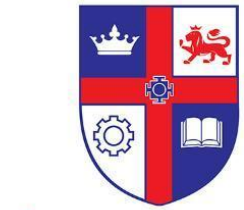
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2023-24 Spring

Student Name: Arpit Gupta

London Met ID: 22067731

College ID: NP01CP4A220145

Group: L1C6

Assignment Due Date: Wednesday, May 10, 2023

Assignment Submission Date: Tuesday, May 9, 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. Introduction to Project Content

This is the second coursework that is given to us for the module “Programming”. Creating a Graphical User Interface (GUI) was the aim of this coursework, which was the continuation of the previous coursework. The previous coursework motive was to write a program that allows to create objects of DebitCard and CreditCard. Withdrawing from DebitCard and setting the credit limit along with cancelling the credit limit for the Credit Card was also the motive of previous coursework. The present coursework has the same three classes present that were created during previous coursework and adding a GUI to it is the aim of this coursework. In the project folder, a new class named BankGUI was created that was used for creating the GUI, along with the three previously created classes.

The DebitCard and CreditCard were the subclasses of the BankCard class, but BankGUI was dependant on all of the classes. The BankGUI uses BankCard class to create an ArrayList that stores the object of DebitCard and CreditCard in it. To access all the methods inside of BankGUI from their classes respectively, Down Casting was employed into the project.

Different JComponents were used for creating the GUI. JFrame, JLabel, JTextField, JComboBox, JButtons and JSeparator are some of the JComponents that has been included in the creation of the GUI. All these components are defined above of the constructor. An ArrayList of BankCard type is also defined before creating the constructor. The constructor is where the GUI’s design has been completed.

There are different text fields for entering data, combo box for selecting the date and different buttons that perform various tasks according to their functionality. There are a total of 8 buttons in the GUI, Add Debit Card Button to add the debit card to the ArrayList, Add Credit Card Button to add the credit card to the ArrayList, Withdraw Button to withdraw a sum of amount from debit card, Set Credit Limit Button to set the credit limit to the credit card, Cancel Credit Card Button to cancel the credit card, Display Debit Button to display the details of debit card stored inside of the ArrayList, Display Credit Button to display the details of credit card stored inside of the ArrayList and a Clear Button

to clear all the JTextField and reset the value displayed in the JComboBox. Not only does it do all the functions, but these buttons also display a suitable message upon every action on them.

Validation for the text fields has also been implemented to make sure that correct type of data is entered in the text fields. A message is displayed regarding the error in entry of data. If the text fields are found empty, then the GUI displays a suitable message for it.

1.2. Tools Used for the Project

BlueJ: BlueJ is a beginner friendly software, specifically for writing Java codes, which makes running Java programs quicker and easier. Visualizing and interacting with objects is made easier and it allows to inspect the value of attributes of the object and call the methods on them (harleenk_99, 2023).

draw.io: draw.io is an open-sourced diagram and flowchart application that is used for designing tables, charts, and creating diagrams such as flowcharts, etc. It is an easy to use platform for creating and editing diagrams, as all the necessary tools are design layouts are available (Hope, 2020).

Moqups: moqups, similar to draw.io, is a web-based designing tool which is used for creating various designs for any sort of application. It is used to create a design for a software before the building of the software. It gives an idea about the looks of the software and allows to make changes before the actual software is created (Pietroluongo, 2022).

MS Word: MS Word is a software published by Microsoft. It allows to edit and create documents, letters and other important reports. It is an easy to use software that was released in 1983, developed by Charles Simonyi and Richard Brodie. It allows to convert the documents to various formats including PDFs, which is one of the widely used format for viewing documents and reports (Hope, 2021).

2. Class Diagram

A class diagram is a type of diagram that visualizes the program structure. It gives a short description about all the classes within it, along with the attributes and method present inside of each class. The bold arrow represents the direction from child class to parent class and it should always be pointed upwards. The other represents (dotted arrow) that the class is dependent on other classes to function properly. (Bhumika_Rani, 2023).

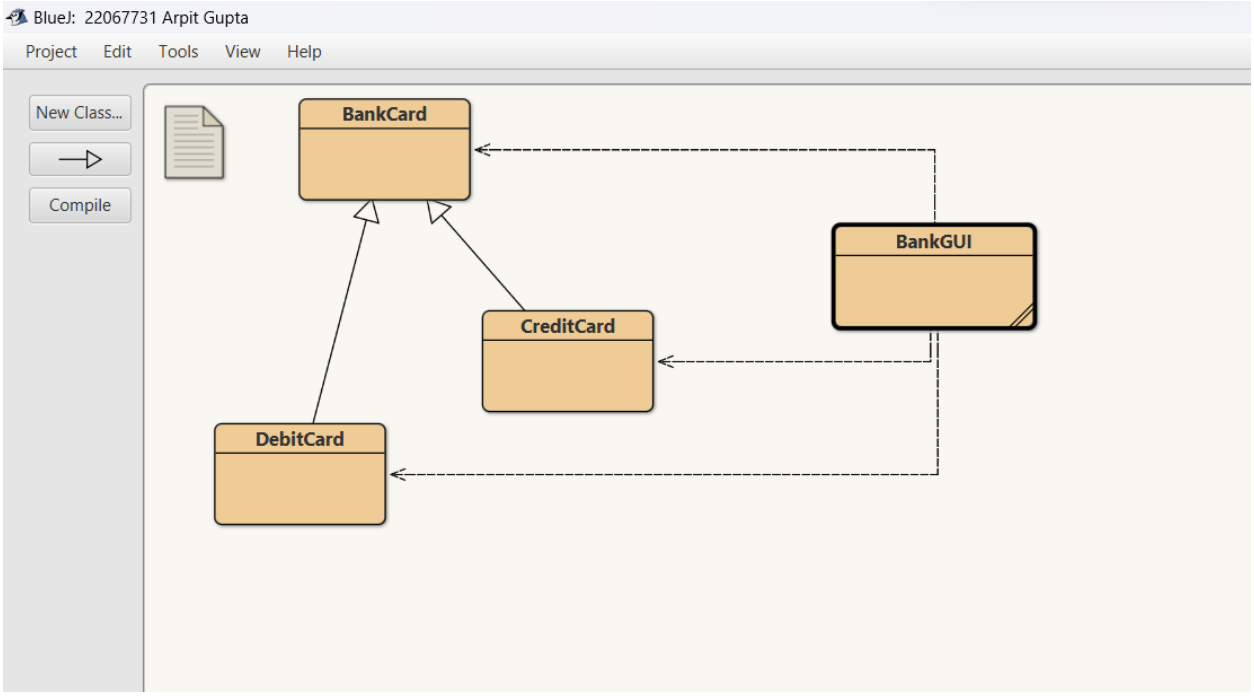


Figure 1. Screenshot of class diagram in BlueJ

2.1. Class Diagram for BankCard

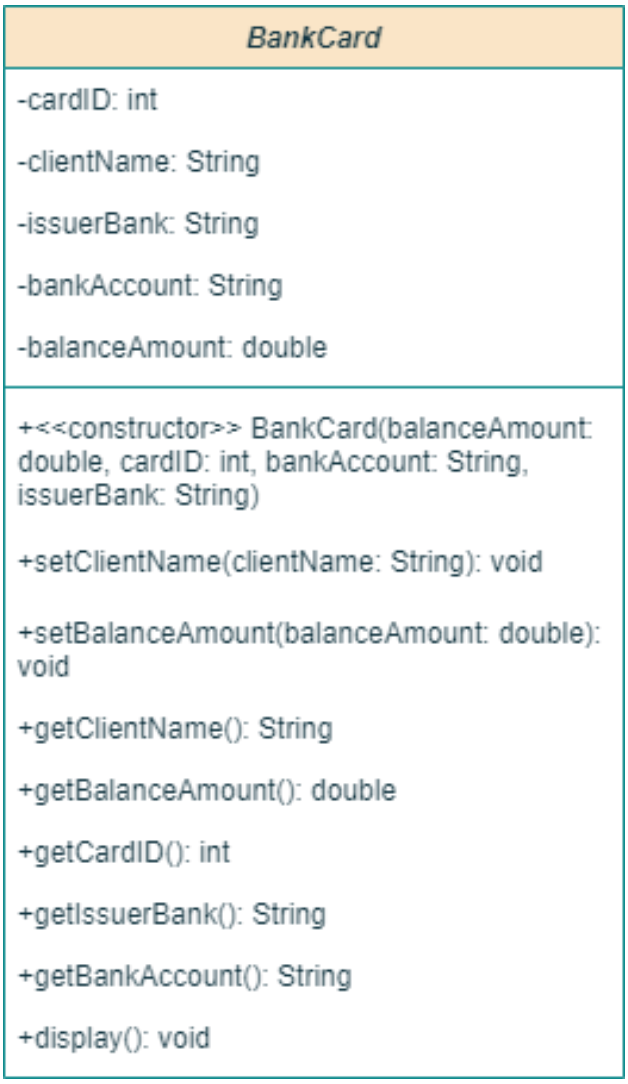


Figure 2. Class Diagram for BankCard

2.2. Class Diagram for DebitCard

Figure 3. Class Diagram for DebitCard

2.3. Class Diagram for CreditCard

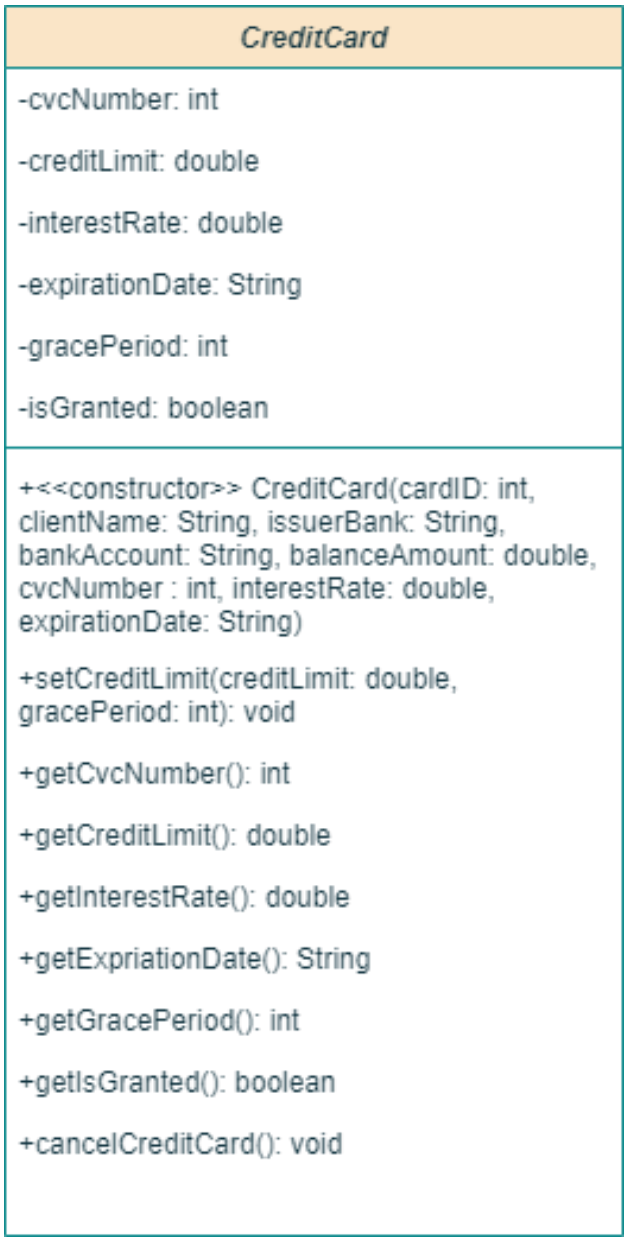


Figure 4. Class Diagram for CreditCard

2.4. Class Diagram for BankGUI

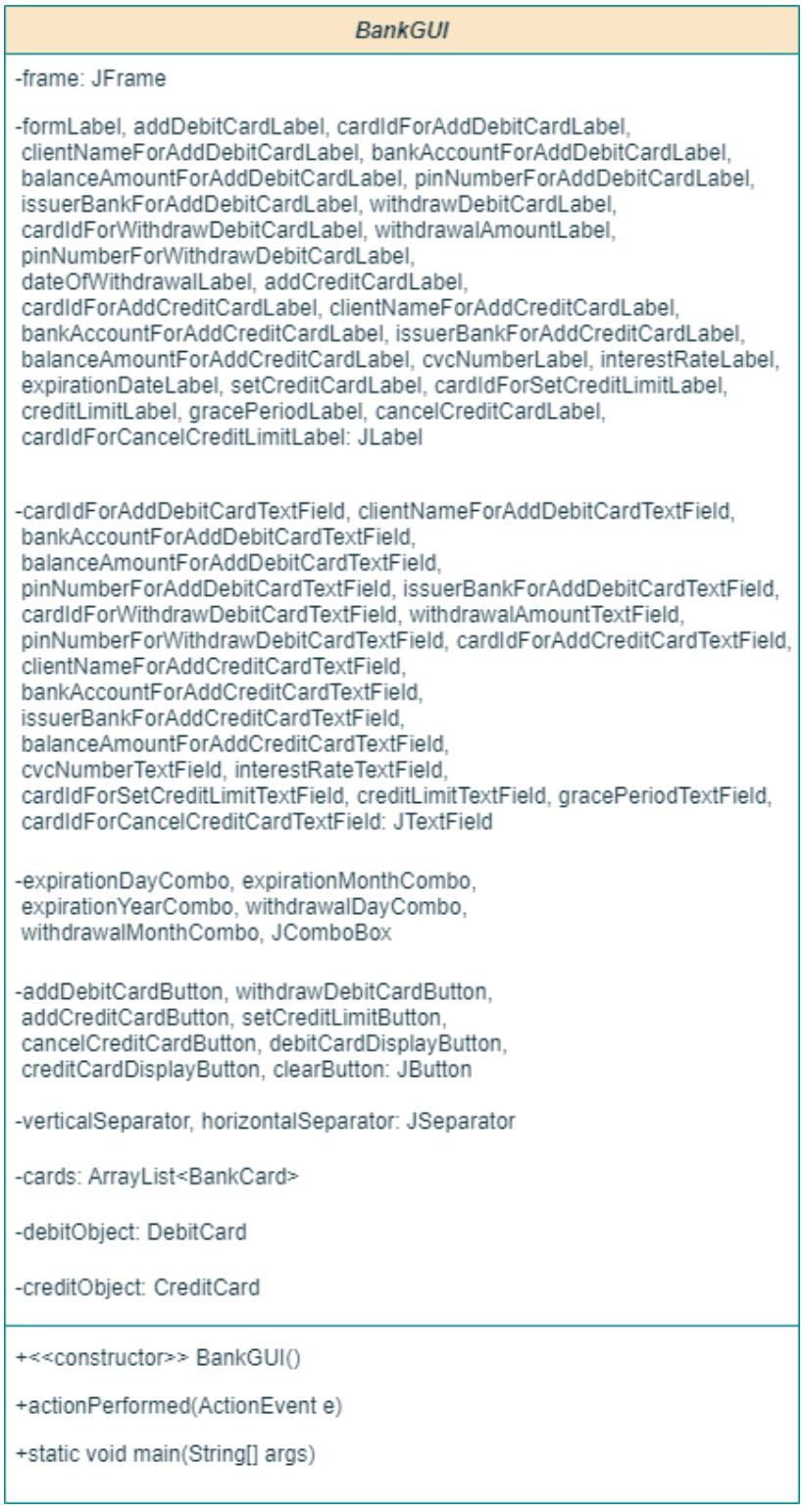


Figure 5. Class Diagram for BankGUI

2.5. Class Diagram

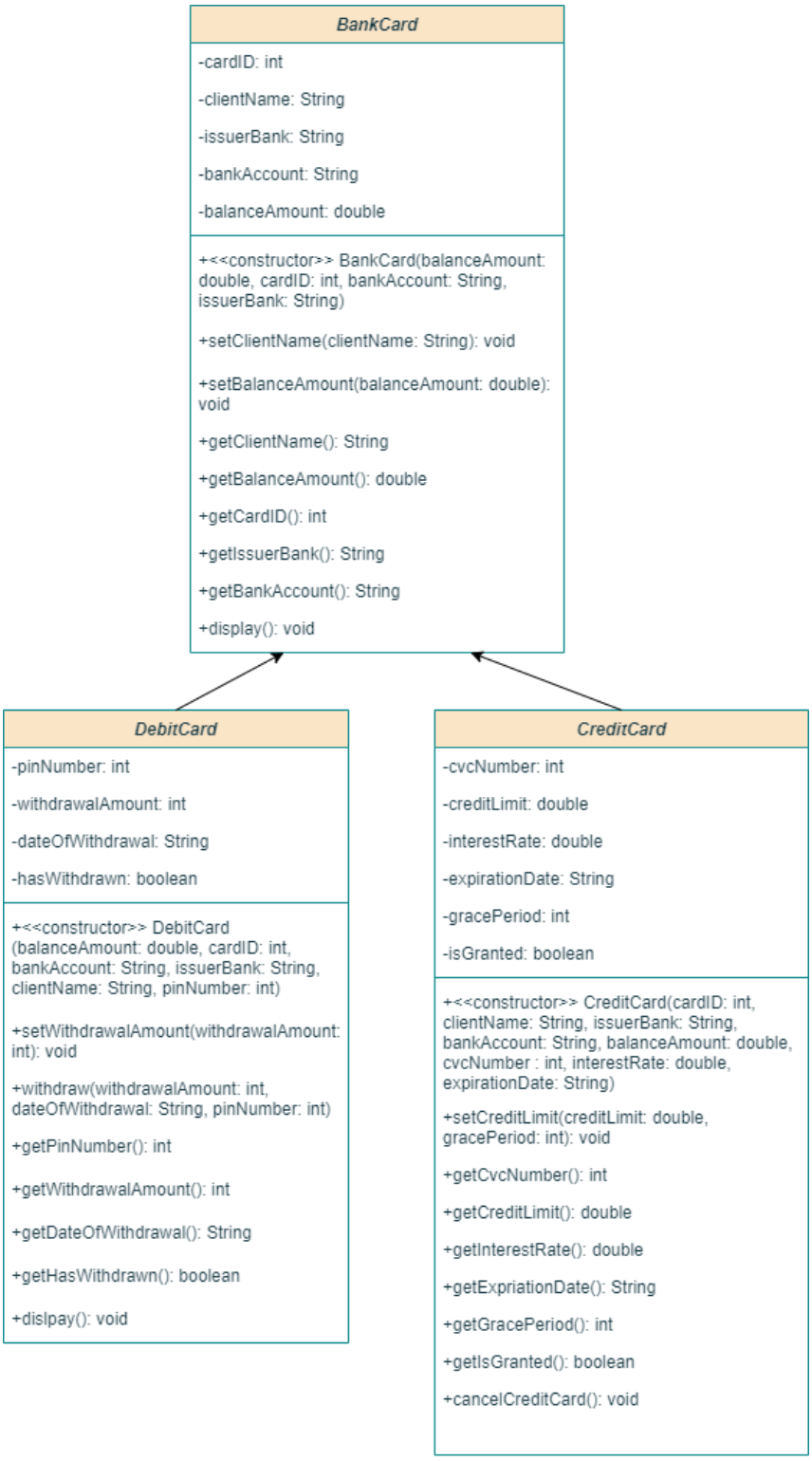


Figure 6. Class Diagram

3. Pseudocode

Pseudocode refers to a precise yet understandable explanation of what a program should do. It is written in natural English-like language rather than using any programming language. It is helpful to get basic idea about the program codes for those who don't know the working process of the code. The use of syntax is almost none as uses English text, so the pseudocode can not be complied or interpreted like codes of any programming languages. However, to show the neatness and proper understanding of the pseudocode, proper indentation is needed. Also, the beginning of each new line should be a verb for the clarity and making it easier to read. (Contributer, 2005)

The pseudocode for BankGUI class is written below:

IMPORT java packages that includes swing, awt, awt.event and util

CREATE class BankGUI implements ActionListener

DO

DECLARE frame as JFrame

DECLARE required number of labels as JLabel

DECLARE required number of text fields as JTextField

DECLARE required number of combo box as JComboBox

DECLARE required number of buttons as JButton

DECLARE verticalSeparator and horizontalSeparator as JSeparator

DECLARE cards as an ArrayList of BankCard

DECLARE debitObject as DebitCard

DECLARE creditObject as CreditCard

CREATE constructor for the class BankGUI

DO

INITIALIZE cards to new ArrayList of BankCard

INITIALIZE frame to a new JFrame with title "Bank GUI"

SET the title, size, position, Font and Color to all the JLabel and add them to the frame

SET the size, position and Font to all the JTextField and add them to the frame

DECLARE days as a list of String and add days from 1 to 31 as String

DECLARE months as a list of String and add all the months as String

DECLARE years as a list of String and years from 2000 to 2023 as String

SET the size, position and Font to all the JComboBox and add them to the frame

SET the name, size, position, Font and Color of all the JButton and add them to the frame

CREATE action listeners for all the JButton

SET the layout of frame to null

SET the background color of frame

SET the size of the frame

SET the default close operation to exit on close

SET the resizing of the frame to false

SET the visibility of the frame to true

END DO

CREATE a method actionPerformed(PASS parameter as ActionEvent e)

DO

IF button clicked is addDebitCardButton

ASSIGN value of cardIdForAddDebitCardTextField.getText() to cardIdForAddDebitCard as String

ASSIGN value of clientNameForAddDebitCardTextField.getText() to clientNameForAddDebitCard as String

ASSIGN value of bankAccountForAddDebitCardTextField.getText() to bankAccountForAddDebitCard as String

ASSIGN value balanceAmountForAddDebitCardTextField.getText() to balanceAmountForAddDebitCard as String

ASSIGN value of pinNumberForAddDebitCardTextField.getText() to pinNumberForAddDebitCard as String

ASSIGN value of issuerBankForAddDebitCardTextField.getText() to issuerBankForAddDebitCard as String

DECLARE hasNoException as boolean and set it to true

DECLARE counter as integer and set it to 0

IF required text fields are empty

DISPLAY “Empty fields have been found. Please fill up the requried fields: Card ID, Client Name, Bank Account, Balance Amount, PIN Number, Issuer Bank.” as an alert message

ELSE

TRY

CONVERT cardIdForAddDebitCard to integer and store the value in cardIdValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of card ID as a whole number." as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT balanceAmountForAddDebitCard to double and store the value in balanceAmountValue as double

CATCH NumberFomatException as ex

DISPLAY “Enter the value of Balance Amount as a decimal number.” as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT pinNumberForAddDebitCard to integer and store the value in pinNumberValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of PIN Number as a whole number.” as a message

SET hasNoException to false

IF hasNoException == true

CONVERT cardIdForAddDebitCard to integer and store the value in cardIdValue as integer

CONVERT balanceAmountForAddDebitCard to double and store the value in balanceAmountValue as double

CONVERT pinNumberForAddDebitCard to integer and store the value in pinNumberValue as integer

IF ArrayList cards is empty

CREATE debitObject of DebitCard and pass the parameters

ADD debitObject to the ArrayList cards

DISPLAY a message “Debit Card has been added.”

ELSE

FOR each card in ArrayList cards

IF card instance of DebitCard

DOWNCAST card as an object of DebitCard as debitObject

IF cardId of debitObject matches the cardId entered

DISPLAY “Debit Card for that Card ID already exists.” as a message

SET counter++

BREAK

END IF

END IF

END FOR

IF counter == 0

CREATE debitObject of DebitCard and pass the parameters

ADD debitObject to the ArrayList cards

DISPLAY a message “Debit Card has been added.”

END IF

END IF

END IF

END IF

END IF

IF button clicked is withdrawDebitCardButton

ASSIGN value of cardIdForWithdrawDebitCardTextField.getText() to cardIdForWithdrawDebitCard as String

ASSIGN value of withdrawalAmountTextField.getText() to withdrawalAmountForWithdrawDebitCard as String

ASSIGN value of pinNumberForWithdrawDebitCardTextField.getText() to pinNumberForWithdrawDebitCard as String

DECLARE withdrawalDay as String and covert the value from withdrawalDayCombo to String and store in it

DECLARE withdrawalMonth as String and covert the value from withdrawalMonthCombo to String and store in it

DECLARE withdrawalYear as String and covert the value from withdrawalYearCombo to String and store in it

SET withdrawalDate as withdrawalMonth + “ “ + withdrawalDay + “, “ + withdrawalYear

CONVERT withdrawalYear to integer and store it in withdrawalYearValue

CONVERT withdrawalDay to integer and store it in withdrawalDayValue

DECLARE hasNoException as boolean and set it to true

DECLARE counter as integer and set it to 0

IF required text fields are empty

DISPLAY “Empty fields have been found. Please fill up the required fields: Card ID, Withdrawal Amount, PIN Number.” as an alert message

ELSE

TRY

CONVERT cardIdForWithdrawDebitCard to integer and store the value in cardIdValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of card ID as a whole number." as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT withdrawalAmountForWithdrawDebitCard to double and store the value in withdrawalAmountValue as double

CATCH NumberFomatException as ex

DISPLAY “Enter the value of Withdrawal Amount as a whole number.” as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT pinNumberForWithdrawDebitCard to integer and store the value in pinNumberValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of PIN Number as a whole number.” as a message

SET hasNoException to false

IF hasNoException == true

IF withdrawalYearValue mod 4 is not equal to 0

IF withdrawalMonth is equal to “Apr”, “Jun”, “Sep” or “Nov”

IF withdrawalDayValue is equal to 31

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

IF withdrawalMonth is equal to “Feb”

IF withdrawalDayValue > 28

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

ELSE

IF withdrawalMonth is equal to “Apr”, “Jun”, “Sep” or “Nov”

IF withdrawalDayValue is equal to 31

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

IF withdrawalMonth is equal to “Feb”

IF withdrawalDayValue > 29

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

END IF

END IF

IF hasNoException == true

CONVERT cardIdForWithdrawDebitCard to integer and store the value in cardIdValue as integer

CONVERT withdrawalAmountForWithdrawDebitCard to double and store the value in withdrawalAmountValue as integer

CONVERT pinNumberForWithdrawDebitCard to integer and store the value in pinNumberValue as integer

FOR each card in ArrayList cards

IF card instance of DebitCard

DOWNCAST card as an object of DebitCard as debitObject

IF cardId of debitObject matches the cardIdValue

CALL withdraw (pass parameters) using debitObject

SET counter++

DECLARE withdrawn as boolean and call getHasWithdrawn() using debitObject and set it to withdrawn

DECLARE PinNumber as int and call getPinNumber() using debitObject and set it to PinNumber

IF withdrawn == false or PinNumber is not equal to pinNumberValue

DISPLAY "The PIN Number or the Withdrawal Amount entered doesn't meet the condition."

ELSE

DISPLAY details of the transaction

END IF

BREAK

END IF

END IF

END FOR

IF counter == 0

DISPLAY "There is no card registered to the provided ID." as a message

END IF

END IF

END IF

END IF

IF button clicked is addCreditCardButton

ASSIGN value of cardIdForAddCreditCardTextField.getText() to cardIdForAddCreditCard as String

ASSIGN value of clientNameForAddCreditCardTextField.getText() to clientNameForAddCreditCard as String

ASSIGN value of bankAccountForAddCreditCardTextField.getText() to bankAccountForAddCreditCard as String

ASSIGN value issuerBankForAddCreditCardTextField.getText() to issuerBankForAddCreditCardas String

ASSIGN value of balanceAmountForAddCreditCardTextField.getText() to balanceAmountForAddCreditCard as String

ASSIGN value of cvcNumberTextField.getText() to cvcNumberForAddCreditCard as String

ASSIGN value of interestRateTextField.getText() to interestRateForAddCreditCard as String

DECLARE expirationDay as String and covert the value from expirationDayCombo to String and store in it

DECLARE expirationMonth as String and covert the value from expirationMonthCombo to String and store in it

DECLARE expirationYear as String and covert the value from expirationYearCombo to String and store in it

SET expirationDate as expirationMonth + " " + expirationDay + ", " + expirationYear

CONVERT expirationDay to integer and store it in expirationDayValue

CONVERT expirationYear to integer and store it in expirationYearValue

DECLARE hasNoException as boolean and set it to true

DECLARE counter as integer and set it to 0

IF required text fields are empty

DISPLAY “Empty fields have been found. Please fill up the requried fields: Card ID, Client Name, Bank Account, Issuer Bank, Balance Amount, CVC Number, Interest Rate.” as an alert message

ELSE

TRY

CONVERT cardIdForAddCreditCard to integer and store the value in cardIdValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of card ID as a whole number." as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT balanceAmountForAddCreditCard to double and store the value in balanceAmountValue as double

CATCH NumberFomatException as ex

DISPLAY “Enter the value of Balance Amount as a decimal number.” as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT cvcNumberForAddCreditCard to integer and store the value in cvcNumberValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of CVC Number as a whole number.” as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT interestRateForAddCreditCard to double and store the value in interestRateValue as double

CATCH NumberFomatException as ex

DISPLAY “Enter the value of Interest Rate as a decimal number.” as a message

SET hasNoException to false

IF hasNoException == true

IF expirationYearValue mod 4 is not equal to 0

IF expirationMonth is equal to “Apr”, “Jun”, “Sep” or “Nov”

IF expirationDayValue is equal to 31

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

IF expirationMonth is equal to “Feb”

IF expirationDayValue > 28

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

ELSE

IF expirationMonth is equal to “Apr”, “Jun”, “Sep” or “Nov”

IF expirationDayValue is equal to 31

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

IF expirationMonth is equal to “Feb”

IF expirationDayValue > 29

DISPLAY "The date you chose doesn't exist. Enter a valid date." as a message

SET hasNoException to false

END IF

END IF

END IF

END IF

IF hasNoException == true

CONVERT cardIdForAddCreditCard to integer and store the value in cardIdValue as integer

CONVERT balanceAmountForAddCreditCard to double and store the value in balanceAmountValue as double

CONVERT cvcNumberForAddCreditCard to integer and store the value in cvcNumberValue as integer

CONVERT interestRateForAddCreditCard to double and store the value in interestRateValue as double

IF ArrayList cards is empty

CREATE creditObject of CreditCard and pass the parameters

ADD creditObject to the ArrayList cards

DISPLAY a message “Credit Card has been added.”

ELSE

FOR each card in ArrayList cards

IF card instance of CreditCard

DOWNCAST card as an object of CreditCard as creditObject

IF cardId of creditObject matches the cardIdValue

DISPLAY “Credit Card for that Card ID already exists.” as a message

SET counter++

BREAK

END IF

END IF

END FOR

IF counter == 0

CREATE creditObject of CreditCard and pass the parameters

ADD creditObject to the ArrayList cards

DISPLAY a message “Credit Card has been added.”

END IF

END IF

END IF

END IF

END IF

IF button clicked is setCreditLimitButton

ASSIGN value of cardIdForSetCreditLimitTextField.getText() to cardIdForSetCreditLimit as String

ASSIGN value of creditLimitTextField.getText() to creditLimitForSetCreditLimit as String

ASSIGN value of gracePeriodTextField.getText() to gracePeriodForSetCreditLimit as String

DECLARE hasNoException as boolean and set it to true

DECLARE counter as integer and set it to 0

IF required text fields are empty

DISPLAY “Empty fields have been found. Please fill up the requried fields: Card ID, Credit Limit, Grace Period.” as an alert message

ELSE

TRY

CONVERT cardIdForSetCreditLimit to integer and store the value in cardIdValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of card ID as a whole number." as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT creditLimitForSetCreditLimit to double and store the value in creditLimitValue as double

CATCH NumberFomatException as ex

DISPLAY "Enter the value of Credit Limit as a decimal number." as a message

SET hasNoException to false

IF hasNoException == true

TRY

CONVERT gracePeriodForSetCreditLimit to integer and store the value in gracePeriodValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of PIN Number as a whole number.” as a message

SET hasNoException to false

IF hasNoException == true

CONVERT cardIdForSetCreditLimit to integer and store the value in cardIdValue as integer

CONVERT creditLimitForSetCreditLimit to double and store the value in creditLimitValue as integer

CONVERT gracePeriodForSetCreditLimit to integer and store the value in gracePeriodValue as integer

FOR each card in ArrayList cards

IF card instance of CreditCard

DOWNCAST card as an object of CreditCard as creditObject

IF cardId of creditObject matches the entered cardIdValue

CALL setCreditLimit (pass parameters) using creditObject

IF creditObject.getIsGranted() == false

DISPLAY a message “The credit limit can not be issued."

ELSE

DISPLAY details of that card

END IF

SET counter++

BREAK

END IF

END IF

END FOR

IF counter == 0

DISPLAY "There is no card registered to the provided ID." as a message

END IF

END IF

END IF

END IF

IF button clicked is cancelCreditCardButton

ASSIGN value of cardIdForCancelCreditCardTextField.getText() to cardIdForCancelCreditCard as String

DECLARE hasNoException as boolean and set it to true

DECLARE counter as integer and set it to 0

IF required text fields are empty

DISPLAY “Empty fields have been found. Please fill the required fields: Card ID.” as an alert message

ELSE

TRY

CONVERT cardIdForCancelCreditCard to integer and store the value in cardIdValue as integer

CATCH NumberFomatException as ex

DISPLAY “Enter the value of card ID as a whole number." as a message

SET hasNoException to false

IF hasNoException == true

CONVERT cardIdForCancelCreditCard to integer and store the value in cardIdValue as integer

FOR each card in ArrayList cards

IF card instance of CreditCard

DOWNCAST card as an object of CreditCard as creditObject

IF cardId of creditObject matches the entered cardIdValue

CALL cancelCreditCard() using creditObject

DISPLAY "The credit card has been canceled.” as a message

SET counter++

BREAK

END IF

END IF

END FOR

IF counter == 0

DISPLAY "There is no card registered to the provided ID." as a message

END IF

END IF

END IF

END IF

IF button clicked is debitCardDisplayButton

DECLARE counter as integer and set it to 0

IF ArrayList cards is empty

DISPLAY "No debit card have been added yet" as message

ELSE

FOR each card in ArrayList cards

IF card instance of DebitCard

DOWNCAST card as an object of DebitCard as debitObject

PRINT “Details of Debit Card: “ + debitObject.getCardID()

CALL display() method using debitObject

PRINT empty line

SET counter++

END IF

END FOR

IF counter == 0

DISPLAY "No debit card have been added yet." as a message

END IF

END IF

END IF

IF button clicked is creditCardDisplayButton

DECLARE counter as integer and set it to 0

IF ArrayList cards is empty

DISPLAY "No credit card have been added yet" as message

ELSE

FOR each card in ArrayList cards

IF card instance of CreditCard

DOWNCAST card as an object of CreditCard as creditObject

PRINT “Details of Credit Card: “ + creditObject.getCardID()

CALL display() method using creditObject

PRINT empty line

SET counter++

END IF

END FOR

IF counter == 0

DISPLAY "No credit card have been added yet." as a message

END IF

END IF

END IF

IF button clicked is clearButton

SET text for cardIdForAddDebitCardTextField to null

SET text for clientNameForAddDebitCardTextField to null

SET text for bankAccountForAddDebitCardTextField to null

SET text for balanceAmountForAddDebitCardTextField to null

SET text for pinNumberForAddDebitCardTextField to null

SET text for issuerBankForAddDebitCardTextField to null

SET text for cardIdForWithdrawDebitCardTextField to null

SET text for withdrawalAmountTextField to null

SET text for pinNumberForWithdrawDebitCardTextField to null

SET text for cardIdForAddCreditCardTextField to null

SET text for clientNameForAddCreditCardTextField to null

SET text for bankAccountForAddCreditCardTextField to null

SET text for issuerBankForAddCreditCardTextField to null

SET text for balanceAmountForAddCreditCardTextField to null

SET text for cvcNumberTextField to null

SET text for interestRateTextField to null

SET text for cardIdForSetCreditLimitTextField to null

SET text for creditLimitTextField to null

SET text for gracePeriodTextField to null

SET text for cardIdForCancelCreditCardTextField to null

SET selected index for expirationDayCombo to 0

SET selected index for expirationMonthCombo to 0

SET selected index for expirationYearCombo to 0

SET selected index for withdrawalDayCombo to 0

SET selected index for withdrawalMonthCombo to 0

SET selected index for withdrawalYearCombo to 0

END IF

END DO

CREATE main method for BankGUI

DO

DECLARE bankGUIObject as a new object of BankGUI

END DO

END DO

4. Method Description

The methods that have been used in BankGUI class are listed below:

4.1. actionPerformed (ActionEvent e)

All the event handling within the program are to be performed using this method. When the button is clicked inside of the GUI, an action is performed related to the button clicked. It occurs as this method searches for the reference of the button and then performs action for that button. For example, when the button of Add Debit Card is clicked on, the program searches for the source of that button and then executes the codes written for that button displaying a suitable message and doing some function.

There is a total of 8 buttons that have some functions present within this method. They are Add Debit Card, Add Credit Card, Withdraw, Set Credit Limit, Cancel Credit Card, Display Debit, Display Credit and Clear. All these buttons have different functions as their name suggests. The descriptions of the buttons are as follows:

addDebitCardButton: On click of the button, the debit card is added to the ArrayList of BankCard. It takes 6 values as input, namely Card ID, Client Name, Bank Account, Balance Amount, PIN Number and Issuer Bank. It checks whether the data entered in the text fields are valid and if valid, then displays a suitable message, else an error message is displayed.

addCrerditCardButton: On click of the button, the credit card is added to the ArrayList of BankCard. It takes 8 values as input, namely Card ID, Client Name, Bank Account, Issuer Bank, Balance Amount, CVC Number, Interest Rate and Expiration Date. It checks whether the data entered in the text fields are valid and if valid, then displays a suitable message, else an error message is displayed. Also, the combo box is restricted to the date that exists; any date such as February 30 chosen throws an error message.

withdrawDebitCardButton: On click of the button, the Card ID, Withdrawal Amount, PIN Number and Date of Withdrawal are taken as input. An error message is displayed whenever data entered in the text field goes beyond the limitation of the program. If not, it checks whether the Card ID entered exists or not. If it exists then it checks for the PIN Number and Withdrawal Amount and then amount is withdrawn if the data entered correctly. If the details entered is not same as that of the existing details, then an error message is displayed.

setCreditLimitButton: On click of the button, the Card ID, Credit Limit and Grace Period are taken as input. It checks whether the data entered in the text field meet the validation or not. If not, then it displays an error message. If the details entered meets the requirements then the Credit Limit is set.

cancelCreditCard: On click of the button, the Card ID is taken as input and the validation is done. If the Card ID entered is of correct data type, then it is checked whether the Card ID has been registered before or no. If registered then the credit card is cancelled. If not registered then an error message is displayed.

debitCardDisplayButton: On click of the button, it displays the information of all the debit card added in the ArrayList. If the ArrayList is empty when the button is clicked, a suitable message is displayed.

creditCardDisplayButton: On click of the button, it displays the information of all the credit card added in the ArrayList. If the ArrayList is empty when the button is clicked, a suitable message is displayed.

clearButton: On click of the button, all the text fields are set to empty and all the combo box are reset to its initial value.

4.2. static void main (String[] args)

This method is the main method of the BankGUI class and is present inside of every class that is created. Inside of this method, an object has been created using BankGUI constructor which has been used to execute the program without any error.

5. Testing

Testing is the process of continuously executing a program in order to detect errors and to check if the program does what it is supposed to do. It is done to prevent the occurrence of an error after the system is in use in daily life. During testing, if the result that occurred is same as the expected result, then it is considered to be a successful test (RajKumar, 2023).

5.1. Test 1: To compile and run the program using command prompt

Test No.	**1**
Objective	To compile and run the program using command prompt
Action	→ Open the command prompt and go to the directory of the project folder. → To compile the codes: javac BankGUI.java was written → To run the codes: java BankGUI.java was written
Expected Result	The codes of the program would be compiled and the GUI would be opened.
Actual Result	The codes of the program were compiled and the GUI was opened.
Conclusion	The test is successful

Table 1. To compile and run the program using command prompt

Output Result:

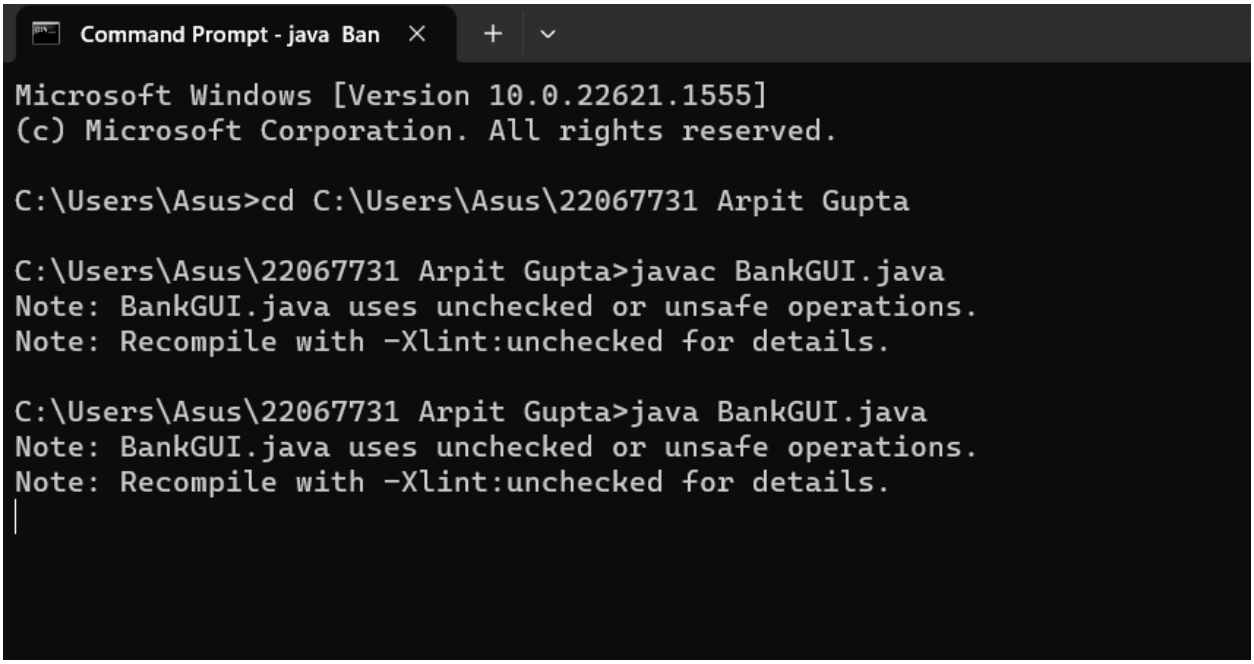


Figure 7. Command to run compile and java program through terminal

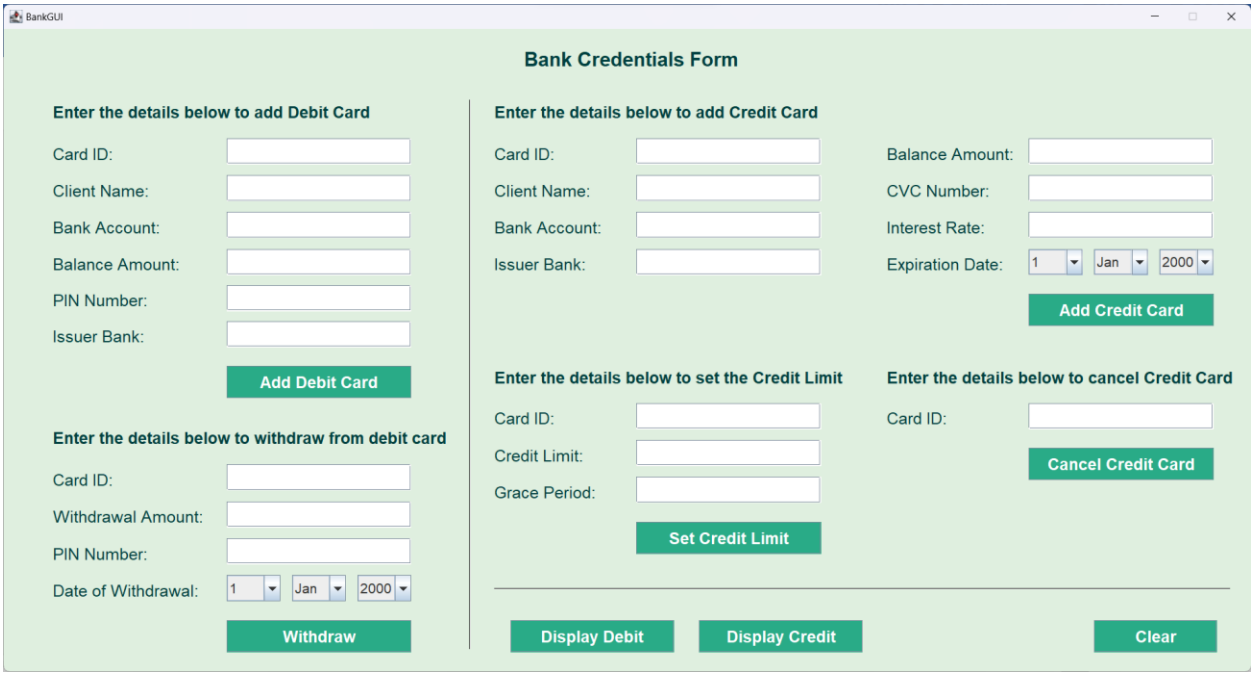


Figure 8. Output after compiling and running through terminal

5.2. Test 2: To check the functionality of the buttons

5.2.1. To add a card for debit card

Test No.	**2.1**
Objective	To add a card for debit card
Action	→ The required text fields are filled for adding debit card Card ID: “7000” Client Name: “John Smith” Bank Account: “900091” Balance Amount: “100000.0” PIN Number: “1014” Issuer Bank: “Nepal Rastriya Bank” →Click on Add Debit Card Button
Expected Result	The card would be added to the ArrayList and a pop-up message telling “Debit Card has been added.” would be displayed.
Actual Result	The card was added to the ArrayList and a pop-up message telling “Debit Card has been added.” was displayed.
Result	The test is successful.

Table 2. Add a debit card

Output Result:

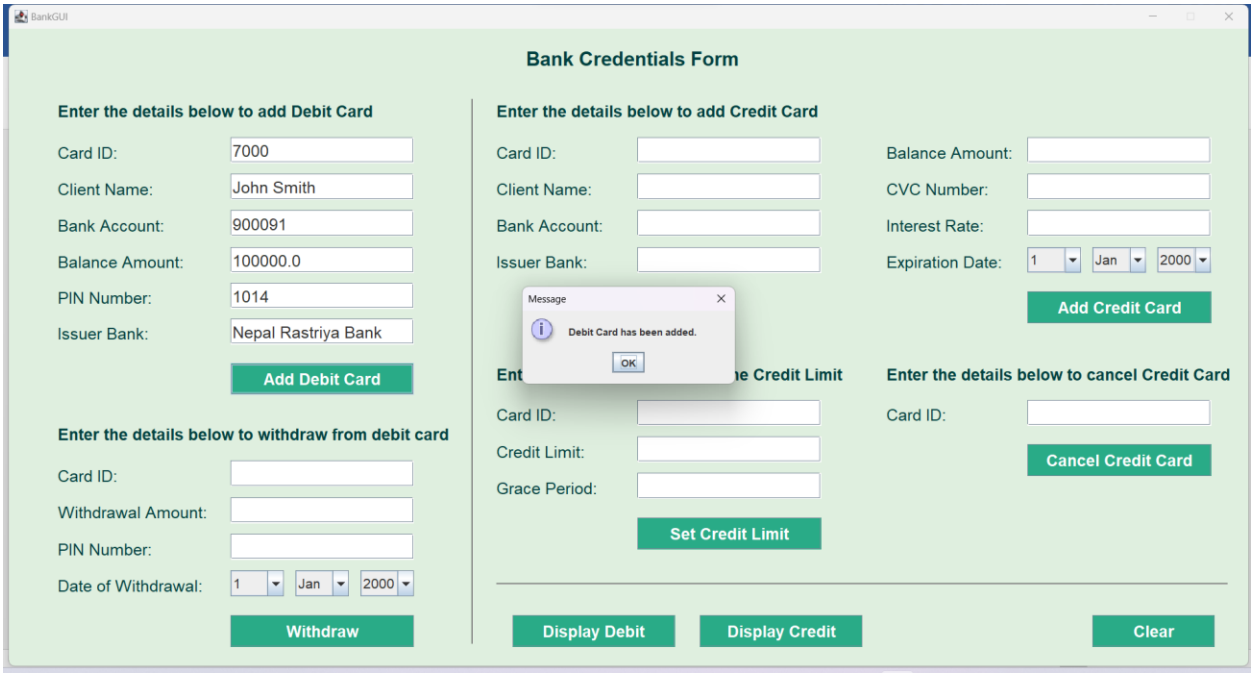


Figure 9. Debit card added

5.2.2. To add a card for credit card

Test No.	**2.2**
Objective	To add a card for credit card
Action	→ The required text fields are filled for adding credit card Card ID: “8000” Client Name: “Harry Brook” Bank Account: “900092” Issuer Bank: “Nepal Rastriya Bank” Balance Amount: “50000.0” CVC Number: “563” Interest Rate: “7.5” Expiration Date: 29-Feb-2020 →Click on Add Credit Card Button
Expected Result	The card would be added to the ArrayList and a pop-up message telling “Credit Card has been added.” would be displayed.
Actual Result	The card was added to the ArrayList and a pop-up message telling “Credit Card has been added.” was displayed.
Result	The test is successful.

Table 3. Add a credit card

Output Result:

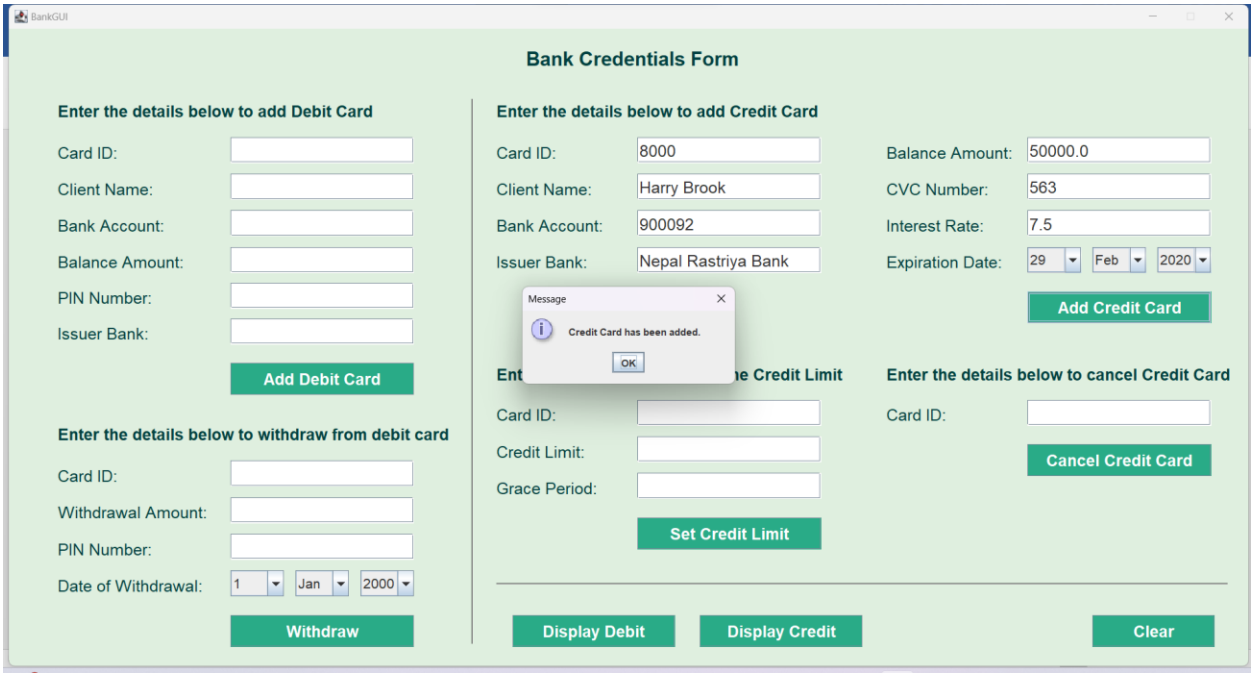


Figure 10. Credit Card added

5.2.3. To withdraw amount from debit card

Test No.	**2.3**
Objective	To withdraw amount from debit card
Action	→ The required text fields are filled for withdrawing from debit card Card ID: “7000” Withdrawal Amount: “20000” PIN Number: “1014” Date of Withdrawal: 5-May-2023 →Click on Withdraw Button
Expected Result	The amount would be withdrawn from the debit card and the details of the transaction would be displayed as a pop-up message.
Actual Result	The amount was withdrawn from the debit card and the details of the transaction was displayed as a pop-up message.
Result	The test is successful.

Table 4. Withdraw from debit card

Output Result:

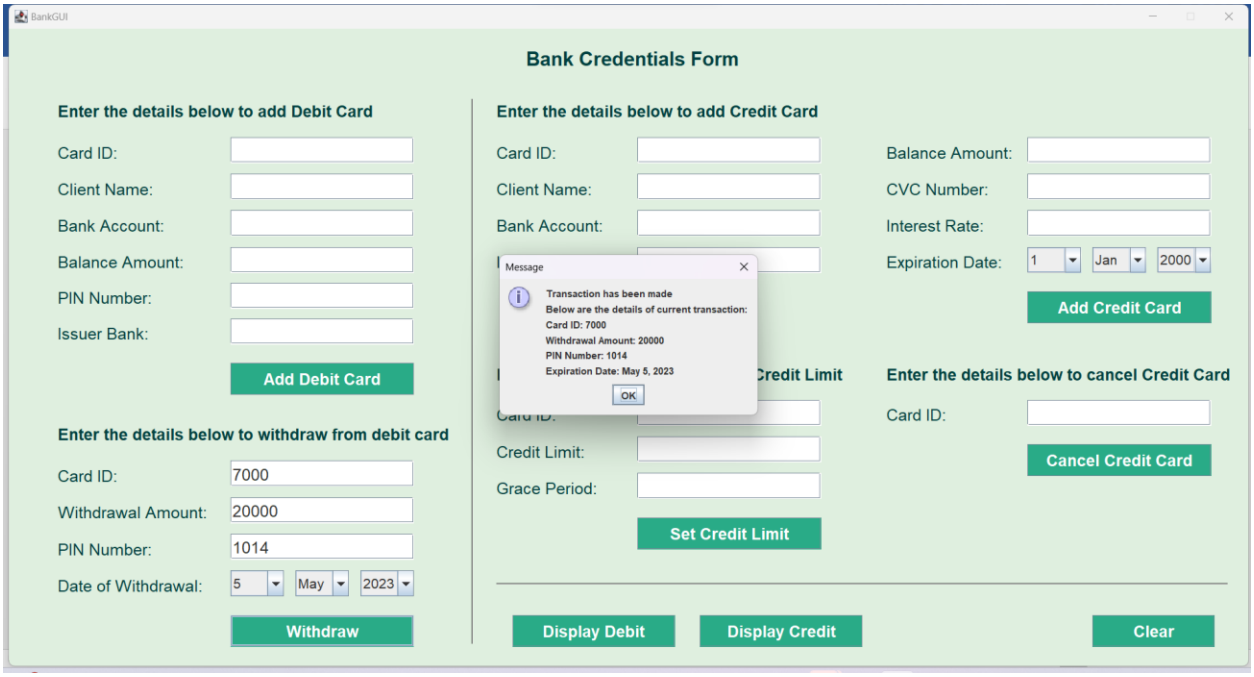


Figure 11. Withdraw from Debit Card

5.2.4. To set the credit limit to a credit card

Test No.	**2.4**
Objective	To set the credit limit to a credit card
Action	→ The required text fields are filled for setting the credit limit Card ID: “8000” Credit Limit: “75000” Grace Period: “30” →Click on Set Credit Limit Button
Expected Result	The credit limit would be set to the credit card and the details for it would be displayed as a pop-up message.
Actual Result	The credit limit was set to the credit card and the details for it was displayed as a pop-up message.
Result	The test is successful.

Table 5. Set the credit limit

Output Result:

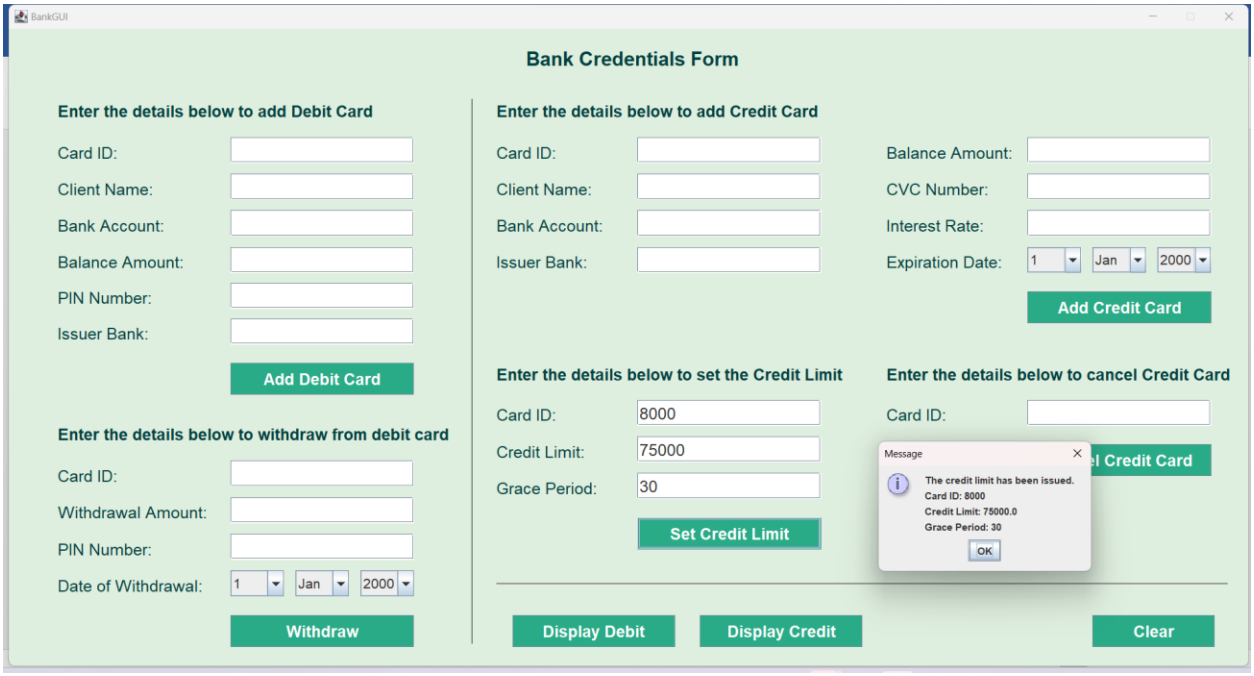


Figure 12. Set the credit limit

5.2.5. To cancel a credit card

Test No.	**2.5**
Objective	To cancel a credit card
Action	→ The required text fields are filled for cancelling a credit card Card ID: “8000” →Click on Cancel Credit Card Button
Expected Result	The credit card would be cancelled and the card ID for it would be displayed as a pop-up message.
Actual Result	The credit card was cancelled and the card ID for it was displayed as a pop-up message.
Result	The test is successful.

Table 6. Cancel a credit card

Output Result

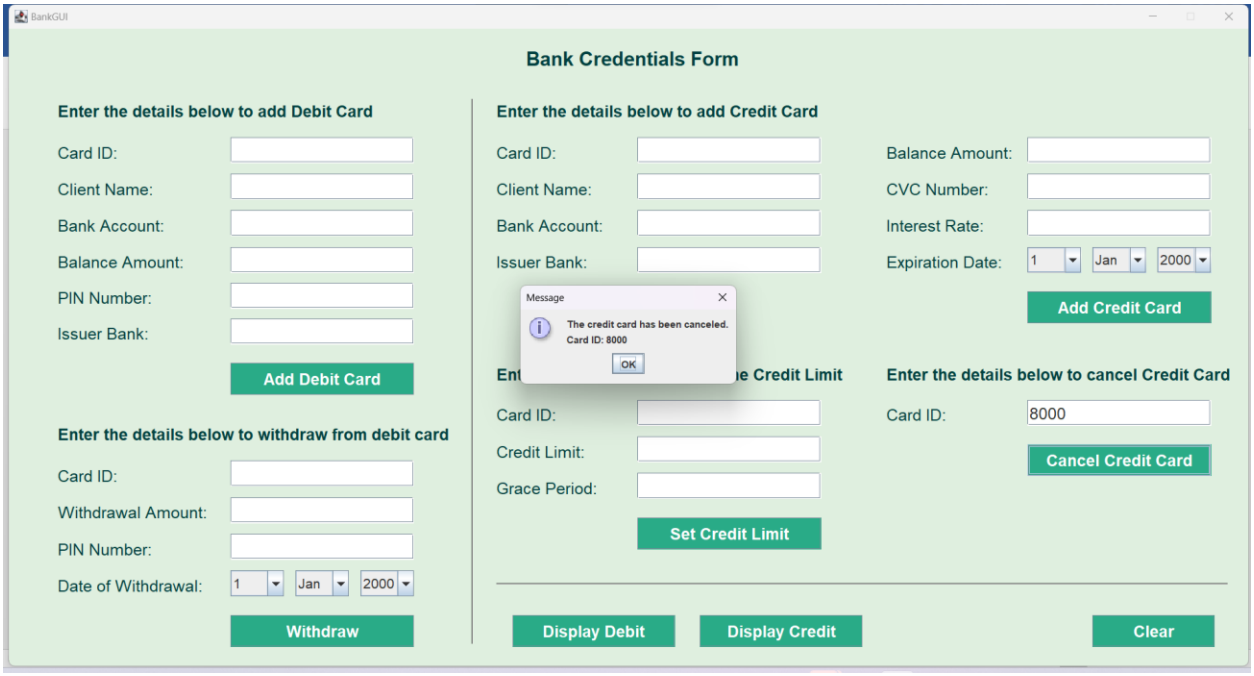


Figure 13. Cancel a credit card

5.3. Test Validations

5.3.1. Tests for empty text fields

Test No.	**3.1**
Objective	To test for empty text fields
Action	→ All the text fields are left empty and the respective buttons are clicked. • Add Debit Card Button • Withdraw Button • Add Credit Card Button • Set Credit Limit Button • Cancel Credit Limit Button • Display Debit Button • Display Credit Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 7. Test for empty text fields.

Output Result 1 (Click on Add Debit Card):

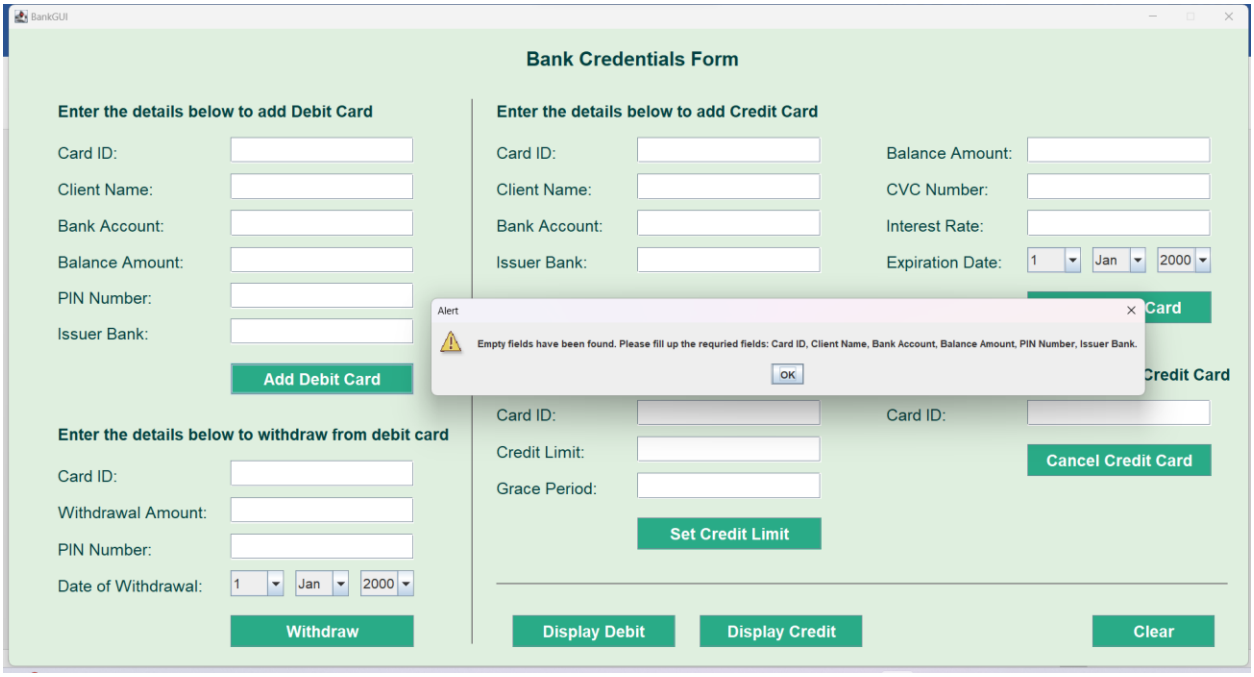


Figure 14. Add Debit Card with empty Text Fields

Output Result 2 (Click on Withdraw):

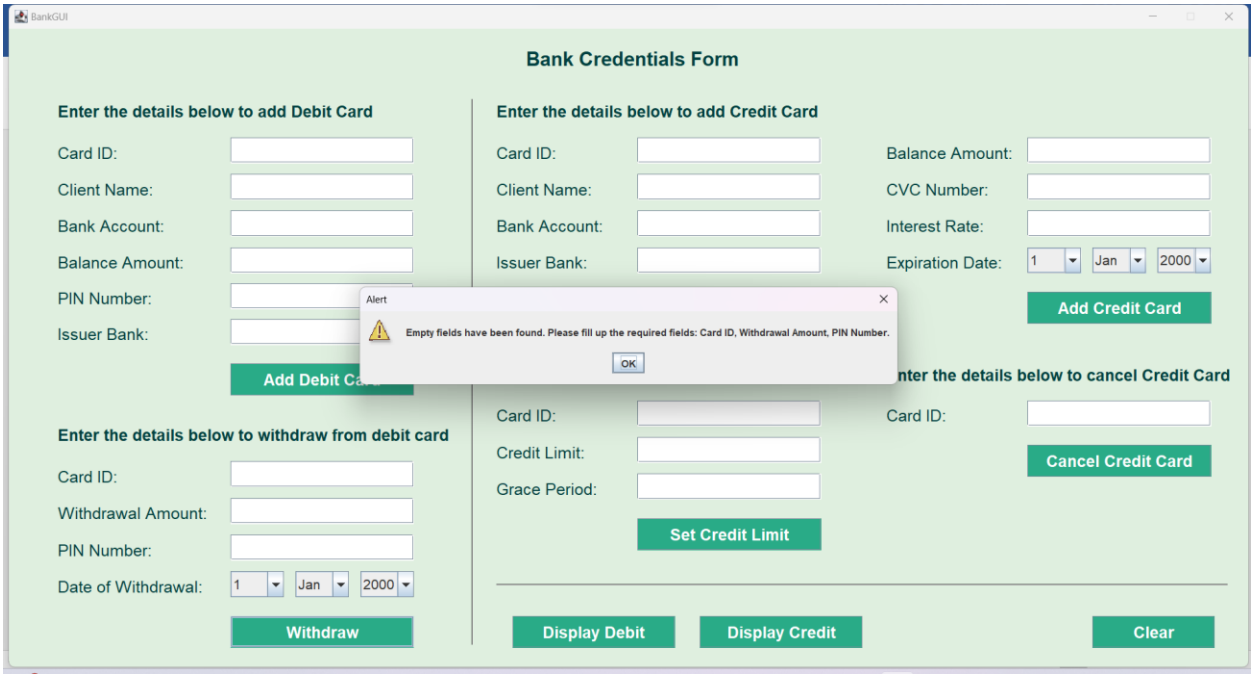


Figure 15. Withdraw with empty Text Fields

Output Result 3 (Click on Add Credit Card):

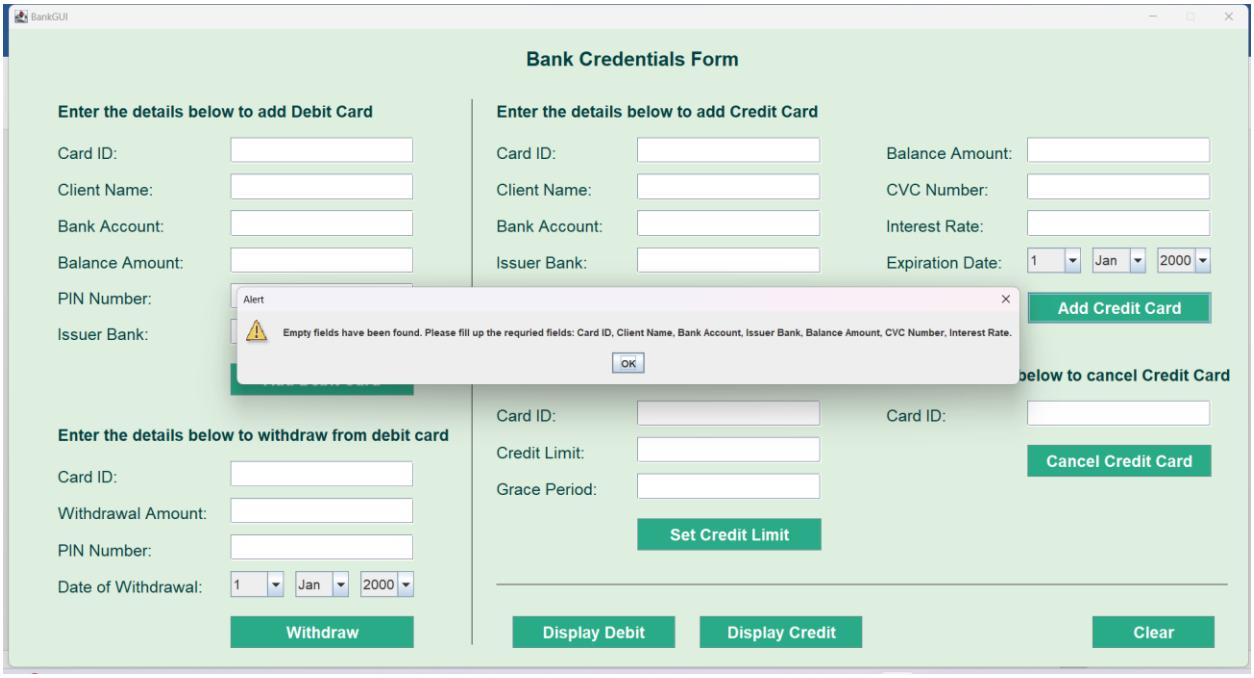


Figure 16. Add Credit Card with empty Text Fields

Output Result 4 (Click on Set Credit Limit):

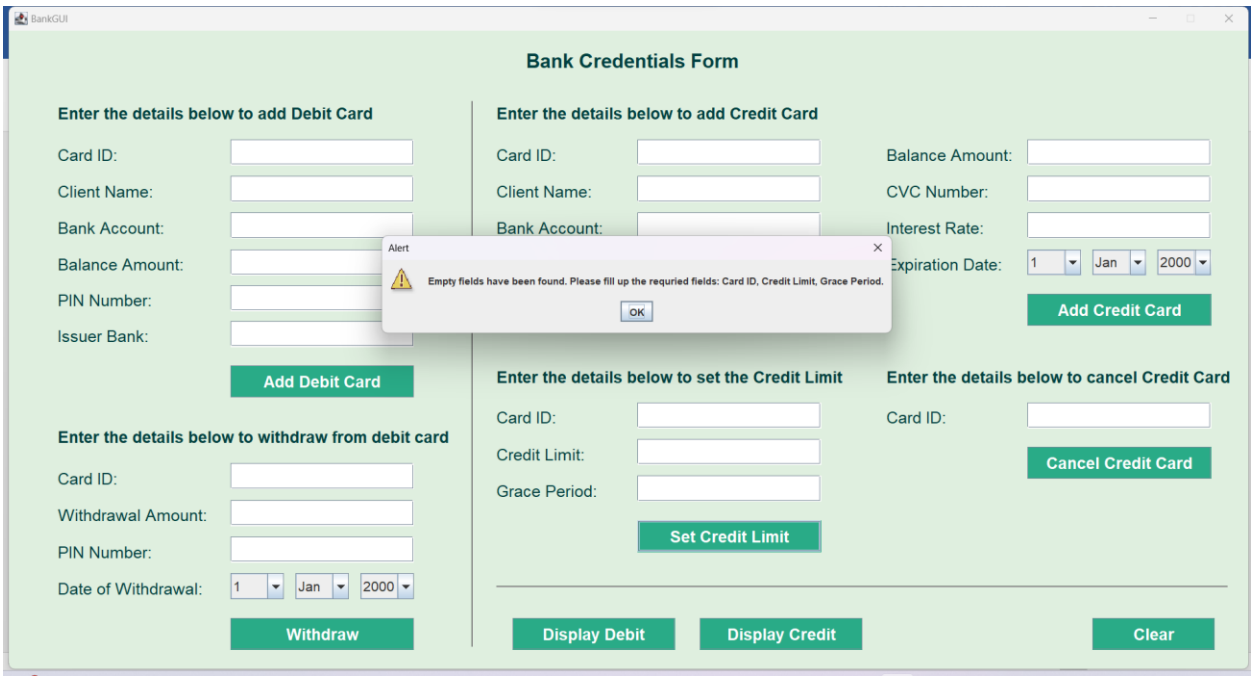


Figure 17. Set Credit Limit with empty Text Fields

Output Result 5 (Click on Cancel Credit Card):

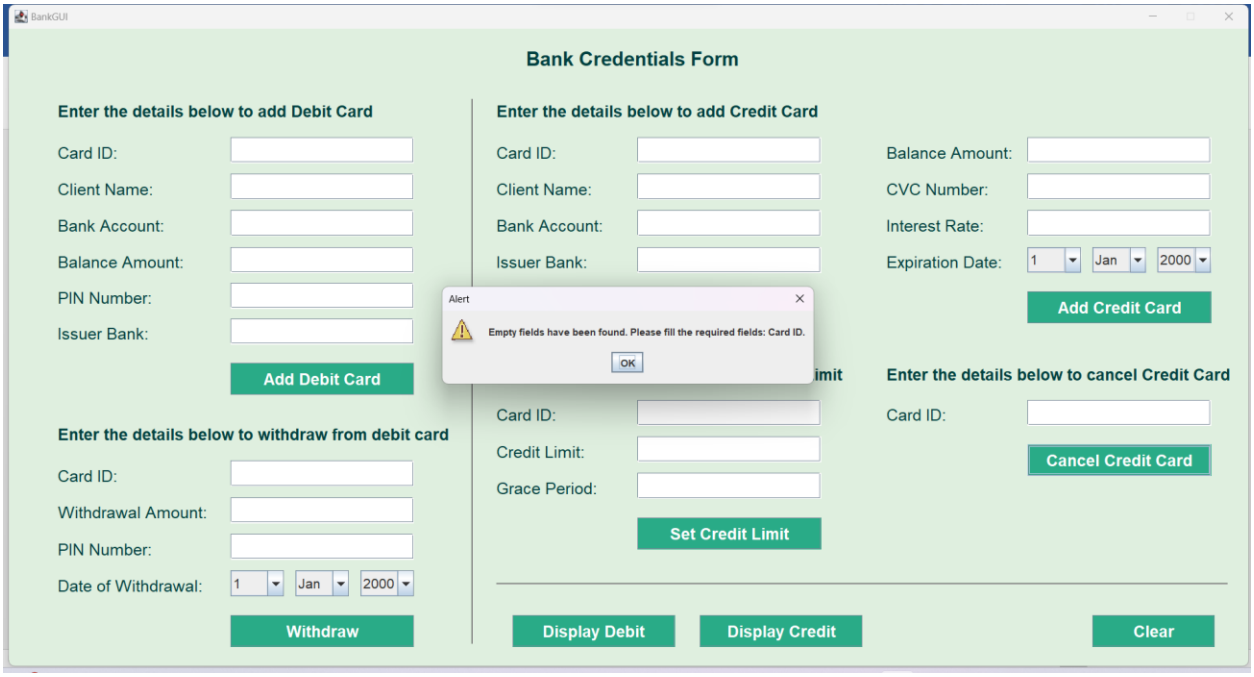


Figure 18. Cancel Credit Card with empty Text Fields

Output Result 6 (Click on Display Debit):

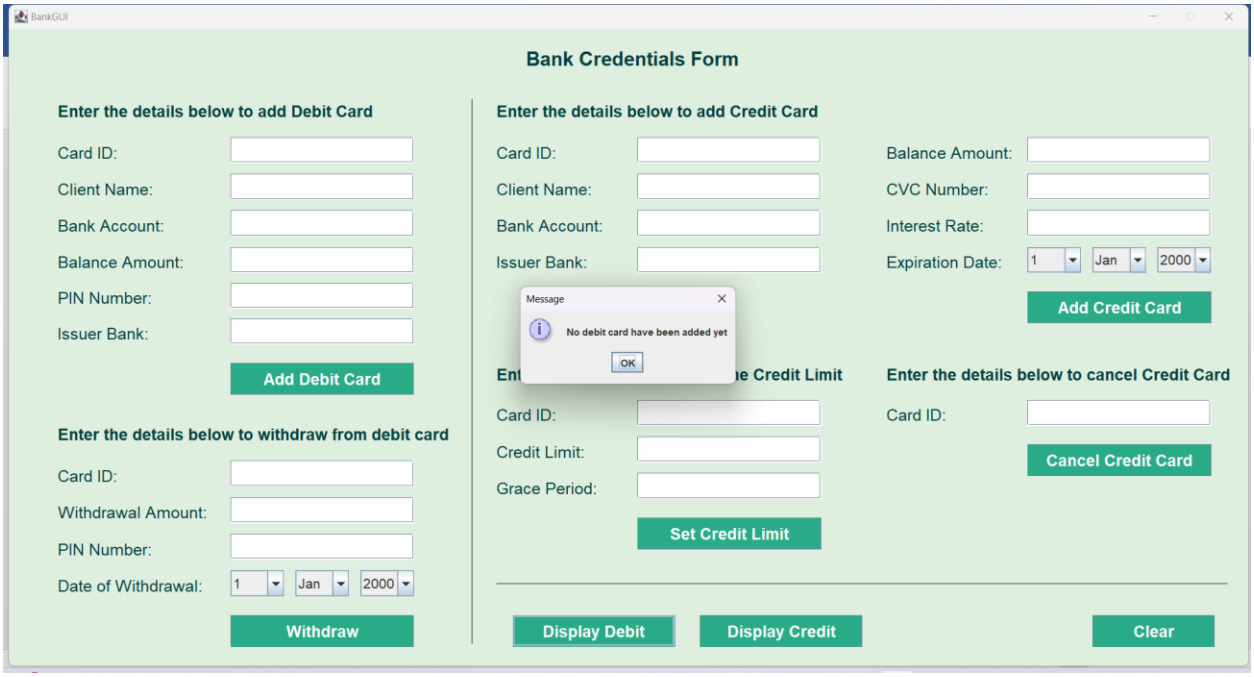


Figure 19. Display Debit without adding any debit card

Output Result 7 (Click on Display Credit):

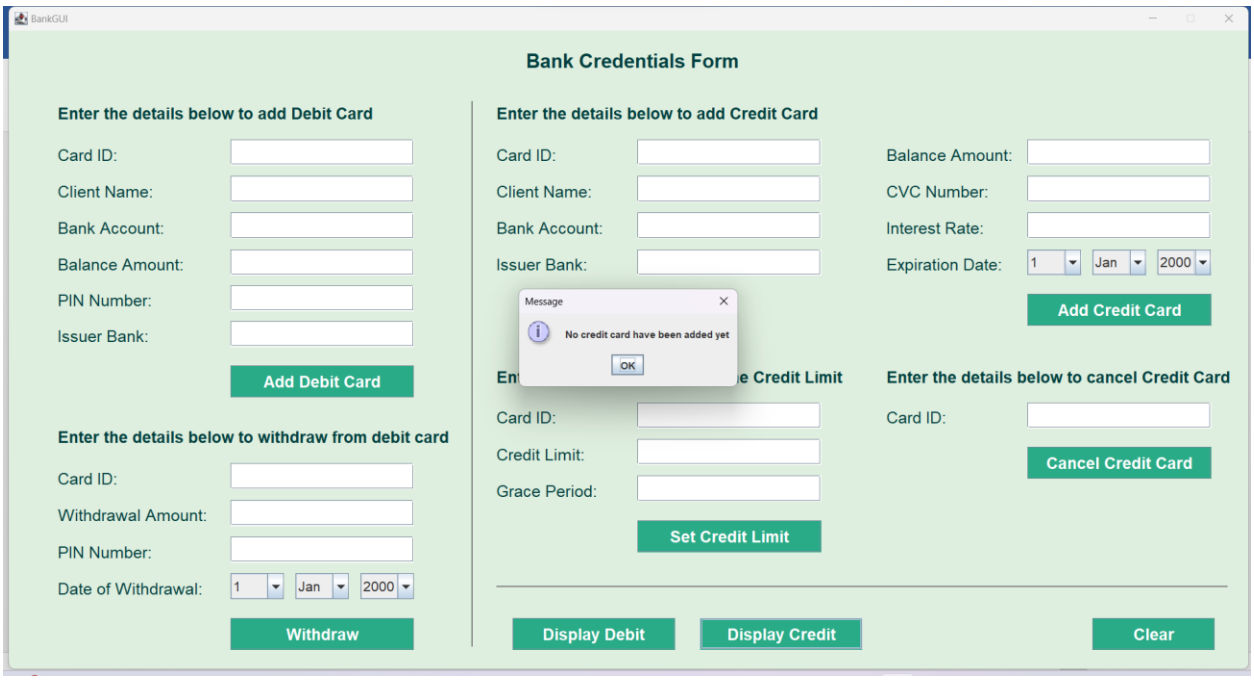


Figure 20. Display Credit without adding any credit card

5.3.2. To enter invalid input for adding Debit Card

a. Input invalid Card ID for adding Debit Card

Test No.	**3.2 (a)**
Objective	To enter invalid Card ID as input for adding Debit Card
Action	→ Wrong value for Card ID is entered for add debit card • Card ID: “AAAA” • Client Name: “John Smith” • Bank Account: “900091” • Balance Amount: “100000.0” • PIN Number: “1014” • Issuer Bank: “Nepal Rastriya Bank” → Click on Add Debit Card Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 8. Invalid Card Id for adding debit card

Output Result:

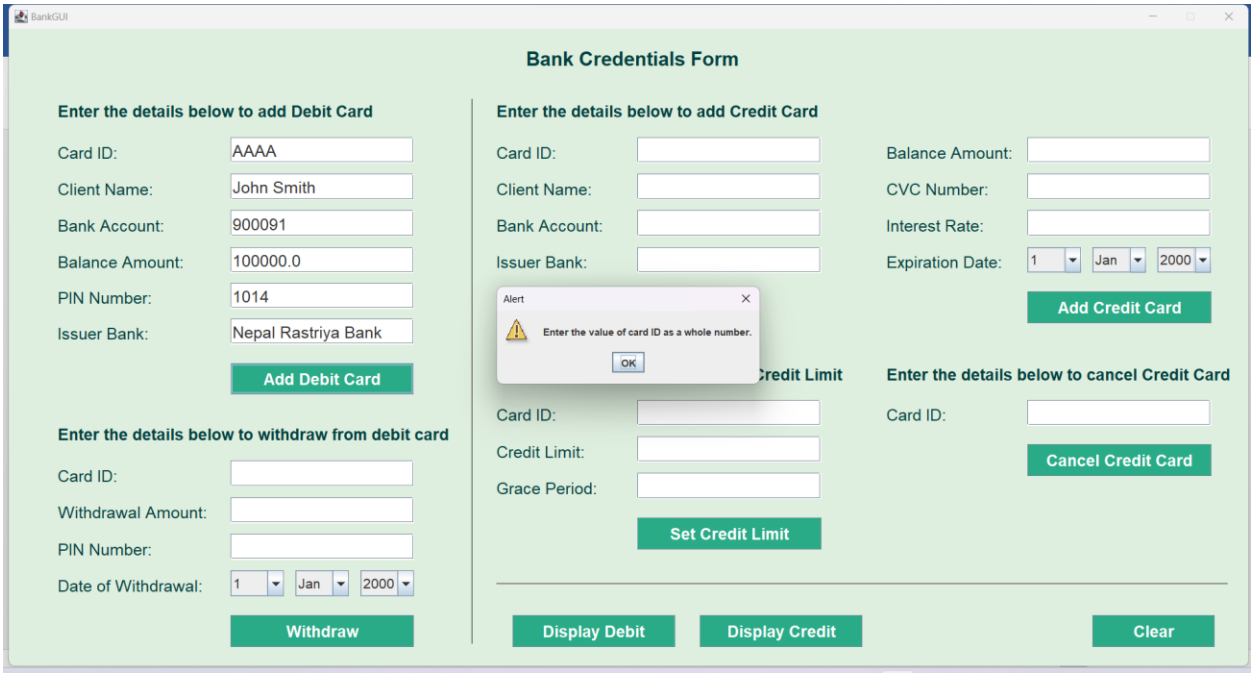


Figure 21. Screenshot of adding invalid Card ID (Debit Card)

b. Input invalid Balance Amount for adding debit card

Test No.	**3.2 (b)**
Objective	To enter invalid Balance Amount as input for adding Debit Card
Action	→ Wrong value for Balance Amount is entered for add debit card • Card ID: “1000” • Client Name: “John Smith” • Bank Account: “900091” • Balance Amount: “AAAA” • PIN Number: “1014” • Issuer Bank: “Nepal Rastriya Bank” → Click on Add Debit Card Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 9. Invalid Balance Amount for adding Debit Card

Output Result:

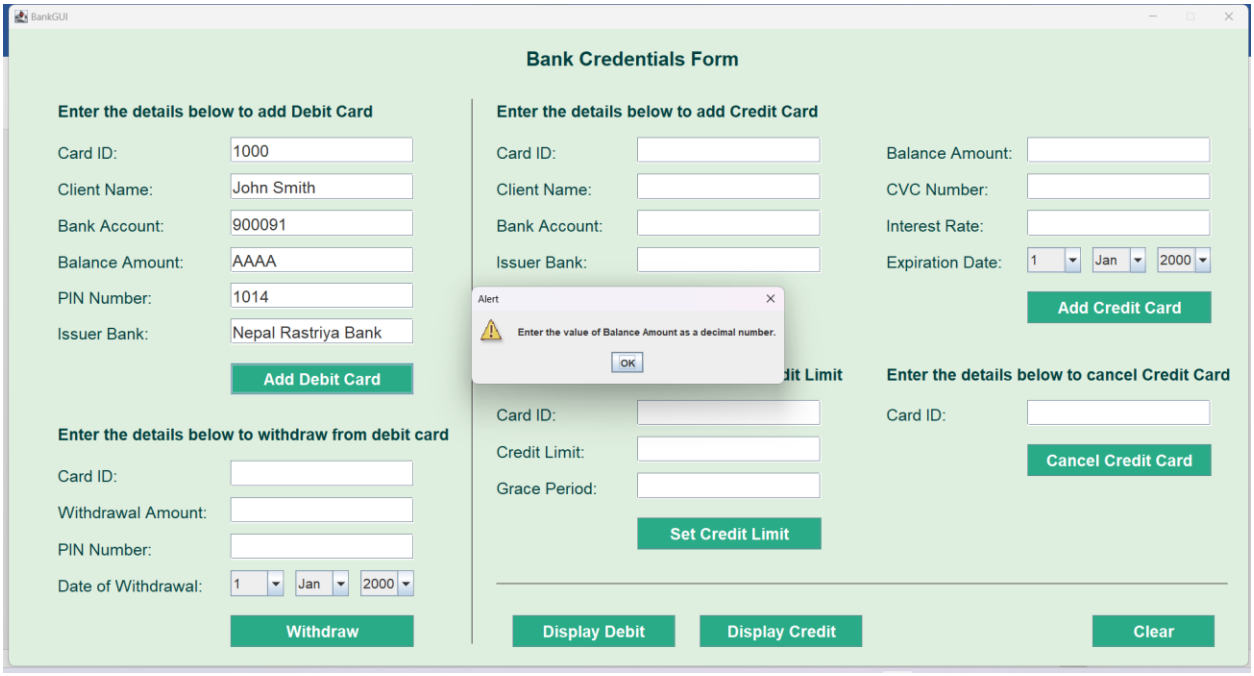


Figure 22. Screenshot of entering invalid Balance Amount (Debit Card)

c. Input invalid PIN Number for adding Debit Card

Test No.	**3.2 (c)**
Objective	To enter invalid PIN Number as input for adding Debit Card
Action	→ Wrong value for Balance Amount is entered for add debit card • Card ID: “1000” • Client Name: “John Smith” • Bank Account: “900091” • Balance Amount: “100000.0” • PIN Number: “AAAA” • Issuer Bank: “Nepal Rastriya Bank” → Click on Add Debit Card Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 10. Invalid PIN Number for adding Debit Card

Output Result:

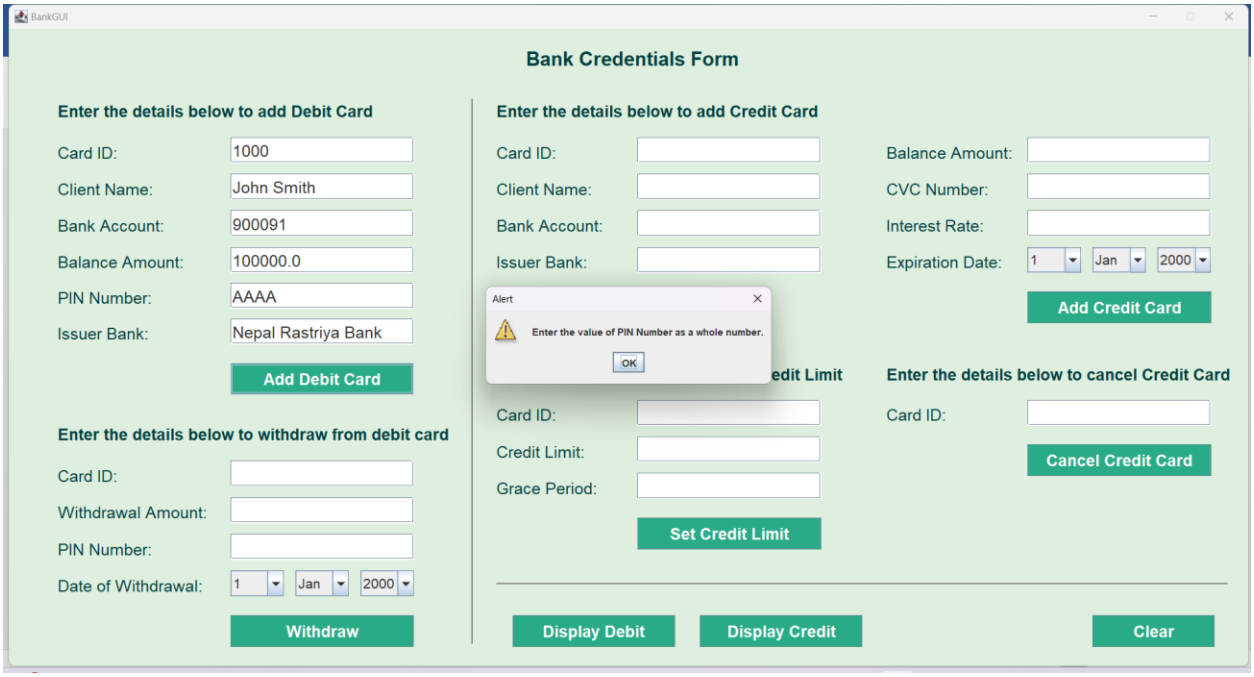


Figure 23. Screenshot of entering invalid PIN Number (Debit Card)

5.3.3. To enter invalid input for adding Credit Card

a. Input invalid Card ID for adding Credit Card

Test No.	**3.3 (a)**
Objective	To enter invalid Card ID as input for adding Credit Card
Action	→ Wrong value for Card ID is entered for add credit card • Card ID: “AAAA” • Client Name: “Harry Brook” • Bank Account: “900092” • Issuer Bank: “Nepal Rastriya Bank” • Balance Amount: “50000.0” • CVC Number: “563” • Interest Rate: “7.5” • Expiration Date: 1-Jan-2000 → Click on Add Credit Card Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 11. Invalid Card ID for adding Credit Card

Output Result:

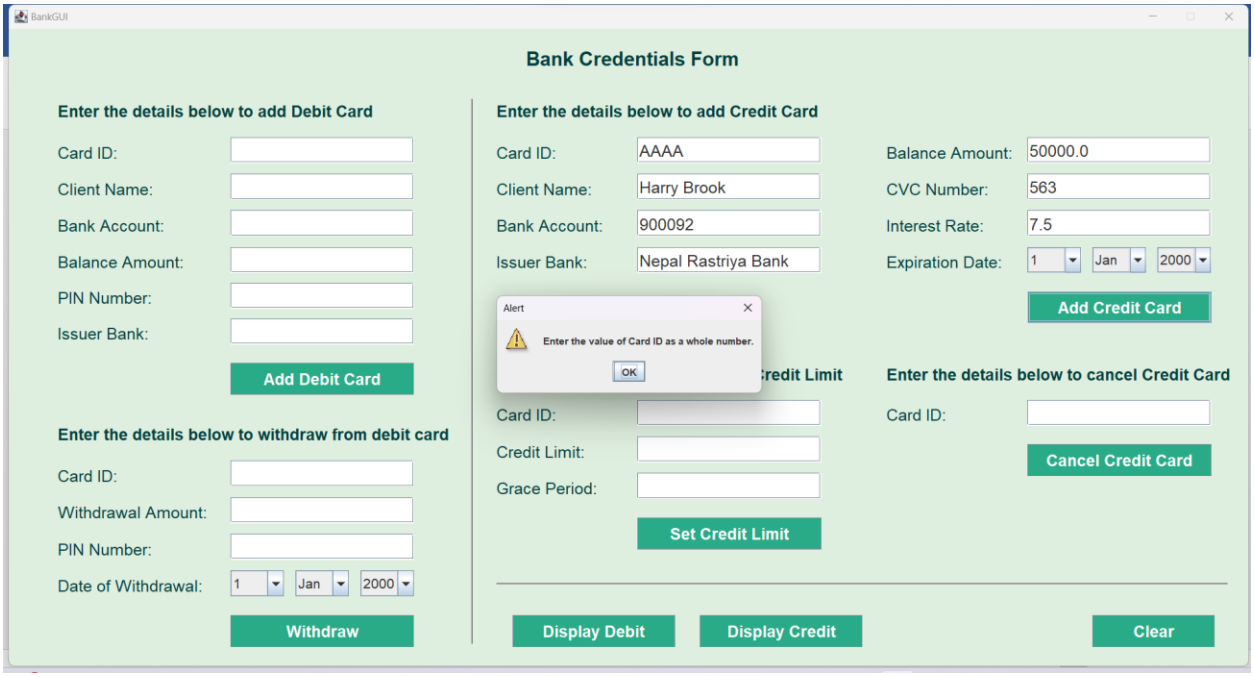


Figure 24. Screenshot of entering invalid Card ID (Credit Card)

b. Input invalid Balance Amount for adding Credit Card

Test No.	**3.3 (b)**
Objective	To enter invalid Balance amount as input for adding Credit Card
Action	→ Wrong value for Balance Amount is entered for add credit card • Card ID: “2000” • Client Name: “Harry Brook” • Bank Account: “900092” • Issuer Bank: “Nepal Rastriya Bank” • Balance Amount: “AAAA” • CVC Number: “563” • Interest Rate: “7.5” • Expiration Date: 1-Jan-2000 → Click on Add Credit Card Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 12. Invalid Balance Amount for adding Credit Card

Output Result:

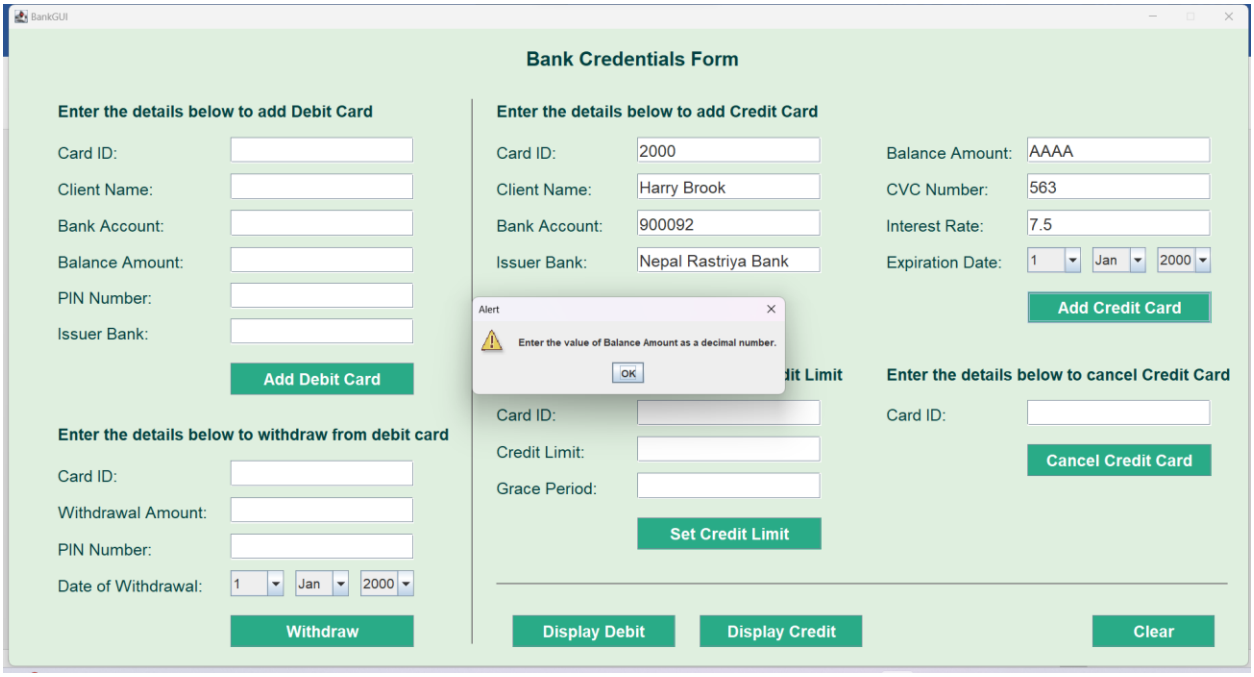


Figure 25. Screenshot of entering invalid Balance Amount (Credit Card)

c. Input invalid CVC Number for adding Credit Card

Test No.	**3.3 (c)**
Objective	To enter invalid CVC Number as input for adding Credit Card
Action	→ Wrong value for CVC Number is entered for add credit card • Card ID: “2000” • Client Name: “Harry Brook” • Bank Account: “900092” • Issuer Bank: “Nepal Rastriya Bank” • Balance Amount: “50000.0” • CVC Number: “AAAA” • Interest Rate: “7.5” • Expiration Date: 1-Jan-2000 → Click on Add Credit Card Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 13. Invalid CVC Number for adding Credit Card

Output Result:

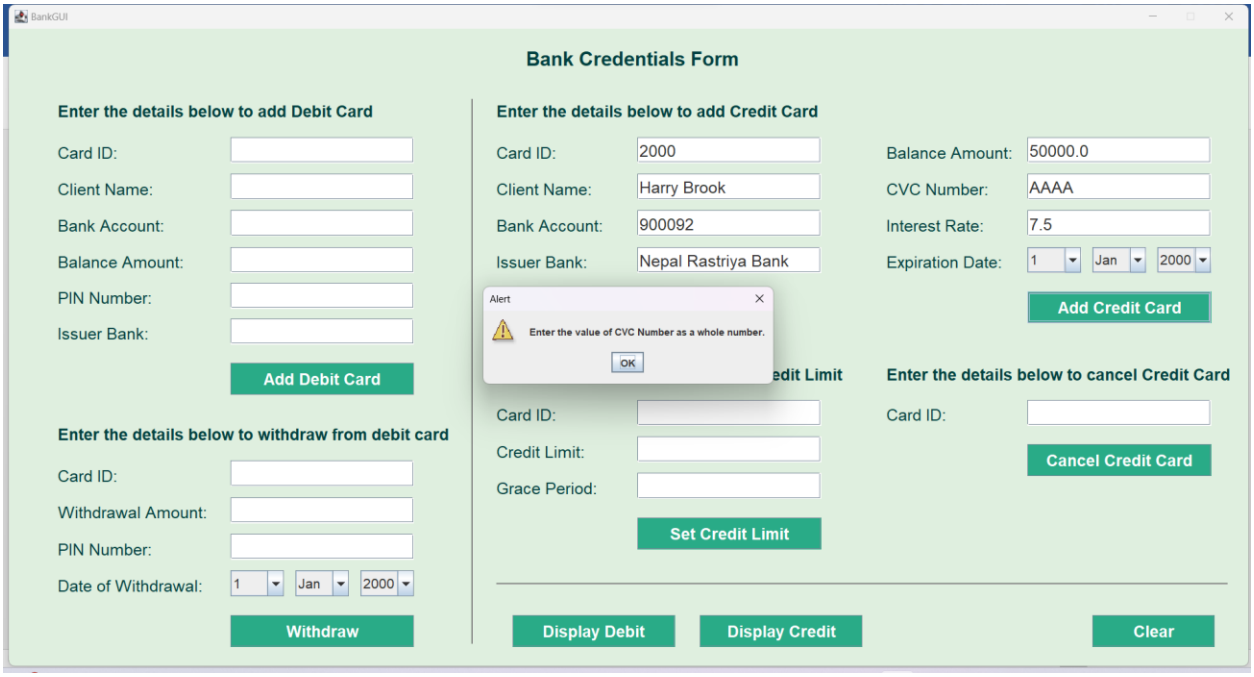


Figure 26. Screenshot of entering invalid CVC Number (Credit Card)

d. Input invalid Interest Rate for adding Credit Card

Test No.	**3.3 (d)**
Objective	To enter invalid Interest Rate as input for adding Credit Card
Action	→ Wrong value for Interest Rate is entered for add credit card • Card ID: “2000” • Client Name: “Harry Brook” • Bank Account: “900092” • Issuer Bank: “Nepal Rastriya Bank” • Balance Amount: “50000.0” • CVC Number: “563” • Interest Rate: “AAAA” • Expiration Date: 1-Jan-2000 → Click on Add Credit Card Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 14. Invalid Interest Rate for adding credit card

Output Result:

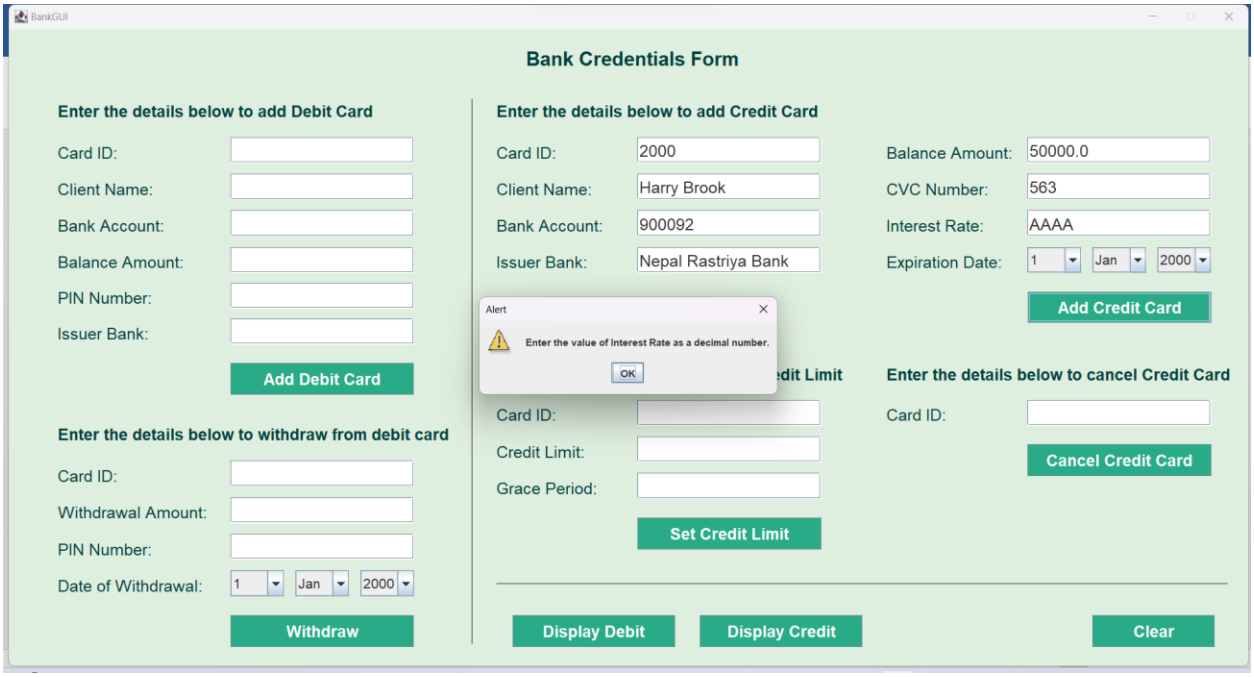


Figure 27. Screenshot of entering invalid Interest Rate (Credit Card)

5.3.4. To enter invalid input while withdrawing from debit card

a. Invalid Card ID for withdrawing from debit card

Test No.	**3.4 (a)**
Objective	To enter invalid Card ID as input for withdrawing from debit card
Action	→ Invalid value for Card ID is entered for withdrawing from debit card • Card ID: “AAAA” • Withdrawal Amount: “4000” • PIN Number: “1014” • Date of Withdrawal: 30-Jul-2005 → Click on Withdraw Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 15. Invalid Card ID for withdraw from debit card

Output Result:

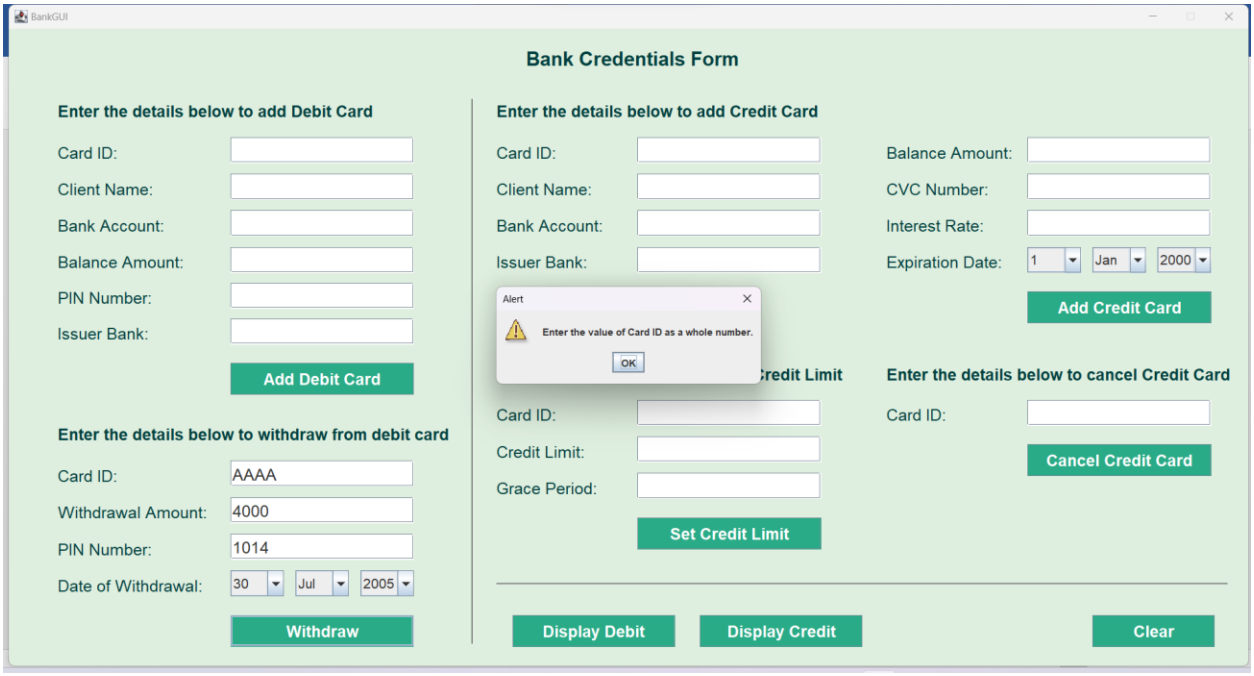


Figure 28. Screenshot of entering invalid Card ID (Withdraw)

b. Invalid Withdrawal Amount for withdrawing from debit card

Test No.	**3.4 (b)**
Objective	To enter invalid Withdrawal Amount as input for withdrawing from debit card
Action	→ Invalid value for PIN Number is entered for withdrawing from debit card • Card ID: “1000” • Withdrawal Amount: “AAAA” • PIN Number: “1014” • Date of Withdrawal: 30-Jul-2005 → Click on Withdraw Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 16. Invalid Withdrawal Amount for withdraw from debit card

Output Result:

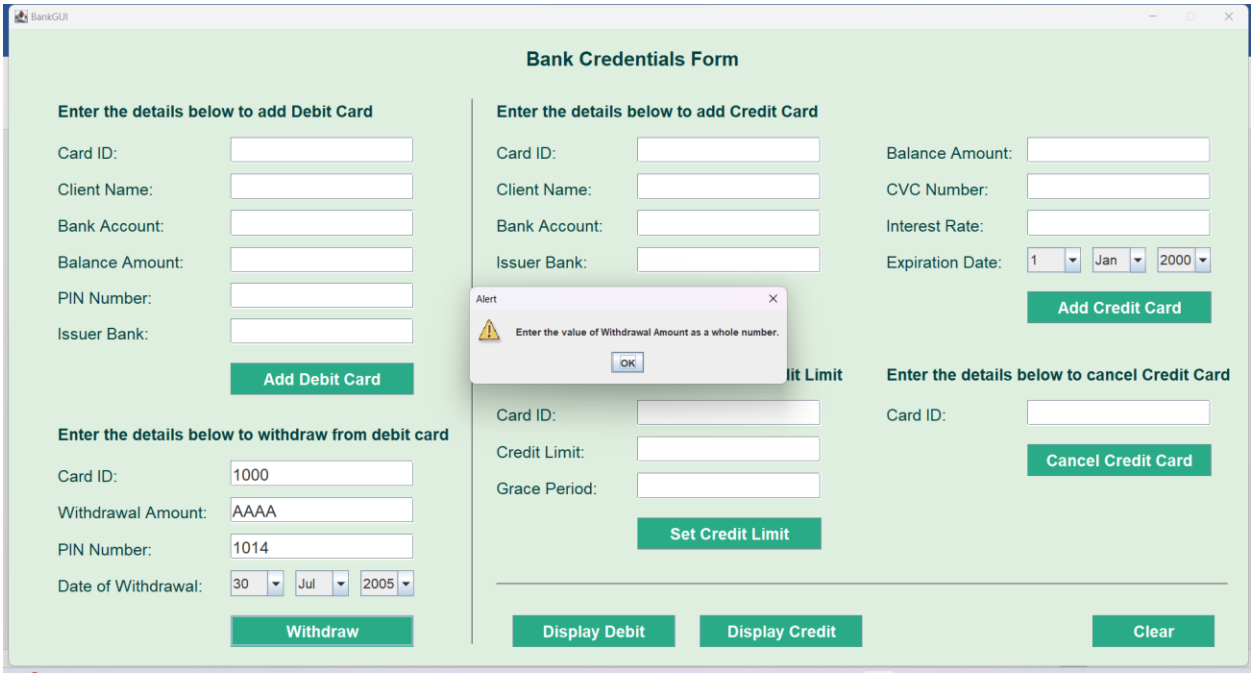


Figure 29. Screenshot of entering invalid Withdrawal Amount (Withdraw)

c. Invalid PIN Number for withdrawing from debit card

Test No.	**3.4 (c)**
Objective	To enter invalid PIN Number as input for withdrawing from debit card
Action	→ Invalid value for PIN Number is entered for withdrawing from debit card • Card ID: “1000” • Withdrawal Amount: “4000” • PIN Number: “AAAA” • Date of Withdrawal: 30-Jul-2005 → Click on Withdraw Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 17. Invalid PIN Number for withdraw from debit card

Output Result:

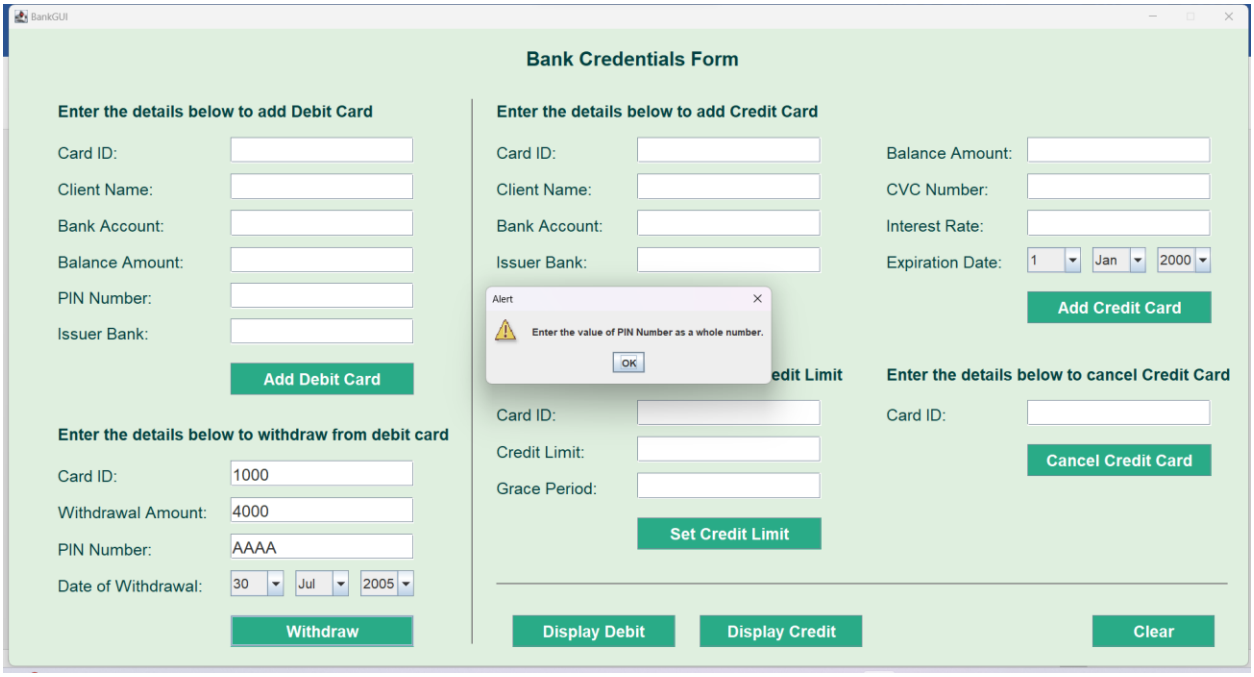


Figure 30. Screenshot of entering invalid PIN Number (Withdraw)

5.3.5. To enter invalid input while setting from credit limit

a. Invalid Card ID for setting the credit limit

Test No.	**3.5 (a)**
Objective	To enter invalid Card ID as input for setting the credit limit
Action	→ Invalid value for Card ID is entered for setting the credit limit • Card ID: “AAAA” • Credit Limit: “40000” • Grace Period: “28” → Click on Set Credit Limit Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 18. Invalid Card ID for setting the credit limit

Output Result:

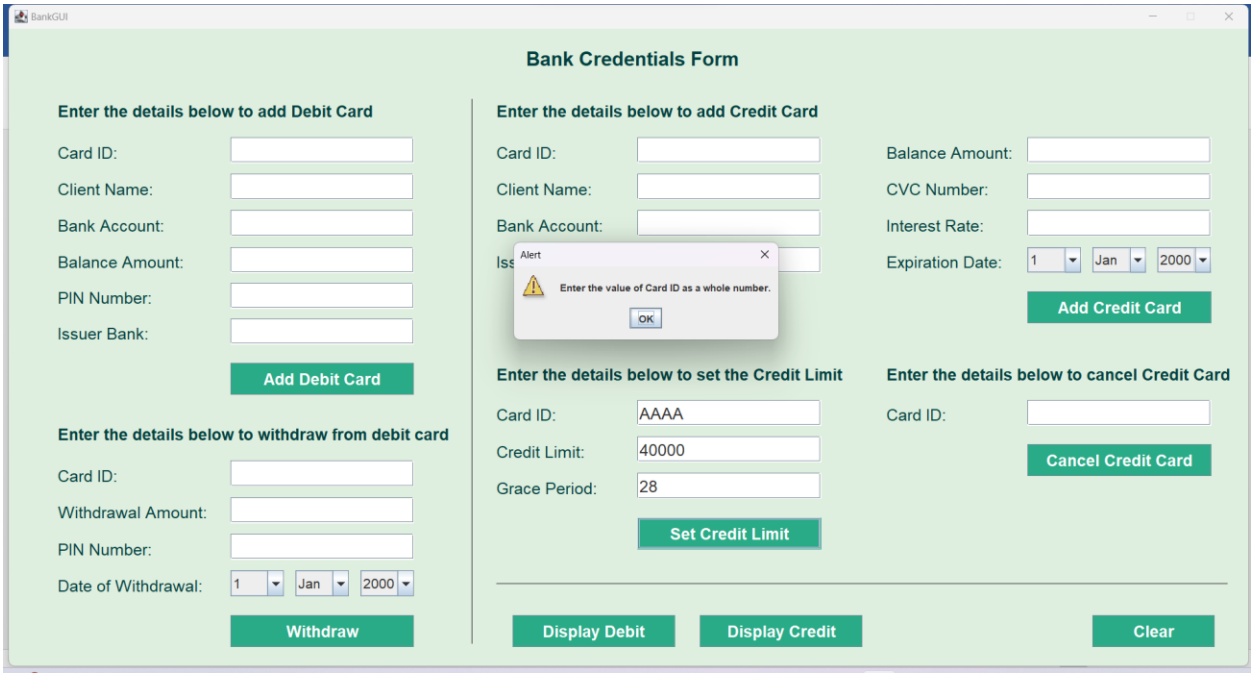


Figure 31. Screenshot of entering invalid Card ID (Set Credit Limit)

a. Invalid Credit Limit for setting the credit limit

Test No.	**3.5 (b)**
Objective	To enter invalid Credit Limit as input for setting the credit limit
Action	→ Invalid value for Credit Limit is entered for setting the credit limit • Card ID: “2000” • Credit Limit: “AAAA” • Grace Period: “28” → Click on Set Credit Limit Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 19. Invalid Credit Limit for setting the credit limit

Output Result:

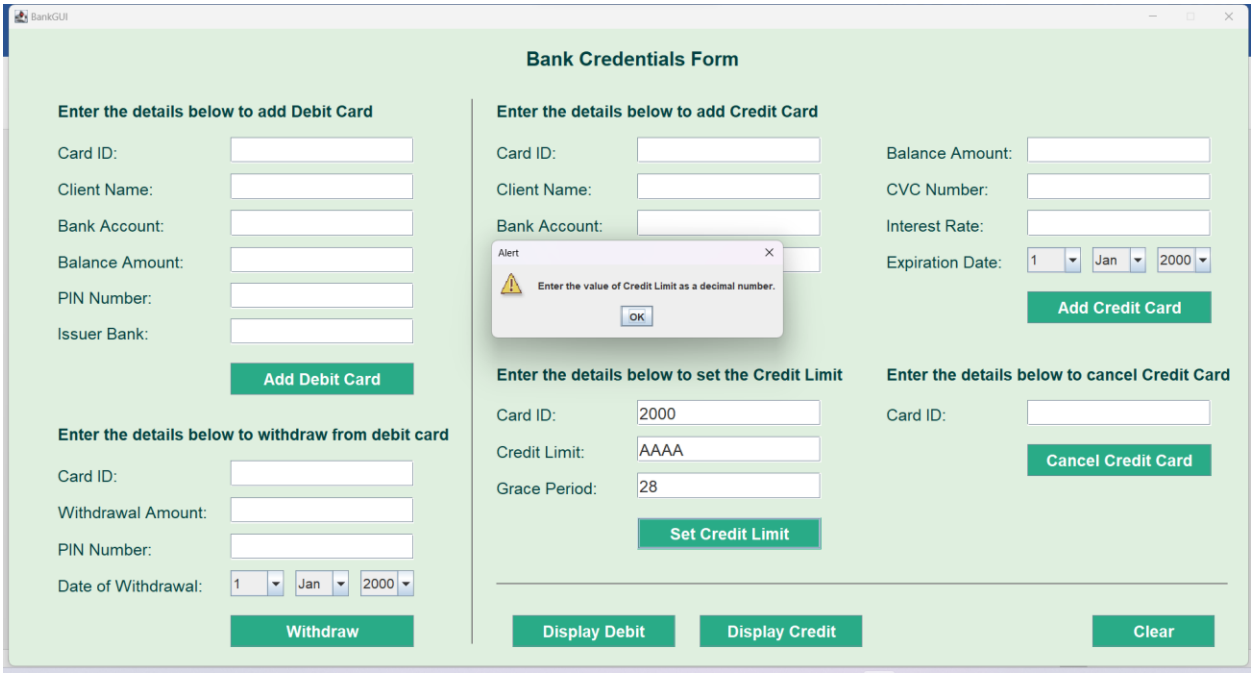


Figure 32. Screenshot of entering invalid Credit Limit (Set Credit Limit)

b. Invalid Grace Period for setting the credit limit

Test No.	**3.5 (c)**
Objective	To enter invalid Grace Period as input for setting the credit limit
Action	→ Invalid value for Grace Period is entered for setting the credit limit • Card ID: “2000” • Credit Limit: “40000” • Grace Period: “AAAA” → Click on Set Credit Limit Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 20. Invalid Grace Period for setting the credit limit

Output Result:

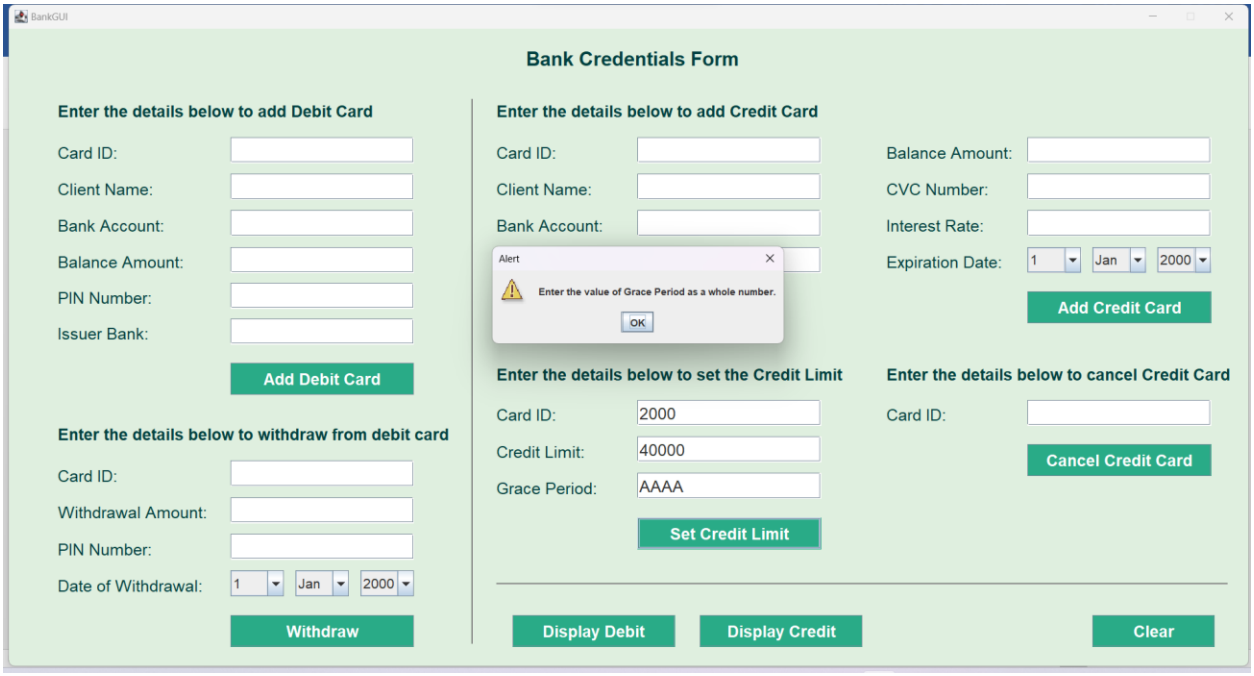


Figure 33. Screenshot of entering invalid Grace Period (Set Credit Limit)

5.3.6. To enter invalid input while cancelling the credit limit

a. Invalid Card ID for cancelling the credit limit

Test No.	**3.6 (a)**
Objective	To enter invalid Card ID as input for cancelling the credit limit
Action	→ Invalid value for Card ID is entered for cancelling the credit limit • Card ID: “AAAA” → Click on Cancel Credit Limit Button
Expected Result	A warning would be displayed as a pop-up message.
Actual Result	A warning was displayed as a pop-up message.
Result	The test is successful.

Table 21. Invalid Card ID for cancelling the Credit Card

Output Result:

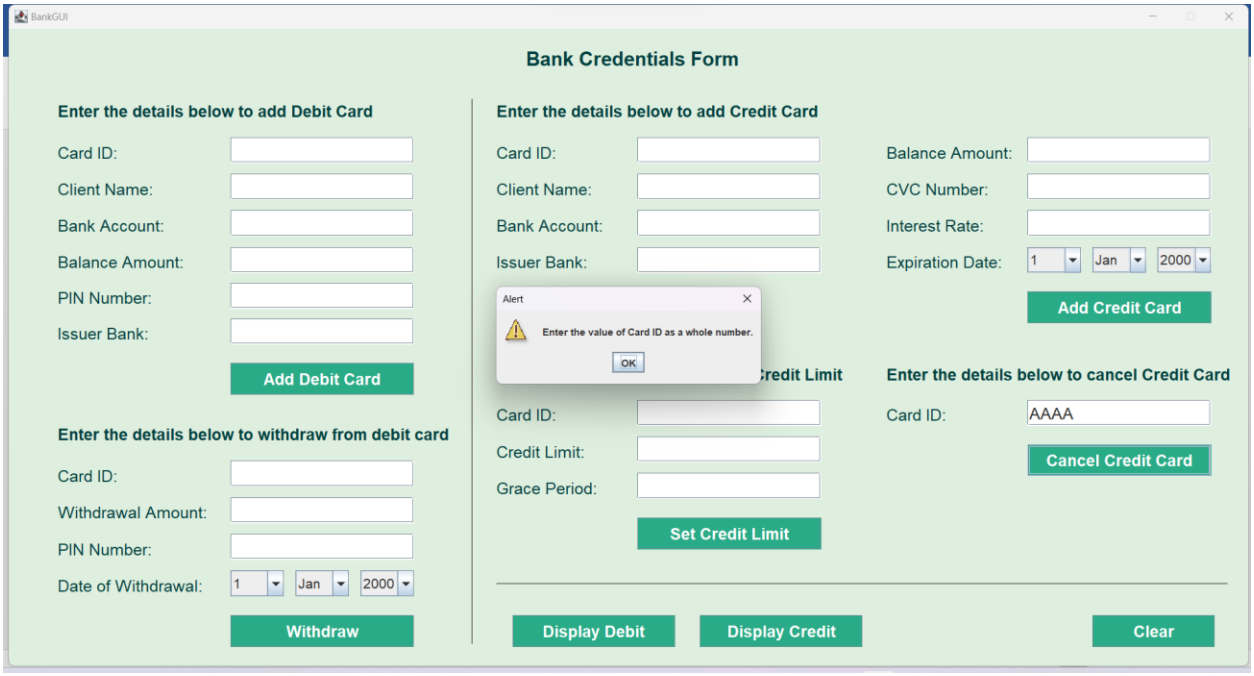


Figure 34. Screenshot of entering invalid Card ID (Cancel Credit Card)

5.3.7. To add already existing Card to ArrayList

a. To add already existing debit card

Test No.	**3.7 (a)**
Objective	To add already existing card for debit card
Action	→ The required text fields are filled for adding debit card Card ID: “7000” Client Name: “John Smith” Bank Account: “900091” Balance Amount: “100000.0” PIN Number: “1014” Issuer Bank: “Nepal Rastriya Bank” →Click on Add Debit Card Button
Expected Result	A message telling “Debit Card for that Card ID already exists” would be displayed.
Actual Result	A message telling “Debit Card for that Card ID already exists” was displayed.
Result	The test is successful.

Table 22. Adding existing Debit Card

Output Result:

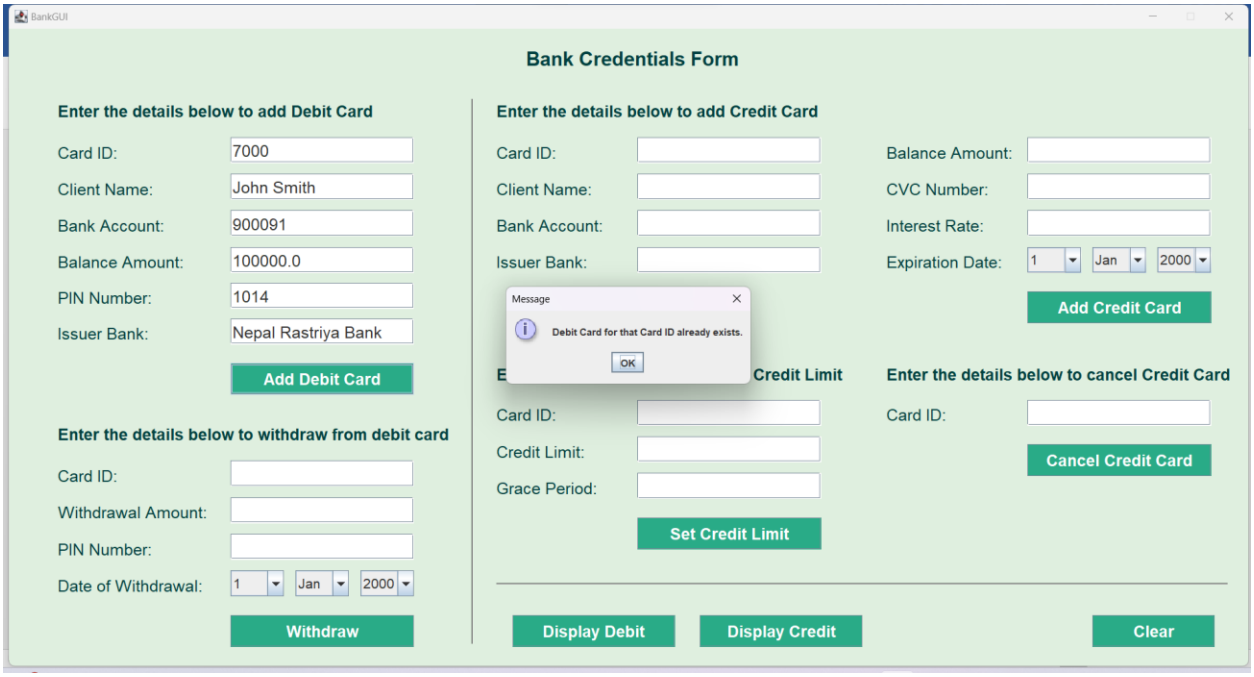


Figure 35. Screenshot of adding already existing Debit Card

a. To add already existing credit card

Test No.	**3.7 (b)**
Objective	To add a card for credit card
Action	→ The required text fields are filled for adding credit card Card ID: “8000” Client Name: “Harry Brook” Bank Account: “900092” Issuer Bank: “Nepal Rastriya Bank” Balance Amount: “50000.0” CVC Number: “563” Interest Rate: “7.5” Expiration Date: 29-Apr-2023 →Click on Add Credit Card Button
Expected Result	A message telling “Credit Card for that Card ID already exists” would be displayed.
Actual Result	A message telling “Credit Card for that Card ID already exists” was displayed.
Result	The test is successful.

Table 23. Adding existing Credit Card

Output Result:

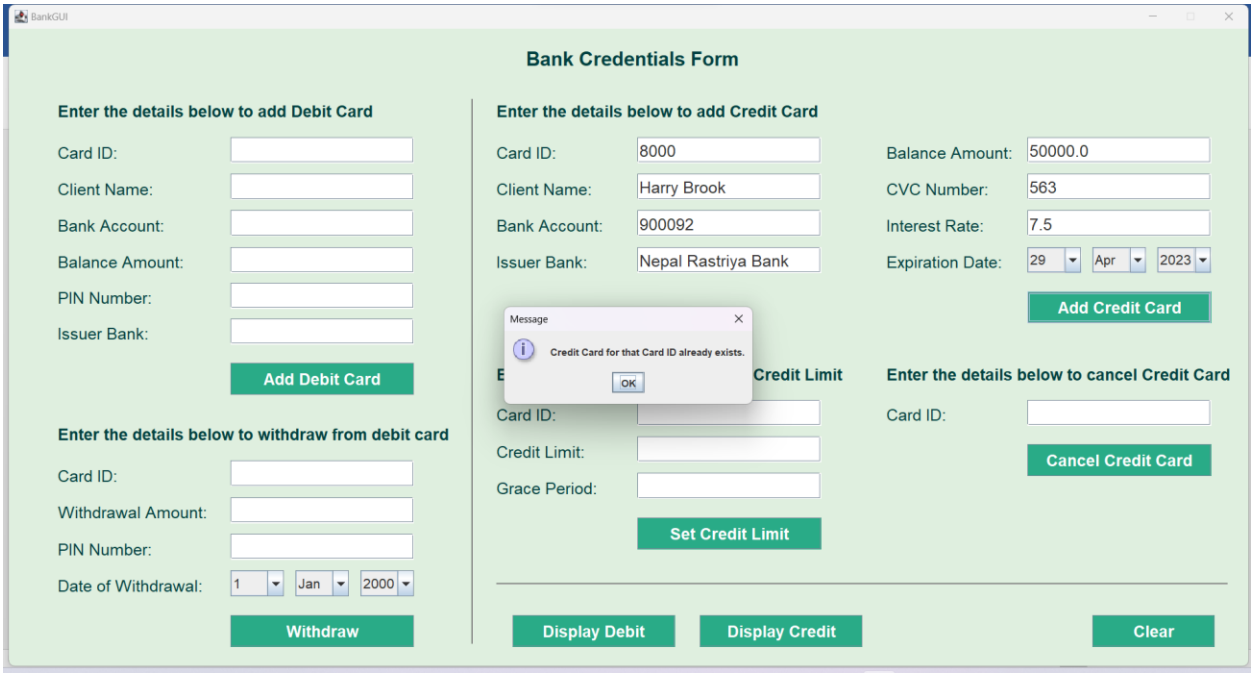


Figure 36. Screenshot of adding already existing Credit Card

5.3.8. Test for withdraw button

a. To withdraw amount from a non-existing card

Test No.	**3.8 (a)**
Objective	To withdraw amount from a non-existing card
Action	→ The required text fields are filled for withdrawing from debit card Card ID: “1200” Withdrawal Amount: “10000” PIN Number: “9098” Date of Withdrawal: 28-Dec-2022 →Click on Withdraw Button
Expected Result	A message telling “There is no card registered to the provided ID.” would be displayed.
Actual Result	A message telling “There is no card registered to the provided ID.” was displayed.
Result	The test is successful.

Table 24. Withdraw Amount from non-existing card

Output Result:

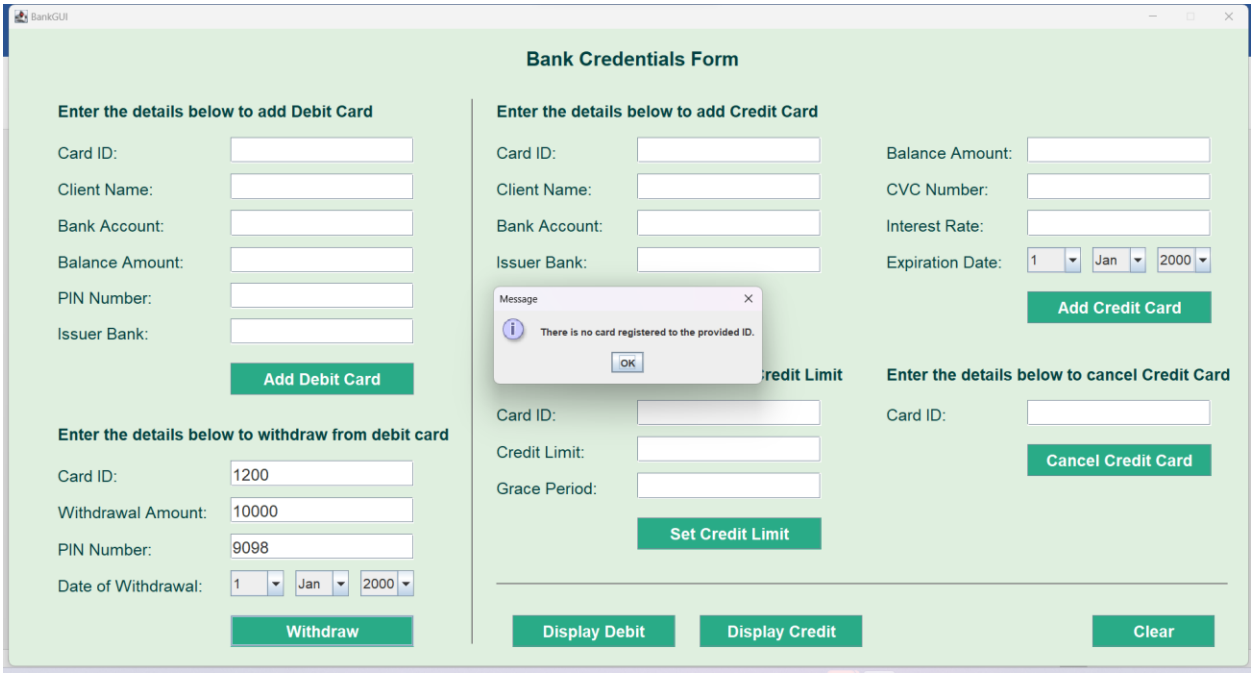


Figure 37. Screenshot of withdrawing amount from a non-existing card

b. To withdraw amount more than the balance amount

Test No.	**3.8 (b)**
Objective	To withdraw amount more than balance amount
Action	→ The required text fields are filled for withdrawing from debit card Card ID: “7000” Withdrawal Amount: “110000” PIN Number: “1014” Date of Withdrawal: 28-Dec-2022 →Click on Withdraw Button
Expected Result	A message telling “The PIN Number or the Withdrawal Amount doesn’t meet the condition” would be displayed.
Actual Result	A message telling “The PIN Number or the Withdrawal Amount doesn’t meet the condition” was displayed.
Result	The test is successful.

Table 25. Withdraw Amount more than Balance Amount

Output Result:

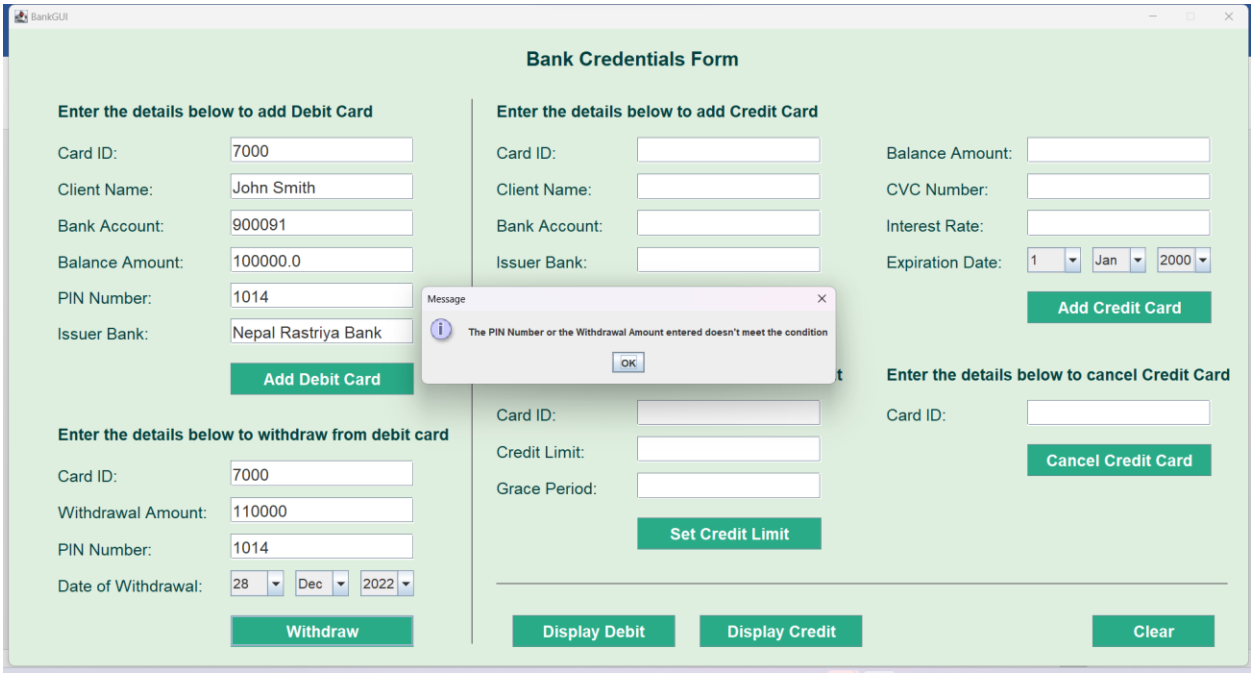


Figure 38. Screenshot of Withdrawal Amount more than Balance Amount

c. To withdraw amount entering the wrong PIN Number

Test No.	**3.8 (c)**
Objective	To withdraw amount entering the wrong PIN Number
Action	→ The required text fields are filled for withdrawing from debit card Card ID: “7000” Withdrawal Amount: “5000” PIN Number: “2000” Date of Withdrawal: 28-Dec-2022 →Click on Withdraw Button
Expected Result	A message telling “The PIN Number or the Withdrawal Amount doesn’t meet the condition” would be displayed.
Actual Result	A message telling “The PIN Number or the Withdrawal Amount doesn’t meet the condition” was displayed.
Result	The test is successful.

Table 26. Withdraw amount entering the wrong PIN Number

Output Result:

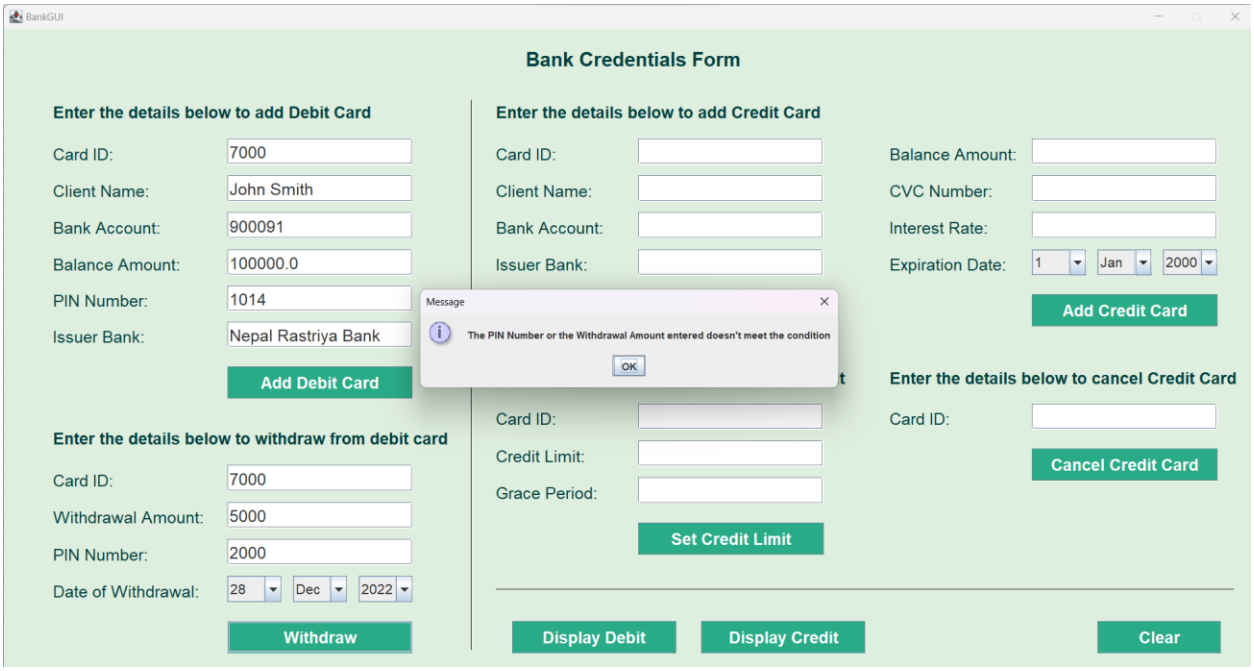


Figure 39. Screenshot of entering wrong PIN Number to withdraw amount

5.3.9. Test for Set Credit Limit Button and Cancel Credit Card Button

a. To set the credit limit to a non-existing card

Test No.	**3.9 (a)**
Objective	To set the credit limit to a non-existing card
Action	→ The required text fields are filled for setting the credit limit Card ID: “1200” Credit Limit: “100000” Grace Period: “28” →Click on Set Credit Limit Button
Expected Result	A message telling “There is no card registered to the provided ID” would be displayed.
Actual Result	A message telling “There is no card registered to the provided ID” was displayed.
Result	The test is successful.

Table 27. Set credit limit to non-existing card

Output Result:

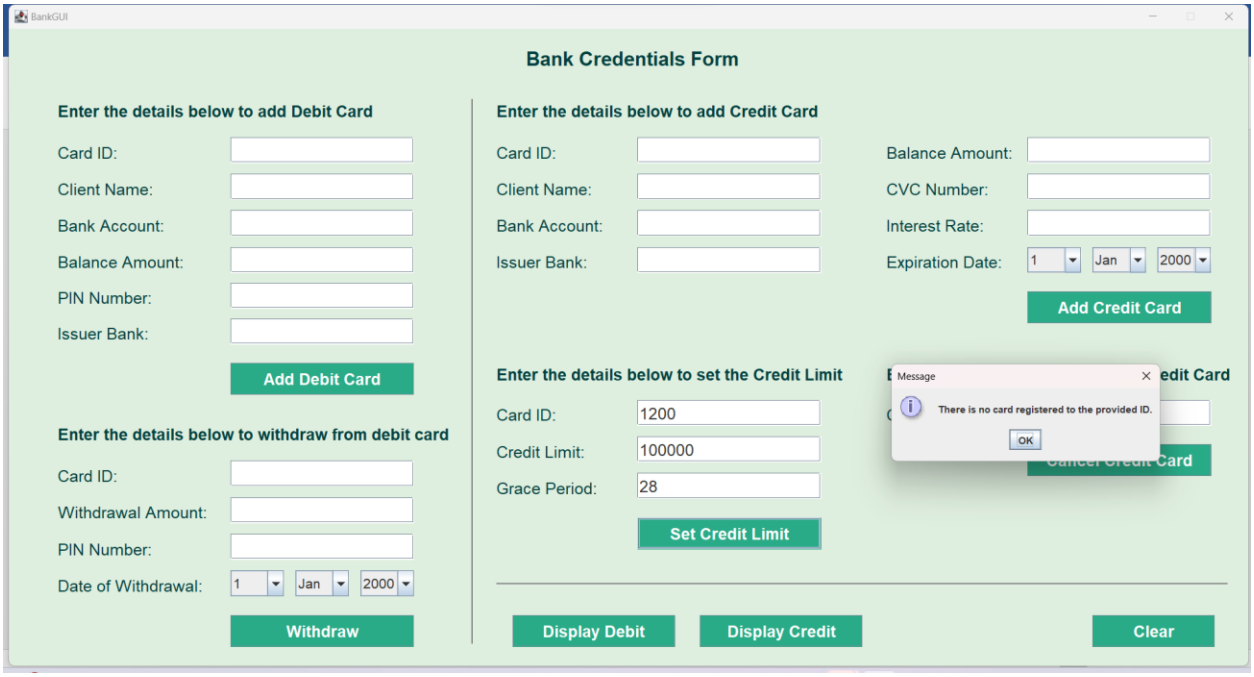


Figure 40. Setting credit limit to a non-existing card

b. To set the credit limit more than the required condition

Test No.	**3.9 (b)**
Objective	To set the credit limit more than the required condition
Action	→ The required text fields are filled for setting the credit limit Card ID: “8000” Credit Limit: “300000” Grace Period: “28” →Click on Set Credit Limit Button
Expected Result	A message telling “The credit limit can not be issued.” would be displayed.
Actual Result	A message telling “The credit limit can not be issued.” was displayed.
Result	The test is successful.

Table 28. Set credit limit more than the required condition

Output Result:

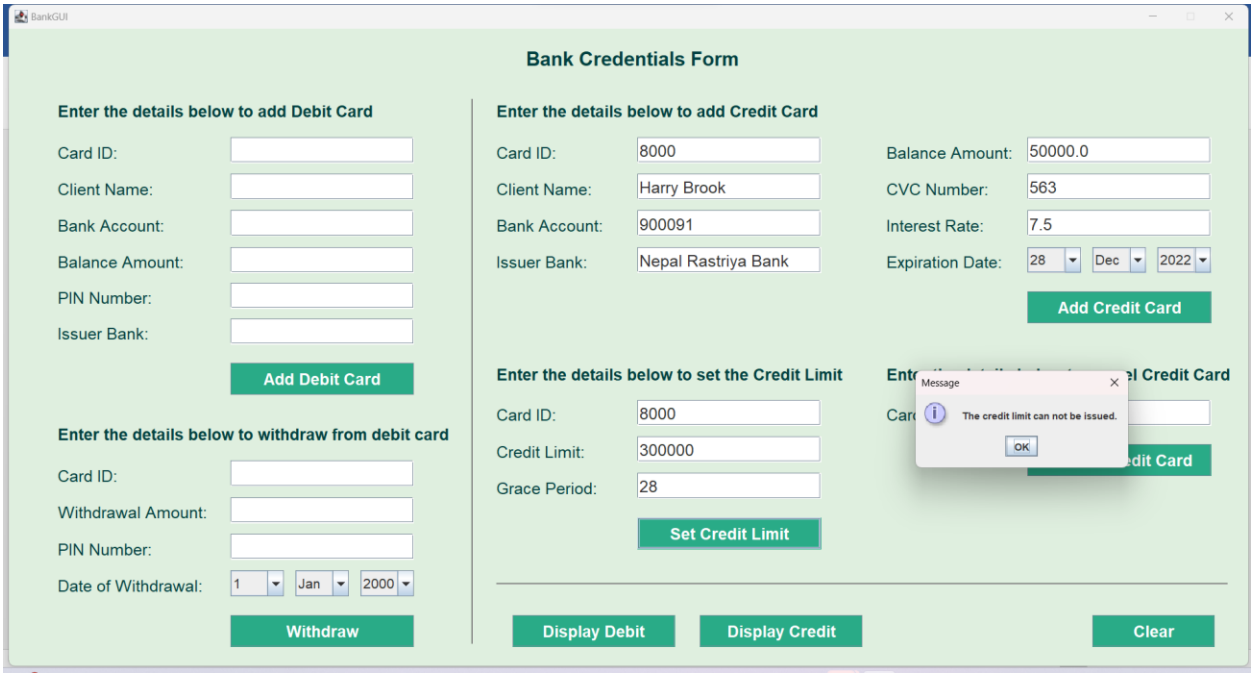


Figure 41. Screenshot of setting credit limit more than the required condition

c. To cancel a non-existing credit card

Test No.	**3.9 (c)**
Objective	To cancel a non-existing credit card
Action	→ The required text fields are filled for setting the credit limit Card ID: “1200” →Click on Cancel Credit Card Button
Expected Result	A message telling “There is no card registered to the provided ID.” would be displayed.
Actual Result	A message telling “There is no card registered to the provided ID.” was displayed.
Result	The test is successful.

Table 29. Cancelling a non-existing credit card

Output Result:

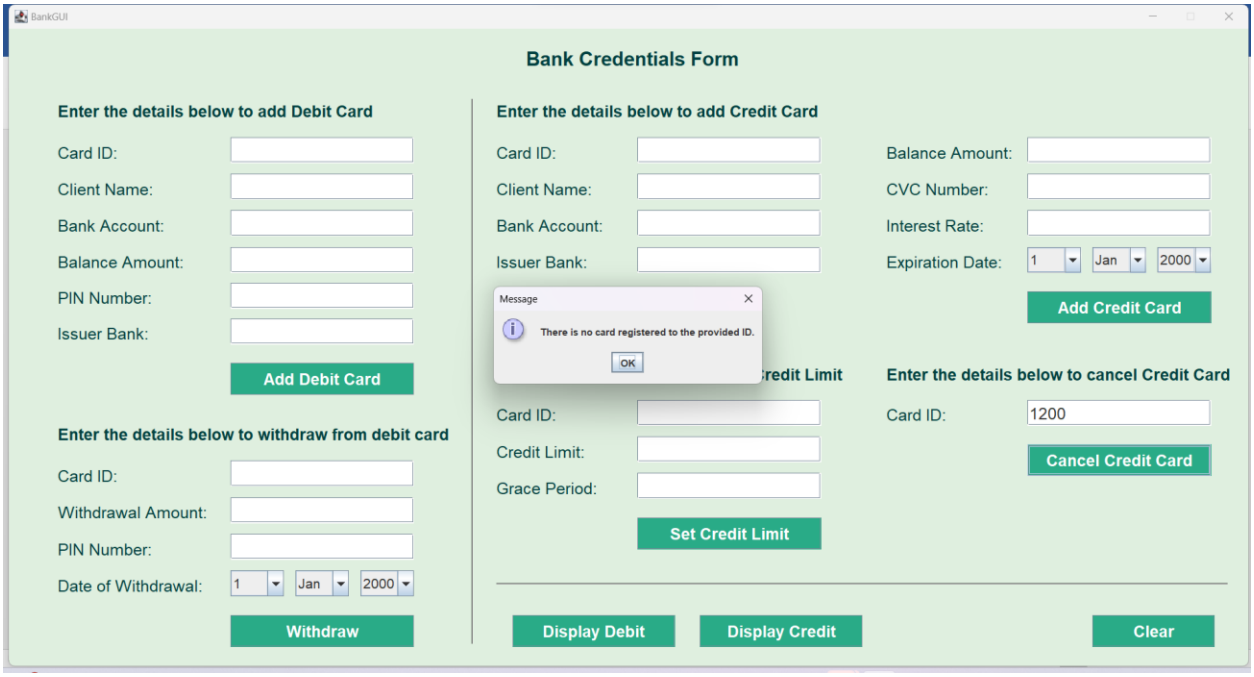


Figure 42. Screenshot of cancelling a non-existing credit card

6. Error Detection

Error is the issue or the fault in a computer program that obstructs the normal flow of program and hamper the execution of the program to get the desired output. The process of detecting those errors is called error detection. There are three types of errors (Syntax error, Semantic error and Logical error) (Codeforwin, 2018).

6.1. Syntax Error

A syntax is considered to be the rules to be followed while writing a program. Hence, syntax error is an error that violates the pre-defined fundamental rule of writing a program. A syntax error interrupt the compilation or interpretation of the program (mnd web docs, 2023).

Syntax error found in the program:

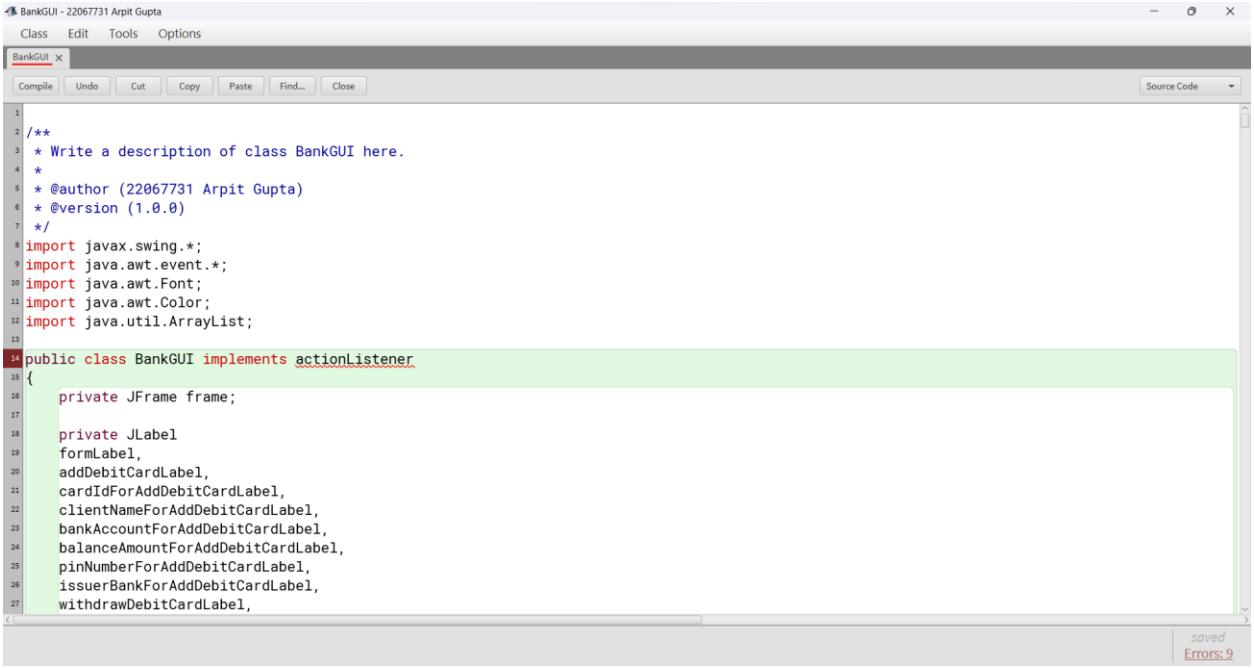


Figure 43. Error 1: Syntax Error

The error was corrected by correcting the spelling of actionPerformed to ActionPerformed.

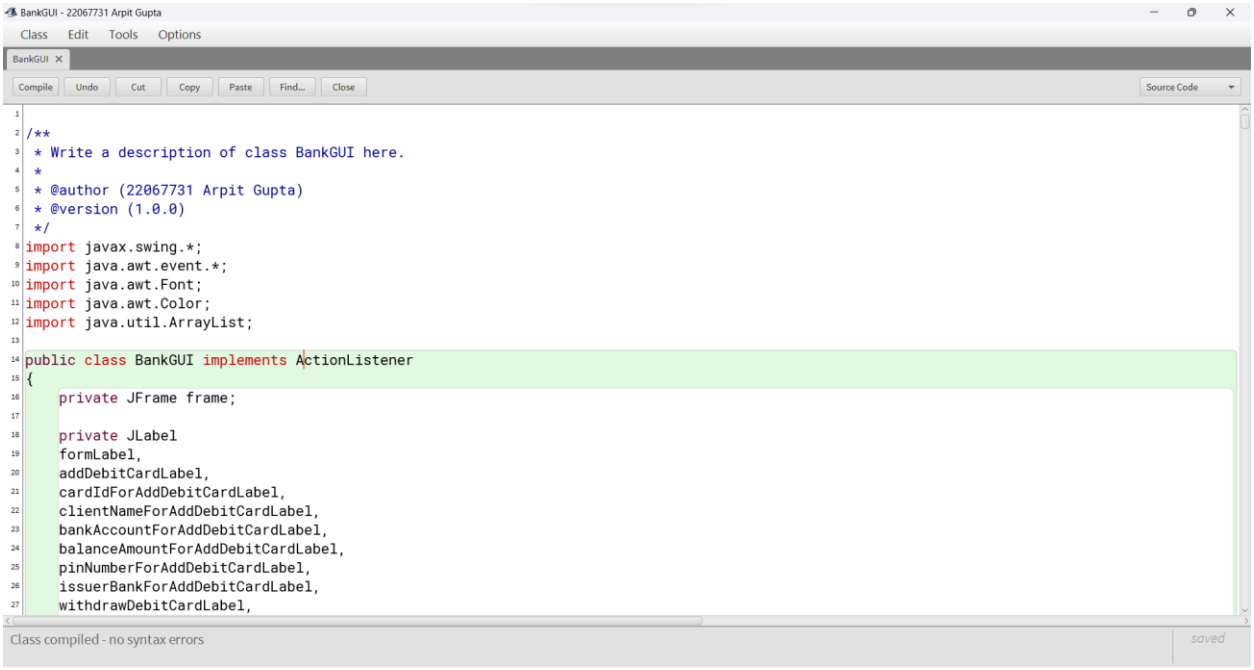


Figure 44. Error 1 solved

6.2. Semantic Error

Semantic errors are the type of errors that result from incorrect use of program statements. Such errors are found during the compilation process. Incompatible operand types, the use of undeclared variables, or the length of the actual argument not matching the formal argument are all examples of semantic mistakes (Javatpoint, 2021).

Semantic error found in the program:

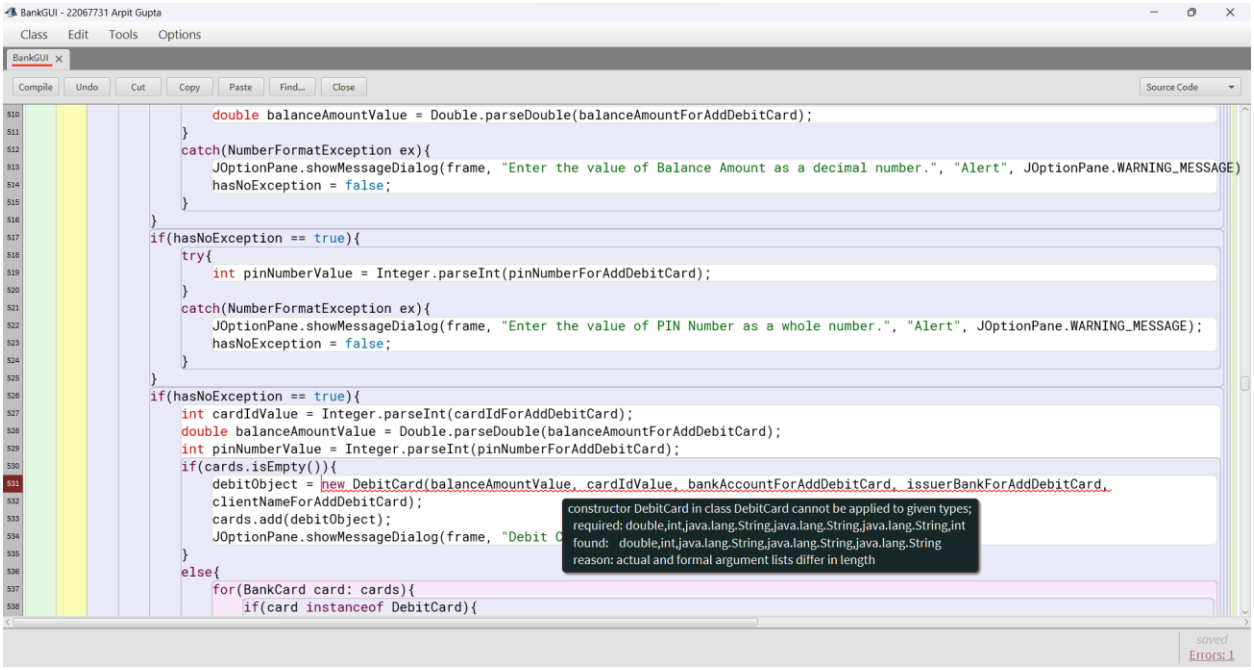


Figure 45. Error 2: Semantic Error

The error was corrected by matching the length of DebitCard present in DebitCard class.

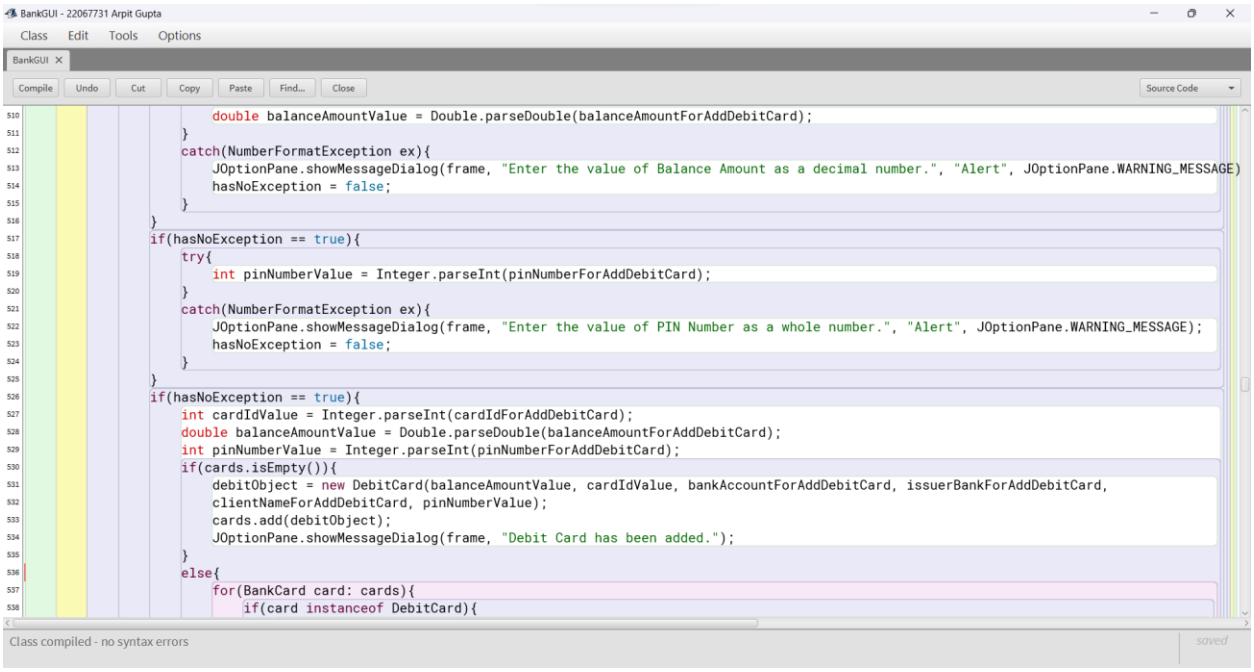


Figure 46. Error 2 solved

6.3. Logical error

Logical errors are the type of errors that occurs when the program shows no syntax and semantic errors and the execution of program is done correctly but the desired output is not achieved. These types of errors are challenging to identify (edureka, 2021).

Logical error found in program:

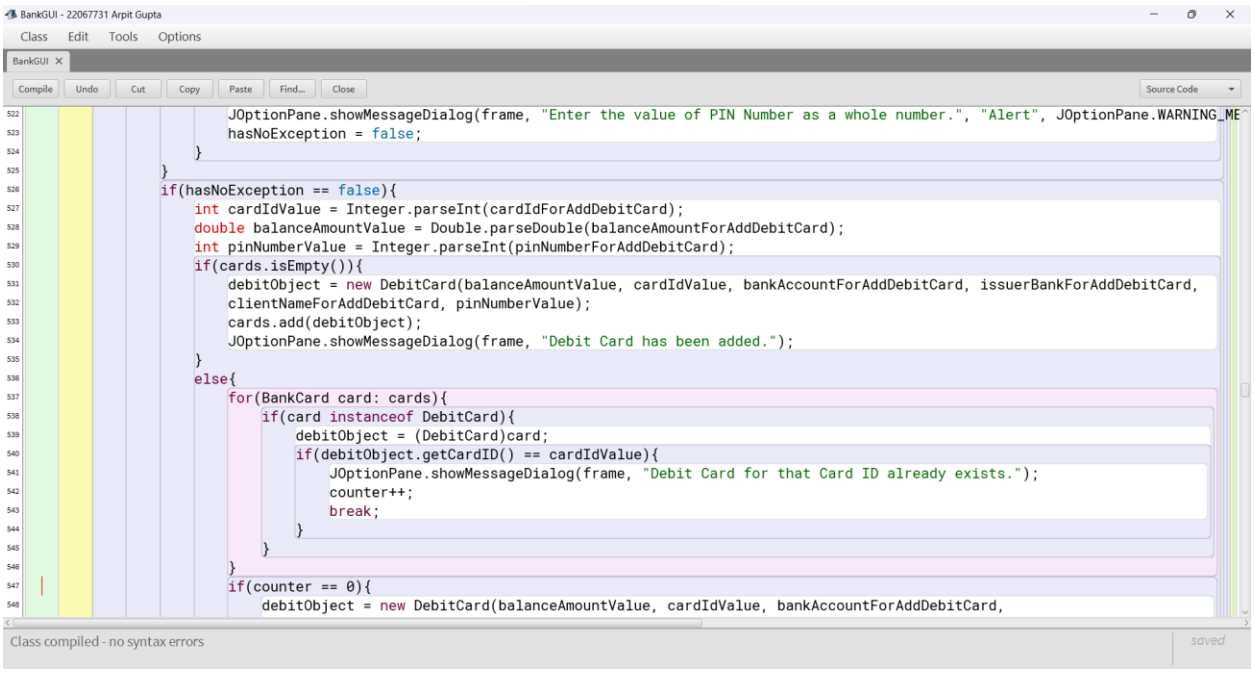


Figure 47. Error 3: Logical error

The debit card was not added.

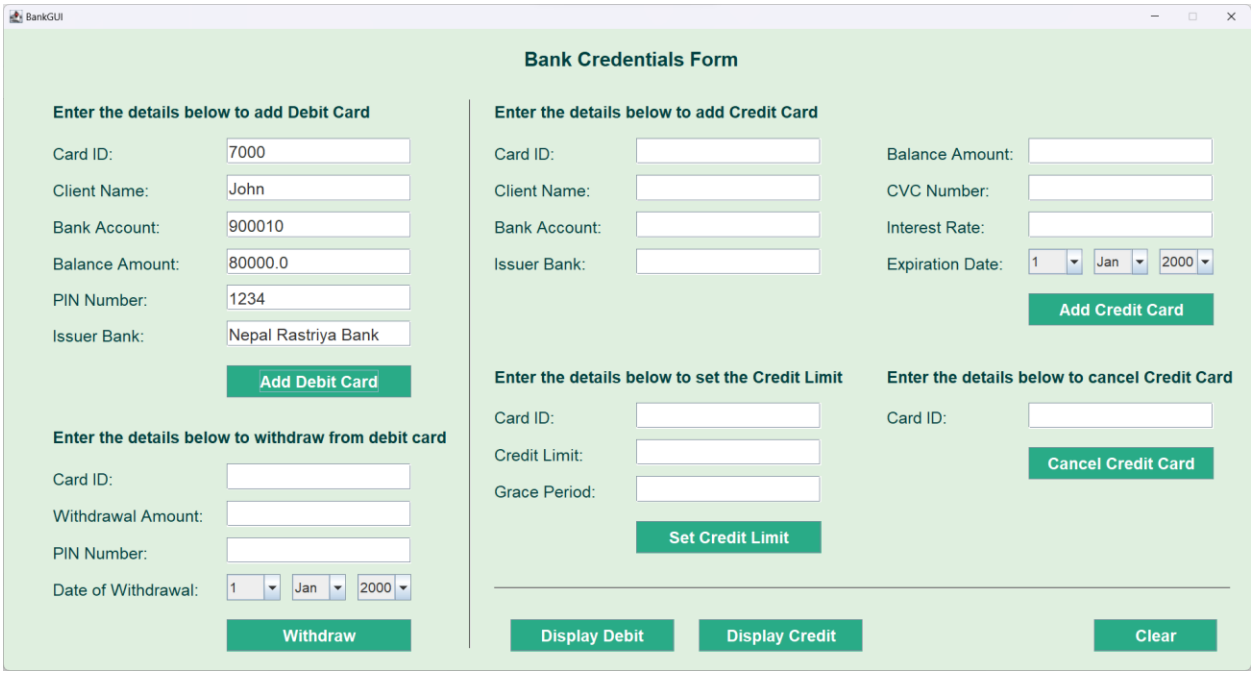


Figure 48. Output due to logical error

The error was solved by changing the code as per the requirements of the program

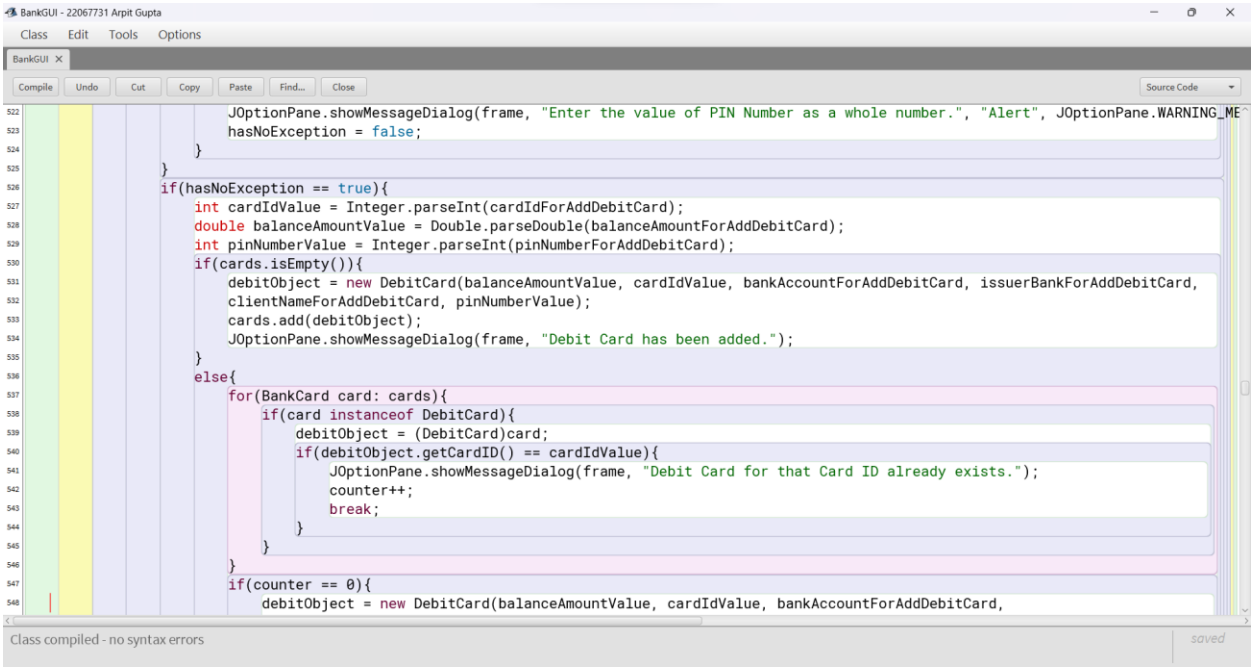


Figure 49. Error 3 solved

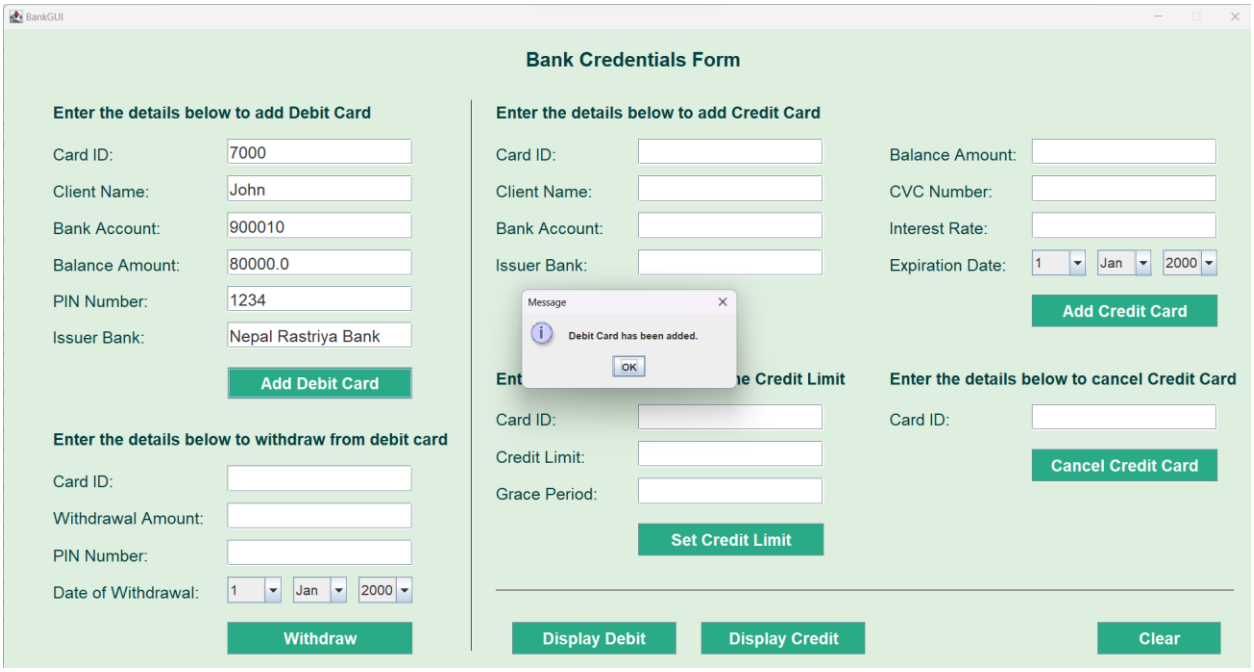


Figure 50. Output after solving logical error

7. Conclusion

This was the second coursework assigned to us for the module “Programming”. The main objective of this coursework was to create a Graphical User Interface (GUI) for the system made in coursework 1. The GUI was made using many JComponents such as JFrame, JLabel, JTextField, JButton, JComboBox, JSeparator and so on. It required us to be able to add Debit Card and Credit Card, for which the inputs were taken from JTextField and JComboBox and functions were performed using JButton.

With the completion of this coursework, many new and interesting concepts such as creating a GUI and adding functionality to it has been understood. Not only a GUI was made, but many important concepts such as Exception Handling, Event Handling and Casting has been understood by me. It helped me understand the importance of Exception Handling as without it, any type of data could have been entered in any text fields. It gave me a clear understanding of Casting, especially Down Casting, and why should it be used. The event handling functionality was one of the most important part of the program as without it, entering data and submitting it would be meaningless. This coursework taught me on how to avoid errors during the creation of a GUI. Errors include simple errors like submitting a form without entering any sort of data, and some complex errors like displaying certain messages when wrong data is entered, or trying to use a button without entering any data and so on. The coursework that I completed helped me understand many new concepts in the field of programming.

While doing this coursework, I faced many difficulties. One of them was making a vertical and horizontal line in GUI. Though being familiar to it, writing pseudocode was a task that I faced many problems on. Another simple yet an issue that can give a lot of trouble was writing attributes names. A lot of attributes were initialized and writing name for each of them along with managing a lot of code along with it was a great issue. So, to solve those problem, I did a lot of research work and sought assistance from the lecturers and tutors. In addition to this, many concepts that were confusing were present in the lecture slide, from which references were taken. All of these greatly helped me in the completion of my coursework.

This coursework helped me understand the basics of creating a GUI and adding functionality to it. Furthermore, I am grateful to the teachers who assisted me with the completion of this coursework. The challenges encountered are now apparent, and the coursework was finished with a greater understanding of what was taught. Creating new GUI seems very much possible now and being able to learn this topic has improved my knowledge in the field of Programming.

8. References

Bhumika_Rani, 2023. *Unified Modeling Language (UML)* | *Class Diagrams - GeeksForGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/
[Accessed 29 April 2023].

Codeforwin, 2018. *Introduction to Programming - Errors - Codeforwin.* [Online]
Available at: https://codeforwin.org/fundamentals/introduction-to-programming-errors
[Accessed 01 May 2023].

Contributer, T. T., 2005. *What is pseudocode?* | *Definition from TechTarget.* [Online]
Available at: https://www.techtarget.com/whatis/definition/pseudocode
[Accessed 29 April 2023].

edureka, 2021. *Introduction to Errors in Java.* [Online]
Available at: https://www.edureka.co/blog/introduction-to-errors-in-java/
[Accessed 07 May 2023].

harleenk_99, 2023. *Introduction of BlueJ - GeeksForGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-of-bluej/
[Accessed 29 April 2023].

Hope, C., 2020. *What is Draw.io?.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 29 April 2023].

Hope, C., 2021. *What is Microsoft Word?.* [Online]
Available at: https://www.computerhope.com/jargon/m/microsoft-word.htm
[Accessed 30 April 2023].

Javatpoint, 2021. *Semantic Error - javatpoint.* [Online]
Available at: https://www.javatpoint.com/semantic-error
[Accessed 07 May 2023].

mnd web docs, 2023. *Syntax error - MDN Web Docs Glossary: Definitions of Web-related terms* | *MDN.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Glossary/Syntax_error
[Accessed 07 May 2023].

Pietroluongo, L., 2022. *Moqups: An Overview and Review.* [Online]
Available at: https://www.elegantthemes.com/blog/design/moqups-an-overview-and-review
[Accessed 29 April 2023].

RajKumar, 2023. *What Is Software Testing* | *Everything You Should Know.* [Online]
Available at: https://www.softwaretestingmaterial.com/software-testing/
[Accessed 01 May 2023].

9. Appendix

```
/**
 * Write a description of class BankGUI here.
 *
 * @author (22067731 Arpit Gupta)
 * @version (1.0.0)
 */
import javax.swing.*;
import java.awt.event.*;
import java.awt.Font;
import java.awt.Color;
import java.util.ArrayList;

public class BankGUI implements ActionListener
{
    private JFrame frame;

    private JLabel
    formLabel,
    addDebitCardLabel,
    cardIdForAddDebitCardLabel,
    clientNameForAddDebitCardLabel,
    bankAccountForAddDebitCardLabel,
    balanceAmountForAddDebitCardLabel,
    pinNumberForAddDebitCardLabel,
    issuerBankForAddDebitCardLabel,
```

withdrawDebitCardLabel,
cardIdForWithdrawDebitCardLabel,
withdrawalAmountLabel,
pinNumberForWithdrawDebitCardLabel,
dateOfWithdrawalLabel,
addCreditCardLabel,
cardIdForAddCreditCardLabel,
clientNameForAddCreditCardLabel,
bankAccountForAddCreditCardLabel,
issuerBankForAddCreditCardLabel,
balanceAmountForAddCreditCardLabel,
cvcNumberLabel,
interestRateLabel,
expirationDateLabel,
setCreditCardLabel,
cardIdForSetCreditLimitLabel,
creditLimitLabel,
gracePeriodLabel,
cancelCreditCardLabel,
cardIdForCancelCreditLimitLabel;

private JTextField
cardIdForAddDebitCardTextField,
clientNameForAddDebitCardTextField,
bankAccountForAddDebitCardTextField,
balanceAmountForAddDebitCardTextField,
pinNumberForAddDebitCardTextField,
issuerBankForAddDebitCardTextField,

```
    cardIdForWithdrawDebitCardTextField,
    withdrawalAmountTextField,
    pinNumberForWithdrawDebitCardTextField,
    cardIdForAddCreditCardTextField,
    clientNameForAddCreditCardTextField,
    bankAccountForAddCreditCardTextField,
    issuerBankForAddCreditCardTextField,
    balanceAmountForAddCreditCardTextField,
    cvcNumberTextField,
    interestRateTextField,
    cardIdForSetCreditLimitTextField,
    creditLimitTextField,
    gracePeriodTextField,
    cardIdForCancelCreditCardTextField;

    private JComboBox
    expirationDayCombo,
    expirationMonthCombo,
    expirationYearCombo,
    withdrawalDayCombo,
    withdrawalMonthCombo,
    withdrawalYearCombo;

    private JButton
    addDebitCardButton,
    withdrawDebitCardButton,
    addCreditCardButton,
    setCreditLimitButton,
```

```
    cancelCreditCardButton,
    debitCardDisplayButton,
    creditCardDisplayButton,
    clearButton;

    private JSeparator verticalSeparator, horizontalSeparator;

    private ArrayList <BankCard> cards;

    private DebitCard debitObject;
    private CreditCard creditObject;

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

        formLabel = new JLabel("Bank Credentials Form");
        formLabel.setBounds(637, 20, 300, 34);
        formLabel.setFont(new Font("Helvetica", Font.BOLD, 24));
        formLabel.setForeground(Color.decode("#004040"));
        frame.add(formLabel);

        addDebitCardLabel = new JLabel("Enter the details below to add Debit Card");
        addDebitCardLabel.setBounds(60, 87, 400, 28);
        addDebitCardLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
        addDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(addDebitCardLabel);
```

```
        cardIdForAddDebitCardLabel = new JLabel("Card ID: ");
        cardIdForAddDebitCardLabel.setBounds(60, 138, 200, 28);
        cardIdForAddDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        cardIdForAddDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(cardIdForAddDebitCardLabel);

        clientNameForAddDebitCardLabel = new JLabel("Client Name: ");
        clientNameForAddDebitCardLabel.setBounds(60, 183, 200, 28);
        clientNameForAddDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        clientNameForAddDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(clientNameForAddDebitCardLabel);

        bankAccountForAddDebitCardLabel = new JLabel("Bank Account: ");
        bankAccountForAddDebitCardLabel.setBounds(60, 228, 200, 28);
        bankAccountForAddDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        bankAccountForAddDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(bankAccountForAddDebitCardLabel);

        balanceAmountForAddDebitCardLabel = new JLabel("Balance Amount: ");
        balanceAmountForAddDebitCardLabel.setBounds(60, 273, 200, 28);
        balanceAmountForAddDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        balanceAmountForAddDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(balanceAmountForAddDebitCardLabel);

        pinNumberForAddDebitCardLabel = new JLabel("PIN Number: ");
        pinNumberForAddDebitCardLabel.setBounds(60, 317, 200, 28);
```

```
        pinNumberForAddDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        pinNumberForAddDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(pinNumberForAddDebitCardLabel);

        issuerBankForAddDebitCardLabel = new JLabel("Issuer Bank: ");
        issuerBankForAddDebitCardLabel.setBounds(60, 361, 200, 28);
        issuerBankForAddDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        issuerBankForAddDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(issuerBankForAddDebitCardLabel);

        withdrawDebitCardLabel = new JLabel("Enter the details below to withdraw from
debit card");
        withdrawDebitCardLabel.setBounds(60, 485, 510, 28);
        withdrawDebitCardLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
        withdrawDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(withdrawDebitCardLabel);

        cardIdForWithdrawDebitCardLabel = new JLabel("Card ID: ");
        cardIdForWithdrawDebitCardLabel.setBounds(60, 536, 200, 28);
        cardIdForWithdrawDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        cardIdForWithdrawDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(cardIdForWithdrawDebitCardLabel);

        withdrawalAmountLabel = new JLabel("Withdrawal Amount: ");
        withdrawalAmountLabel.setBounds(60, 581, 200, 28);
        withdrawalAmountLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        withdrawalAmountLabel.setForeground(Color.decode("#004040"));
        frame.add(withdrawalAmountLabel);
```

```
        pinNumberForWithdrawDebitCardLabel = new JLabel("PIN Number: ");
        pinNumberForWithdrawDebitCardLabel.setBounds(60, 626, 200, 28);
        pinNumberForWithdrawDebitCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        pinNumberForWithdrawDebitCardLabel.setForeground(Color.decode("#004040"));
        frame.add(pinNumberForWithdrawDebitCardLabel);

        dateOfWithdrawalLabel = new JLabel("Date of Withdrawal: ");
        dateOfWithdrawalLabel.setBounds(60, 671, 200, 28);
        dateOfWithdrawalLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        dateOfWithdrawalLabel.setForeground(Color.decode("#004040"));
        frame.add(dateOfWithdrawalLabel);

        addCreditCardLabel = new JLabel("Enter the details below to add Credit Card");
        addCreditCardLabel.setBounds(600, 87, 450, 28);
        addCreditCardLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
        addCreditCardLabel.setForeground(Color.decode("#004040"));
        frame.add(addCreditCardLabel);

        cardIdForAddCreditCardLabel = new JLabel("Card ID: ");
        cardIdForAddCreditCardLabel.setBounds(600, 138, 155, 28);
        cardIdForAddCreditCardLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        cardIdForAddCreditCardLabel.setForeground(Color.decode("#004040"));
        frame.add(cardIdForAddCreditCardLabel);

        clientNameForAddCreditCardLabel = new JLabel("Client Name: ");
        clientNameForAddCreditCardLabel.setBounds(600, 183, 155, 28);
```

```
        clientNameForAddCreditCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        clientNameForAddCreditCardLabel.setForeground(Color.decode("#004040"));
        frame.add(clientNameForAddCreditCardLabel);

        bankAccountForAddCreditCardLabel = new JLabel("Bank Account: ");
        bankAccountForAddCreditCardLabel.setBounds(600, 228, 155, 28);
        bankAccountForAddCreditCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        bankAccountForAddCreditCardLabel.setForeground(Color.decode("#004040"));
        frame.add(bankAccountForAddCreditCardLabel);

        issuerBankForAddCreditCardLabel = new JLabel("Issuer Bank: ");
        issuerBankForAddCreditCardLabel.setBounds(600, 273, 155, 28);
        issuerBankForAddCreditCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        issuerBankForAddCreditCardLabel.setForeground(Color.decode("#004040"));
        frame.add(issuerBankForAddCreditCardLabel);

        balanceAmountForAddCreditCardLabel = new JLabel("Balance Amount: ");
        balanceAmountForAddCreditCardLabel.setBounds(1080, 138, 170, 28);
        balanceAmountForAddCreditCardLabel.setFont(new Font("Helvetica", Font.PLAIN,
20));
        balanceAmountForAddCreditCardLabel.setForeground(Color.decode("#004040"));
        frame.add(balanceAmountForAddCreditCardLabel);

        cvcNumberLabel = new JLabel("CVC Number: ");
        cvcNumberLabel.setBounds(1080, 183, 170, 28);
        cvcNumberLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        cvcNumberLabel.setForeground(Color.decode("#004040"));
```

```
frame.add(cvcNumberLabel);

interestRateLabel = new JLabel("Interest Rate: ");
interestRateLabel.setBounds(1080, 228, 170, 28);
interestRateLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
interestRateLabel.setForeground(Color.decode("#004040"));
frame.add(interestRateLabel);

expirationDateLabel = new JLabel("Expiration Date: ");
expirationDateLabel.setBounds(1080, 273, 170, 28);
expirationDateLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
expirationDateLabel.setForeground(Color.decode("#004040"));
frame.add(expirationDateLabel);

setCreditCardLabel = new JLabel("Enter the details below to set the Credit Limit");
setCreditCardLabel.setBounds(600, 411, 450, 28);
setCreditCardLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
setCreditCardLabel.setForeground(Color.decode("#004040"));
frame.add(setCreditCardLabel);

cardIdForSetCreditLimitLabel = new JLabel("Card ID: ");
cardIdForSetCreditLimitLabel.setBounds(600, 461, 155, 28);
cardIdForSetCreditLimitLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
cardIdForSetCreditLimitLabel.setForeground(Color.decode("#004040"));
frame.add(cardIdForSetCreditLimitLabel);

creditLimitLabel = new JLabel("Credit Limit: ");
creditLimitLabel.setBounds(600, 506, 155, 28);
```

```
        creditLimitLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        creditLimitLabel.setForeground(Color.decode("#004040"));
        frame.add(creditLimitLabel);

        gracePeriodLabel = new JLabel("Grace Period: ");
        gracePeriodLabel.setBounds(600, 551, 155, 28);
        gracePeriodLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        gracePeriodLabel.setForeground(Color.decode("#004040"));
        frame.add(gracePeriodLabel);

        cancelCreditCardLabel = new JLabel("Enter the details below to cancel Credit
Card");
        cancelCreditCardLabel.setBounds(1080, 411, 440, 28);
        cancelCreditCardLabel.setFont(new Font("Helvetica", Font.BOLD, 20));
        cancelCreditCardLabel.setForeground(Color.decode("#004040"));
        frame.add(cancelCreditCardLabel);

        cardIdForCancelCreditLimitLabel = new JLabel("Card ID: ");
        cardIdForCancelCreditLimitLabel.setBounds(1080, 461, 155, 28);
        cardIdForCancelCreditLimitLabel.setFont(new Font("Helvetica", Font.PLAIN, 20));
        cardIdForCancelCreditLimitLabel.setForeground(Color.decode("#004040"));
        frame.add(cardIdForCancelCreditLimitLabel);

        cardIdForAddDebitCardTextField = new JTextField();
        cardIdForAddDebitCardTextField.setBounds(273, 134, 226, 32);
        cardIdForAddDebitCardTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
        frame.add(cardIdForAddDebitCardTextField);

        clientNameForAddDebitCardTextField = new JTextField();
```

```
        clientNameForAddDebitCardTextField.setBounds(273, 179, 226, 32);
        clientNameForAddDebitCardTextField.setFont(new Font("Helvetica", Font.PLAIN,
20));
        frame.add(clientNameForAddDebitCardTextField);


        bankAccountForAddDebitCardTextField = new JTextField();
        bankAccountForAddDebitCardTextField.setBounds(273, 224, 226, 32);
        bankAccountForAddDebitCardTextField.setFont(new Font("Helvetica",
Font.PLAIN, 20));
        frame.add(bankAccountForAddDebitCardTextField);


        balanceAmountForAddDebitCardTextField = new JTextField();
        balanceAmountForAddDebitCardTextField.setBounds(273, 269, 226, 32);
        balanceAmountForAddDebitCardTextField.setFont(new Font("Helvetica",
Font.PLAIN, 20));
        frame.add(balanceAmountForAddDebitCardTextField);


        pinNumberForAddDebitCardTextField = new JTextField();
        pinNumberForAddDebitCardTextField.setBounds(273, 313, 226, 32);
        pinNumberForAddDebitCardTextField.setFont(new Font("Helvetica", Font.PLAIN,
20));
        frame.add(pinNumberForAddDebitCardTextField);


        issuerBankForAddDebitCardTextField = new JTextField();
        issuerBankForAddDebitCardTextField.setBounds(273, 357, 226, 32);
        issuerBankForAddDebitCardTextField.setFont(new Font("Helvetica", Font.PLAIN,
20));
        frame.add(issuerBankForAddDebitCardTextField);


        cardIdForWithdrawDebitCardTextField = new JTextField();
```

```
        cardIdForWithdrawDebitCardTextField.setBounds(273, 532, 226, 32);
        cardIdForWithdrawDebitCardTextField.setFont(new Font("Helvetica", Font.PLAIN,
20));
        frame.add(cardIdForWithdrawDebitCardTextField);

        withdrawalAmountTextField = new JTextField();
        withdrawalAmountTextField.setBounds(273, 577, 226, 32);
        withdrawalAmountTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
        frame.add(withdrawalAmountTextField);

        pinNumberForWithdrawDebitCardTextField = new JTextField();
        pinNumberForWithdrawDebitCardTextField.setBounds(273, 622, 226, 32);
        pinNumberForWithdrawDebitCardTextField.setFont(new Font("Helvetica",
Font.PLAIN, 20));
        frame.add(pinNumberForWithdrawDebitCardTextField);

        cardIdForAddCreditCardTextField = new JTextField();
        cardIdForAddCreditCardTextField.setBounds(774, 134, 226, 32);
        cardIdForAddCreditCardTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
        frame.add(cardIdForAddCreditCardTextField);

        clientNameForAddCreditCardTextField = new JTextField();
        clientNameForAddCreditCardTextField.setBounds(774, 179, 226, 32);
        clientNameForAddCreditCardTextField.setFont(new Font("Helvetica", Font.PLAIN,
20));
        frame.add(clientNameForAddCreditCardTextField);

        bankAccountForAddCreditCardTextField = new JTextField();
        bankAccountForAddCreditCardTextField.setBounds(774, 224, 226, 32);
```

```
        bankAccountForAddCreditCardTextField.setFont(new Font("Helvetica",
Font.PLAIN, 20));
        frame.add(bankAccountForAddCreditCardTextField);


        issuerBankForAddCreditCardTextField = new JTextField();
        issuerBankForAddCreditCardTextField.setBounds(774, 269, 226, 32);
        issuerBankForAddCreditCardTextField.setFont(new Font("Helvetica", Font.PLAIN,
20));
        frame.add(issuerBankForAddCreditCardTextField);


        balanceAmountForAddCreditCardTextField = new JTextField();
        balanceAmountForAddCreditCardTextField.setBounds(1254, 134, 226, 32);
        balanceAmountForAddCreditCardTextField.setFont(new Font("Helvetica",
Font.PLAIN, 20));
        frame.add(balanceAmountForAddCreditCardTextField);


        cvcNumberTextField = new JTextField();
        cvcNumberTextField.setBounds(1254, 179, 226, 32);
        cvcNumberTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
        frame.add(cvcNumberTextField);


        interestRateTextField = new JTextField();
        interestRateTextField.setBounds(1254, 224, 226, 32);
        interestRateTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
        frame.add(interestRateTextField);


        cardIdForSetCreditLimitTextField = new JTextField();
        cardIdForSetCreditLimitTextField.setBounds(774, 457, 226, 32);
        cardIdForSetCreditLimitTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
```

```
        frame.add(cardIdForSetCreditLimitTextField);

        creditLimitTextField = new JTextField();
        creditLimitTextField.setBounds(774, 502, 226, 32);
        creditLimitTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
        frame.add(creditLimitTextField);

        gracePeriodTextField = new JTextField();
        gracePeriodTextField.setBounds(774, 547, 226, 32);
        gracePeriodTextField.setFont(new Font("Helvetica", Font.PLAIN, 20));
        frame.add(gracePeriodTextField);

        cardIdForCancelCreditCardTextField = new JTextField();
        cardIdForCancelCreditCardTextField.setBounds(1254, 457, 226, 32);
        cardIdForCancelCreditCardTextField.setFont(new Font("Helvetica", Font.PLAIN,
20));
        frame.add(cardIdForCancelCreditCardTextField);

        String[] days = {"1","2","3","4","5","6","7","8","9", "10",
"11","12","13","14","15","16","17","18","19","20","21","22","23","24",
            "25","26","27","28","29","30","31"};
        String[] months = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
"Oct", "Nov", "Dec"};
        String[] years = {"2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007",
"2008", "2009", "2010", "2011", "2012", "2013",
            "2014", "2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023"};

        expirationDayCombo = new JComboBox(days);
        expirationDayCombo.setBounds(1254, 269, 66, 32);
        expirationDayCombo.setFont(new Font("Helvetica", Font.PLAIN, 17));
```

```
frame.add(expirationDayCombo);

withdrawalDayCombo = new JComboBox(days);
withdrawalDayCombo.setBounds(273, 667, 66, 32);
withdrawalDayCombo.setFont(new Font("Helvetica", Font.PLAIN, 17));
frame.add(withdrawalDayCombo);

expirationMonthCombo = new JComboBox(months);
expirationMonthCombo.setBounds(1334, 269, 66, 32);
expirationMonthCombo.setFont(new Font("Helvetica", Font.PLAIN, 17));
frame.add(expirationMonthCombo);

withdrawalMonthCombo = new JComboBox(months);
withdrawalMonthCombo.setBounds(353, 667, 66, 32);
withdrawalMonthCombo.setFont(new Font("Helvetica", Font.PLAIN, 17));
frame.add(withdrawalMonthCombo);

expirationYearCombo = new JComboBox(years);
expirationYearCombo.setBounds(1414, 269, 66, 32);
expirationYearCombo.setFont(new Font("Helvetica", Font.PLAIN, 17));
frame.add(expirationYearCombo);

withdrawalYearCombo = new JComboBox(years);
withdrawalYearCombo.setBounds(433, 667, 66, 32);
withdrawalYearCombo.setFont(new Font("Helvetica", Font.PLAIN, 17));
frame.add(withdrawalYearCombo);

addDebitCardButton = new JButton("Add Debit Card");
```

```
addDebitCardButton.setBounds(273, 412, 226, 39);
addDebitCardButton.setFont(new Font("Helvetica", Font.BOLD, 20));
addDebitCardButton.setForeground(Color.white);
addDebitCardButton.setBackground(Color.decode("#29ab87"));
frame.add(addDebitCardButton);

withdrawDebitCardButton = new JButton("Withdraw");
withdrawDebitCardButton.setBounds(273, 722, 226, 39);
withdrawDebitCardButton.setFont(new Font("Helvetica", Font.BOLD, 20));
withdrawDebitCardButton.setForeground(Color.white);
withdrawDebitCardButton.setBackground(Color.decode("#29ab87"));
frame.add(withdrawDebitCardButton);

addCreditCardButton = new JButton("Add Credit Card");
addCreditCardButton.setBounds(1254, 324, 226, 39);
addCreditCardButton.setFont(new Font("Helvetica", Font.BOLD, 20));
addCreditCardButton.setForeground(Color.white);
addCreditCardButton.setBackground(Color.decode("#29ab87"));
frame.add(addCreditCardButton);

setCreditLimitButton = new JButton("Set Credit Limit");
setCreditLimitButton.setBounds(774, 602, 226, 39);
setCreditLimitButton.setFont(new Font("Helvetica", Font.BOLD, 20));
setCreditLimitButton.setForeground(Color.white);
setCreditLimitButton.setBackground(Color.decode("#29ab87"));
frame.add(setCreditLimitButton);

cancelCreditCardButton = new JButton("Cancel Credit Card");
```

```
cancelCreditCardButton.setBounds(1254, 512, 226, 39);
cancelCreditCardButton.setFont(new Font("Helvetica", Font.BOLD, 20));
cancelCreditCardButton.setForeground(Color.white);
cancelCreditCardButton.setBackground(Color.decode("#29ab87"));
frame.add(cancelCreditCardButton);

debitCardDisplayButton = new JButton("Display Debit");
debitCardDisplayButton.setBounds(620, 722, 200, 39);
debitCardDisplayButton.setFont(new Font("Helvetica", Font.BOLD, 20));
debitCardDisplayButton.setForeground(Color.white);
debitCardDisplayButton.setBackground(Color.decode("#29ab87"));
frame.add(debitCardDisplayButton);

creditCardDisplayButton = new JButton("Display Credit");
creditCardDisplayButton.setBounds(851, 722, 200, 39);
creditCardDisplayButton.setFont(new Font("Helvetica", Font.BOLD, 20));
creditCardDisplayButton.setForeground(Color.white);
creditCardDisplayButton.setBackground(Color.decode("#29ab87"));
frame.add(creditCardDisplayButton);

clearButton = new JButton("Clear");
clearButton.setBounds(1334, 722, 150, 39);
clearButton.setFont(new Font("Helvetica", Font.BOLD, 20));
clearButton.setForeground(Color.white);
clearButton.setBackground(Color.decode("#29ab87"));
frame.add(clearButton);

verticalSeparator = new JSeparator(SwingConstants.VERTICAL);
```

```
        verticalSeparator.setBounds(570, 87, 5, 670);
        verticalSeparator.setForeground(Color.BLACK);
        frame.add(verticalSeparator);

        horizontalSeparator = new JSeparator(SwingConstants.HORIZONTAL);
        horizontalSeparator.setBounds(600, 683, 900, 5);
        horizontalSeparator.setForeground(Color.BLACK);
        frame.add(horizontalSeparator);

        addDebitCardButton.addActionListener(this);
        withdrawDebitCardButton.addActionListener(this);
        addCreditCardButton.addActionListener(this);
        setCreditLimitButton.addActionListener(this);
        cancelCreditCardButton.addActionListener(this);
        debitCardDisplayButton.addActionListener(this);
        creditCardDisplayButton.addActionListener(this);
        clearButton.addActionListener(this);

        frame.setLayout(null);
        frame.getContentPane().setBackground(Color.decode("#dfefdf"));
        frame.setSize(1540, 822);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);
        frame.setVisible(true);
    }

    public void actionPerformed(ActionEvent e){
        if(e.getSource() == addDebitCardButton){
```

```
        String cardIdForAddDebitCard = cardIdForAddDebitCardTextField.getText();
        String clientNameForAddDebitCard =
clientNameForAddDebitCardTextField.getText();
        String bankAccountForAddDebitCard =
bankAccountForAddDebitCardTextField.getText();
        String balanceAmountForAddDebitCard =
balanceAmountForAddDebitCardTextField.getText();
        String pinNumberForAddDebitCard =
pinNumberForAddDebitCardTextField.getText();
        String issuerBankForAddDebitCard =
issuerBankForAddDebitCardTextField.getText();
        boolean hasNoException = true;
        int counter = 0;
        if(cardIdForAddDebitCard.isEmpty() || clientNameForAddDebitCard.isEmpty() ||
bankAccountForAddDebitCard.isEmpty() ||
        balanceAmountForAddDebitCard.isEmpty() ||
pinNumberForAddDebitCard.isEmpty() || issuerBankForAddDebitCard.isEmpty()){
            JOptionPane.showMessageDialog(frame, "Empty fields have been found.
Please fill up the requried fields: Card ID, Client Name, Bank Account, Balance Amount,
PIN Number, Issuer Bank.", "Alert ", JOptionPane.WARNING_MESSAGE);
        }
        else{
            try{
                int cardIdValue = Integer.parseInt(cardIdForAddDebitCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of card ID as a
whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
            if(hasNoException == true){
                try{
```

```
                double balanceAmountValue =
Double.parseDouble(balanceAmountForAddDebitCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of Balance
Amount as a decimal number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException == true){
            try{
                int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of PIN
Number as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException == true){
            int cardIdValue = Integer.parseInt(cardIdForAddDebitCard);
            double balanceAmountValue =
Double.parseDouble(balanceAmountForAddDebitCard);
            int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
            if(cards.isEmpty()){
                debitObject = new DebitCard(balanceAmountValue, cardIdValue,
bankAccountForAddDebitCard, issuerBankForAddDebitCard,
clientNameForAddDebitCard, pinNumberValue);
                cards.add(debitObject);
                JOptionPane.showMessageDialog(frame, "Debit Card has been
added.");
```

```
                }
                else{
                    for(BankCard card: cards){
                        if(card instanceof DebitCard){
                            debitObject = (DebitCard)card;
                            if(debitObject.getCardID() == cardIdValue){
                                JOptionPane.showMessageDialog(frame, "Debit Card for that
Card ID already exists.");
                                counter++;
                                break;
                            }
                        }
                    }
                    if(counter == 0){
                        debitObject = new DebitCard(balanceAmountValue, cardIdValue,
bankAccountForAddDebitCard,
                        issuerBankForAddDebitCard, clientNameForAddDebitCard,
pinNumberValue);
                        cards.add(debitObject);
                        JOptionPane.showMessageDialog(frame, "Debit Card has been
added.");
                    }
                }
            }
        }
    }


    if(e.getSource() == withdrawDebitCardButton){
        String cardIdForWithdrawDebitCard =
cardIdForWithdrawDebitCardTextField.getText();
```

```
        String withdrawalAmountForWithdrawDebitCard =
withdrawalAmountTextField.getText();

        String pinNumberForWithdrawDebitCard =
pinNumberForWithdrawDebitCardTextField.getText();

        String withdrawalDay = (String)withdrawalDayCombo.getSelectedItem();

        String withdrawalMonth = (String)withdrawalMonthCombo.getSelectedItem();

        String withdrawalYear = (String)withdrawalYearCombo.getSelectedItem();

        String withdrawalDate = withdrawalMonth + " " + withdrawalDay + ", " +
withdrawalYear;

        int withdrawalYearValue = Integer.parseInt(withdrawalYear);

        int withdrawalDayValue = Integer.parseInt(withdrawalDay);

        boolean hasNoException = true;

        int counter = 0;

        if(cardIdForWithdrawDebitCard.isEmpty() ||
withdrawalAmountForWithdrawDebitCard.isEmpty() ||
pinNumberForWithdrawDebitCard.isEmpty()){

            JOptionPane.showMessageDialog(frame, "Empty fields have been found.
Please fill up the required fields: Card ID, Withdrawal Amount, PIN Number.", "Alert",
JOptionPane.WARNING_MESSAGE);

        }

        else{

            try{

                int cardIdValue = Integer.parseInt(cardIdForWithdrawDebitCard);

            }

            catch(NumberFormatException ex){

                JOptionPane.showMessageDialog(frame, "Enter the value of Card ID as a
whole number.", "Alert", JOptionPane.WARNING_MESSAGE);

                hasNoException = false;

            }

            if(hasNoException == true){

                try{
```

```
                int withdrawalAmountValue =
Integer.parseInt(withdrawalAmountForWithdrawDebitCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of Withdrawal
Amount as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException == true){
            try{
                int pinNumberValue =
Integer.parseInt(pinNumberForWithdrawDebitCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of PIN
Number as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException == true){
            if(withdrawalYearValue%4 != 0){
                if(withdrawalMonth == "Apr" || withdrawalMonth == "Jun" ||
withdrawalMonth == "Sep" || withdrawalMonth == "Nov"){
                    if(withdrawalDayValue == 31){
                        JOptionPane.showMessageDialog(frame, "The date you chose
doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
```

```
                if(withdrawalMonth == "Feb"){
                    if(withdrawalDayValue > 28){
                        JOptionPane.showMessageDialog(frame, "The date you chose doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
            }
            else{
                if(withdrawalMonth == "Apr" || withdrawalMonth == "Jun" || withdrawalMonth == "Sep" || withdrawalMonth == "Nov"){
                    if(withdrawalDayValue == 31){
                        JOptionPane.showMessageDialog(frame, "The date you chose doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
                if(withdrawalMonth == "Feb"){
                    if(withdrawalDayValue > 29){
                        JOptionPane.showMessageDialog(frame, "The date you chose doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
            }
        }


        if(hasNoException){
            int cardIdValue = Integer.parseInt(cardIdForWithdrawDebitCard);
```

```
            int withdrawalAmountValue =
Integer.parseInt(withdrawalAmountForWithdrawDebitCard);
            int pinNumberValue = Integer.parseInt(pinNumberForWithdrawDebitCard);


            for(BankCard card: cards){
               if(card instanceof DebitCard){
                  debitObject = (DebitCard)card;
                  if(debitObject.getCardID() == cardIdValue){
                     debitObject.withdraw(withdrawalAmountValue, withdrawalDate,
pinNumberValue);
                     counter++;
                     boolean withdrawn = debitObject.getHasWithdrawn();
                     int PinNumber = debitObject.getPinNumber();
                     if(withdrawn == false || PinNumber != pinNumberValue){
                        JOptionPane.showMessageDialog(frame, "The PIN Number or
the Withdrawal Amount entered doesn't meet the condition");
                     }
                     else{
                        JOptionPane.showMessageDialog(frame, "Transaction has been
made\nBelow are the details of current transaction: "
                        + "\nCard ID: " + cardIdValue + "\nWithdrawal Amount: " +
withdrawalAmountValue + "\nPIN Number: " +
                        pinNumberValue + "\nExpiration Date: " + withdrawalDate);
                     }
                     break;
                  }
               }
            }


            if(counter == 0){
```

```
                    JOptionPane.showMessageDialog(frame, "There is no card registered to
the provided ID.");
                }
            }
        }
    }


    if(e.getSource() == addCreditCardButton){
        String cardIdForAddCreditCard = cardIdForAddCreditCardTextField.getText();
        String clientNameForAddCreditCard =
clientNameForAddCreditCardTextField.getText();
        String bankAccountForAddCreditCard =
bankAccountForAddCreditCardTextField.getText();
        String issuerBankForAddCreditCard =
issuerBankForAddCreditCardTextField.getText();
        String balanceAmountForAddCreditCard =
balanceAmountForAddCreditCardTextField.getText();
        String cvcNumberForAddCreditCard = cvcNumberTextField.getText();
        String interestRateForAddCreditCard = interestRateTextField.getText();
        String expirationDay = (String)expirationDayCombo.getSelectedItem();
        String expirationMonth = (String)expirationMonthCombo.getSelectedItem();
        String expirationYear = (String)expirationYearCombo.getSelectedItem();
        String expirationDate =  expirationMonth + " " + expirationDay + ", " +
expirationYear;
        int expirationDayValue = Integer.parseInt(expirationDay);
        int expirationYearValue = Integer.parseInt(expirationYear);
        boolean hasNoException = true;
        int counter = 0;
        if(cardIdForAddCreditCard.isEmpty() || clientNameForAddCreditCard.isEmpty()
|| bankAccountForAddCreditCard.isEmpty() ||
```

```
        issuerBankForAddCreditCard.isEmpty() ||
balanceAmountForAddCreditCard.isEmpty() || cvcNumberForAddCreditCard.isEmpty()
||
        interestRateForAddCreditCard.isEmpty()){
            JOptionPane.showMessageDialog(frame, "Empty fields have been found.
Please fill up the requried fields: Card ID, Client Name, Bank Account, Issuer Bank,
Balance Amount, CVC Number, Interest Rate.", "Alert ",
JOptionPane.WARNING_MESSAGE);
        }
        else{
            try{
                int cardIdValue = Integer.parseInt(cardIdForAddCreditCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of Card ID as a
whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
            if(hasNoException == true){
                try{
                    double balanceAmountValue =
Double.parseDouble(balanceAmountForAddCreditCard);
                }
                catch(NumberFormatException ex){
                    JOptionPane.showMessageDialog(frame, "Enter the value of Balance
Amount as a decimal number.", "Alert", JOptionPane.WARNING_MESSAGE);
                    hasNoException = false;
                }
            }
            if(hasNoException == true){
                try{
```

```
                int cvcNumberValue = Integer.parseInt(cvcNumberForAddCreditCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of CVC
Number as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException == true){
            try{
                double interestRateValue =
Double.parseDouble(interestRateForAddCreditCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of Interest
Rate as a decimal number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException == true){
            if(expirationYearValue%4 != 0){
                if(expirationMonth == "Apr" || expirationMonth == "Jun" ||
expirationMonth == "Sep" || expirationMonth == "Nov"){
                    if(expirationDayValue == 31){
                        JOptionPane.showMessageDialog(frame, "The date you chose
doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
                if(expirationMonth == "Feb"){
```

```
                    if(expirationDayValue > 28){
                        JOptionPane.showMessageDialog(frame, "The date you chose doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
            }
            else{
                if(expirationMonth == "Apr" || expirationMonth == "Jun" || expirationMonth == "Sep" || expirationMonth == "Nov"){
                    if(expirationDayValue == 31){
                        JOptionPane.showMessageDialog(frame, "The date you chose doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
                if(expirationMonth == "Feb"){
                    if(expirationDayValue > 29){
                        JOptionPane.showMessageDialog(frame, "The date you chose doesn't exist. Enter a valid date.", "Alert", JOptionPane.WARNING_MESSAGE);
                        hasNoException = false;
                    }
                }
            }
            if(hasNoException == true){
                int cardIdValue = Integer.parseInt(cardIdForAddCreditCard);
                double balanceAmountValue = Double.parseDouble(balanceAmountForAddCreditCard);
                int cvcNumberValue = Integer.parseInt(cvcNumberForAddCreditCard);
```

```
            double interestRateValue =
Double.parseDouble(interestRateForAddCreditCard);
            if(cards.isEmpty()){
                creditObject = new CreditCard(cardIdValue,
clientNameForAddCreditCard, issuerBankForAddCreditCard,
                bankAccountForAddCreditCard, balanceAmountValue,
cvcNumberValue, interestRateValue, expirationDate);
                cards.add(creditObject);
                JOptionPane.showMessageDialog(frame, "Credit Card has been
added.");
            }
            else{
                for(BankCard card: cards){
                    if(card instanceof CreditCard){
                        creditObject = (CreditCard)card;
                        if(creditObject.getCardID() == cardIdValue){
                            JOptionPane.showMessageDialog(frame, "Credit Card for that
Card ID already exists.");
                            counter++;
                            break;
                        }
                    }
                }
                if(counter == 0){
                    creditObject = new CreditCard(cardIdValue,
clientNameForAddCreditCard, issuerBankForAddCreditCard,
                    bankAccountForAddCreditCard, balanceAmountValue,
cvcNumberValue, interestRateValue, expirationDate);
                    cards.add(creditObject);
                    JOptionPane.showMessageDialog(frame, "Credit Card has been
added.");
```

```
                }
            }
        }
    }
}

if(e.getSource() == setCreditLimitButton){
    String cardIdForSetCreditLimit = cardIdForSetCreditLimitTextField.getText();
    String creditLimitForSetCreditLimit = creditLimitTextField.getText();
    String gracePeriodForSetCreditLimit = gracePeriodTextField.getText();
    boolean hasNoException = true;
    int counter = 0;
    if(cardIdForSetCreditLimit.isEmpty() || creditLimitForSetCreditLimit.isEmpty() || gracePeriodForSetCreditLimit.isEmpty()){
            JOptionPane.showMessageDialog(frame, "Empty fields have been found. Please fill up the requried fields: Card ID, Credit Limit, Grace Period.", "Alert ", JOptionPane.WARNING_MESSAGE);
    }
    else{
        try{
            int cardIdValue = Integer.parseInt(cardIdForSetCreditLimit);
        }
        catch(NumberFormatException ex){
            JOptionPane.showMessageDialog(frame, "Enter the value of Card ID as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
            hasNoException = false;
        }
        if(hasNoException == true){
            try{
```

```
                double creditLimitValue =
Double.parseDouble(creditLimitForSetCreditLimit);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of Credit Limit
as a decimal number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException == true){
            try{
                int gracePeriodValue = Integer.parseInt(gracePeriodForSetCreditLimit);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of Grace
Period as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
        }
        if(hasNoException){
            int cardIdValue = Integer.parseInt(cardIdForSetCreditLimit);
            double creditLimitValue =
Double.parseDouble(creditLimitForSetCreditLimit);
            int gracePeriodValue = Integer.parseInt(gracePeriodForSetCreditLimit);

            for(BankCard card: cards){
                if(card instanceof CreditCard){
                    creditObject = (CreditCard)card;
                    if(creditObject.getCardID() == cardIdValue){
                        creditObject.setCreditLimit(creditLimitValue, gracePeriodValue);
```

```
                    if(creditObject.getIsGranted() == false){
                        JOptionPane.showMessageDialog(frame, "The credit limit can
not be issued.");
                    }
                    else{
                        JOptionPane.showMessageDialog(frame, "The credit limit has
been issued.\n" + "Card ID: " + cardIdValue + "\nCredit Limit: " + creditLimitValue +
                        "\nGrace Period: " + gracePeriodValue);
                    }
                    counter++;
                    break;
                }
            }
        }
        if(counter == 0){
            JOptionPane.showMessageDialog(frame, "There is no card registered to
the provided ID.");
        }
    }
}


    if(e.getSource() == cancelCreditCardButton){
        String cardIdForCancelCreditCard =
cardIdForCancelCreditCardTextField.getText();
        boolean hasNoException = true;
        int counter = 0;
        if(cardIdForCancelCreditCard.isEmpty()){
            JOptionPane.showMessageDialog(frame, "Empty fields have been found.
Please fill the required fields: Card ID.", "Alert", JOptionPane.WARNING_MESSAGE);
```

```
        }
        else{
            try{
                int cardIdValue = Integer.parseInt(cardIdForCancelCreditCard);
            }
            catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(frame, "Enter the value of Card ID as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
                hasNoException = false;
            }
            if(hasNoException){
                int cardIdValue = Integer.parseInt(cardIdForCancelCreditCard);
                for(BankCard card: cards){
                    if(card instanceof CreditCard){
                        creditObject = (CreditCard)card;
                        if(creditObject.getCardID() == cardIdValue){
                            creditObject.cancelCreditCard();
                            JOptionPane.showMessageDialog(frame, "The credit card has been canceled.\nCard ID: " + cardIdValue);
                            counter++;
                            break;
                        }
                    }
                }
                if(counter == 0){
                    JOptionPane.showMessageDialog(frame, "There is no card registered to the provided ID.");
                }
            }
```

```
        }
    }

    if(e.getSource() == debitCardDisplayButton){
        int counter = 0;
        if(cards.isEmpty()){
            JOptionPane.showMessageDialog(frame, "No debit card have been added
yet");
        }
        else{
            for(BankCard card: cards){
                if(card instanceof DebitCard){
                    debitObject = (DebitCard)card;
                    System.out.println("Details of Debit Card: " + debitObject.getCardID());
                    debitObject.display();
                    System.out.println();
                    counter++;
                }
            }
            if(counter == 0){
                JOptionPane.showMessageDialog(frame, "No debit card have been added
yet.");
            }
        }
    }

    if(e.getSource() == creditCardDisplayButton){
        int counter = 0;
        if(cards.isEmpty()){
```

```
            JOptionPane.showMessageDialog(frame, "No credit card have been added
yet");
        }
        else{
            for(BankCard card: cards){
                if(card instanceof CreditCard){
                    creditObject = (CreditCard)card;
                    System.out.println("Details of Credit Card: " + creditObject.getCardID());
                    creditObject.display();
                    System.out.println();
                    counter++;
                }
            }
            if(counter == 0){
                JOptionPane.showMessageDialog(frame, "No credit card have been added
yet");
            }
        }
    }

    if(e.getSource() == clearButton){
        cardIdForAddDebitCardTextField.setText(null);
        clientNameForAddDebitCardTextField.setText(null);
        bankAccountForAddDebitCardTextField.setText(null);
        balanceAmountForAddDebitCardTextField.setText(null);
        pinNumberForAddDebitCardTextField.setText(null);
        issuerBankForAddDebitCardTextField.setText(null);

        cardIdForWithdrawDebitCardTextField.setText(null);
```

```
        withdrawalAmountTextField.setText(null);
        pinNumberForWithdrawDebitCardTextField.setText(null);

        cardIdForAddCreditCardTextField.setText(null);
        clientNameForAddCreditCardTextField.setText(null);
        bankAccountForAddCreditCardTextField.setText(null);
        issuerBankForAddCreditCardTextField.setText(null);
        balanceAmountForAddCreditCardTextField.setText(null);
        cvcNumberTextField.setText(null);
        interestRateTextField.setText(null);

        cardIdForSetCreditLimitTextField.setText(null);
        creditLimitTextField.setText(null);
        gracePeriodTextField.setText(null);

        cardIdForCancelCreditCardTextField.setText(null);

        expirationDayCombo.setSelectedIndex(0);
        expirationMonthCombo.setSelectedIndex(0);
        expirationYearCombo.setSelectedIndex(0);
        withdrawalDayCombo.setSelectedIndex(0);
        withdrawalMonthCombo.setSelectedIndex(0);
        withdrawalYearCombo.setSelectedIndex(0);
    }
}

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

```
    }
}
```