
            System.out.println();
        }

    }

}
// for display method credit card
if (a.getSource() == display_button2) {

    for (BankCard card : bankCards) {
        if (card instanceof CreditCard) {
            System.out.println();
            System.out.println();
            System.out.println("CARD TYPE: CREDIT CARD");
            ((CreditCard) card).display();
            System.out.println();
            System.out.println();
```

```
            }

        }

    }

  }

  public static void main(String args[]) {
    new BankGui();

  }
}
```