



**Islington college**

(इस्लिंग्टन कॉलेज)

## **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

1.	Introduction .....	1
1.1.	Introduction to Project Content.....	1
1.2.	Tools Used for the Project.....	2
2.	Class Diagram.....	3
2.1.	Class Diagram for BankCard.....	4
2.2.	Class Diagram for DebitCard .....	5
2.3.	Class Diagram for CreditCard .....	6
2.4.	Class Diagram for BankGUI .....	7
2.5.	Class Diagram.....	8
3.	Pseudocode .....	9
4.	Method Description .....	30
4.1.	actionPerformed (ActionEvent e) .....	30
4.2.	static void main (String[] args).....	31
5.	Testing .....	32
5.1.	Test 1: To compile and run the program using command prompt.....	32
5.2.	Test 2: To check the functionality of the buttons .....	34
5.2.1.	To add a card for debit card.....	34
5.2.2.	To add a card for credit card.....	36
5.2.3.	To withdraw amount from debit card .....	38
5.2.4.	To set the credit limit to a credit card.....	40
5.2.5.	To cancel a credit card .....	42
5.3.	Test Validations.....	44
5.3.1.	Tests for empty text fields.....	44
5.3.2.	To enter invalid input for adding Debit Card .....	49
5.3.3.	To enter invalid input for adding Credit Card .....	55
5.3.4.	To enter invalid input while withdrawing from debit card .....	63
5.3.5.	To enter invalid input while setting from credit limit .....	69
5.3.6.	To enter invalid input while cancelling the credit limit .....	75
5.3.7.	To add already existing Card to ArrayList.....	77
5.3.8.	Test for withdraw button .....	81

5.3.9. Test for Set Credit Limit Button and Cancel Credit Card Button .....	87
6. Error Detection .....	93
6.1. Syntax Error .....	93
6.2. Semantic Error .....	95
6.3. Logical error .....	97
7. Conclusion .....	100
8. References.....	102
9. Appendix .....	104

## Table of Figures

Figure 1. Screenshot of class diagram in BlueJ .....	3
Figure 2. Class Diagram for BankCard.....	4
Figure 3. Class Diagram for DebitCard .....	5
Figure 4. Class Diagram for CreditCard .....	6
Figure 5. Class Diagram for BankGUI.....	7
Figure 6. Class Diagram.....	8
Figure 7. Command to run compile and java program through terminal.....	33
Figure 8. Output after compiling and running through terminal .....	33
Figure 9. Debit card added.....	35
Figure 10. Credit Card added.....	37
Figure 11. Withdraw from Debit Card .....	39
Figure 12. Set the credit limit.....	41
Figure 13. Cancel a credit card .....	43
Figure 14. Add Debit Card with empty Text Fields .....	45
Figure 15. Withdraw with empty Text Fields.....	45
Figure 16. Add Credit Card with empty Text Fields .....	46
Figure 17. Set Credit Limit with empty Text Fields .....	46
Figure 18. Cancel Credit Card with empty Text Fields .....	47
Figure 19. Display Debit without adding any debit card.....	48
Figure 20. Display Credit without adding any credit card .....	48
Figure 21. Screenshot of adding invalid Card ID (Debit Card) .....	50
Figure 22. Screenshot of entering invalid Balance Amount (Debit Card) .....	52
Figure 23. Screenshot of entering invalid PIN Number (Debit Card) .....	54
Figure 24. Screenshot of entering invalid Card ID (Credit Card) .....	56
Figure 25. Screenshot of entering invalid Balance Amount (Credit Card) .....	58
Figure 26. Screenshot of entering invalid CVC Number (Credit Card) .....	60
Figure 27. Screenshot of entering invalid Interest Rate (Credit Card) .....	62
Figure 28. Screenshot of entering invalid Card ID (Withdraw) .....	64
Figure 29. Screenshot of entering invalid Withdrawal Amount (Withdraw) .....	66
Figure 30. Screenshot of entering invalid PIN Number (Withdraw) .....	68

Figure 31. Screenshot of entering invalid Card ID (Set Credit Limit) .....	70
Figure 32. Screenshot of entering invalid Credit Limit (Set Credit Limit) .....	72
Figure 33. Screenshot of entering invalid Grace Period (Set Credit Limit) .....	74
Figure 34. Screenshot of entering invalid Card ID (Cancel Credit Card) .....	76
Figure 35. Screenshot of adding already existing Debit Card .....	78
Figure 36. Screenshot of adding already existing Credit Card .....	80
Figure 37. Screenshot of withdrawing amount from a non-existing card.....	82
Figure 38. Screenshot of Withdrawal Amount more than Balance Amount.....	84
Figure 39. Screenshot of entering wrong PIN Number to withdraw amount.....	86
Figure 40. Setting credit limit to a non-existing card.....	88
Figure 41. Screenshot of setting credit limit more than the required condition .....	90
Figure 42. Screenshot of cancelling a non-existing credit card .....	92
Figure 43. Error 1: Syntax Error .....	93
Figure 44. Error 1 solved.....	94
Figure 45. Error 2: Semantic Error .....	95
Figure 46. Error 2 solved.....	96
Figure 47. Error 3: Logical error .....	97
Figure 48. Output due to logical error.....	98
Figure 49. Error 3 solved.....	98
Figure 50. Output after solving logical error .....	99

## Table of Tables

Table 1. To compile and run the program using command prompt .....	32
Table 2. Add a debit card .....	34
Table 3. Add a credit card .....	36
Table 4. Withdraw from debit card.....	38
Table 5. Set the credit limit.....	40
Table 6. Cancel a credit card .....	42
Table 7. Test for empty text fields. ....	44
Table 8. Invalid Card Id for adding debit card.....	49
Table 9. Invalid Balance Amount for adding Debit Card.....	51
Table 10. Invalid PIN Number for adding Debit Card .....	53
Table 11. Invalid Card ID for adding Credit Card .....	55
Table 12. Invalid Balance Amount for adding Credit Card.....	57
Table 13. Invalid CVC Number for adding Credit Card.....	59
Table 14. Invalid Interest Rate for adding credit card.....	61
Table 15. Invalid Card ID for withdraw from debit card.....	63
Table 16. Invalid Withdrawal Amount for withdraw from debit card .....	65
Table 17. Invalid PIN Number for withdraw from debit card .....	67
Table 18. Invalid Card ID for setting the credit limit.....	69
Table 19. Invalid Credit Limit for setting the credit limit .....	71
Table 20. Invalid Grace Period for setting the credit limit .....	73
Table 21. Invalid Card ID for cancelling the Credit Card .....	75
Table 22. Adding existing Debit Card .....	77
Table 23. Adding existing Credit Card.....	79
Table 24. Withdraw Amount from non-existing card.....	81
Table 25. Withdraw Amount more than Balance Amount.....	83
Table 26. Withdraw amount entering the wrong PIN Number .....	85
Table 27. Set credit limit to non-existing card.....	87
Table 28. Set credit limit more than the required condition .....	89
Table 29. Cancelling a non-existing credit card.....	91

## 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).

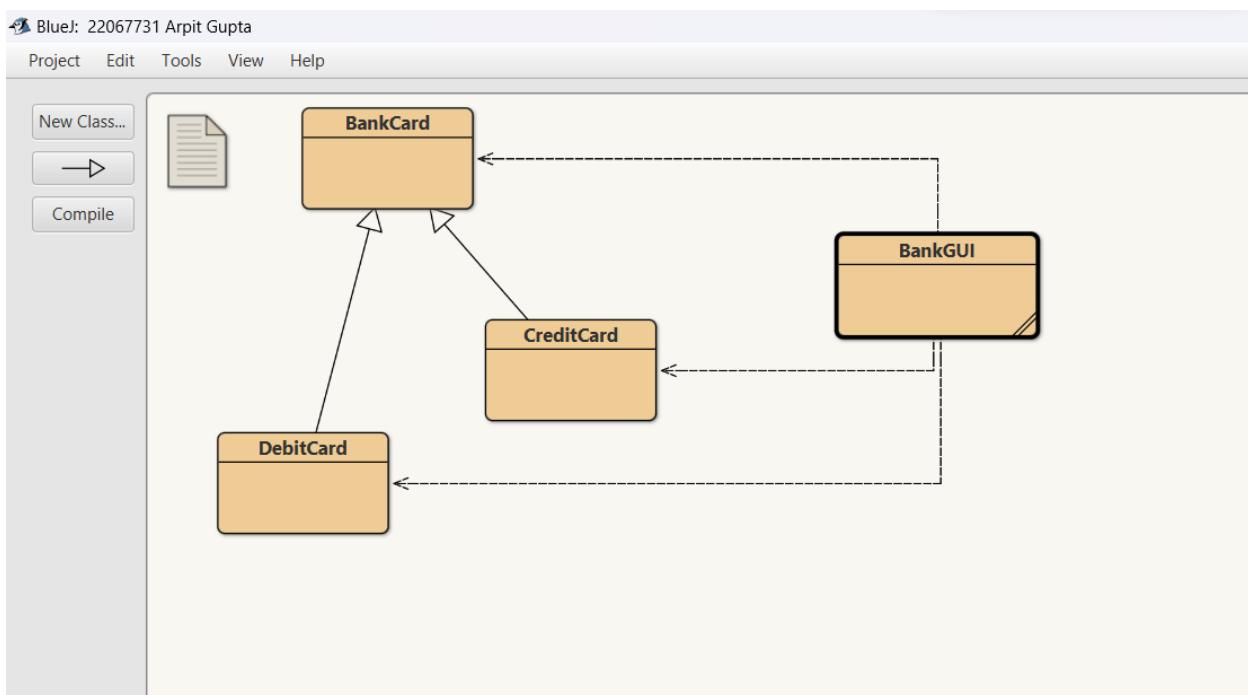


Figure 1. Screenshot of class diagram in BlueJ

## 2.1. Class Diagram for BankCard

<i>BankCard</i>	
-cardID: int	
-clientName: String	
-issuerBank: String	
-bankAccount: String	
-balanceAmount: double	
+<<constructor>>	BankCard(balanceAmount: double, cardID: int, bankAccount: String, issuerBank: String)
+setClientName(clientName: String): void	
+setBalanceAmount(balanceAmount: double): void	
+getClientName(): String	
+getBalanceAmount(): double	
+getCardID(): int	
+getIssuerBank(): String	
+getBankAccount(): String	
+display(): void	

Figure 2. Class Diagram for BankCard

## 2.2. Class Diagram for DebitCard

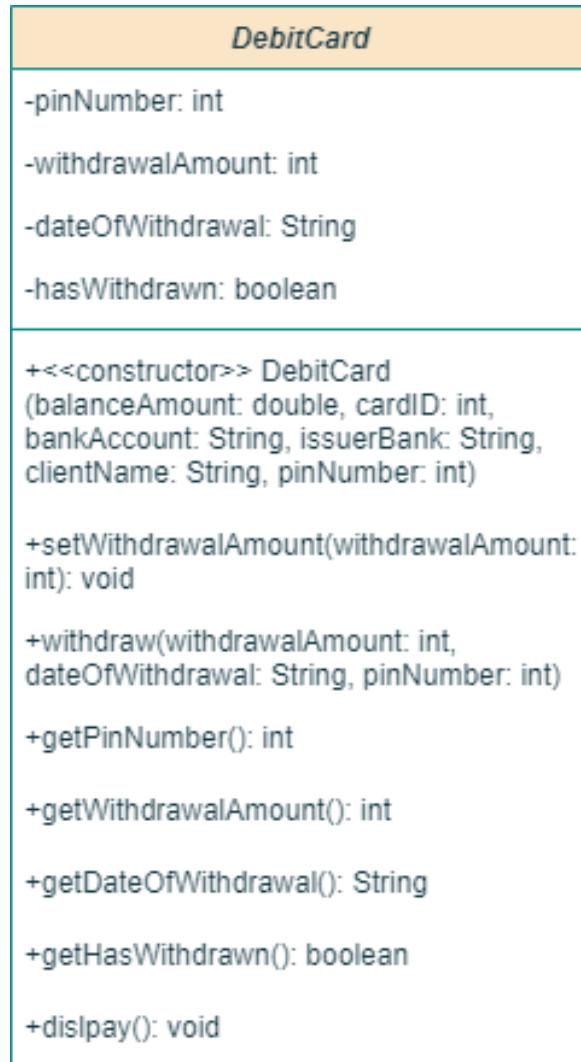


Figure 3. Class Diagram for DebitCard

### 2.3. Class Diagram for CreditCard

<i>CreditCard</i>	
-cvcNumber: int	
-creditLimit: double	
-interestRate: double	
-expirationDate: String	
-gracePeriod: int	
-isGranted: boolean	
+<<constructor>>	CreditCard(cardID: int, clientName: String, issuerBank: String, bankAccount: String, balanceAmount: double, cvcNumber : int, interestRate: double, expirationDate: String)
+setCreditLimit(creditLimit: double, gracePeriod: int): void	
+getCvcNumber(): int	
+getCreditLimit(): double	
+getInterestRate(): double	
+getExpirationDate(): String	
+getGracePeriod(): int	
+getIsGranted(): boolean	
+cancelCreditCard(): void	

Figure 4. Class Diagram for CreditCard

## 2.4. Class Diagram for BankGUI

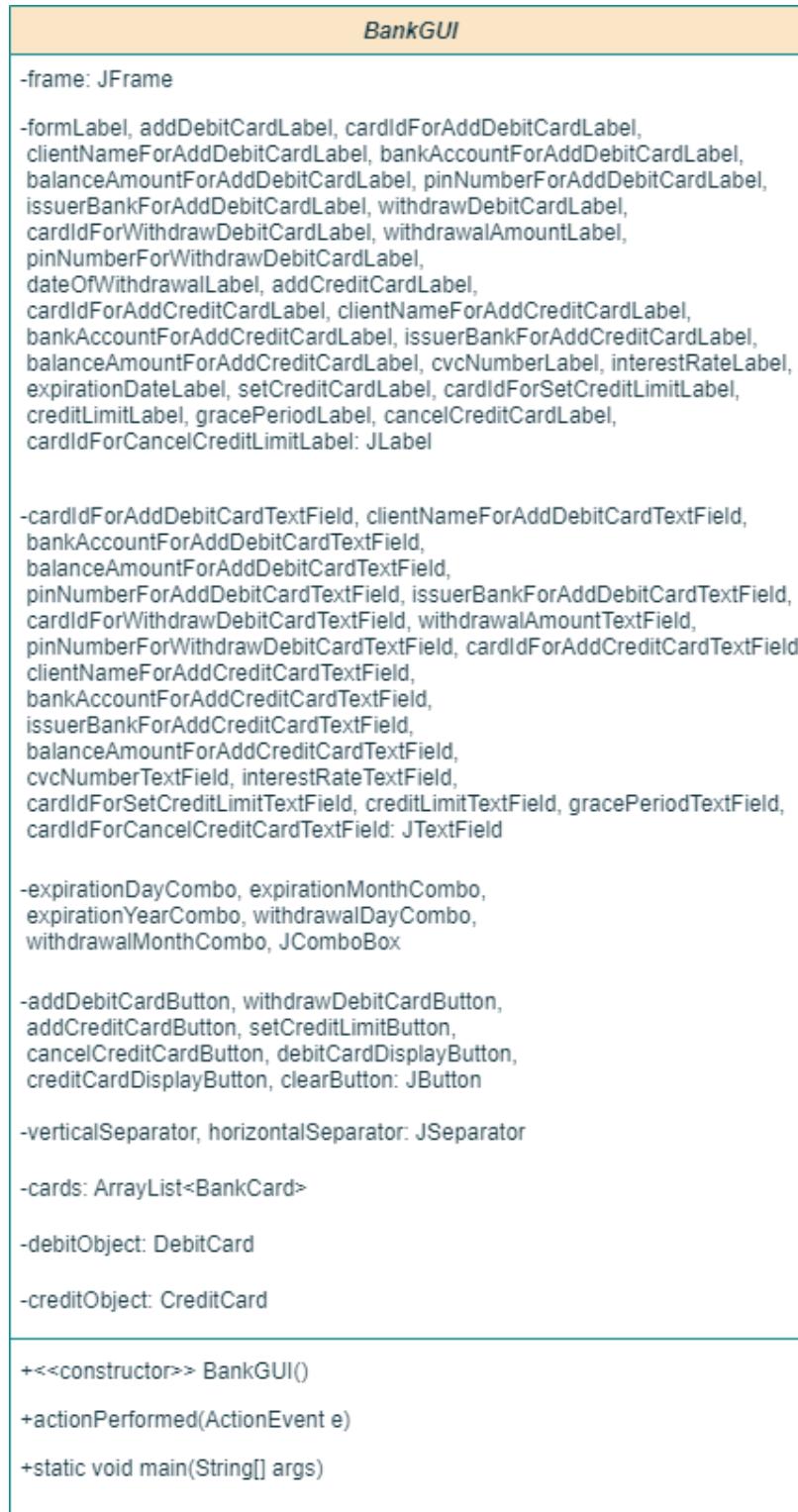


Figure 5. Class Diagram for BankGUI

## 2.5. Class Diagram

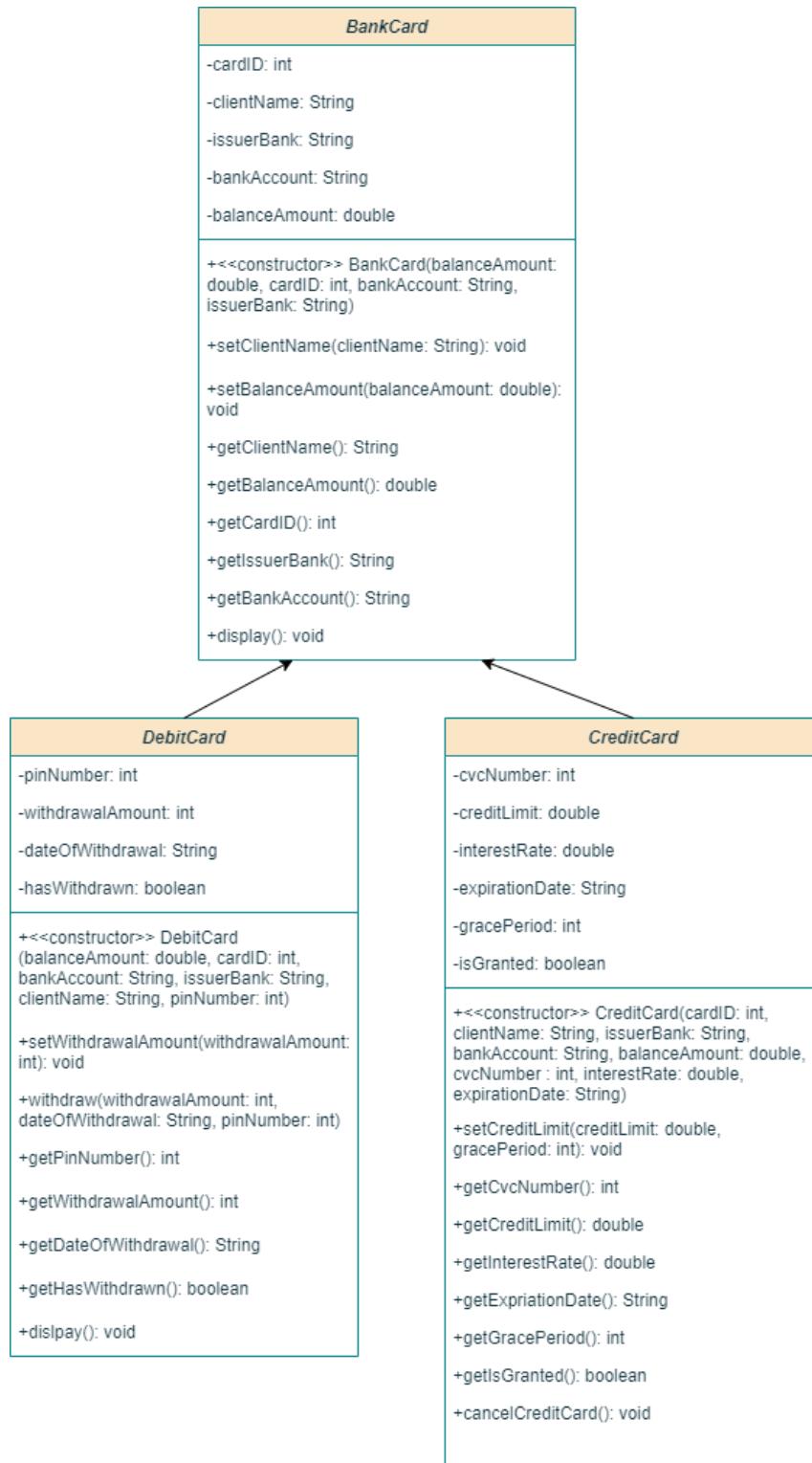


Figure 6. Class Diagram

### 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 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 required 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** NumberFormatException 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** NumberFormatException 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 NumberFormatException 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
                    DOWNGCAST 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

```

**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 convert the value from
        withdrawalDayCombo to String and store in it

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

**DECLARE** withdrawalYear as String and convert 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** NumberFormatException 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** NumberFormatException 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 NumberFormatException 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 issuerBankForAddCreditCard as 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 convert the value from expirationDayCombo to String and store in it

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

**DECLARE** expirationYear as String and convert 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 required 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** NumberFormatException 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** NumberFormatException 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** NumberFormatException 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 NumberFormatException 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
    END IF
```

```

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

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**

**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** NumberFormatException 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** NumberFormatException 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 NumberFormatException 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
    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 NumberFormatException 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
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

        DOWNCASET 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

                DOWNCASET 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 closeButton

    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.

**addCreditCardButton:** 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

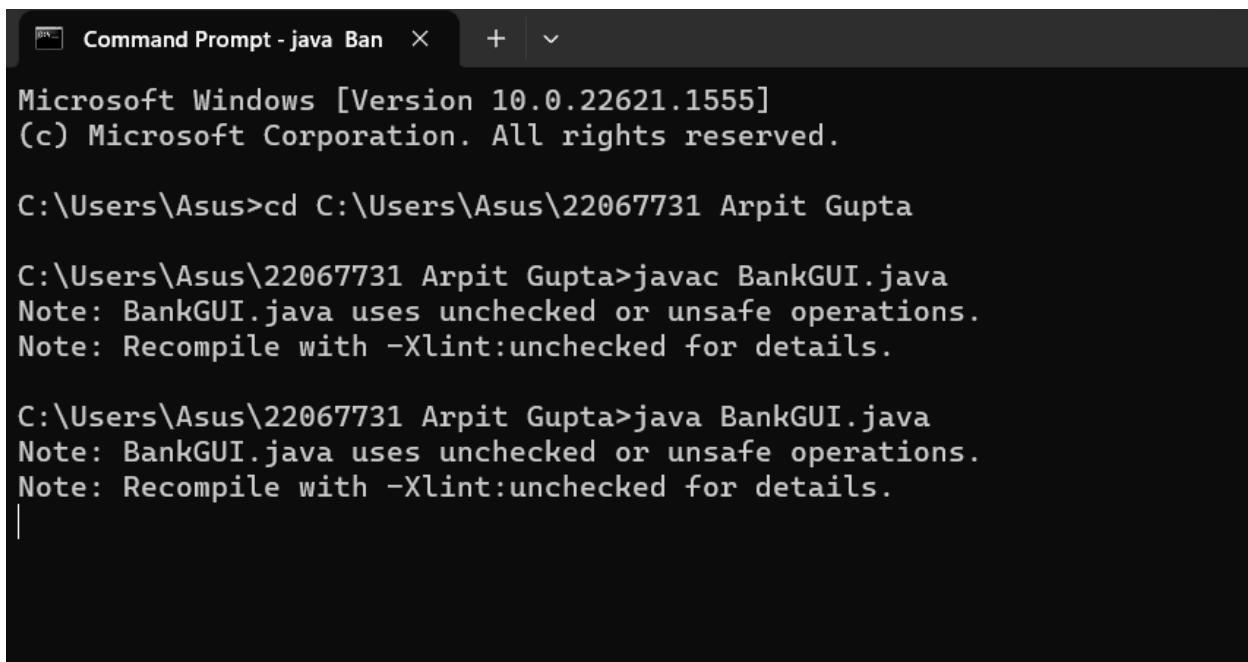
Testing is the process of continuously executing a program in order to detect errors and to check if the program does what it is supposed to do. It is done to prevent the occurrence of an error after the system is in use in daily life. During testing, if the result that occurred is same as the expected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

<b>Test No.</b>	1
Objective	To compile and run the program using command prompt
Action	<ul style="list-style-type: none"> <li>→ Open the command prompt and go to the directory of the project folder.</li> <li>→ To compile the codes: javac BankGUI.java was written</li> <li>→ To run the codes: java BankGUI.java was written</li> </ul>
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:



```

Command Prompt - java Ban  X  +  ▾

Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Asus>cd C:\Users\Asus\22067731 Arpit Gupta

C:\Users\Asus\22067731 Arpit Gupta>javac BankGUI.java
Note: BankGUI.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\Asus\22067731 Arpit Gupta>java BankGUI.java
Note: BankGUI.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
|
```

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

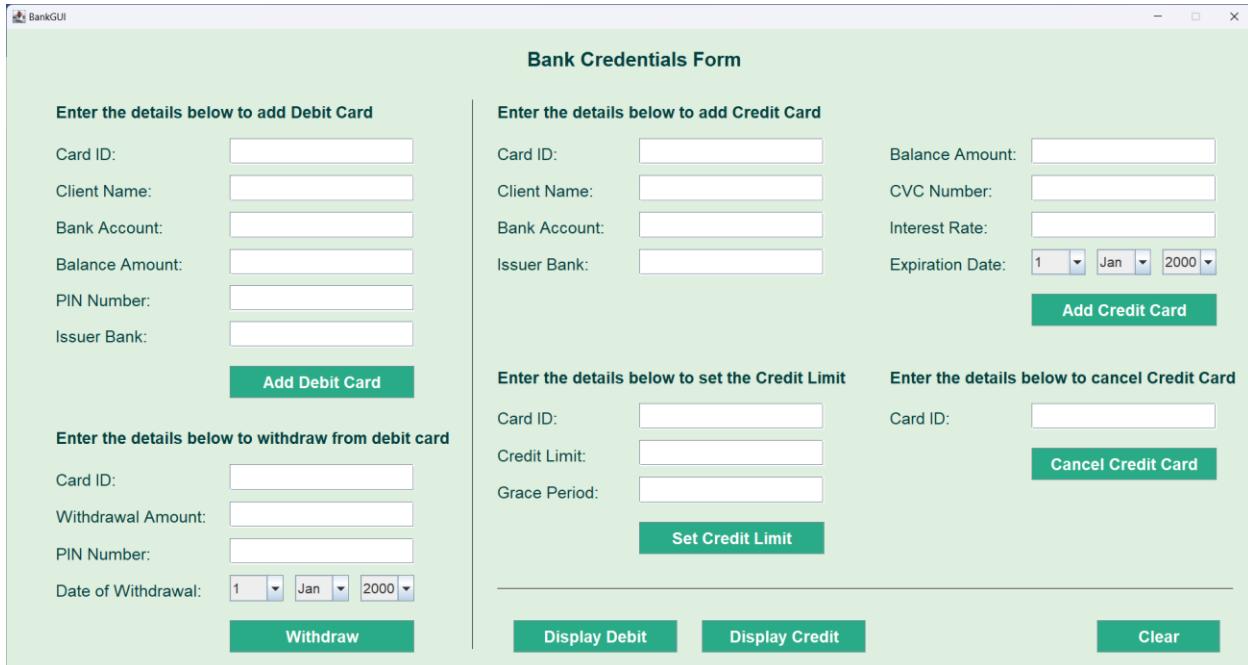


Figure 8. Output after compiling and running through terminal

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

### 5.2.1. To add a card for debit card

<b>Test No.</b>	<b>2.1</b>
Objective	To add a card for debit card
Action	<p>→ The required text fields are filled for adding debit card</p> <p>Card ID: "7000"</p> <p>Client Name: "John Smith"</p> <p>Bank Account: "900091"</p> <p>Balance Amount: "100000.0"</p> <p>PIN Number: "1014"</p> <p>Issuer Bank: "Nepal Rastriya Bank"</p> <p>→ Click on Add Debit Card Button</p>
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:

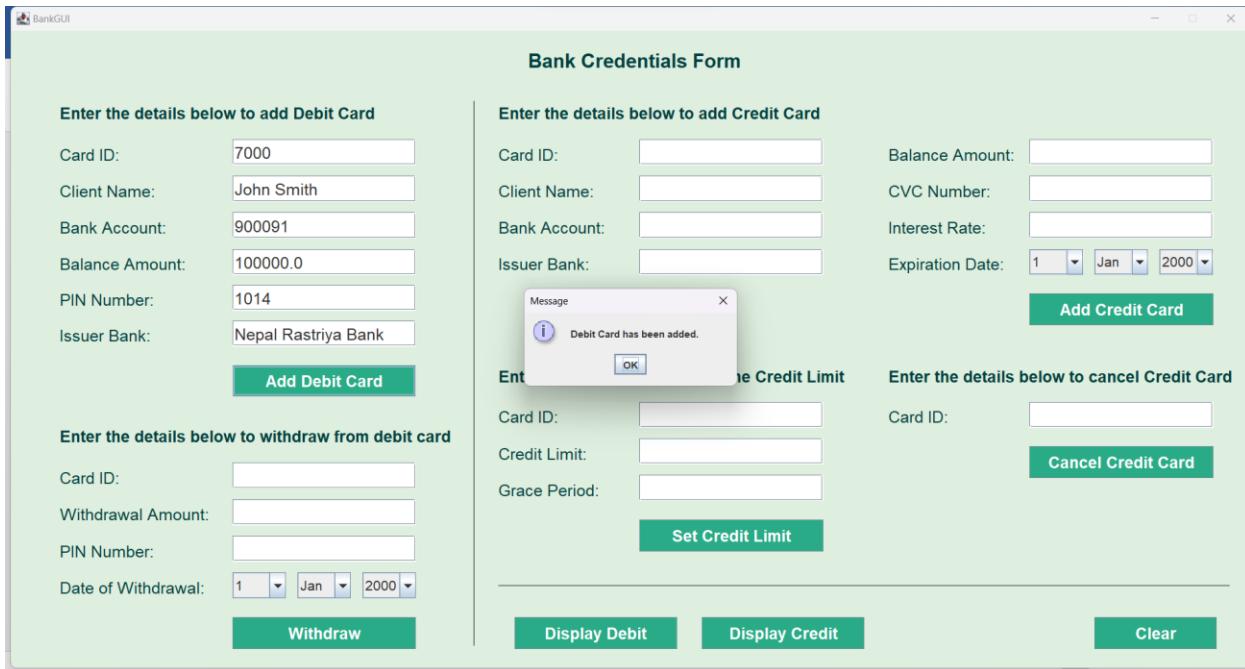


Figure 9. Debit card added

5.2.2. To add a card for credit card

<b>Test No.</b>	<b>2.2</b>
Objective	To add a card for credit card
Action	<p>→ The required text fields are filled for adding credit card</p> <p>Card ID: "8000"</p> <p>Client Name: "Harry Brook"</p> <p>Bank Account: "900092"</p> <p>Issuer Bank: "Nepal Rastriya Bank"</p> <p>Balance Amount: "50000.0"</p> <p>CVC Number: "563"</p> <p>Interest Rate: "7.5"</p> <p>Expiration Date: 29-Feb-2020</p> <p>→ Click on Add Credit Card Button</p>
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:

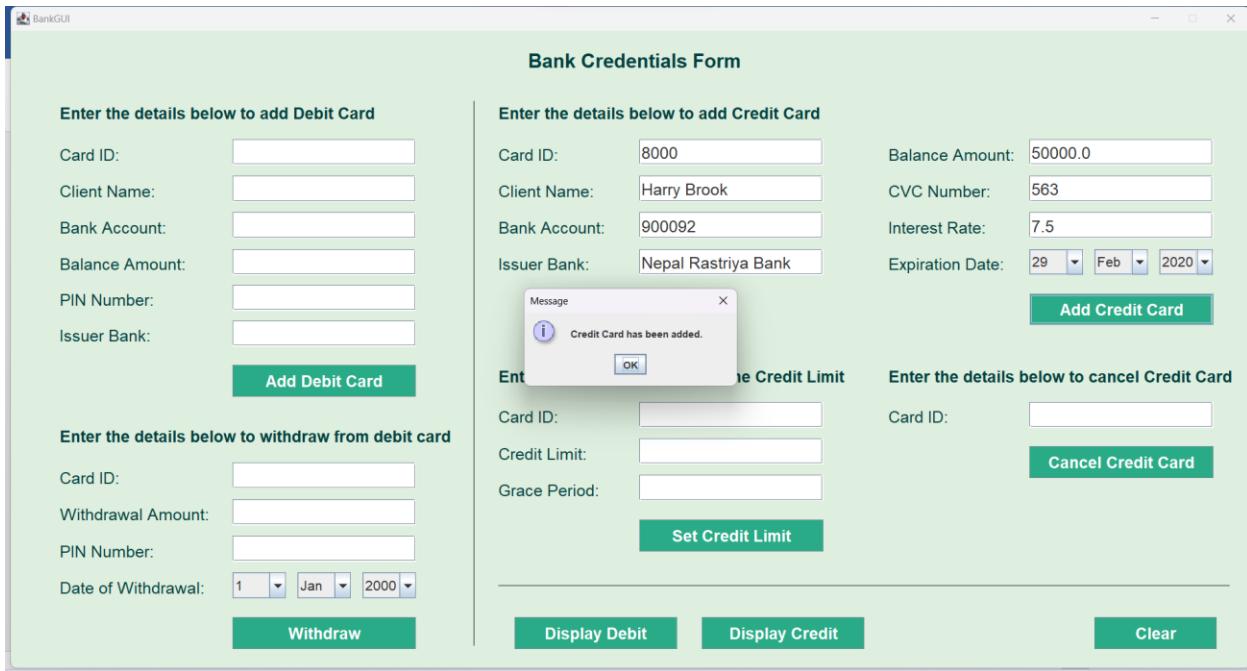


Figure 10. Credit Card added

## 5.2.3. To withdraw amount from debit card

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

*Table 4. Withdraw from debit card*

## Output Result:

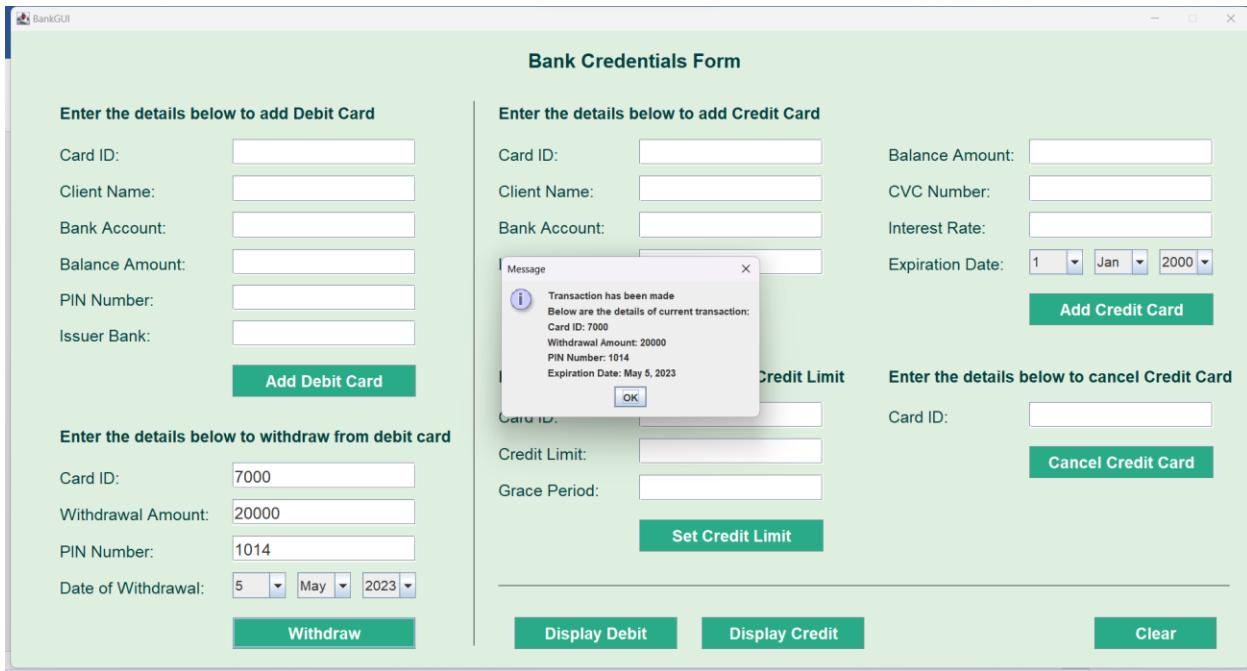


Figure 11. Withdraw from Debit Card

5.2.4. To set the credit limit to a credit card

<b>Test No.</b>	<b>2.4</b>
Objective	To set the credit limit to a credit card
Action	<p>→ The required text fields are filled for setting the credit limit</p> <p>Card ID: "8000"</p> <p>Credit Limit: "75000"</p> <p>Grace Period: "30"</p> <p>→ Click on Set Credit Limit Button</p>
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 were displayed as a pop-up message.
Result	The test is successful.

*Table 5. Set the credit limit*

## Output Result:

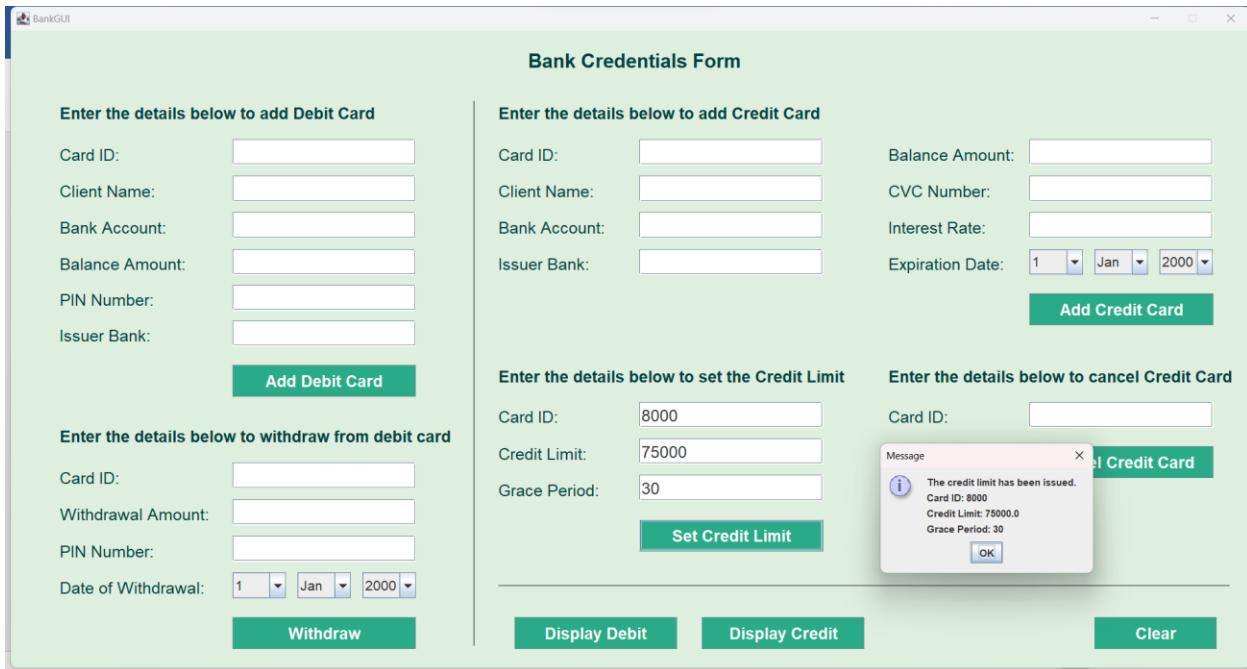


Figure 12. Set the credit limit

## 5.2.5. To cancel a credit card

<b>Test No.</b>	<b>2.5</b>
Objective	To cancel a credit card
Action	<p>→ The required text fields are filled for cancelling a credit card</p> <p>Card ID: "8000"</p> <p>→ Click on Cancel Credit Card Button</p>
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

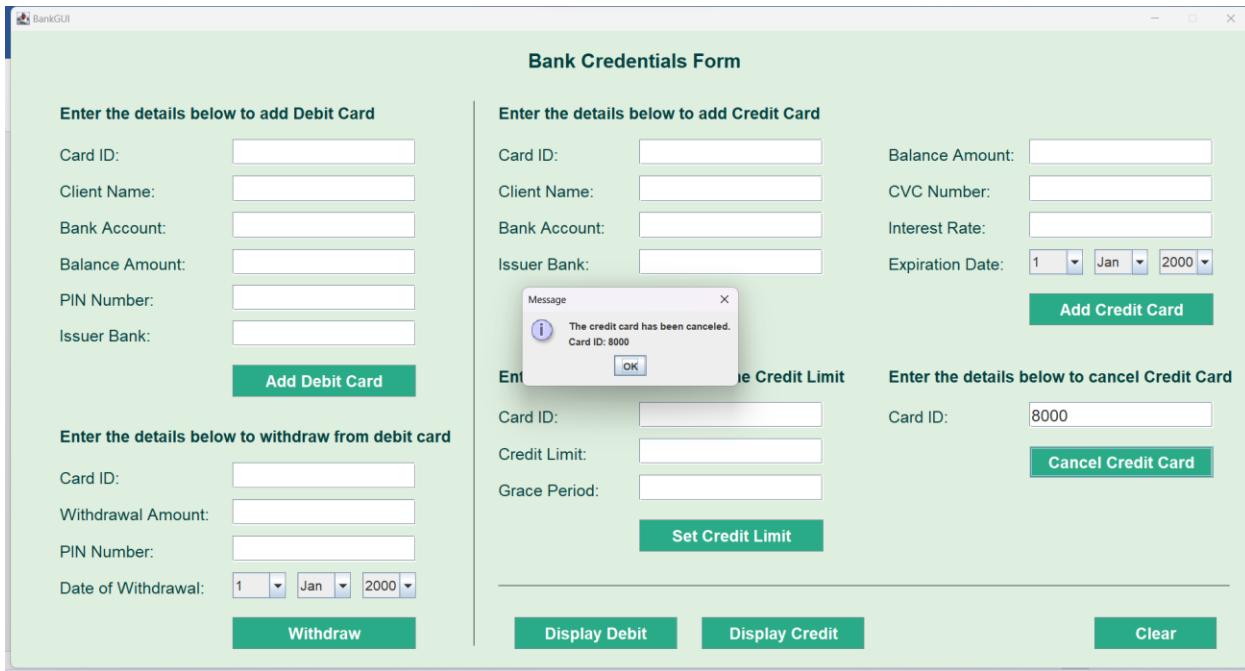


Figure 13. Cancel a credit card

### 5.3. Test Validations

#### 5.3.1. Tests for empty text fields

<b>Test No.</b>	<b>3.1</b>
Objective	To test for empty text fields
Action	<p>→ All the text fields are left empty and the respective buttons are clicked.</p> <ul style="list-style-type: none"> <li>• Add Debit Card Button</li> <li>• Withdraw Button</li> <li>• Add Credit Card Button</li> <li>• Set Credit Limit Button</li> <li>• Cancel Credit Limit Button</li> <li>• Display Debit Button</li> <li>• Display Credit Button</li> </ul>
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):

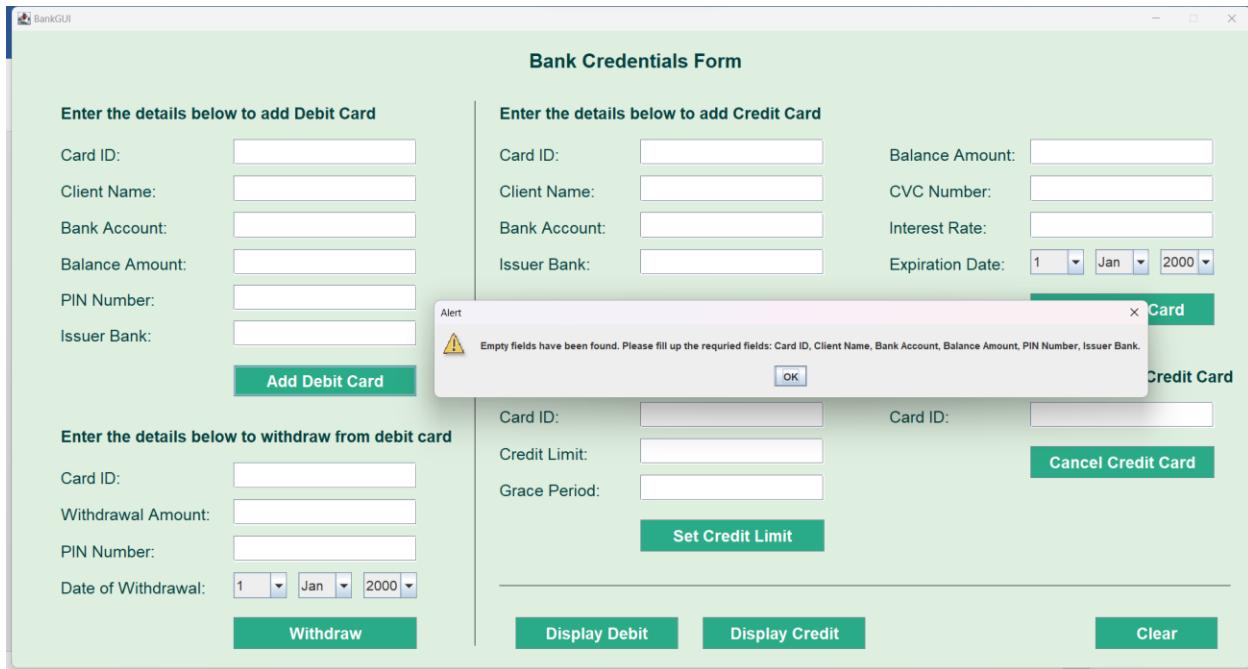


Figure 14. Add Debit Card with empty Text Fields

## Output Result 2 (Click on Withdraw):

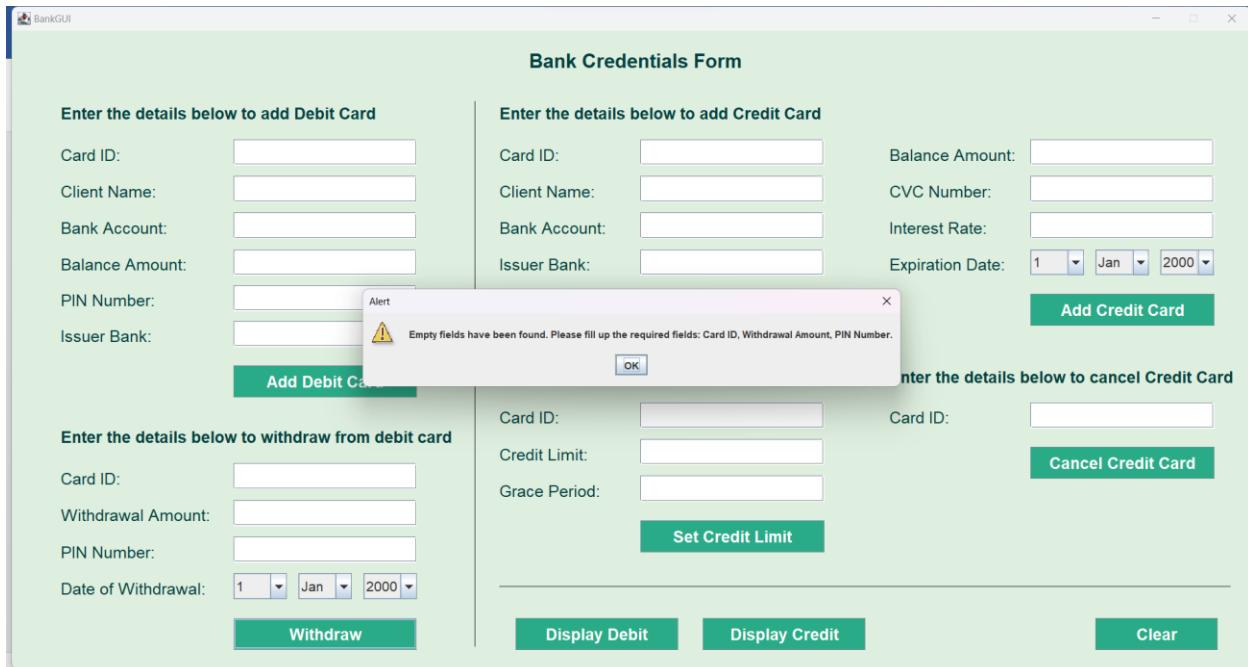


Figure 15. Withdraw with empty Text Fields

## Output Result 3 (Click on Add Credit Card):

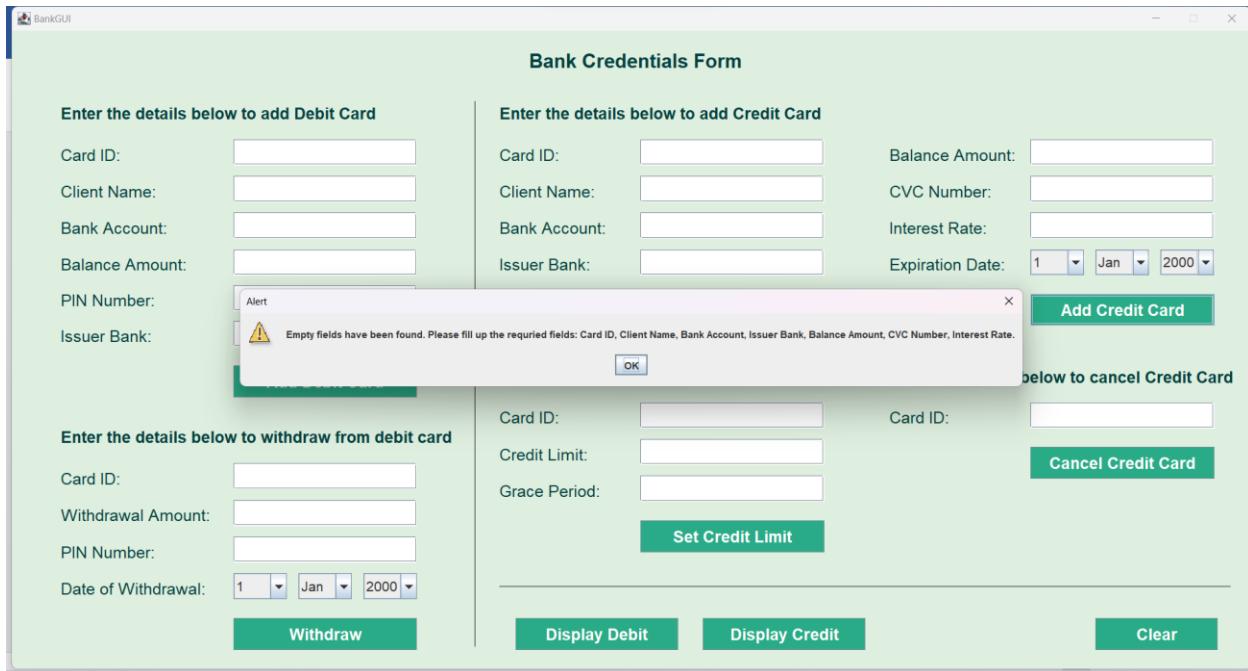


Figure 16. Add Credit Card with empty Text Fields

## Output Result 4 (Click on Set Credit Limit):

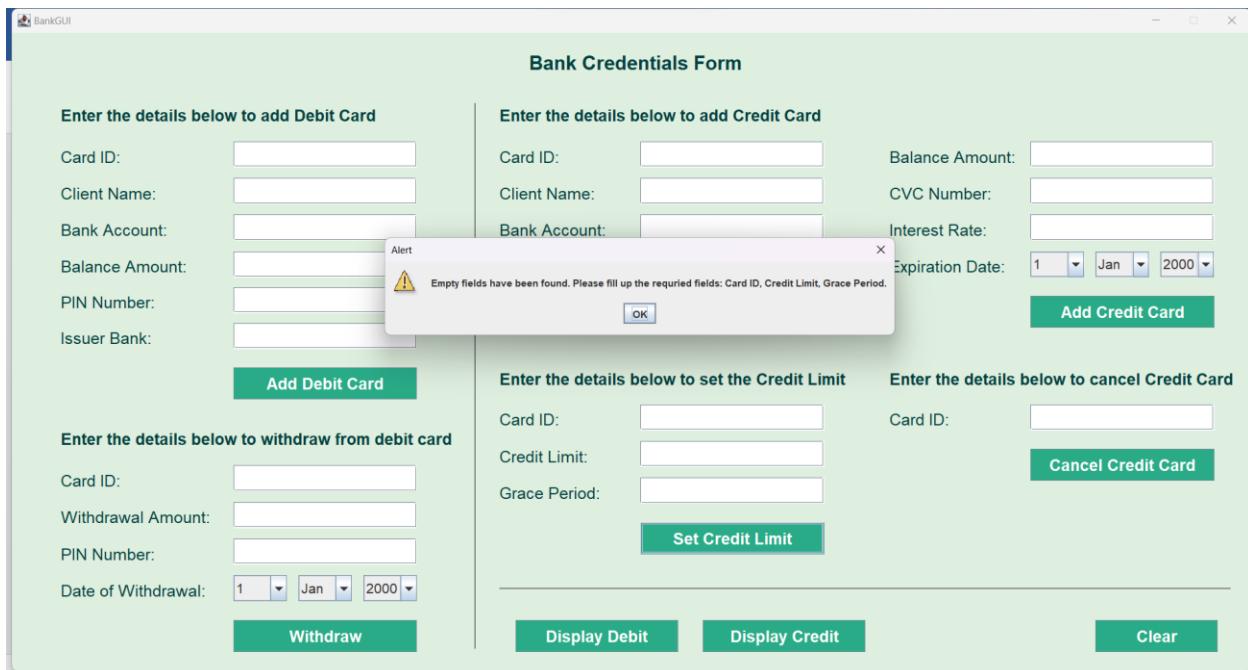


Figure 17. Set Credit Limit with empty Text Fields

## Output Result 5 (Click on Cancel Credit Card):

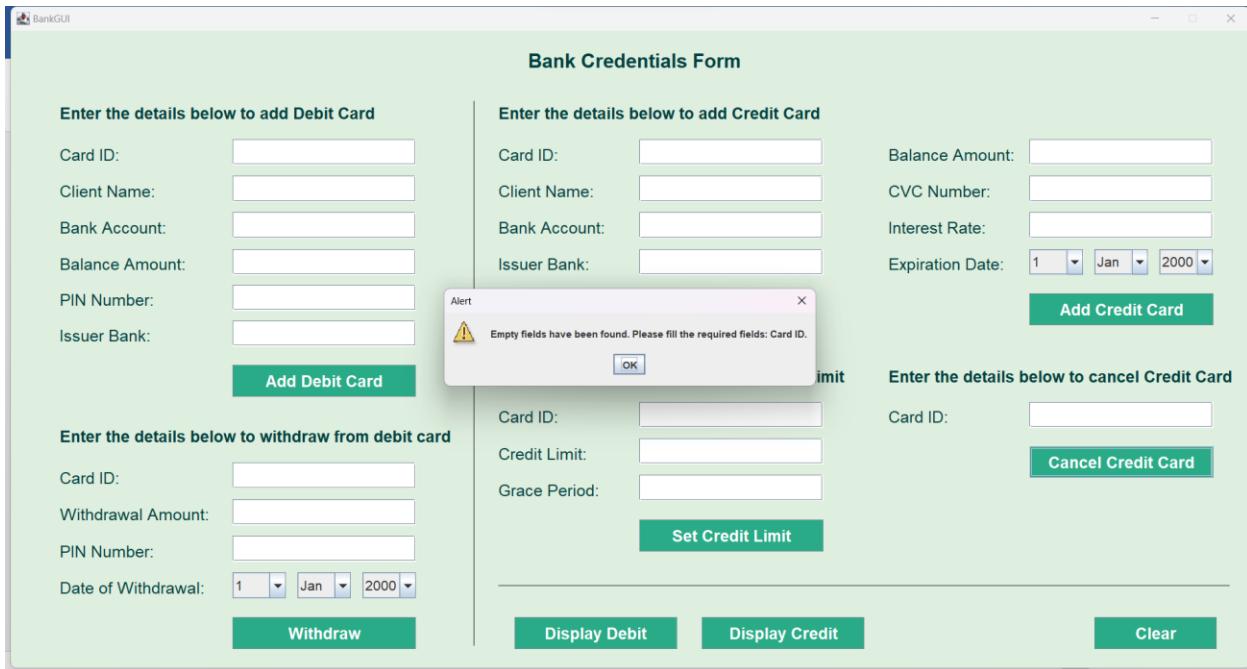


Figure 18. Cancel Credit Card with empty Text Fields

## Output Result 6 (Click on Display Debit):

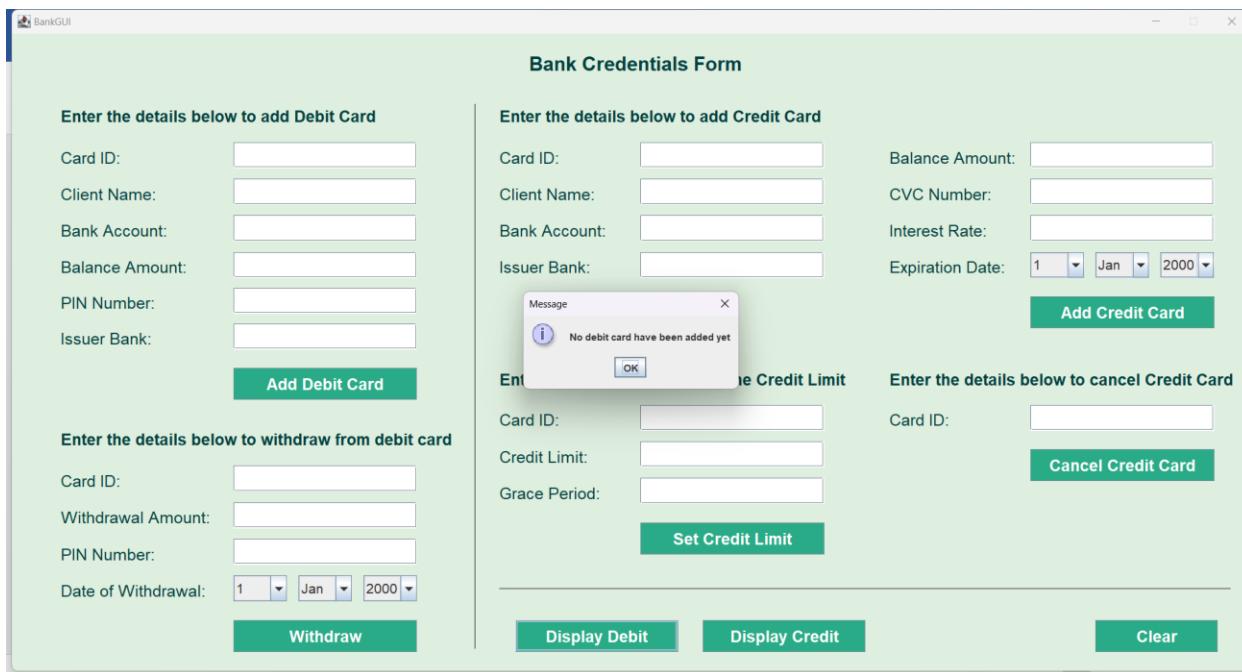


Figure 19. Display Debit without adding any debit card

## Output Result 7 (Click on Display Credit):

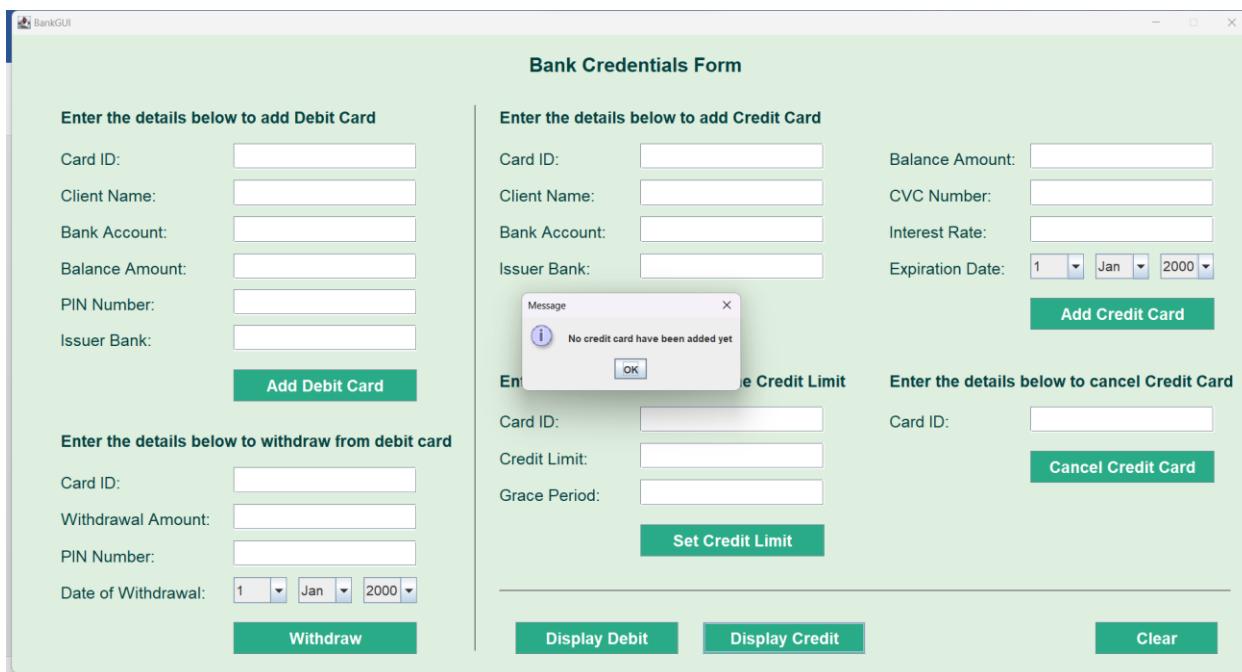


Figure 20. Display Credit without adding any credit card

5.3.2. To enter invalid input for adding Debit Card

a. Input invalid Card ID for adding Debit Card

<b>Test No.</b>	<b>3.2 (a)</b>
Objective	To enter invalid Card ID as input for adding Debit Card
Action	<p>→ Wrong value for Card ID is entered for add debit card</p> <ul style="list-style-type: none"> <li>• Card ID: “AAAA”</li> <li>• Client Name: “John Smith”</li> <li>• Bank Account: “900091”</li> <li>• Balance Amount: “100000.0”</li> <li>• PIN Number: “1014”</li> <li>• Issuer Bank: “Nepal Rastriya Bank”</li> </ul> <p>→ Click on Add Debit Card Button</p>
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:

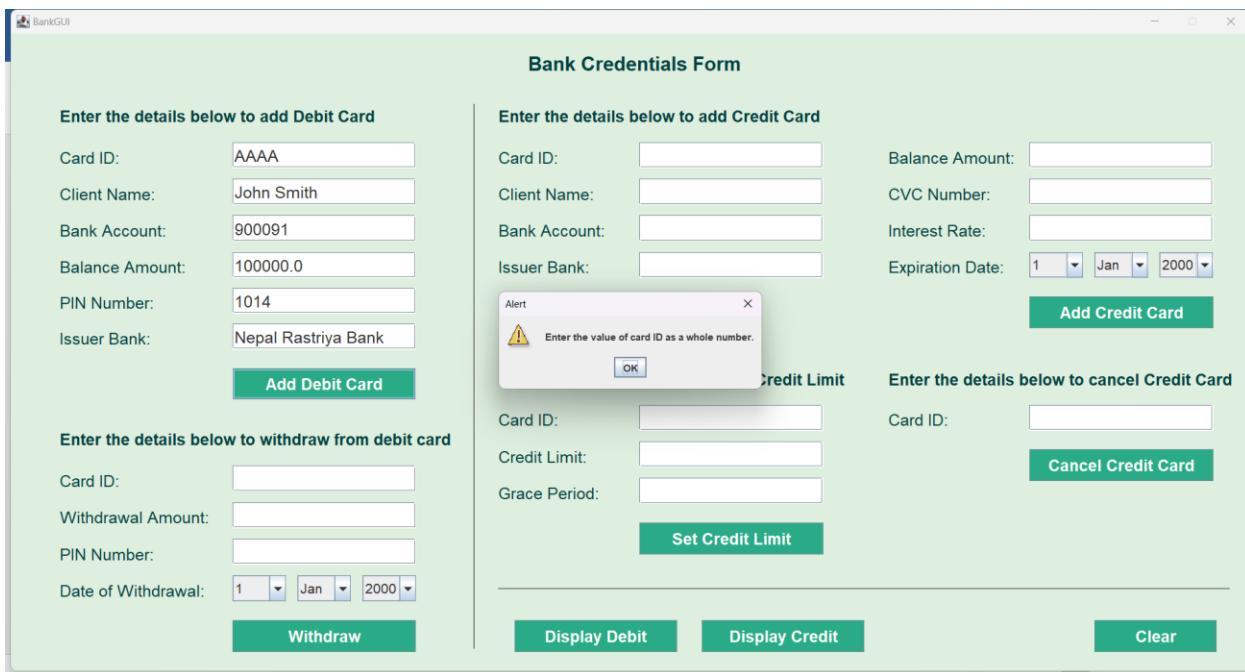


Figure 21. Screenshot of adding invalid Card ID (Debit Card)

## b. Input invalid Balance Amount for adding debit card

<b>Test No.</b>	<b>3.2 (b)</b>
Objective	To enter invalid Balance Amount as input for adding Debit Card
Action	<p>→ Wrong value for Balance Amount is entered for add debit card</p> <ul style="list-style-type: none"> <li>• Card ID: “1000”</li> <li>• Client Name: “John Smith”</li> <li>• Bank Account: “900091”</li> <li>• Balance Amount: “AAAA”</li> <li>• PIN Number: “1014”</li> <li>• Issuer Bank: “Nepal Rastriya Bank”</li> </ul> <p>→ Click on Add Debit Card Button</p>
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:

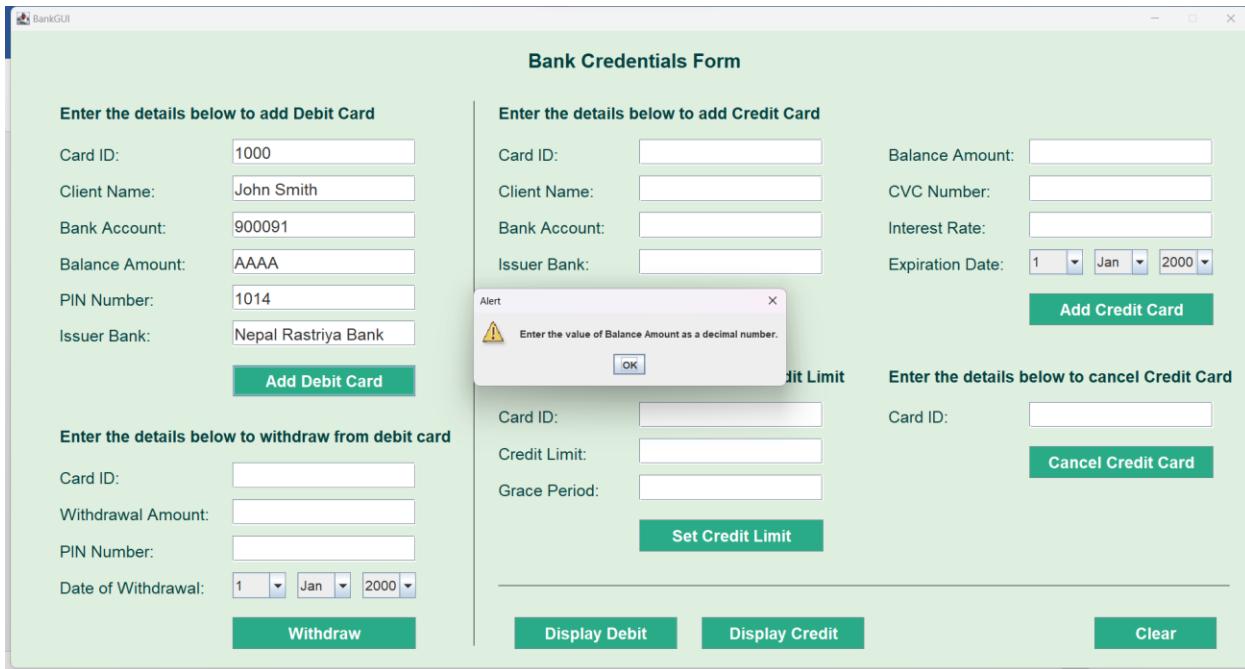


Figure 22. Screenshot of entering invalid Balance Amount (Debit Card)

## c. Input invalid PIN Number for adding Debit Card

<b>Test No.</b>	<b>3.2 (c)</b>
Objective	To enter invalid PIN Number as input for adding Debit Card
Action	<p>→ Wrong value for Balance Amount is entered for add debit card</p> <ul style="list-style-type: none"> <li>• Card ID: “1000”</li> <li>• Client Name: “John Smith”</li> <li>• Bank Account: “900091”</li> <li>• Balance Amount: “100000.0”</li> <li>• PIN Number: “AAAA”</li> <li>• Issuer Bank: “Nepal Rastriya Bank”</li> </ul> <p>→ Click on Add Debit Card Button</p>
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:

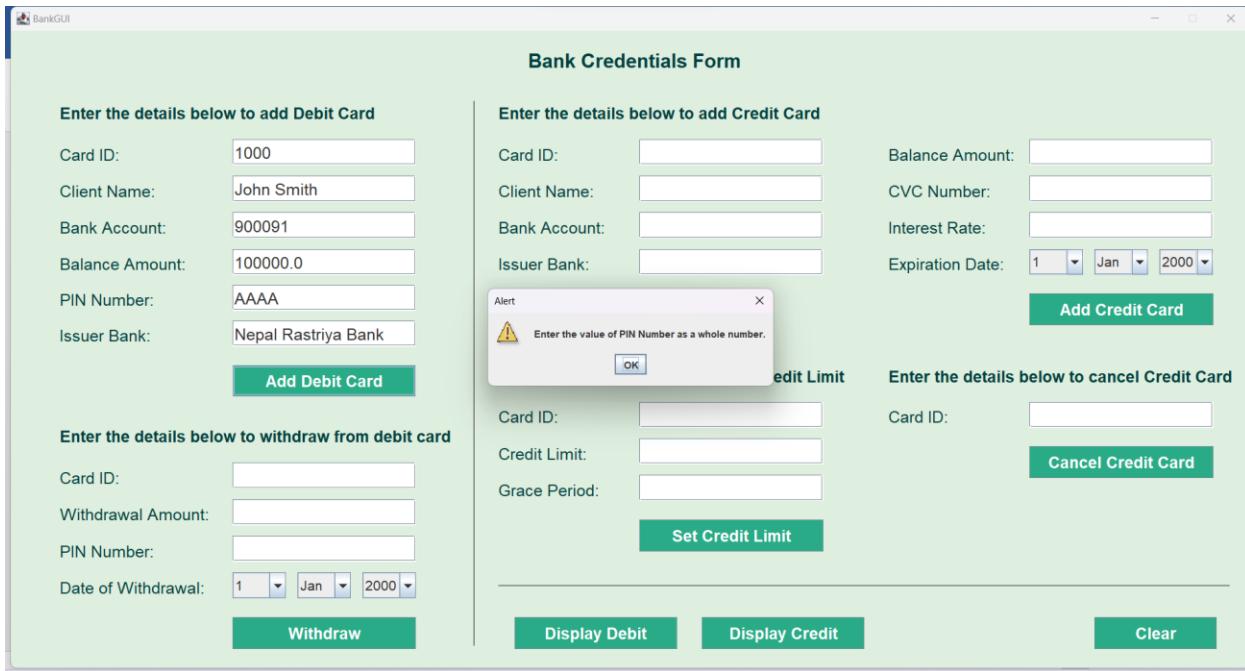


Figure 23. Screenshot of entering invalid PIN Number (Debit Card)

5.3.3. To enter invalid input for adding Credit Card

a. Input invalid Card ID for adding Credit Card

<b>Test No.</b>	<b>3.3 (a)</b>
Objective	To enter invalid Card ID as input for adding Credit Card
Action	<p>→ Wrong value for Card ID is entered for add credit card</p> <ul style="list-style-type: none"> <li>• Card ID: "AAAA"</li> <li>• Client Name: "Harry Brook"</li> <li>• Bank Account: "900092"</li> <li>• Issuer Bank: "Nepal Rastriya Bank"</li> <li>• Balance Amount: "50000.0"</li> <li>• CVC Number: "563"</li> <li>• Interest Rate: "7.5"</li> <li>• Expiration Date: 1-Jan-2000</li> </ul> <p>→ Click on Add Credit Card Button</p>
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:

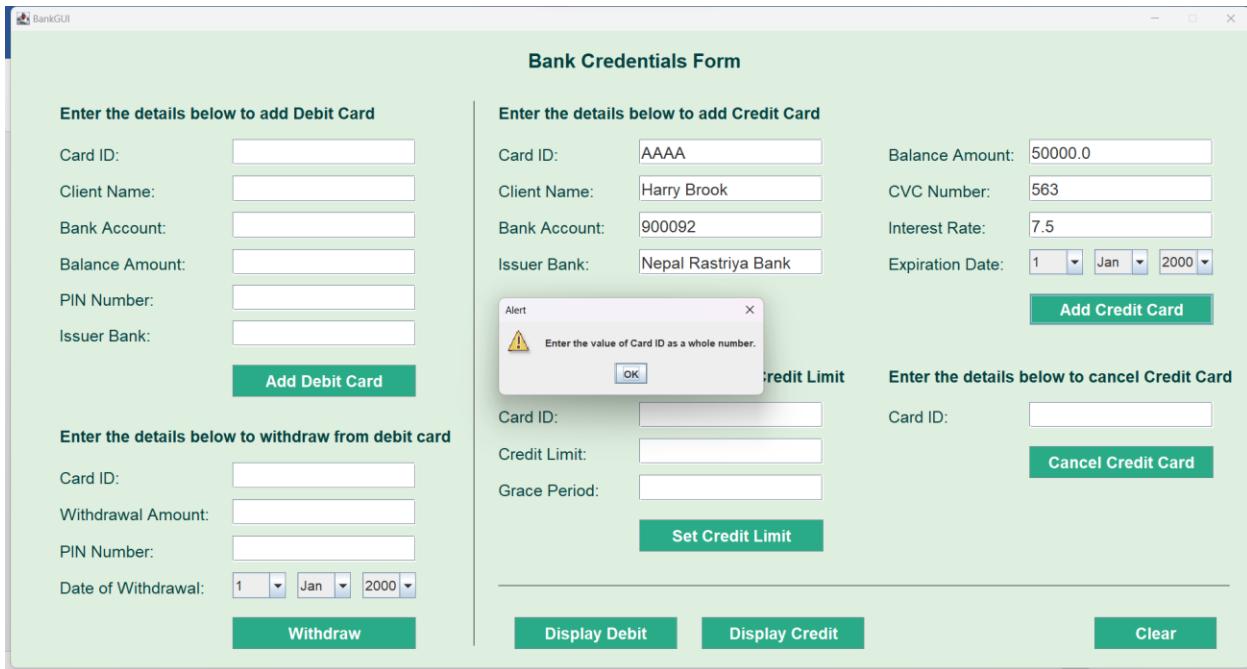


Figure 24. Screenshot of entering invalid Card ID (Credit Card)

b. Input invalid Balance Amount for adding Credit Card

<b>Test No.</b>	<b>3.3 (b)</b>
Objective	To enter invalid Balance amount as input for adding Credit Card
Action	<p>→ Wrong value for Balance Amount is entered for add credit card</p> <ul style="list-style-type: none"> <li>• Card ID: “2000”</li> <li>• Client Name: “Harry Brook”</li> <li>• Bank Account: “900092”</li> <li>• Issuer Bank: “Nepal Rastriya Bank”</li> <li>• Balance Amount: “AAAA”</li> <li>• CVC Number: “563”</li> <li>• Interest Rate: “7.5”</li> <li>• Expiration Date: 1-Jan-2000</li> </ul> <p>→ Click on Add Credit Card Button</p>
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:

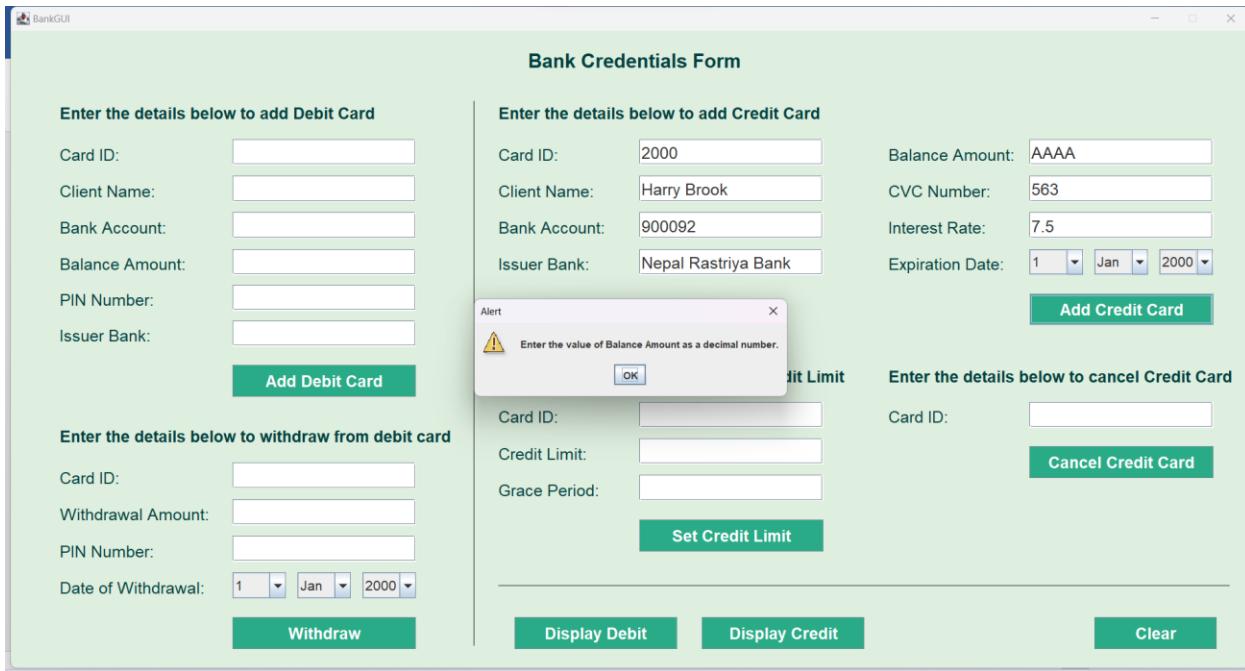


Figure 25. Screenshot of entering invalid Balance Amount (Credit Card)

## c. Input invalid CVC Number for adding Credit Card

<b>Test No.</b>	<b>3.3 (c)</b>
Objective	To enter invalid CVC Number as input for adding Credit Card
Action	<p>→ Wrong value for CVC Number is entered for add credit card</p> <ul style="list-style-type: none"> <li>• Card ID: “2000”</li> <li>• Client Name: “Harry Brook”</li> <li>• Bank Account: “900092”</li> <li>• Issuer Bank: “Nepal Rastriya Bank”</li> <li>• Balance Amount: “50000.0”</li> <li>• CVC Number: “AAAA”</li> <li>• Interest Rate: “7.5”</li> <li>• Expiration Date: 1-Jan-2000</li> </ul> <p>→ Click on Add Credit Card Button</p>
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:

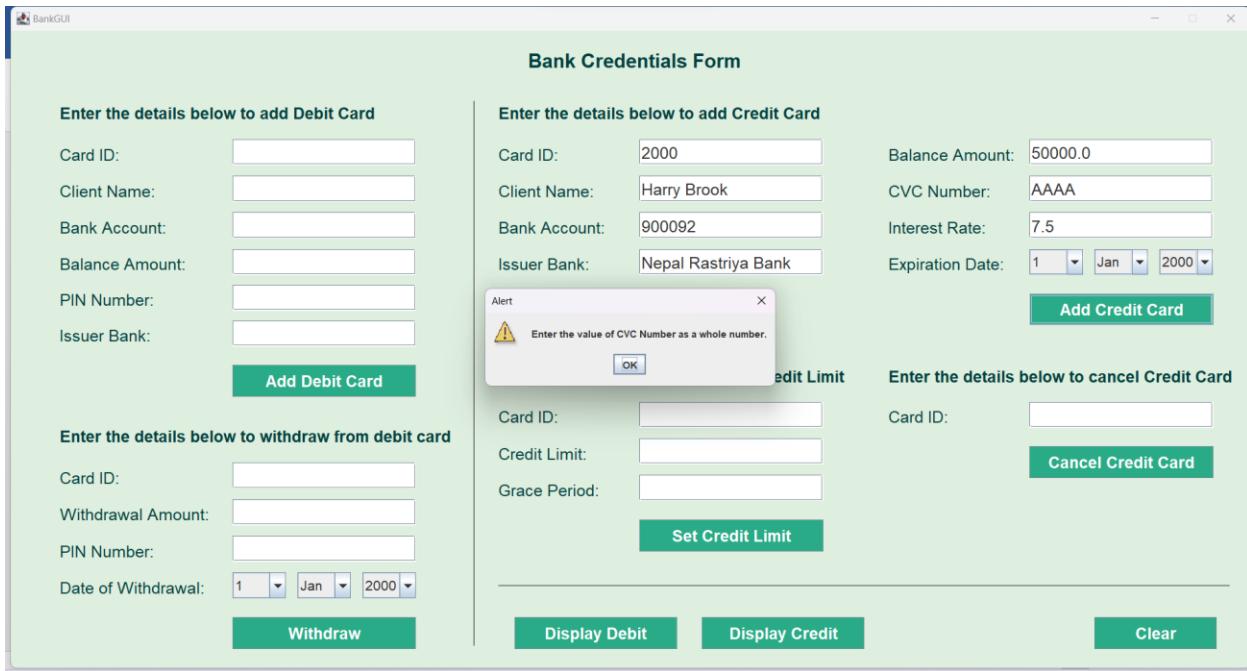


Figure 26. Screenshot of entering invalid CVC Number (Credit Card)

## d. Input invalid Interest Rate for adding Credit Card

<b>Test No.</b>	<b>3.3 (d)</b>
Objective	To enter invalid Interest Rate as input for adding Credit Card
Action	<p>→ Wrong value for Interest Rate is entered for add credit card</p> <ul style="list-style-type: none"> <li>• Card ID: “2000”</li> <li>• Client Name: “Harry Brook”</li> <li>• Bank Account: “900092”</li> <li>• Issuer Bank: “Nepal Rastriya Bank”</li> <li>• Balance Amount: “50000.0”</li> <li>• CVC Number: “563”</li> <li>• Interest Rate: “AAAA”</li> <li>• Expiration Date: 1-Jan-2000</li> </ul> <p>→ Click on Add Credit Card Button</p>
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:

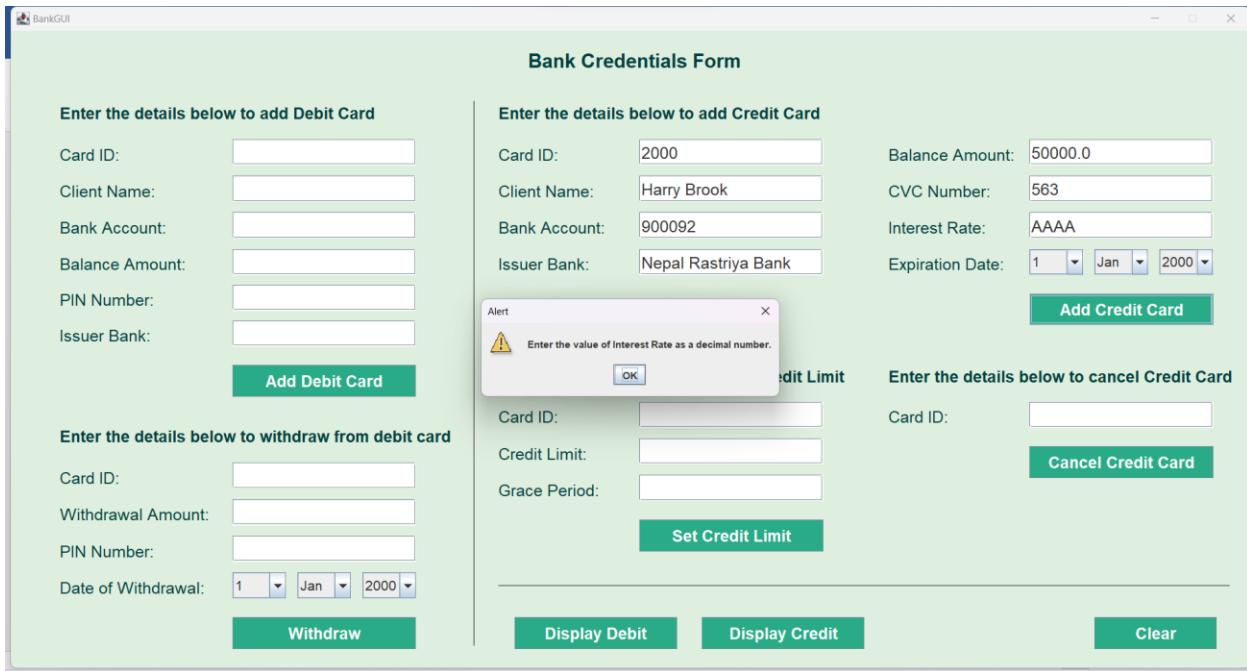


Figure 27. Screenshot of entering invalid Interest Rate (Credit Card)

5.3.4. To enter invalid input while withdrawing from debit card

a. Invalid Card ID for withdrawing from debit card

<b>Test No.</b>	<b>3.4 (a)</b>
Objective	To enter invalid Card ID as input for withdrawing from debit card
Action	<p>→ Invalid value for Card ID is entered for withdrawing from debit card</p> <ul style="list-style-type: none"> <li>• Card ID: "AAAA"</li> <li>• Withdrawal Amount: "4000"</li> <li>• PIN Number: "1014"</li> <li>• Date of Withdrawal: 30-Jul-2005</li> </ul> <p>→ Click on Withdraw Button</p>
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:

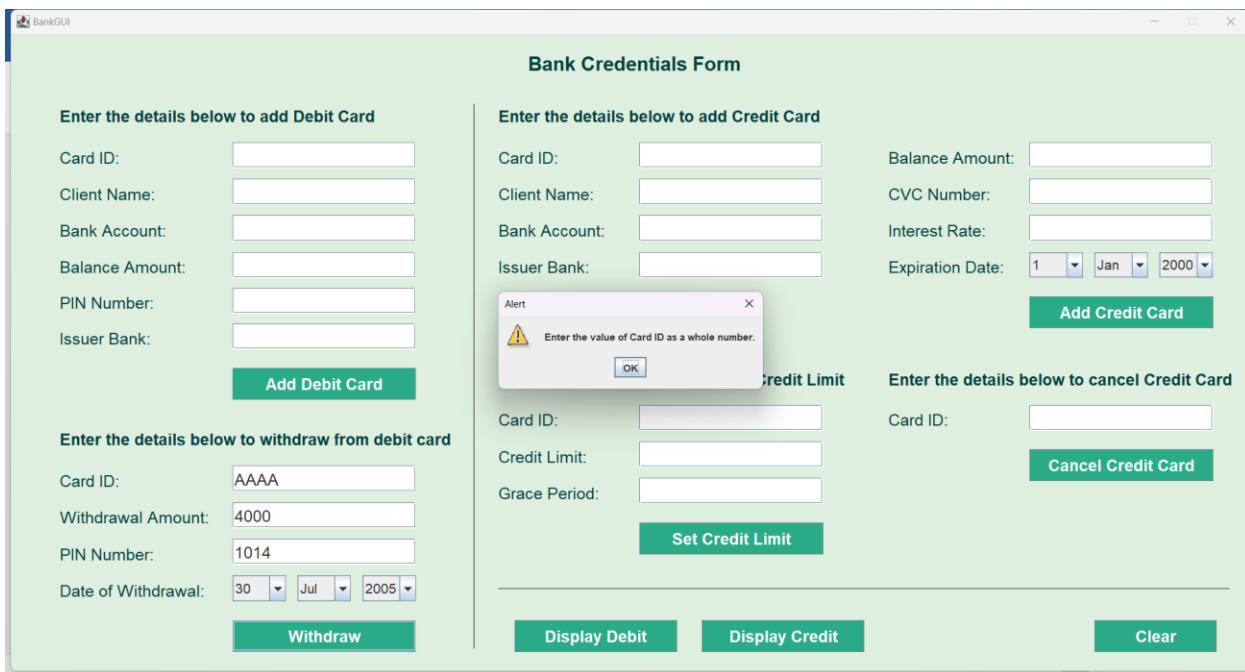


Figure 28. Screenshot of entering invalid Card ID (Withdraw)

## b. Invalid Withdrawal Amount for withdrawing from debit card

<b>Test No.</b>	<b>3.4 (b)</b>
Objective	To enter invalid Withdrawal Amount as input for withdrawing from debit card
Action	<p>→ Invalid value for PIN Number is entered for withdrawing from debit card</p> <ul style="list-style-type: none"> <li>• Card ID: “1000”</li> <li>• Withdrawal Amount: “AAAA”</li> <li>• PIN Number: “1014”</li> <li>• Date of Withdrawal: 30-Jul-2005</li> </ul> <p>→ Click on Withdraw Button</p>
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:

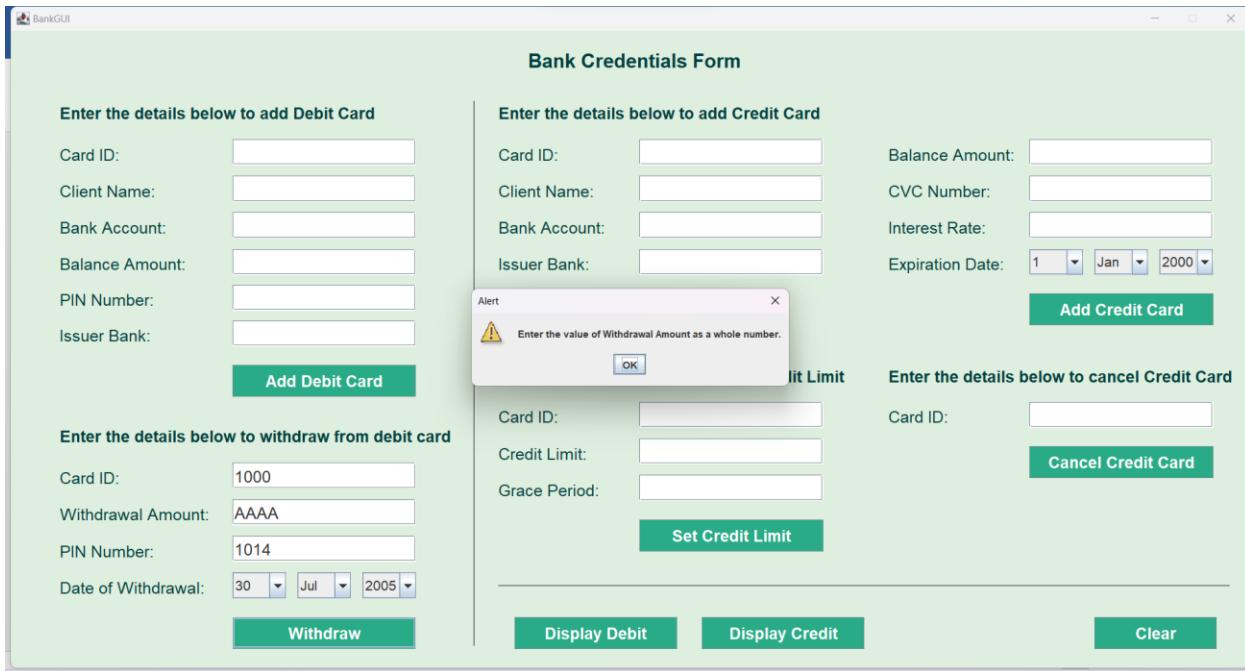


Figure 29. Screenshot of entering invalid Withdrawal Amount (Withdraw)

## c. Invalid PIN Number for withdrawing from debit card

<b>Test No.</b>	<b>3.4 (c)</b>
Objective	To enter invalid PIN Number as input for withdrawing from debit card
Action	<p>→ Invalid value for PIN Number is entered for withdrawing from debit card</p> <ul style="list-style-type: none"> <li>• Card ID: “1000”</li> <li>• Withdrawal Amount: “4000”</li> <li>• PIN Number: “AAAA”</li> <li>• Date of Withdrawal: 30-Jul-2005</li> </ul> <p>→ Click on Withdraw Button</p>
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:

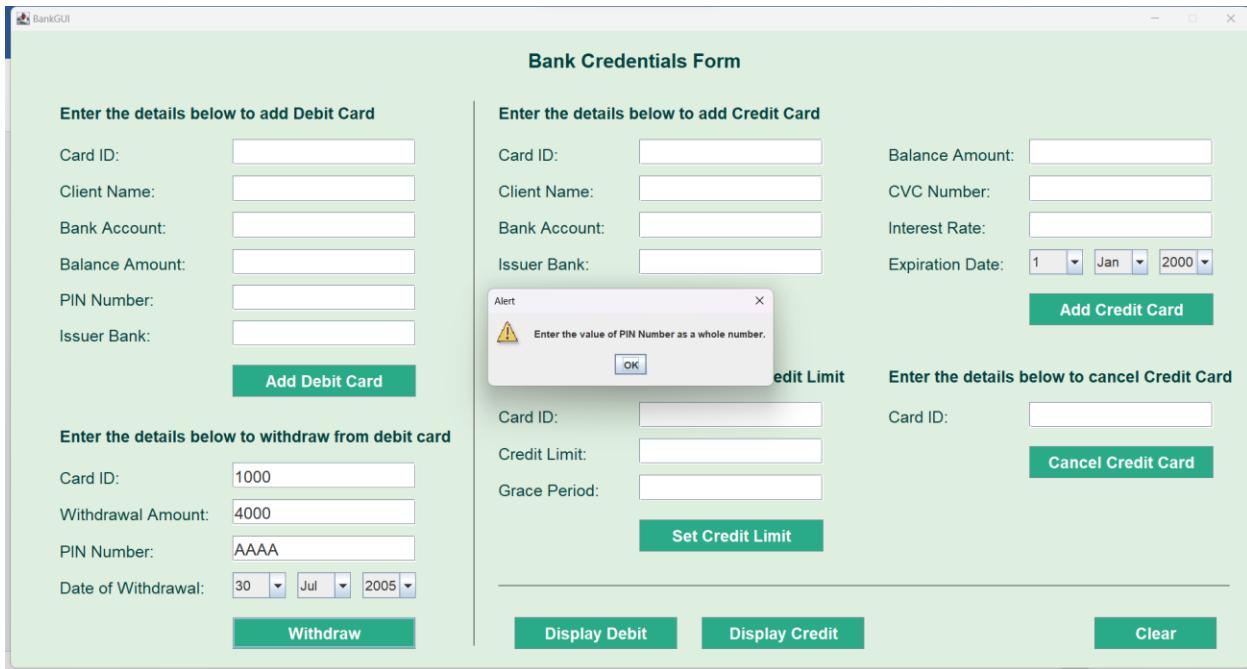


Figure 30. Screenshot of entering invalid PIN Number (Withdraw)

5.3.5. To enter invalid input while setting from credit limit

- a. Invalid Card ID for setting the credit limit

<b>Test No.</b>	<b>3.5 (a)</b>
Objective	To enter invalid Card ID as input for setting the credit limit
Action	<p>→ Invalid value for Card ID is entered for setting the credit limit</p> <ul style="list-style-type: none"> <li>• Card ID: “AAAA”</li> <li>• Credit Limit: “40000”</li> <li>• Grace Period: “28”</li> </ul> <p>→ Click on Set Credit Limit Button</p>
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:

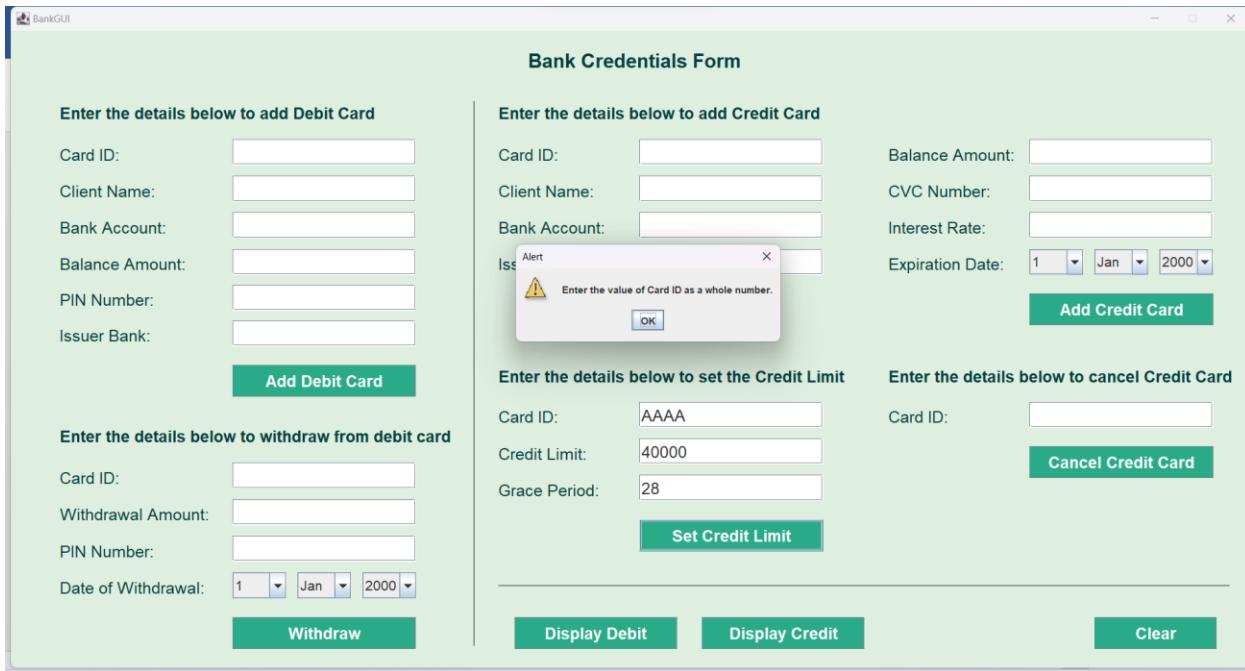


Figure 31. Screenshot of entering invalid Card ID (Set Credit Limit)

a. Invalid Credit Limit for setting the credit limit

<b>Test No.</b>	<b>3.5 (b)</b>
Objective	To enter invalid Credit Limit as input for setting the credit limit
Action	<p>→ Invalid value for Credit Limit is entered for setting the credit limit</p> <ul style="list-style-type: none"> <li>• Card ID: “2000”</li> <li>• Credit Limit: “AAAA”</li> <li>• Grace Period: “28”</li> </ul> <p>→ Click on Set Credit Limit Button</p>
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:

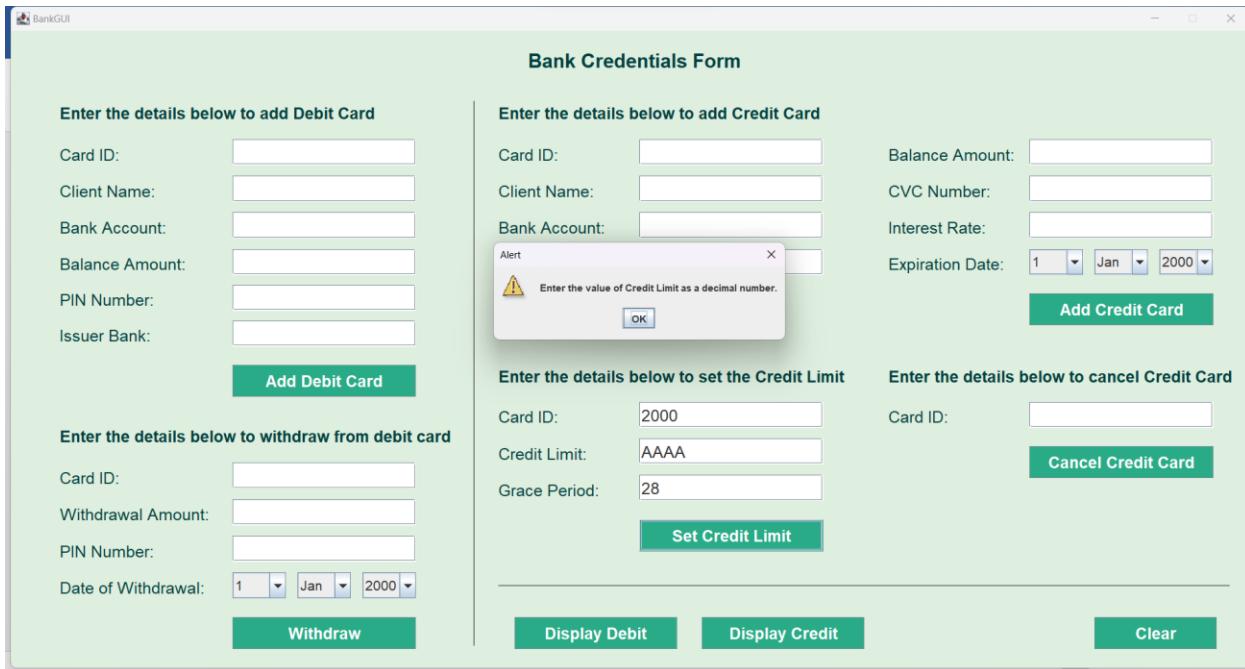


Figure 32. Screenshot of entering invalid Credit Limit (Set Credit Limit)

b. Invalid Grace Period for setting the credit limit

<b>Test No.</b>	<b>3.5 (c)</b>
Objective	To enter invalid Grace Period as input for setting the credit limit
Action	<p>→ Invalid value for Grace Period is entered for setting the credit limit</p> <ul style="list-style-type: none"> <li>• Card ID: “2000”</li> <li>• Credit Limit: “40000”</li> <li>• Grace Period: “AAAA”</li> </ul> <p>→ Click on Set Credit Limit Button</p>
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:

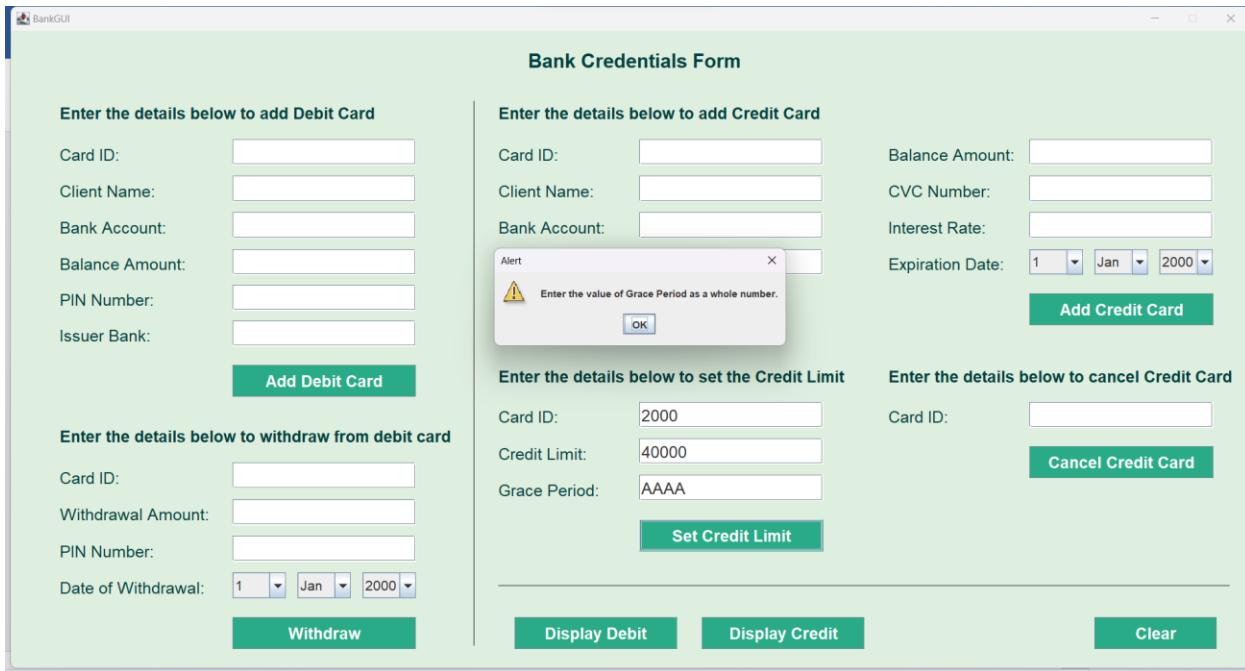


Figure 33. Screenshot of entering invalid Grace Period (Set Credit Limit)

5.3.6. To enter invalid input while cancelling the credit limit

- a. Invalid Card ID for cancelling the credit limit

<b>Test No.</b>	<b>3.6 (a)</b>
Objective	To enter invalid Card ID as input for cancelling the credit limit
Action	<p>→ Invalid value for Card ID is entered for cancelling the credit limit</p> <ul style="list-style-type: none"> <li>• Card ID: "AAAA"</li> </ul> <p>→ Click on Cancel Credit Limit Button</p>
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:

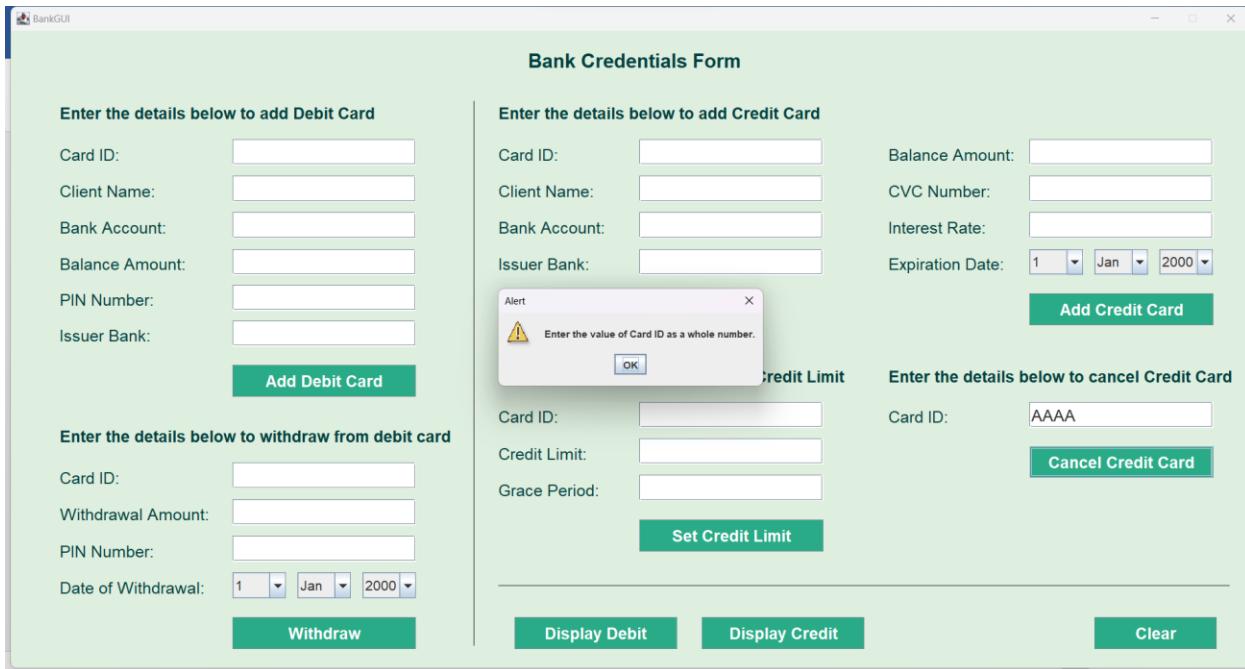


Figure 34. Screenshot of entering invalid Card ID (Cancel Credit Card)

5.3.7. To add already existing Card to ArrayList

- a. To add already existing debit card

<b>Test No.</b>	<b>3.7 (a)</b>
Objective	To add already existing card for debit card
Action	<p>→ The required text fields are filled for adding debit card</p> <p>Card ID: "7000"</p> <p>Client Name: "John Smith"</p> <p>Bank Account: "900091"</p> <p>Balance Amount: "100000.0"</p> <p>PIN Number: "1014"</p> <p>Issuer Bank: "Nepal Rastriya Bank"</p> <p>→ Click on Add Debit Card Button</p>
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:

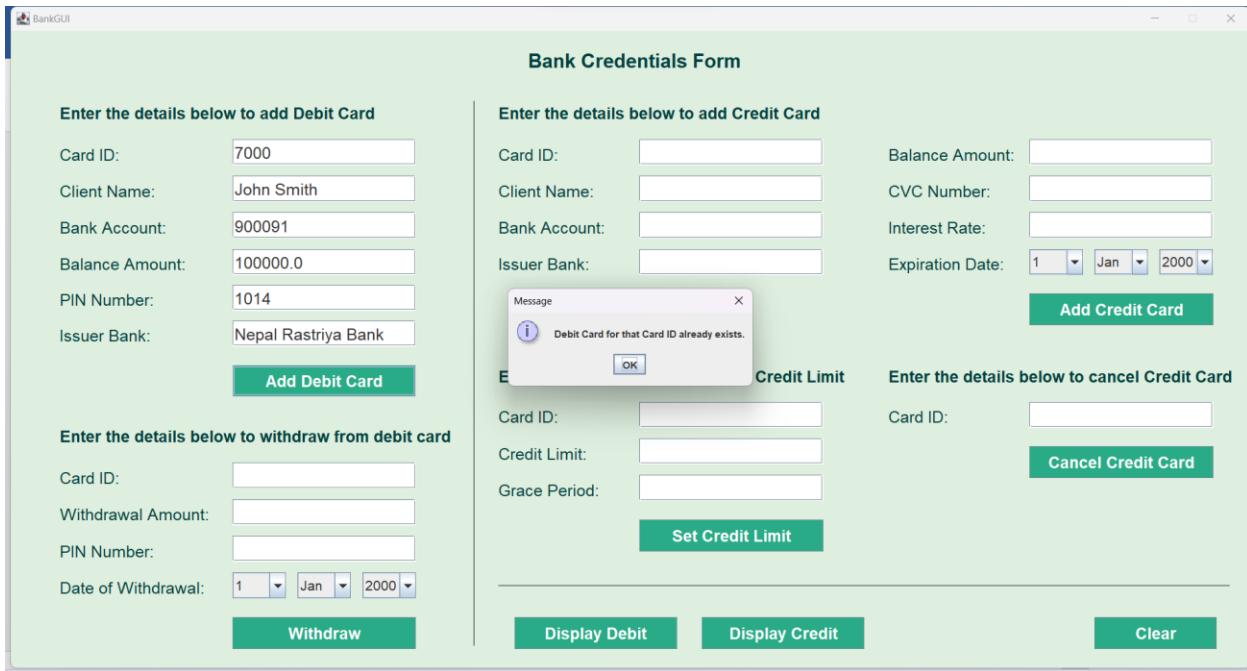


Figure 35. Screenshot of adding already existing Debit Card

- a. To add already existing credit card

<b>Test No.</b>	<b>3.7 (b)</b>
Objective	To add a card for credit card
Action	<p>→ The required text fields are filled for adding credit card</p> <p>Card ID: "8000"</p> <p>Client Name: "Harry Brook"</p> <p>Bank Account: "900092"</p> <p>Issuer Bank: "Nepal Rastriya Bank"</p> <p>Balance Amount: "50000.0"</p> <p>CVC Number: "563"</p> <p>Interest Rate: "7.5"</p> <p>Expiration Date: 29-Apr-2023</p> <p>→ Click on Add Credit Card Button</p>
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:

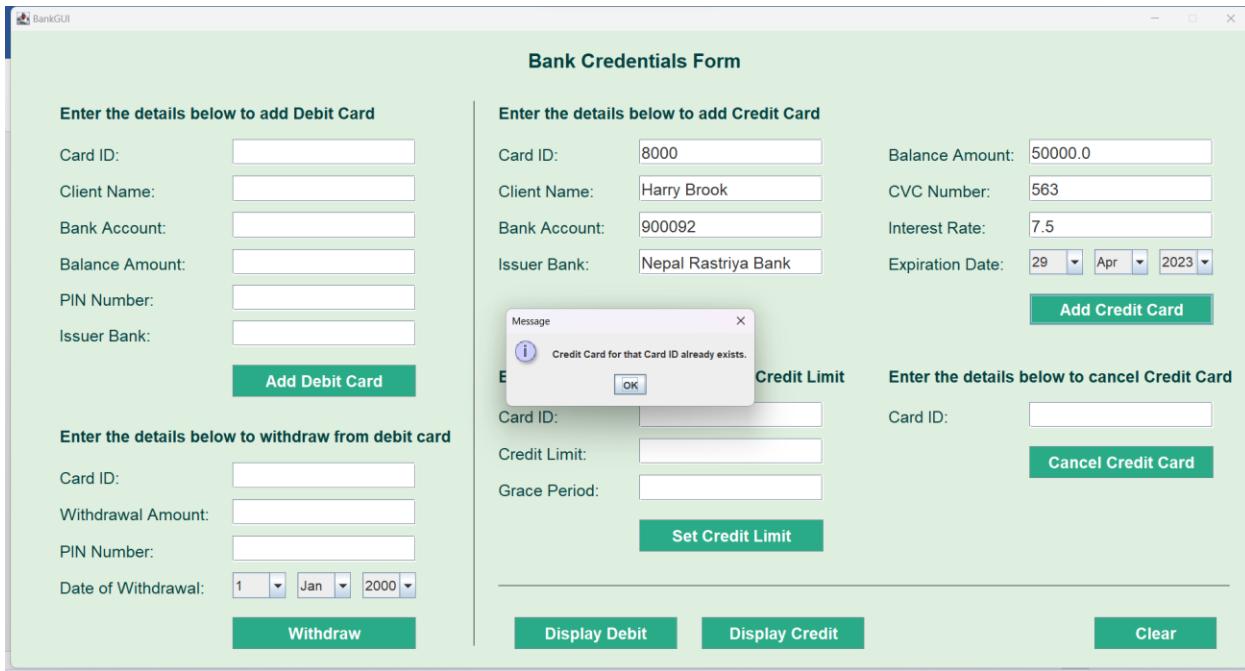


Figure 36. Screenshot of adding already existing Credit Card

### 5.3.8. Test for withdraw button

- a. To withdraw amount from a non-existing card

<b>Test No.</b>	<b>3.8 (a)</b>
Objective	To withdraw amount from a non-existing card
Action	<p>→ The required text fields are filled for withdrawing from debit card</p> <p>Card ID: "1200"</p> <p>Withdrawal Amount: "10000"</p> <p>PIN Number: "9098"</p> <p>Date of Withdrawal: 28-Dec-2022</p> <p>→ Click on Withdraw Button</p>
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:

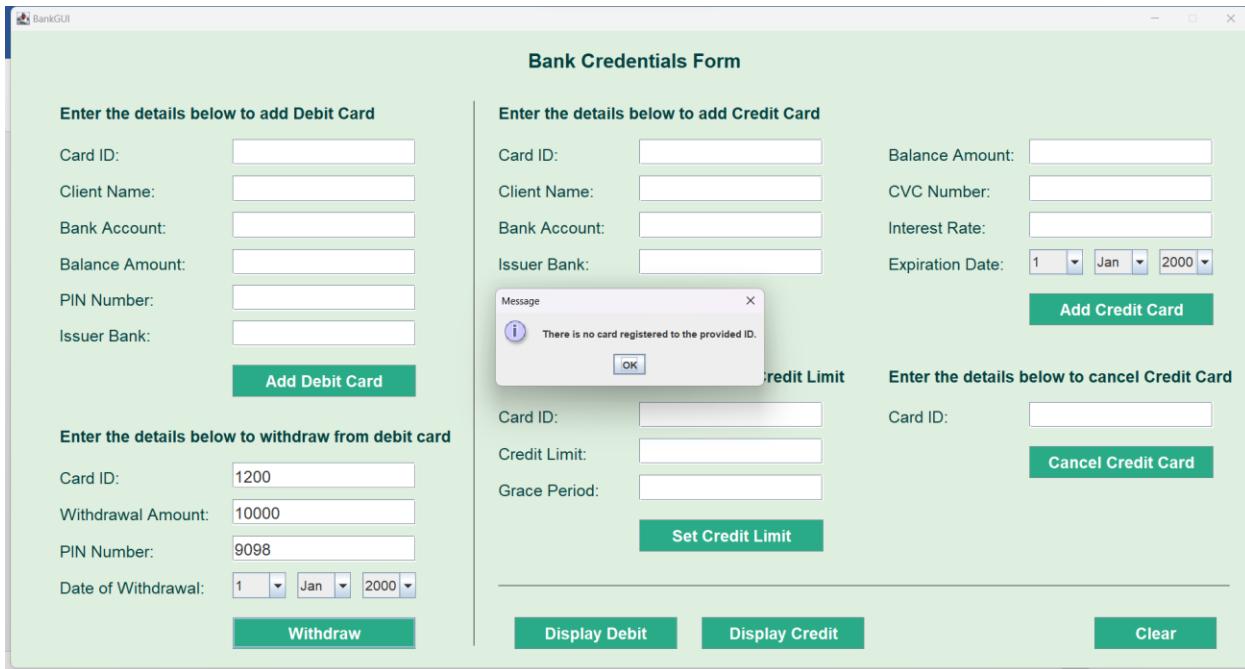


Figure 37. Screenshot of withdrawing amount from a non-existing card

b. To withdraw amount more than the balance amount

<b>Test No.</b>	<b>3.8 (b)</b>
Objective	To withdraw amount more than balance amount
Action	<p>→ The required text fields are filled for withdrawing from debit card</p> <p>Card ID: "7000"</p> <p>Withdrawal Amount: "110000"</p> <p>PIN Number: "1014"</p> <p>Date of Withdrawal: 28-Dec-2022</p> <p>→ Click on Withdraw Button</p>
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:

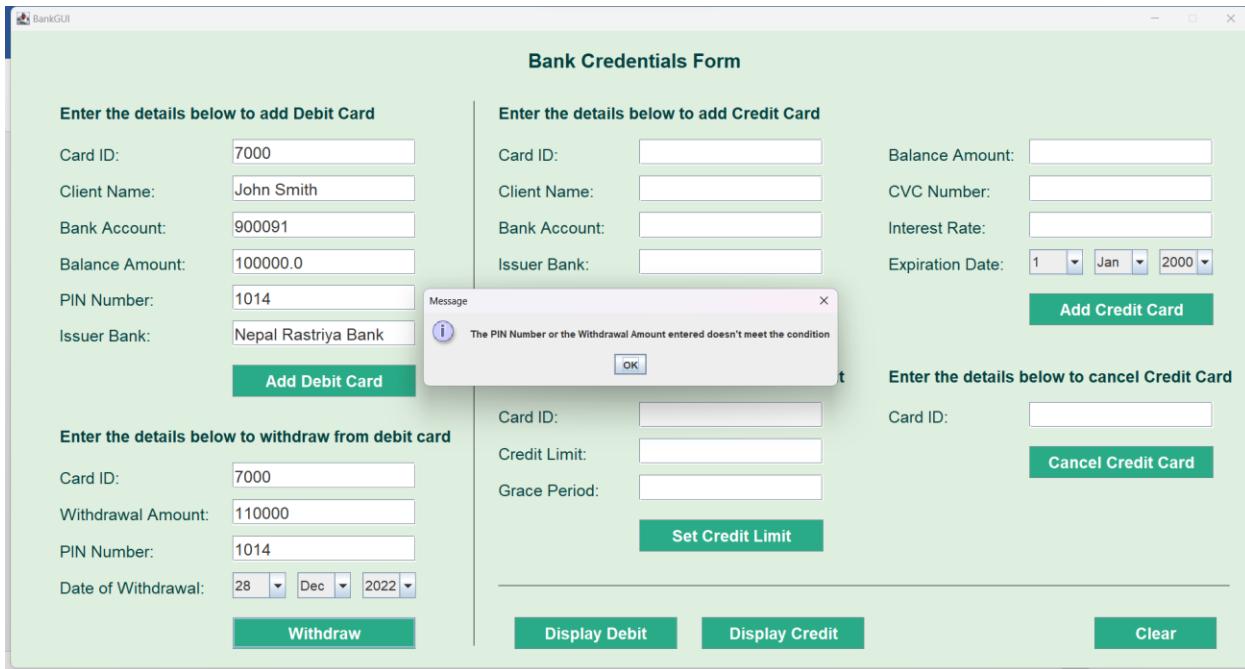


Figure 38. Screenshot of Withdrawal Amount more than Balance Amount

- c. To withdraw amount entering the wrong PIN Number

<b>Test No.</b>	<b>3.8 (c)</b>
Objective	To withdraw amount entering the wrong PIN Number
Action	<p>→ The required text fields are filled for withdrawing from debit card</p> <p>Card ID: "7000"</p> <p>Withdrawal Amount: "5000"</p> <p>PIN Number: "2000"</p> <p>Date of Withdrawal: 28-Dec-2022</p> <p>→ Click on Withdraw Button</p>
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:

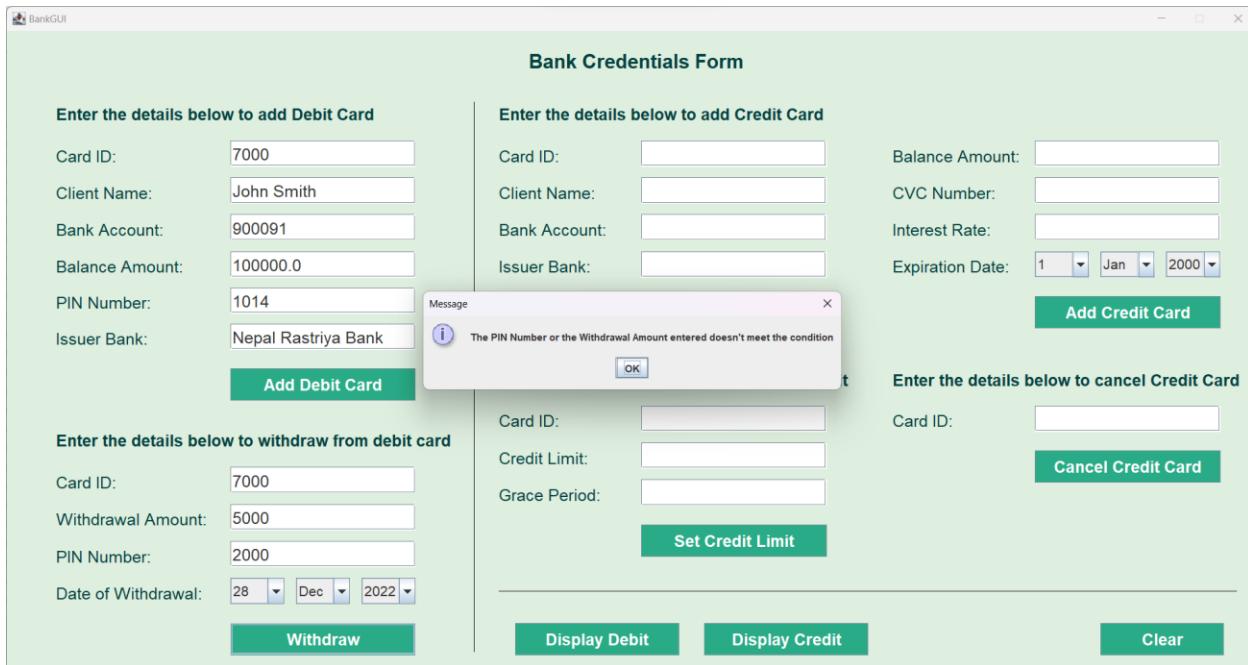


Figure 39. Screenshot of entering wrong PIN Number to withdraw amount

### 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

<b>Test No.</b>	<b>3.9 (a)</b>
Objective	To set the credit limit to a non-existing card
Action	<p>→ The required text fields are filled for setting the credit limit</p> <p>Card ID: “1200”</p> <p>Credit Limit: “100000”</p> <p>Grace Period: “28”</p> <p>→ Click on Set Credit Limit Button</p>
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:

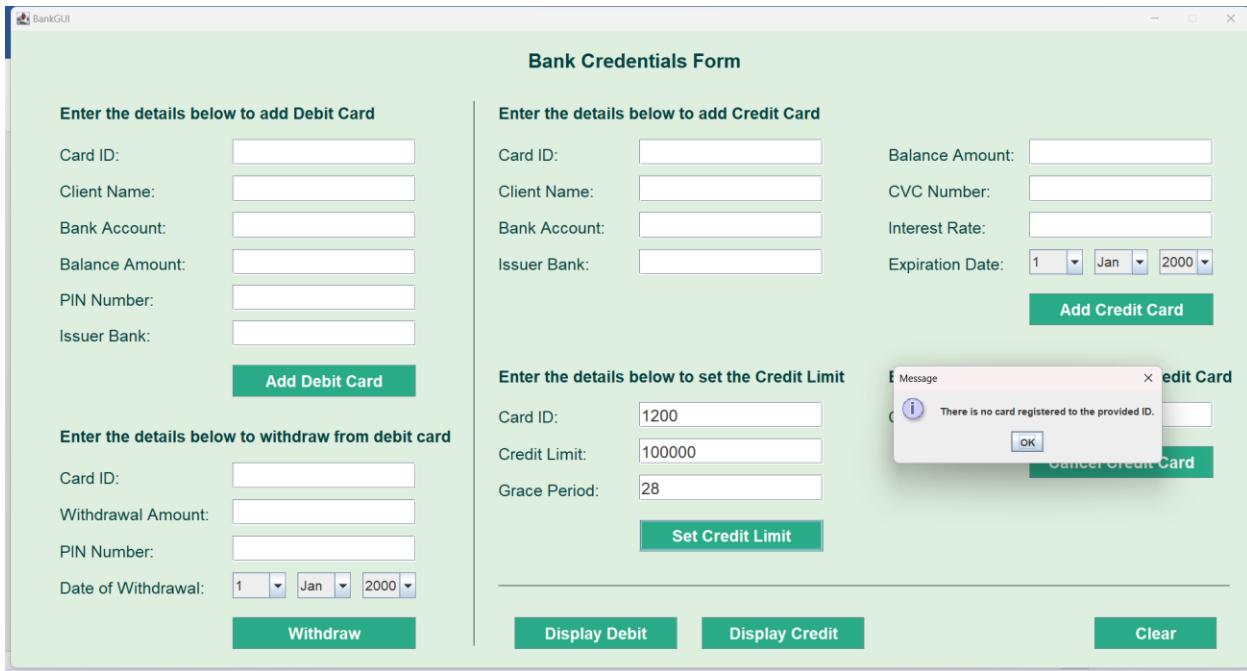


Figure 40. Setting credit limit to a non-existing card

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

<b>Test No.</b>	<b>3.9 (b)</b>
Objective	To set the credit limit more than the required condition
Action	<p>→ The required text fields are filled for setting the credit limit</p> <p>Card ID: "8000"</p> <p>Credit Limit: "300000"</p> <p>Grace Period: "28"</p> <p>→ Click on Set Credit Limit Button</p>
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:

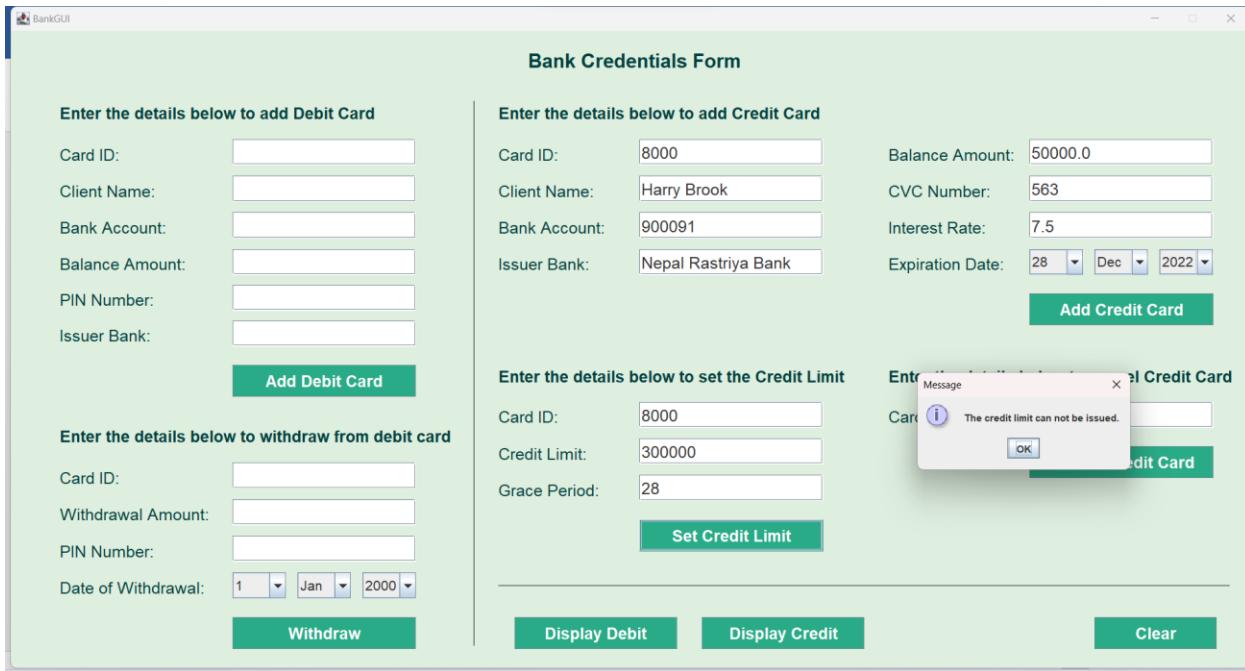


Figure 41. Screenshot of setting credit limit more than the required condition

c. To cancel a non-existing credit card

<b>Test No.</b>	<b>3.9 (c)</b>
Objective	To cancel a non-existing credit card
Action	<p>→ The required text fields are filled for setting the credit limit</p> <p>Card ID: “1200”</p> <p>→ Click on Cancel Credit Card Button</p>
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:

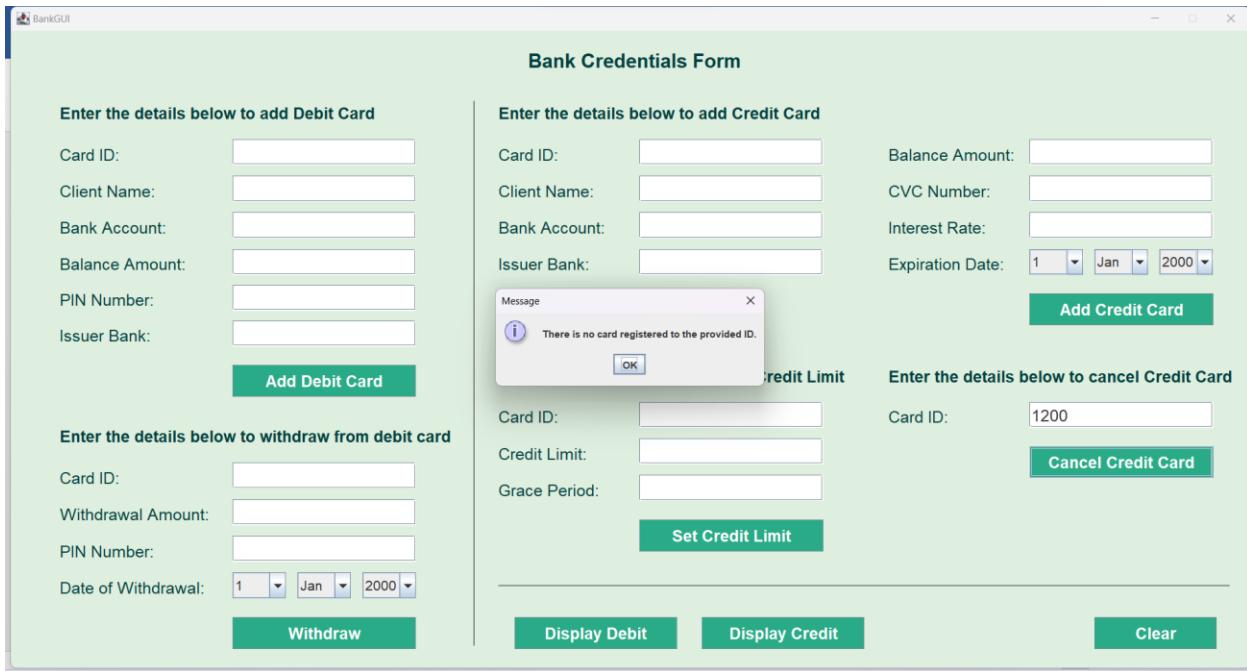


Figure 42. Screenshot of cancelling a non-existing credit card

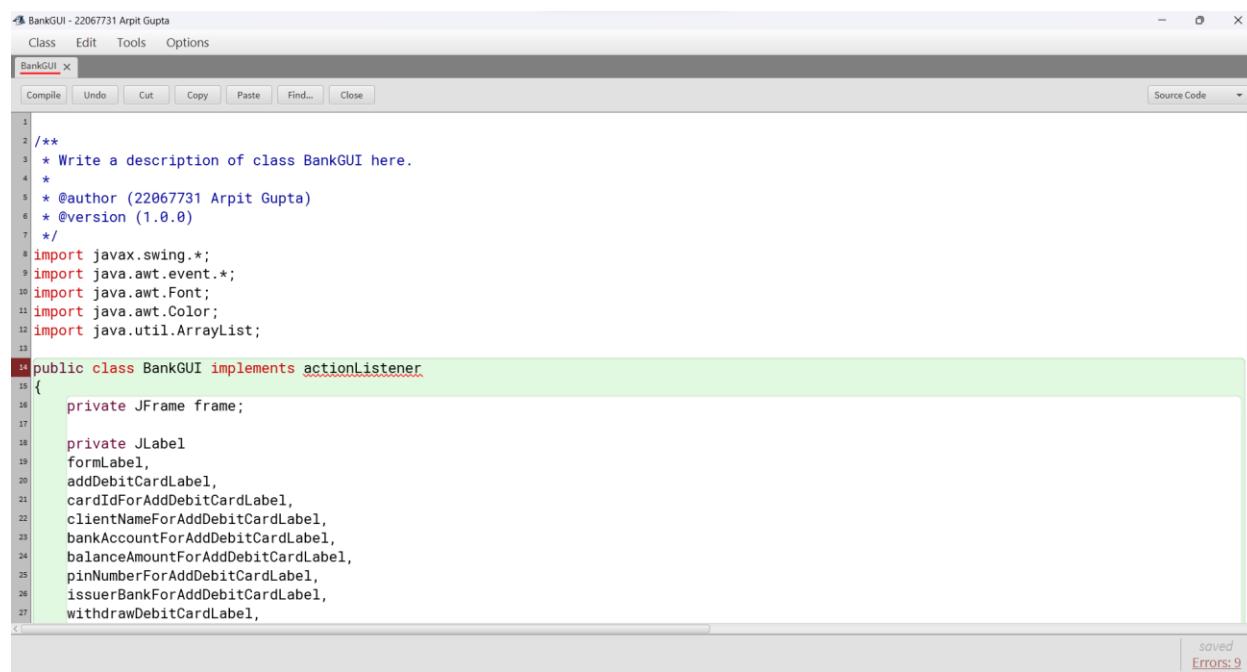
## 6. Error Detection

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:



The screenshot shows a Java code editor window titled "BankGUI - 22067731 Arpit Gupta". The menu bar includes "Class", "Edit", "Tools", and "Options". Below the menu is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". On the right side of the editor, there is a "Source Code" dropdown menu. The code area contains the following Java code:

```

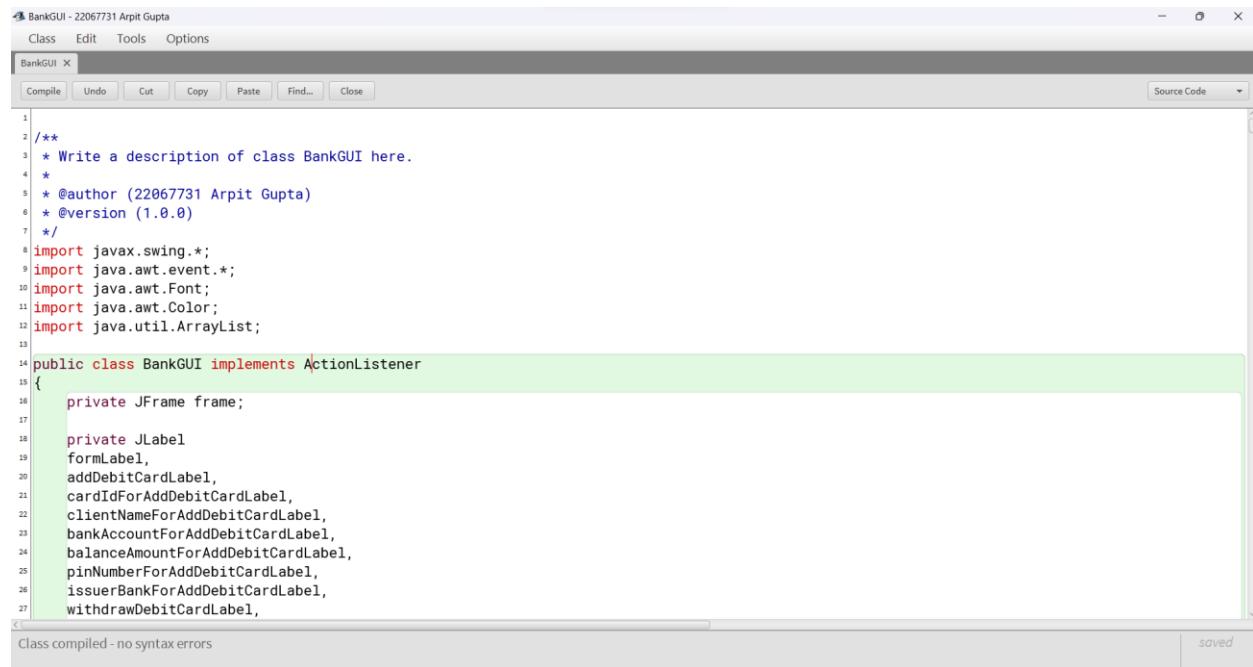
1 /**
2  * Write a description of class BankGUI here.
3  *
4  * @author (22067731 Arpit Gupta)
5  * @version (1.0.0)
6  */
7
8 import javax.swing.*;
9 import java.awt.event.*;
10 import java.awt.Font;
11 import java.awt.Color;
12 import java.util.ArrayList;
13
14 public class BankGUI implements ActionListener
15 {
16     private JFrame frame;
17
18     private JLabel
19         formLabel,
20         addDebitCardLabel,
21         cardIdForAddDebitCardLabel,
22         clientNameForAddDebitCardLabel,
23         bankAccountForAddDebitCardLabel,
24         balanceAmountForAddDebitCardLabel,
25         pinNumberForAddDebitCardLabel,
26         issuerBankForAddDebitCardLabel,
27         withdrawDebitCardLabel,

```

The word "ActionListener" in line 14 is underlined with a red squiggly line, indicating a syntax error. In the bottom right corner of the editor, there is a status bar with the text "saved" and "Errors: 9".

Figure 43. Error 1: Syntax Error

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



```
1 /**
2  * Write a description of class BankGUI here.
3  *
4  * @author (22067731 Arpit Gupta)
5  * @version (1.0.0)
6 */
7 import javax.swing.*;
8 import java.awt.event.*;
9 import java.awt.Font;
10 import java.awt.Color;
11 import java.util.ArrayList;
12
13 public class BankGUI implements ActionListener
14 {
15     private JFrame frame;
16
17     private JLabel
18         formLabel,
19         addDebitCardLabel,
20         cardIdForAddDebitCardLabel,
21         clientNameForAddDebitCardLabel,
22         bankAccountForAddDebitCardLabel,
23         balanceAmountForAddDebitCardLabel,
24         pinNumberForAddDebitCardLabel,
25         issuerBankForAddDebitCardLabel,
26         withdrawDebitCardLabel,
```

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:

BankGUI - 22067731 Arpit Gupta

Class Edit Tools Options

BankGUI.x

Compile Undo Cut Copy Paste Find... Close

Source Code

```
S10    double balanceAmountValue = Double.parseDouble(balanceAmountForAddDebitCard);
S11}
S12catch(NumberFormatException ex){
S13    JOptionPane.showMessageDialog(frame, "Enter the value of Balance Amount as a decimal number.", "Alert", JOptionPane.WARNING_MESSAGE);
S14    hasNoException = false;
S15}
S16}
S17if(hasNoException == true){
S18    try{
S19        int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
S20    }
S21    catch(NumberFormatException ex){
S22        JOptionPane.showMessageDialog(frame, "Enter the value of PIN Number as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
S23        hasNoException = false;
S24    }
S25}
S26if(hasNoException == true){
S27    int cardIdValue = Integer.parseInt(cardIdForAddDebitCard);
S28    double balanceAmountValue = Double.parseDouble(balanceAmountForAddDebitCard);
S29    int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
S30    if(cards.isEmpty()){
S31        debitObject = new DebitCard(balanceAmountValue, cardIdValue, bankAccountForAddDebitCard, issuerBankForAddDebitCard,
S32            clientNameForAddDebitCard);
S33        cards.add(debitObject);
S34        JOptionPane.showMessageDialog(frame, "Debit Card Added");
S35    }
S36    else{
S37        for(BankCard card: cards){
S38            if(card instanceof DebitCard){
```

*Figure 45. Error 2: Semantic Error*

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

```

BankGUI - 22067731 Arpit Gupta
Class Edit Tools Options
BankGUI X
Compile Undo Cut Copy Paste Find... Close Source Code
510     double balanceAmountValue = Double.parseDouble(balanceAmountForAddDebitCard);
511 }
512 catch(NumberFormatException ex){
513     JOptionPane.showMessageDialog(frame, "Enter the value of Balance Amount as a decimal number.", "Alert", JOptionPane.WARNING_MESSAGE);
514     hasNoException = false;
515 }
516 }
517 if(hasNoException == true){
518     try{
519         int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
520     }
521     catch(NumberFormatException ex){
522         JOptionPane.showMessageDialog(frame, "Enter the value of PIN Number as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
523         hasNoException = false;
524     }
525 }
526 if(hasNoException == true){
527     int cardIdValue = Integer.parseInt(cardIdForAddDebitCard);
528     double balanceAmountValue = Double.parseDouble(balanceAmountForAddDebitCard);
529     int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
530     if(cards.isEmpty()){
531         debitObject = new DebitCard(balanceAmountValue, cardIdValue, bankAccountForAddDebitCard, issuerBankForAddDebitCard,
532             clientNameForAddDebitCard, pinNumberValue);
533         cards.add(debitObject);
534         JOptionPane.showMessageDialog(frame, "Debit Card has been added.");
535     }
536     else{
537         for(BankCard card: cards){
538             if(card instanceof DebitCard){
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
823
824
825
825
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1710
1711
1711
1712
1712
1713
1713
1714
1714
1715
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1720
1721
1721
1722
1722
1723
1723
1724
1724
1725
1725
1726
1726
1727
1727
1728
1728
1729
1729
1730
1730
1731
1731
1732
1732
1733
1733
1734
1734
1735
1735
1736
1736
1737
1737
1738
1738
1739
1739
1740
1740
1741
1741
1742
1742
1743
1743
1744
1744
1745
1745
1746
1746
1747
1747
1748
1748
1749
1749
1750
1750
1751
1751
1752
1752
1753
1753
1754
1754
1755
1755
1756
1756
1757
1757
1758
1758
1759
1759
1760
1760
1761
1761
1762
1762
1763
1763
1764
1764
1765
1765
1766
1766
1767
1767
1768
1768
1769
1769
1770
1770
1771
1771
1772
1772
1773
1773
1774
1774
1775
1775
1776
1776
1777
1777
1778
1778
1779
1779
1780
1780
1781
1781
1782
1782
1783
1783
1784
1784
1785
1785
1786
1786
1787
1787
1788
1788
1789
1789
1790
1790
1791
1791
1792
1792
1793
1793
1794
1794
1795
1795
1796
1796
1797
1797
1798
1798
1799
1799
1800
1800
1801
1801
1802
1802
1803
1803
1804
1804
1805
1805
1806
1806
1807
1807
1808
1808
1809
1809
1810
1810
1811
1811
1812
1812
1813
1813
1814
1814
1815
1815
1816
1816
181
```

### 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:

The screenshot shows a Java Swing application titled "BankGUI - 22067731 Arpit Gupta". The menu bar includes "File", "Class", "Edit", "Tools", and "Options". A toolbar below the menu has buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". On the right, there's a "Source Code" dropdown. The main area contains Java code for adding a debit card. The code uses JOptionPane to get user input for PIN number and checks if a card with the same ID already exists. It then creates a new DebitCard object and adds it to a list.

```
522     JOptionPane.showMessageDialog(frame, "Enter the value of PIN Number as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
523     hasNoException = false;
524 }
525 }
526 if(hasNoException == false){
527     int cardIdValue = Integer.parseInt(cardIdForAddDebitCard);
528     double balanceAmountValue = Double.parseDouble(balanceAmountForAddDebitCard);
529     int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
530     if(cards.isEmpty()){
531         debitObject = new DebitCard(balanceAmountValue, cardIdValue, bankAccountForAddDebitCard, issuerBankForAddDebitCard,
532             clientNameForAddDebitCard, pinNumberValue);
533         cards.add(debitObject);
534         JOptionPane.showMessageDialog(frame, "Debit Card has been added.");
535     }
536     else{
537         for(BankCard card: cards){
538             if(card instanceof DebitCard){
539                 debitObject = (DebitCard)card;
540                 if(debitObject.getCardID() == cardIdValue){
541                     JOptionPane.showMessageDialog(frame, "Debit Card for that Card ID already exists.");
542                     counter++;
543                     break;
544                 }
545             }
546         }
547         if(counter == 0){
548             debitObject = new DebitCard(balanceAmountValue, cardIdValue, bankAccountForAddDebitCard,
```

*Figure 47. Error 3: Logical error*

The debit card was not added.

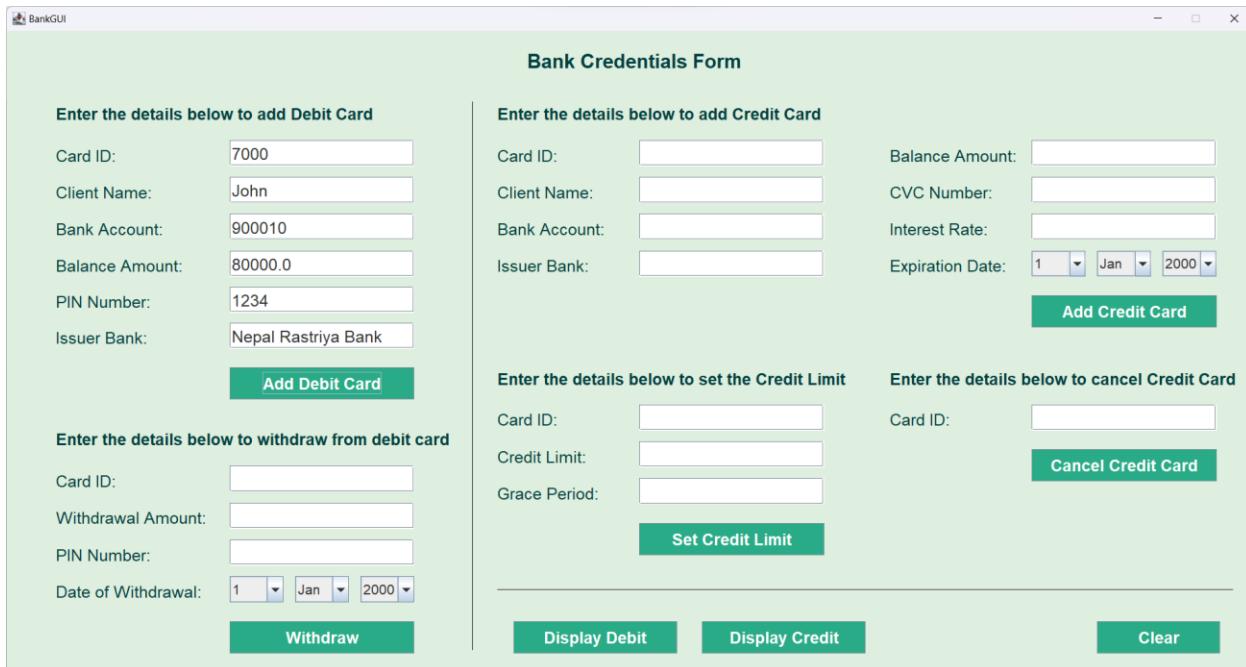


Figure 48. Output due to logical error

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

```

522     JOptionPane.showMessageDialog(frame, "Enter the value of PIN Number as a whole number.", "Alert", JOptionPane.WARNING_MESSAGE);
523     hasNoException = false;
524   }
525 }
526 if(hasNoException == true){
527   int cardIdValue = Integer.parseInt(cardIdForAddDebitCard);
528   double balanceAmountValue = Double.parseDouble(balanceAmountForAddDebitCard);
529   int pinNumberValue = Integer.parseInt(pinNumberForAddDebitCard);
530   if(cards.isEmpty()){
531     debitObject = new DebitCard(balanceAmountValue, cardIdValue, bankAccountForAddDebitCard, issuerBankForAddDebitCard,
532     clientNameForAddDebitCard, pinNumberValue);
533     cards.add(debitObject);
534     JOptionPane.showMessageDialog(frame, "Debit Card has been added.");
535   } else{
536     for(BankCard card: cards){
537       if(card instanceof DebitCard){
538         debitObject = (DebitCard)card;
539         if(debitObject.getCardID() == cardIdValue){
540           JOptionPane.showMessageDialog(frame, "Debit Card for that Card ID already exists.");
541           counter++;
542           break;
543         }
544       }
545     }
546     if(counter == 0){
547       debitObject = new DebitCard(balanceAmountValue, cardIdValue, bankAccountForAddDebitCard,
548       clientNameForAddDebitCard, pinNumberValue);
549       cards.add(debitObject);
550       JOptionPane.showMessageDialog(frame, "Debit Card has been added.");
551     }
552   }
553 }
554 
```

Class compiled - no syntax errors

Figure 49. Error 3 solved

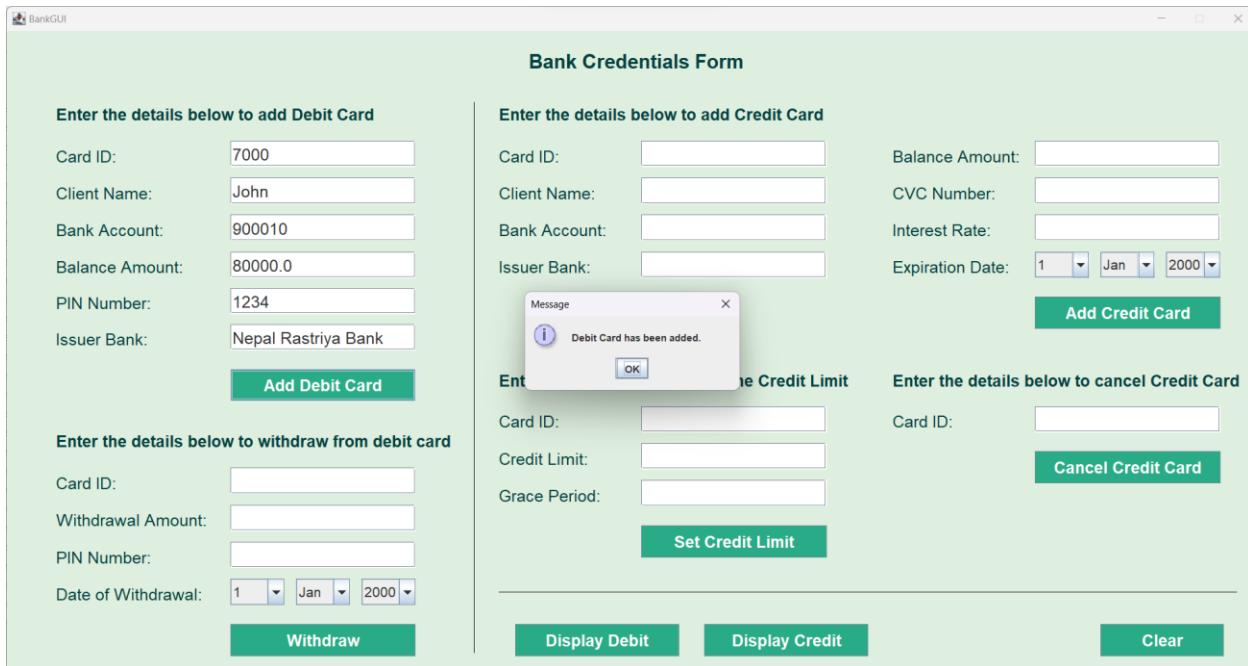


Figure 50. Output after solving logical error

## 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](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 required 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 required 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 required 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() == closeButton){  
  
    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();
```

```
    }  
}
```