LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

30% Individual Coursework

2022-23 Spring

Student Name: Aadesh Shrestha

London Met ID: 22067566

College ID: NP01CP4A220063

Group: C3

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

List of Figure

List of Tables

1. Introduction

1.1 About Project

While developing a program after the functionality, the graphical interface is the most important factor of the program. Graphical User Interface(GUI) is used to make the program easily usable by the general public rather than just those with higher computer knowledge. It also makes the program visually appealing to the user. Even if the functionality of a program is well developed but the GUI is hideous, the program would be very difficult for the user to operate. This coursework aims to create a user-friendly GUI for the application developed in the previous coursework.

Previously, an application Bank Card that adds debit cards and credit cards was developed. The program had a few more features like withdrawing from debit cards and setting the credit limit of credit cards. All the functions were taken from the Bank Card to create a form GUI called BankGUI to simplify the data gathering and extraction process. The content of the form and their functions also change according to the category of the card i.e. Debit or Credit Card. Furthermore, exception handling was also implemented to somewhat reduce human errors. Alerts and warnings were displayed when invalid input was found with a description of the error and instructions to fix them. A single Array List was used to store the both types of cards and casting was performed to retrieve the required card when a function was called. Moreover, adding a card is very simple and so is using other functions. Implementing this application at a bank would increase efficiency and appending additional information would be easier.

1.2 Tools Used

Basic programming knowledge of the GUI was required to develop this application. An integrated development environment (IDE) called BlueJ was used to write the codes for this project. BlueJ was preferred to other IDEs because it is a development environment which is interactive, portable, innovative, and simple to use. (BlueJ org, n.d.)Draw.io was used to create the class diagrams as it is easy to use and more convenient than its alternatives. Finally, MS. Word was used for the documentation of this project

2. Class diagram

2.1 BankCard

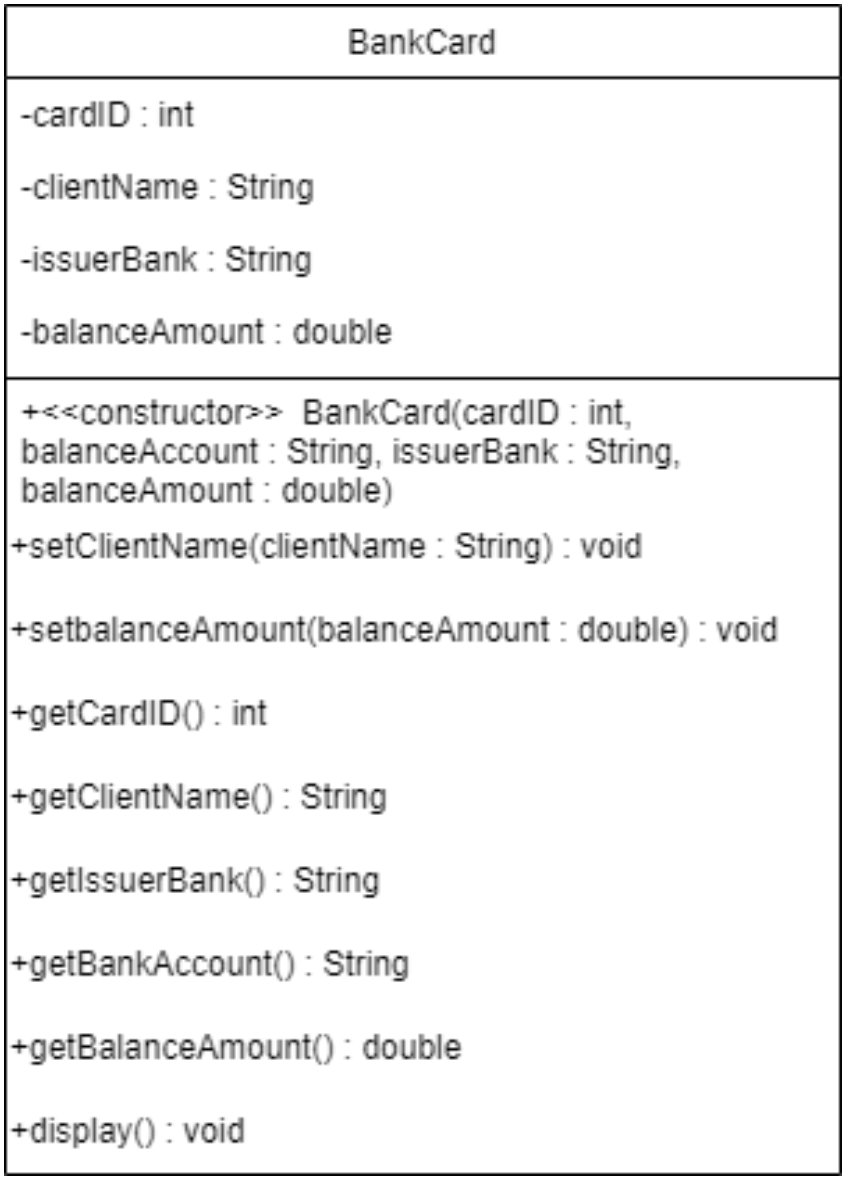

Figure 1: Class diagram of BankCard

2.2 DebitCard

Figure 2: Class diagram of DebitCard

2.3 CreditCard

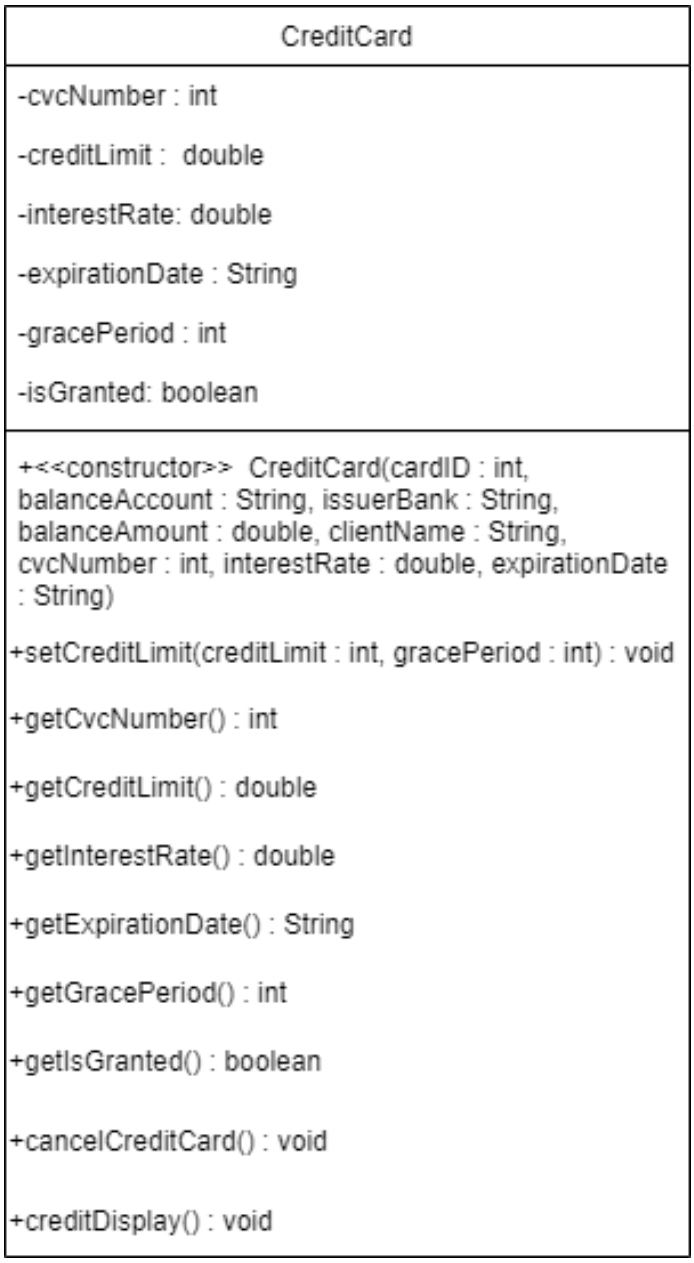

Figure 3: Class diagram of CreditCard

2.4 Inheritance

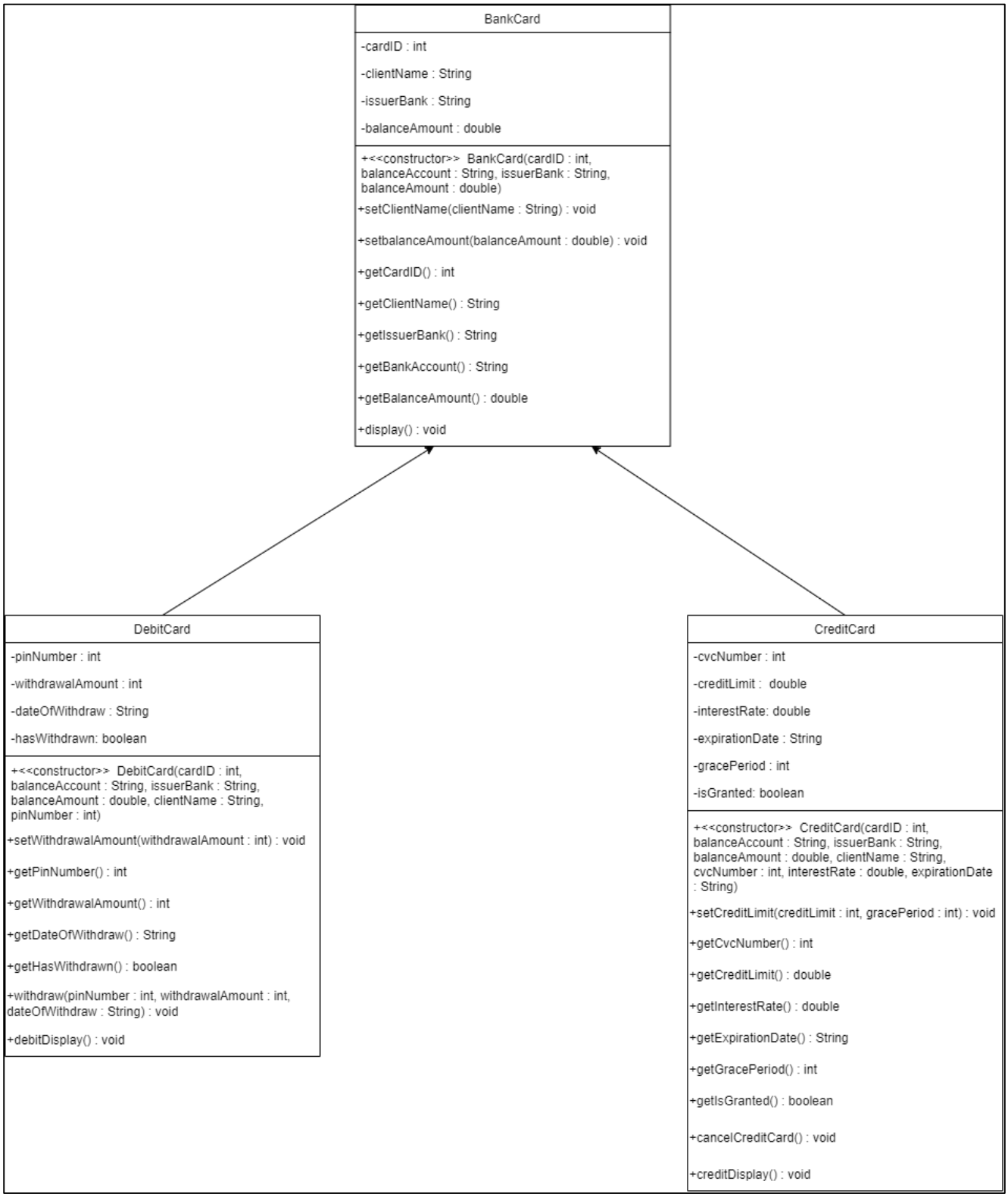

Figure 4: Class diagram of Inheritance

2.5 BankGUI

BankGUI

-frame : JFrame

-panelDC, panelCC : JPanel

-mainTitleDC, debitCardTitle, addDebitCardTitle, withdraw, mainTitleCC, creditCardTitle, addCreditCardTitle, setCreditLimit, creditCardCancel, debitCardIDL, debitClientNameL, debitIssuerBankL, debitBankAccountL, debitBalanceAmountL, debitPinNumberL, cardIDWithdrawL, pinNumberWithdrawL, withdrawalAmountL, dateOfWithdrawL, creditCardIDL, creditClientNameL, creditIssuerBankL, creditBankAccountL, creditBalanceAmountL, creditCvcNumberL, creditInterestRateL, creditExpirationDateL, cardIDcreditLimitL, creditLimitL, gracePeriodL, cardIDCreditCancelL : JLabel

-debitCardIDTF, debitClientNameTF, debitIssuerBankTF, debitBankAccountTF, debitBalanceAmountTF, debitPinNumberTF, cardIDWithdrawTF, pinNumberWithdrawTF, withdrawalAmountTF, creditCardIDTF, creditClientNameTF, creditIssuerBankTF, creditBankAccountTF, creditBalanceAmountTF, creditCvcNumberTF, creditInterestRateTF, cardIDcreditLimitTF, creditLimitTF, gracePeriodTF, cardIDCreditCancelTF : JTextField

-dayDateOfWithdrawCB, monthDateOfWithdrawCB, yearDateOfWithdrawCB, dayExpirationDateCB, monthExpirationDateCB, yearExpirationDateCB : JComboBox

-switchToDC, switchToCC, addDebitCardBtn, withdrawDCBtn, clearDCBtn, displayDCBtn, addCreditCardBtn, setCreditLimitCCBtn, creditCardCancelCCBtn, clearCCBtn, displayCCBtn : JButton

-cards : ArrayList<BankCard>

-mon[] : String

+<<constructor>> BankGUI()

+actionPerformed(ActionEvent e): void

Figure 5: Class diagram of BankGUI

3. Pseudocode

IMPORT javax.swing.*;

IMPORT java.awt.*;

IMPORT java.awt.event.*;

IMPORT java.util.ArrayList;

CREATE class BankGUI THAT implements ActionListener

DECLARE JFrame frame

DECLARE JPanel panelDC

DECLARE JLabel mainTitleDC, debitCardTitle, addDebitCardTitle, withdraw

DECLARE JLabel debitCardIDL, debitClientNameL, debitIssuerBankL, debitBankAccountL, debitBalanceAmountL, debitPinNumberL

DECLARE JTextField debitCardIDTF, debitClientNameTF, debitIssuerBankTF, debitBankAccountTF, debitBalanceAmountTF, debitPinNumberTF

DECLARE JLabel cardIDWithdrawL, pinNumberWithdrawL, withdrawalAmountL, dateOfWithdrawL

DECLARE JTextField cardIDWithdrawTF, pinNumberWithdrawTF, withdrawalAmountTF

DECLARE JComboBox dayDateOfWithdrawCB, monthDateOfWithdrawCB, yearDateOfWithdrawCB

DECLARE JButton switchToDC, switchToCC, addDebitCardBtn, withdrawDCBtn, clearDCBtn, displayDCBtn

DECLARE JPanel panelCC

DECLARE JLabel mainTitleCC, creditCardTitle, addCreditCardTitle, setCreditLimit, creditCardCancel

DECLARE JLabel creditCardIDL, creditClientNameL, creditIssuerBankL, creditBankAccountL, creditBalanceAmountL, creditCvcNumberL, creditInterestRateL, creditExpirationDateL

DECLARE JTextField creditCardIDTF, creditClientNameTF, creditIssuerBankTF, creditBankAccountTF, creditBalanceAmountTF, creditCvcNumberTF, creditInterestRateTF

DECLARE JComboBox dayExpirationDateCB, monthExpirationDateCB, yearExpirationDateCB

DECLARE JLabel cardIDcreditLimitL, creditLimitL, gracePeriodL

DECLARE JTextField cardIDcreditLimitTF, creditLimitTF, gracePeriodTF

DECLARE JButton addCreditCardBtn, setCreditLimitCCBtn, creditCardCancelCCBtn, clearCCBtn, displayCCBtn

DECLARE ArrayList cards OF type BankCard

String [] mon = {"Jan", "Feb", "Mar", "April", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov", "Dec"};

CREATE constructor BankGUI()

DO

CREATE frame as JFrame

SETSIZE of frame

CREATE panelDC as JPanel

SETSIZE of panelDC

CREATE all the components (JLabel, JTextField, JCombox, JButtons) of panelDC

SET the respective bounds of all components on panelDC

ADD all the components on panelDC

CREATE panelCC as JPanel

SETSIZE of panelCC

CREATE all the components (JLabel, JTextField, JCombox, JButtons) of panelCC

SET the respective bounds of all components on panelCC

ADD all the components on panelCC

SET the layout of panelDC to null

SET the layout of panelCC to null

ADD panelDC to frame

```
            SET the layout of frame to null
            SET the default close operation of the frame to exit on close (3)
            SET the visibility of the frame to true
        END DO

    CREATE method actionPerformed(ActionEvent e)
    DO
        IF (e.getSource() == switchToCC)
        DO
            REMOVE panelDC from frame
            DO validate frame
            DO repaint frame
            ADD panelCC to frame
            DO validate frame
            DO repaint frame
        END DO
        END IF

        IF (e.getSource() == switchToDC)
            DO
                REMOVE panelCC from frame
                DO validate frame
                DO repaint frame
```

ADD panelDC to frame

DO validate frame

DO repaint frame

END DO

END IF

IF (e.getSource() == addDebitCardBtn)

IF (debitCardIDTF.getText() is empty OR debitClientNameTF.getText() is empty OR debitIssuerBankTF.getText() is empty OR debitBankAccountTF.getText() is empty OR debitBalanceAmountTF.getText() is empty OR debitPinNumberTF.getText() is empty)

DISPLAY an error message dialog box with the message "All values must be entered."

END IF

ELSE

TRY

ASSIGN cardIDText TO the text from debitCardIDTF

ASSIGN clientName TO the text from debitClientNameTF

ASSIGN issuerBank TO the text from debitIssuerBankTF

ASSIGN bankAccount TO the text from debitBankAccountTF

ASSIGN balanceAmountText TO the text from debitBalanceAmountTF

ASSIGN pinNumberText TO the text from debitPinNumberTF

ASSIGN cardID TO the integer value of cardIDText

ASSIGN balanceAmount TO the integer value of balanceAmountText

ASSIGN pinNumber TO the integer value of pinNumberText

SET debitObj AS NEW DebitCard(cardID, issuerBank, bankAccount, balanceAmount, clientName, pinNumber)

IF (cards.size() is equal to 0)

ADD "debitObj" to the "cards" list.

DISPLAY an information message dialog box with the message "Debit Card has been Added."

END IF

ELSE

FOR each card in cards list DO

IF card is an instance of DebitCard

DOWNCAST card to DebitCard and ASSIGN to debitCard

IF (debitCard.getCardID() is equal to cardID)

DISPLAY an error message dialog box with the message "The Debit Card is already present."

RETURN

END IF

END IF

END FOR

ADD debitObj to cards list

DISPLAY an information message dialog box with the message " Debit Card Added."

END ELSE

END TRY

CATCH (NumberFormatException em)

DISPLAY an error message dialog box with the message "Please enter valid numbers in Card ID, Balance Amount and PIN number."

END CATCH

END ELSE

END IF

IF (e.getSource() == addCreditCardBtn)

IF (creditCardIDTF.getText() is empty OR creditClientNameTF.getText() is empty OR creditIssuerBankTF.getText() is empty OR creditBankAccountTF.getText() is empty OR creditBalanceAmountTF.getText() is empty OR creditCvcNumberTF.getText() is empty OR creditInterestRateTF.getText() is empty)

DISPLAY an error message dialog box with the message "All values must be entered."

END IF

ELSE

TRY

ASSIGN cardIDText TO the text from creditCardIDTF

ASSIGN clientName TO the text from creditClientNameTF

ASSIGN issuerBank TO the text from creditIssuerBankTF

ASSIGN bankAccount TO the text from creditBankAccountTF

ASSIGN balanceAmountText TO the text from creditBalanceAmountTF

ASSIGN cvcNumberText TO the text from creditCvcNumberTF

ASSIGN interestRateText TO the text from creditInterestRateTF

ASSIGN dayCB to the selected item from dayExpirationDateCB

ASSIGN dayCB to the selected item from monthExpirationDateCB

ASSIGN dayCB to the selected item from yearExpirationDateCB

ASSIGN expirationDate to monthCB + " " + dayCB + "," + yearCB

ASSIGN cardID TO the integer value of cardIDText

ASSIGN balanceAmount TO the integer value of balanceAmountText

ASSIGN cvcNumber TO the integer value of cvcNumberText

ASSIGN interestRate TO the double value of interestRateText

SET creditObj AS NEW CreditCard (cardID, issuerBank, bankAccount, balanceAmount, clientName, cvcNumber, interestRate, expirationDate)

IF (cards.size() is equal to 0)

ADD " creditObj" to the "cards" list.

DISPLAY an information message dialog box with the message "Credit Card has been Added."

END IF

ELSE

FOR each card in cards list DO

IF card is an instance of CreditCard

DOWNCAST card to CreditCard and ASSIGN to creditCard

IF (creditCard.getCardID() is equal to cardID)

DISPLAY an error message dialog box with the message "The Credit Card is already present."

RETURN

END IF

END IF

END FOR

ADD creditObj to cards list

DISPLAY an information message dialog box with the message "Credit Card Added."

END ELSE

END TRY

CATCH (NumberFormatException em)

DISPLAY an error message dialog box with the message " Please enter valid numbers in Card ID, Balance Amount, CVC Number and Interest Rate."

END CATCH

END ELSE

END IF

IF (e.getSource() == withdrawDCBtn)

IF (cardIDWithdrawTF.getText() is empty OR withdrawalAmountTF.getText() is empty OR pinNumberWithdrawTF.getText() is empty)

DISPLAY an error message dialog box with the message "All values must be entered."

END IF

ELSE

TRY

ASSIGN correctCard to false

ASSIGN cardIDWithdrawText to the text from cardIDWithdrawTF

ASSIGN withdrawalAmountText to the text from withdrawalAmountTF

ASSIGN pinNumberTextWithdraw to the text from pinNumberWithdrawTF

ASSIGN dayCB to the selected item from dayDateOfWithdrawCB

ASSIGN monthCB to the selected item from monthDateOfWithdrawCB

ASSIGN yearCB to the selected item from yearDateOfWithdrawCB

ASSIGN dateOfWithdraw to monthCB + " " + dayCB + "," + yearCB

ASSIGN cardIDWithdraw to the integer value of cardIDWithdrawText

ASSIGN pinNumberWithdraw to the integer value of pinNumberTextWithdraw

ASSIGN withdrawalAmount to the integer value of withdrawalAmountText

FOR each card in cards DO

IF card is an instance of DebitCard

DOWNCAST card to DebitCard and ASSIGN to debitCard

IF (debitCard.getCardID() equals cardIDWithdraw)

SET correctCard to true

CALL withdraw method of DebitCard with parameters pinNumberWithdraw, withdrawalAmount and dateOfWithdraw

IF pinNumberWithdraw equals debitCard.getPinNumber()

IF (withdrawalAmount < debitCard.getBalanceAmount() AND withdrawalAmount > 0)

DISPLAY an information message dialog box with the message " Your amount is withdrawn."

BREAK from loop

END IF

ELSE

DISPLAY a warning message dialog box with the message " Your Withdrawal Amount exceeds your Balance Amount."

END ELSE

END IF

ELSE

DISPLAY a warning message dialog box with the message "You have entered an incorrect PIN Number."

END ELSE

END IF

END IF

END FOR

IF correctCard is false

DISPLAY an error message dialog box with the message "The Card ID is invalid.”

END IF

END TRY

CATCH

DISPLAY an error message dialog box with the message " Please enter valid numbers in Card ID, PIN number and Withdrawal Amount.”

END CATCH

END ELSE

END IF

IF (e.getSource() == setCreditLimitCCBtn)

IF (cardIDcreditLimitTF.getText() is empty OR creditLimitTF.getText() is empty OR gracePeriodTF.getText() is empty)

DISPLAY an error message dialog box with the message "All values must be entered."

END IF

ELSE

TRY

ASSIGN correctCard to false

ASSIGN cardIDcreditLimitText to the text from cardIDcreditLimitTF

ASSIGN creditLimitText to the text from creditLimitTF

ASSIGN gracePeriodText to the text from gracePeriodTF

ASSIGN cardIDcreditLimit to the integer value of cardIDcreditLimitText

ASSIGN creditLimit to the integer value of creditLimitText

ASSIGN gracePeriod to the integer value of gracePeriodText

FOR each card in cards DO

IF card is an instance of CreditCard

DOWNCAST card to CreditCard and ASSIGN to creditCard

IF (creditCard.getCardID() equals cardIDcreditLimit)

SET correctCard to true

CALL setCreditLimit method of CreditCard with parameters creditLimit, gracePeriod

IF (creditLimit <= creditCard.getBalanceAmount() multiplied by 2.5 AND creditLimit >= creditCard.getBalanceAmount() multiplied by 2)

DISPLAY an information message dialog box with the message "The Credit Limit is set. "

BREAK from loop

END IF

ELSE

DISPLAY a warning message dialog box with the message "The requested Amount exceeds the Credit Limit. It should be lesser than 2.5 times your balance amount and greater than 2 times your balance amount."

END ELSE

END IF

END IF

END FOR

IF correctCard is false

DISPLAY an error message dialog box with the message "The Card ID is invalid."

END IF

END TRY

CATCH

DISPLAY an error message dialog box with the message "Please enter valid numbers in Card ID, Credit Limit and Grace Period."

END CATCH

END ELSE

END IF

IF (e.getSource() == creditCardCancelCCBtn)

IF (cardIDCreditCancelTF.getText() is empty)

DISPLAY an error message dialog box with the message "Card ID not entered."

END IF

ELSE

TRY

ASSIGN correctCard to false

ASSIGN cardIDCancelText to the text from cardIDCreditCancelTF

ASSIGN cardIDCancel to the integer value of cardIDCancelText

FOR each card in cards DO

IF card is an instance of CreditCard

DOWNCAST card to CreditCard and ASSIGN to creditCard

IF (cardIDCancel equals creditCard.getCardID())

SET correctCard to true

CALL cancelCreditCard method of CreditCard

DISPLAY an information message dialog box with the message "The Card has been cancelled."

BREAK from loop

END IF

END IF

END FOR

IF correctCard is false

DISPLAY an error message dialog box with the message "The Card ID is invalid."

END IF

END TRY

CATCH

```
                    DISPLAY an error message dialog box with the
        message "Please enter valid number in Card ID."
            END CATCH
        END ELSE
    END IF

    IF (e.getSource() == displayDCBtn)
        FOR each card in cards DO
            IF card is an instance of DebitCard
                CALL debitDisplay() method on the DebitCard object
            END IF
        END FOR
    END IF

    IF (e.getSource() == displayCCBtn)
        FOR each card in cards DO
            IF card is an instance of CreditCard
                CALL creditDisplay() method on the CreditCard
            object
            END IF
        END FOR
    END IF
```

```
IF (e.getSource() == clearDCBtn)
        SET debitCardIDTF to empty string
        SET debitClientNameTF to empty string
        SET debitIssuerBankTF to empty string
        SET debitBankAccountTF to empty string
        SET debitBalanceAmountTF to empty string
        SET debitPinNumberTF to empty string

        SET cardIDWithdrawTF to empty string
        SET pinNumberWithdrawTF to empty string
        SET withdrawalAmountTF to empty string
        SET selected item of dayDateOfWithdrawCB to 1
        SET selected item of monthDateOfWithdrawCB to “Jan”
        SET selected item of yearDateOfWithdrawCB to 2000
END IF

IF (e.getSource() == clearCCBtn)
        SET creditCardIDTF to empty string
        SET creditClientNameTF to empty string
```

```
            SET creditIssuerBankTF to empty string
            SET creditBankAccountTF to empty string
            SET creditBalanceAmountTF to empty string
            SET creditCvcNumberTF to empty string
            SET creditInterestRateTF to empty string
            SET selected item of dayExpirationDateCB to 1
            SET selected item of monthExpirationDateCB to “Jan”
            SET selected item of yearExpirationDateCB to 2000

            SET cardIDcreditLimitTF to empty string
            SET creditLimitTF to empty string
            SET gracePeriodTF to empty string

            SET cardIDCreditCancelTF to empty string
        END IF
    END DO

    FUNCTION main
    DO
        CREATE BankGUI object obj
    END DO
```

4. Method Description of BankGUI

4.1 BankGUI ()

This is a constructor of BankGUI class which takes no parameter. This method consists of all the components present in the GUI including JFrame, JPanel, JLabel, JTextField, JCombox, JButtons. It also set the size of JFrame (frame) and JPanel (panelDC and panelCC) and set the bounds of all components and adds them to their respective JPanels. Finally, the JPanel (panelDC) is added to the frame.

4.2 Method actionPerformed ()

This is the only method of Java ActionListener and is automatically initialized whenever a button or a menu is clicked. ActionEvent and the name of the event should be written in the parenthesis. (yuvatimankar, 2023) This method is called when any of one of the buttons is clicked and is action is preformed according to the clicked button.

4.3 Button switchToDC

When this button is clicked, it switches to Debit Card panel (panelDC). It removes the Credit Card panel (panelCC) from the frame and adds Debit Card panel (panelDC) to the frame.

4.4 Button switchToCC

When this button is clicked, it switches to Credit Card panel (panelCC). It removes the Debit Card panel (panelDC) from the frame and adds Credit Card panel (panelCC) to the frame.

4.5 Button addDebitCardBtn

When the button is clicked, it adds a new Debit Card object to the Array List cards. All the required text fields should be filled with suitable values of Add Debit Card. If any text field is empty or inappropriate values are given an error message is displayed. It also checks whether the card already exists or not by comparing the Card ID. Finally, the card is added.

4.6 Button withdrawDCBtn

When the button is clicked, withdraw method is called and withdraws according to the given value. All the required text fields should be filled with suitable values. If any text field is empty or inappropriate values are given an error message is displayed. Then, it checks if the card exists or not. If it does, then it compares the PIN Number and Withdrawal Amount is compatible or not. If they are then withdraw method is called and a successful message will be displayed. If any error occurs, then the user will be notified accordingly.

4.7 Button clearDCBtn

When the button is clicked, it clears all text field of the Debit Card panel (panelDC).

4.8 Button displayDCBtn

When the button is clicked, it displays all the cards present in cards Array List belonging to the Debit card class. This checks every card from the Array List and only when a card matches with the Debit card, it calls debitDisplay method.

4.9 Button addCreditCardBtn

When the button is clicked, it adds a new Credit Card object to the Array List cards. All the required text fields should be filled with suitable values of Add Credit Card. If any text field is empty or inappropriate values are given an error message is displayed. It also checks whether the card already exists or not by comparing the Card ID. Finally, the card is added.

4.10 Button setCreditLimitCCBtn

When the button is clicked, setCreditLimit method is called and sets the Credit Limit according to the given value. All the required text fields should be filled with suitable values of the Set Credit Limit. If any text field is empty or inappropriate values are given an error message is displayed. Then, it checks if the card exists or not by comparing Card Id form the existing card in Array List. If it does, then it compares whether the Credit Limit is compatible or not. If it is then setCreditLimit method is called and a successful message will be displayed. If any error occurs, then the user will be notified accordingly.

4.11 Button creditCardCancelCCBtn

When the button is clicked, cancelCreditCard method is called and removes the credit card. The required text field is Card ID should be filled with suitable values of the Credit Card Cancellation. If text field is left empty or inappropriate value is given an error message is displayed. Then, it checks if the card exists or not by comparing Card Id form the existing card in Array List. If it does, then cancelCreditCard method is called and a successful message will be displayed. If an invalid Card ID is given, then the user will be notified.

4.12 Button clearCCBtn

When the button is clicked, it clears all text field of the Debit Card panel (panelDC).

4.13 Button displayCCBtn

When the button is clicked, it displays all the cards present in cards Array List belonging to the Credit card class. This checks every card from the Array List and only when a card matches with the Credit card, it calls creditDisplay method.

5. Testing

5.1 Test 1

Test that the program can be compiled and run using the command prompt, including a screenshot like Figure 1 from the command prompt learning aid.

Objective:	To ensure the application can be compiled and run using the command prompt
Action	- Command prompt is opened with the folder in which the java program is present: cd path = C:\Users\Windows 10\Documents\Islington\Programming\Sem1_Programming\22067566 Aadesh Shrestha - BankGUI is run by using the following command: java BankGUI.java
Expected Result	The GUI would be displayed.
Actual Result	The GUI was displayed.
Conclusion	The test was successful.

Table 1: To ensure the application can be compiled and run using the command prompt

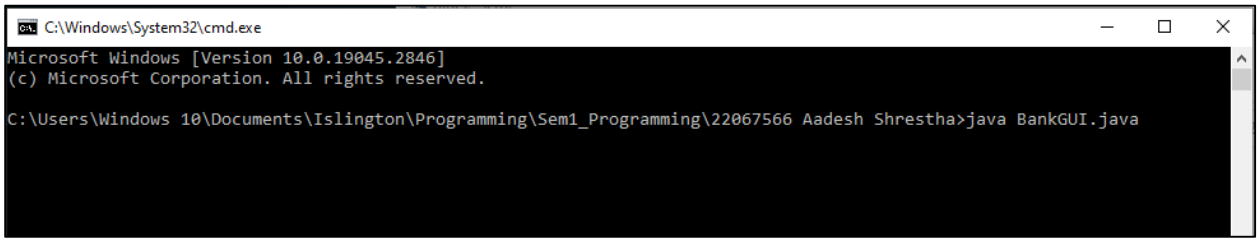

Figure 6: Screenshot of Command Prompt in java program folder and opening BankGUI.java

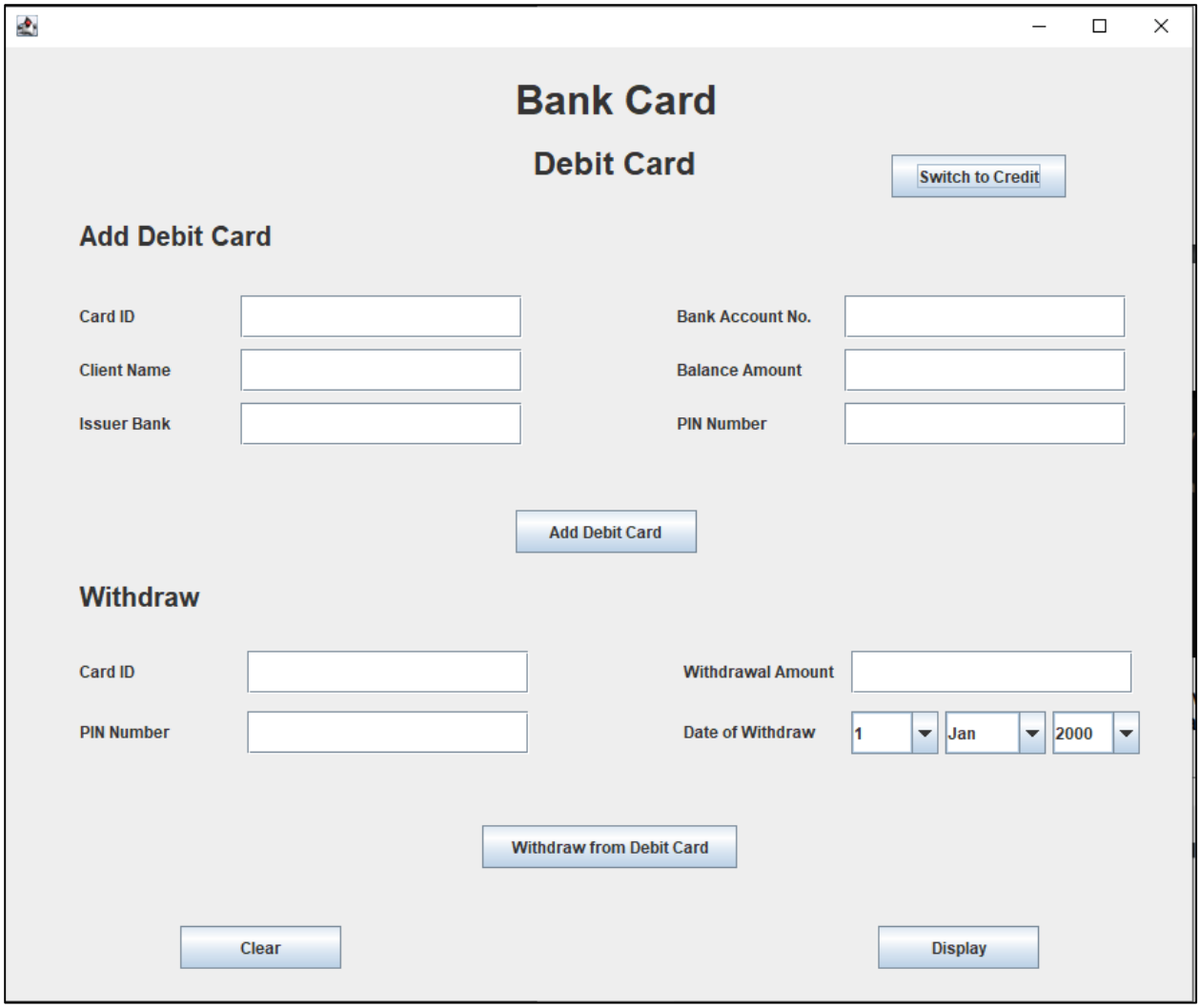

Figure 7: Screenshot of BankGUI class opened through command prompt (panel of Debit Card)

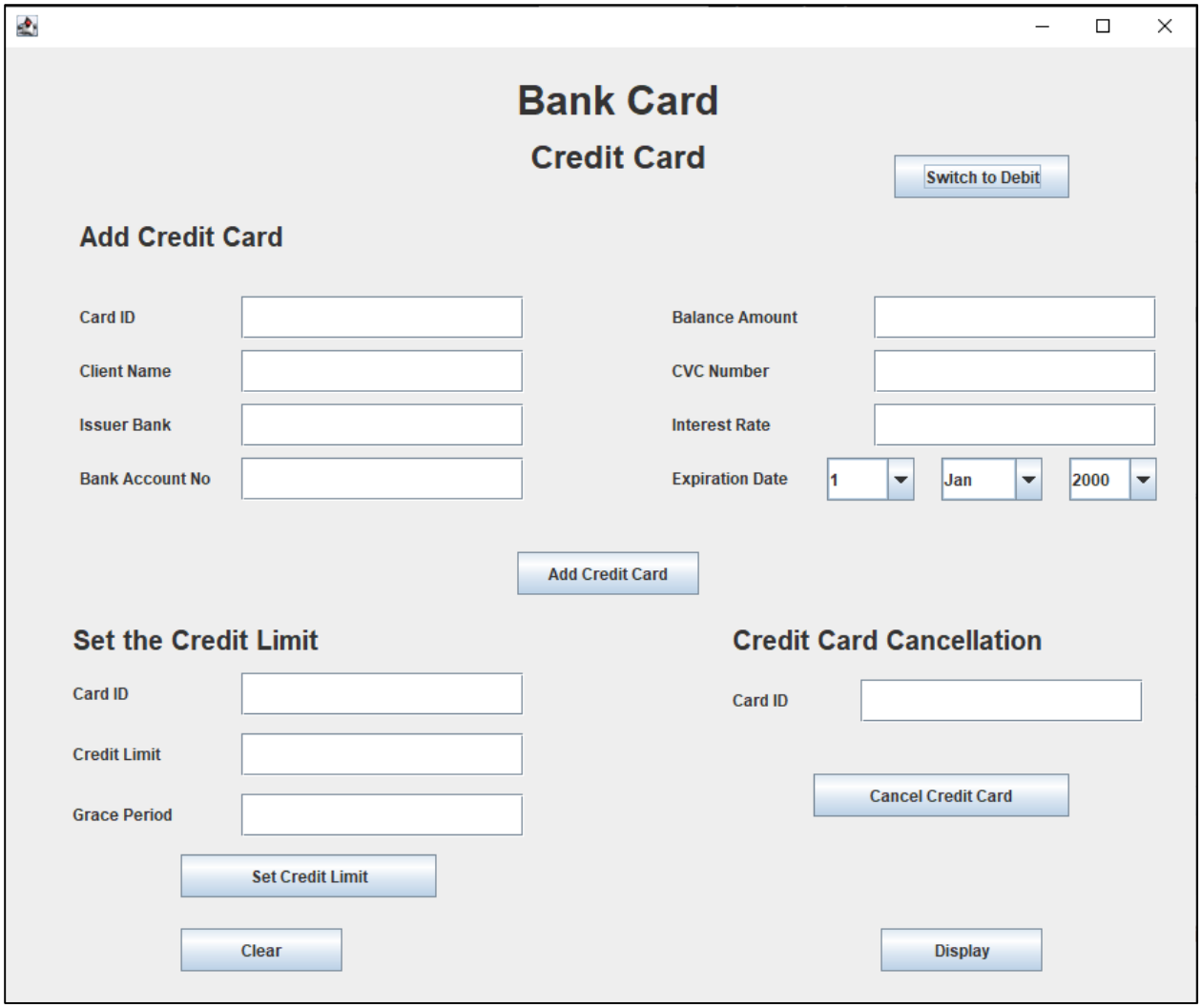

Figure 8: Screenshot of BankGUI class opened through command prompt (panel of Credit Card)

5.2 Test 2

Evidences should be shown of:

a. Add DebitCard

Objective:	To add a debit card
Action	- BankGUI form is opened. - Debit Card panel is loaded. - Details of Debit card is filled in the text fields of Add Debit Card as follows: • Card ID = 1 • Client Name = “Aadesh” • Issuer Bank = “Sanima” • Bank Account No. = “3546632” • Balance Amount = 12300 • PIN Number = 3241 - Note that the details should be accurate or an error or warning message will be given. - Click on the “Add Debit Card” button.
Expected Result	A dialog box stating that the card has been added would be displayed.
Actual Result	A dialog box stating that the card has been added was displayed.
Conclusion	The test was successful.

Table 2: To add a debit card

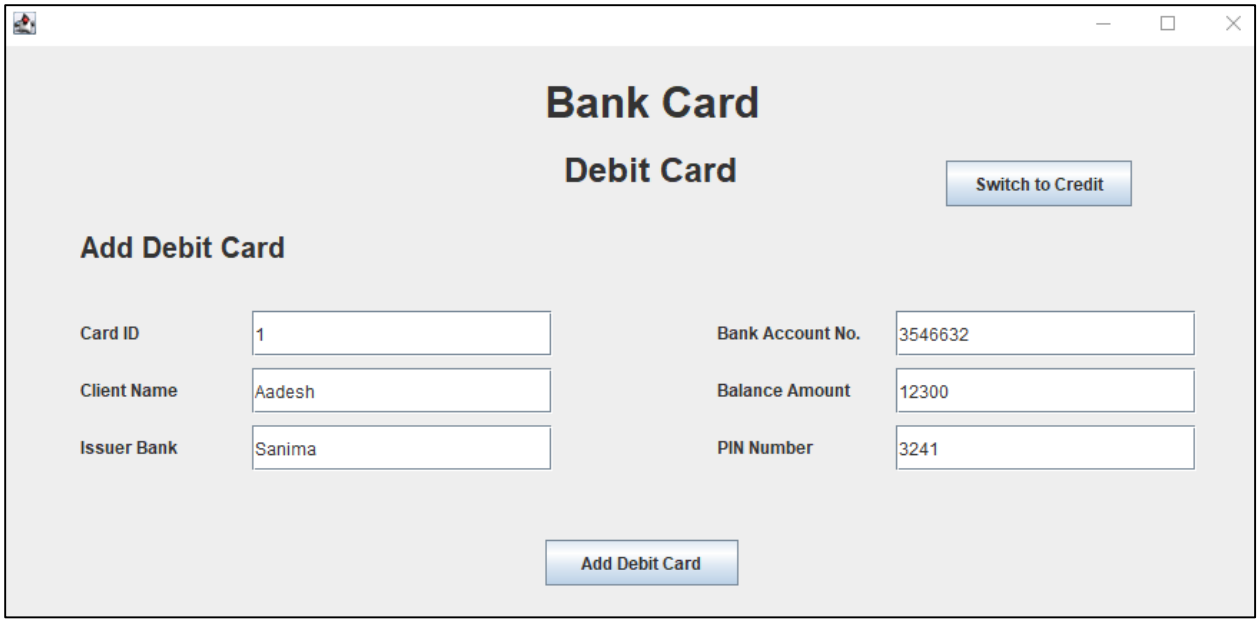

Figure 9: Screenshot to add a debit card(i)

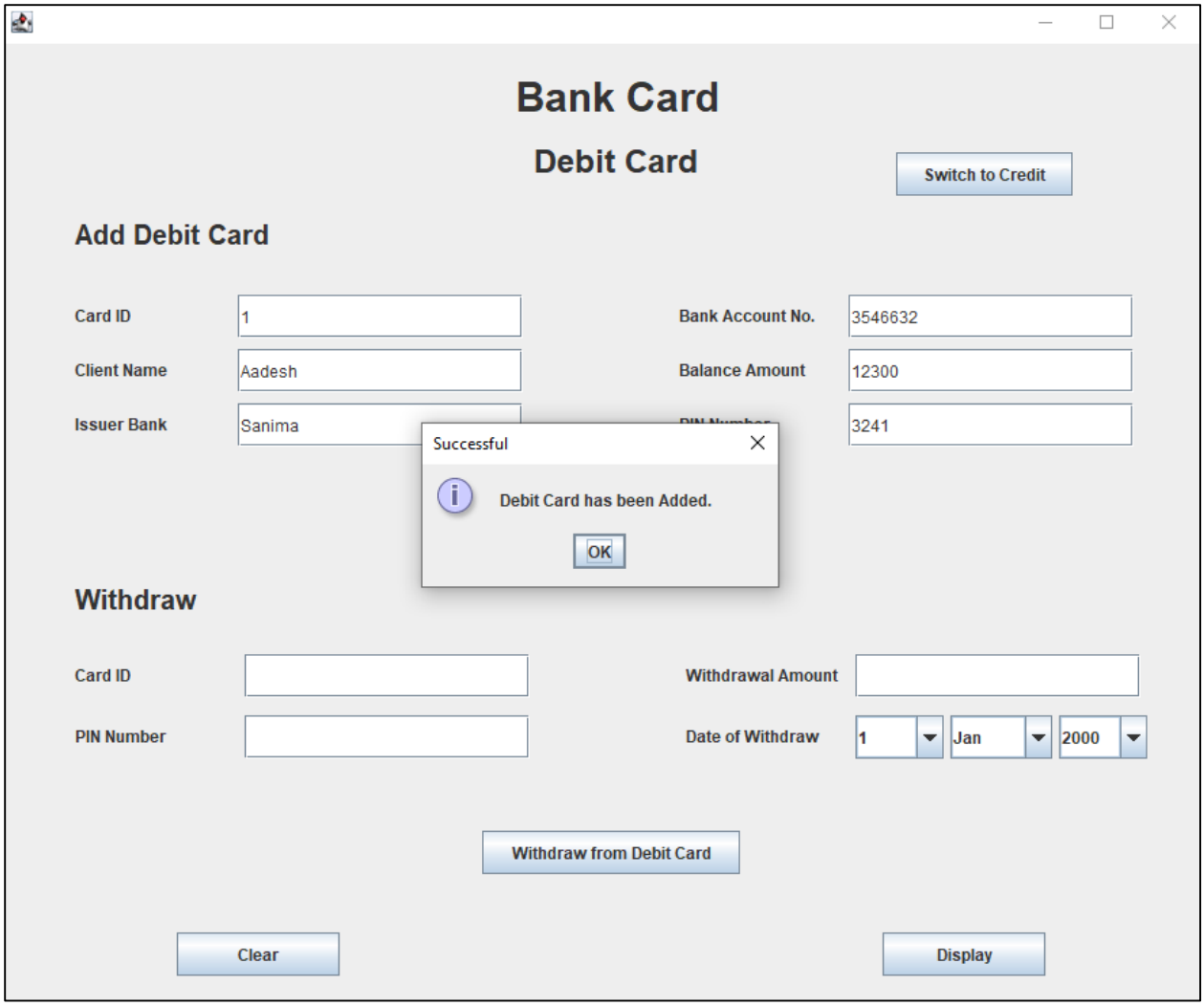

Figure 10: Screenshot to add a debit card(ii)

b. Add CreditCard

Objective:	To add a credit card
Action	- BankGUI form is opened. - Debit Card panel is loaded. - Switch to Credit Card panel by clicking the “Switch to Credit” Button. - Credit Card panel is loaded - Details of Credit card is filled in the text fields of Add Credit Card as follows: • Card ID = 1 • Client Name = “Anless” • Issuer Bank = “Sagarmatha” • Bank Account No. = “2531432” • Balance Amount = 213400 • CVC Number = 5677 • Interest Rate = 12.5 • Expiration Date = “6 April 2023” - Note that the details should be accurate or an error or warning message will be given. - Click on the “Add Credit Card” button.
Expected Result	A dialog box stating that the card has been added would be displayed.
Actual Result	A dialog box stating that the card has been added was displayed.
Conclusion	The test was successful.

Table 3: To add a credit card

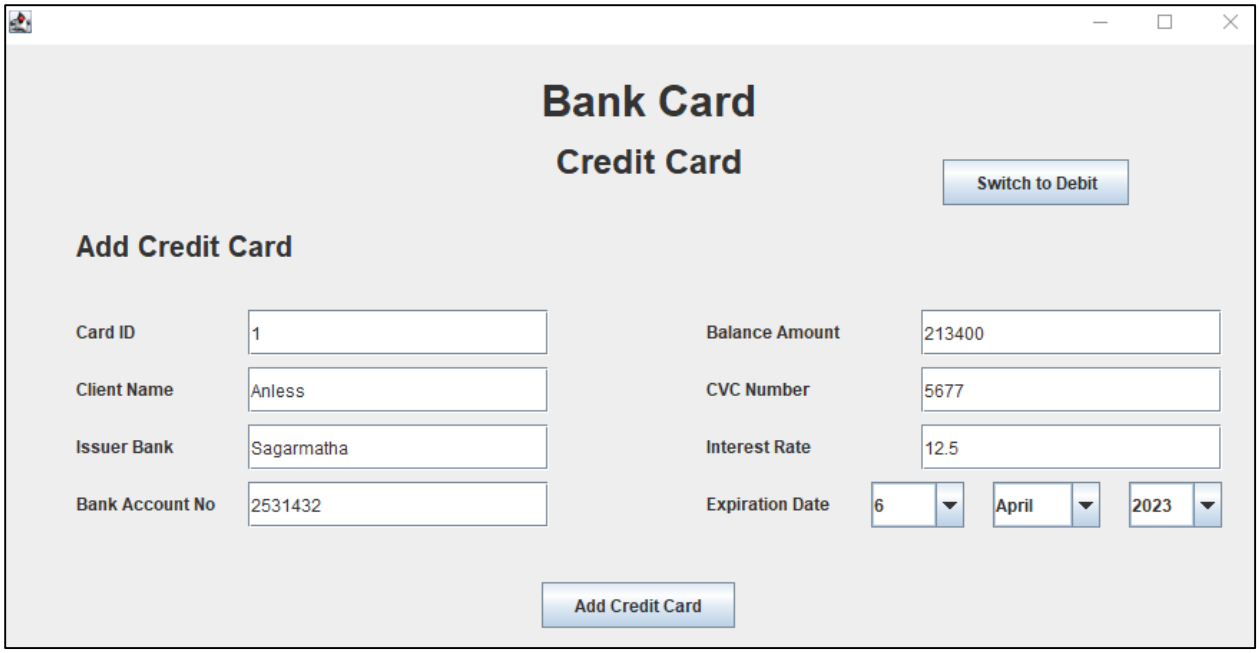

Figure 11: Screenshot of adding a credit card(i)

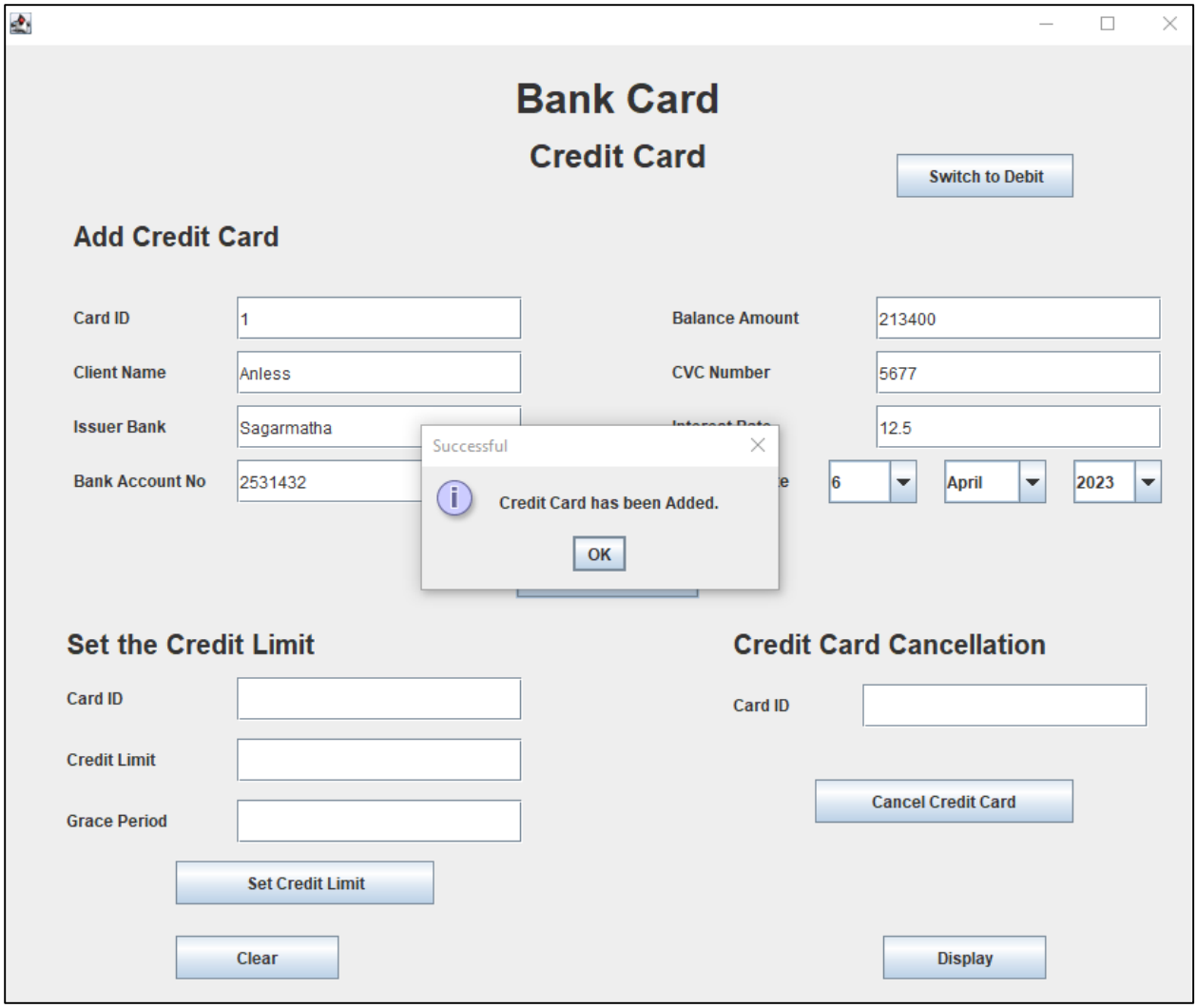

Figure 12: Screenshot of adding a credit card(ii)

c. Withdraw amount from Debit card

Objective:	To withdraw from debit card
Action	- BankGUI form is opened. - Debit Card panel is loaded. - Details of Debit card is filled in the text fields of Withdraw section as follows: • Card ID = 1 • PIN Number = 3241 • Withdrawal Amount = 2300 • Date of Withdraw = “4 May 2023” - Note that the details should be accurate or an error or warning message will be given. - Click on the “Withdraw from Debit Card” button.
Expected Result	A dialog box stating that the given amount is withdrawn would be displayed.
Actual Result	A dialog box stating that the given amount is withdrawn was displayed.
Conclusion	The test was successful.

Table 4: To withdraw from debit card

Figure 13: Screenshot of withdrawing from debit card(i)

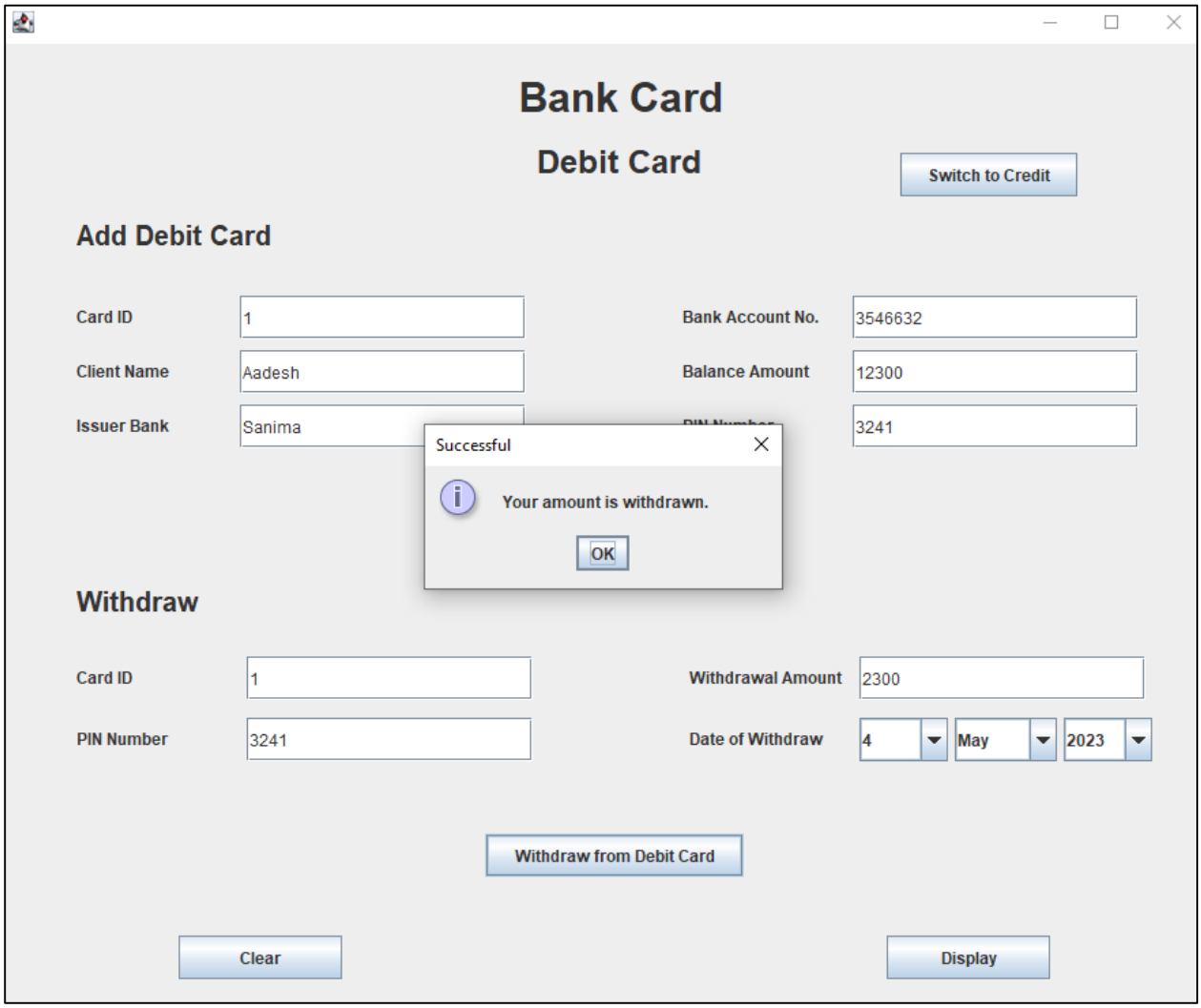

Figure 14: Screenshot of withdrawing from debit card(ii)

d. Set the credit limit

Objective:	To set the credit limit of a credit card
Action	- BankGUI form is opened. - Debit Card panel is loaded. - Switch to Credit Card panel by clicking the “Switch to Credit” Button. - Credit Card panel is loaded - Details of Credit card is filled in the text fields of Set the Credit Limit as follows: • Card ID = 1 • Credit Limit = 533000 • Grace Period = 30 - Note that the details should be accurate or an error or warning message will be given. - Click on the “Set Credit Limit” button.
Expected Result	A dialog box stating that the credit limit of the card has been set would be displayed.
Actual Result	A dialog box stating that the credit limit of the card has been set was displayed.
Conclusion	The test was successful.

Table 5: To set the credit limit of a credit card

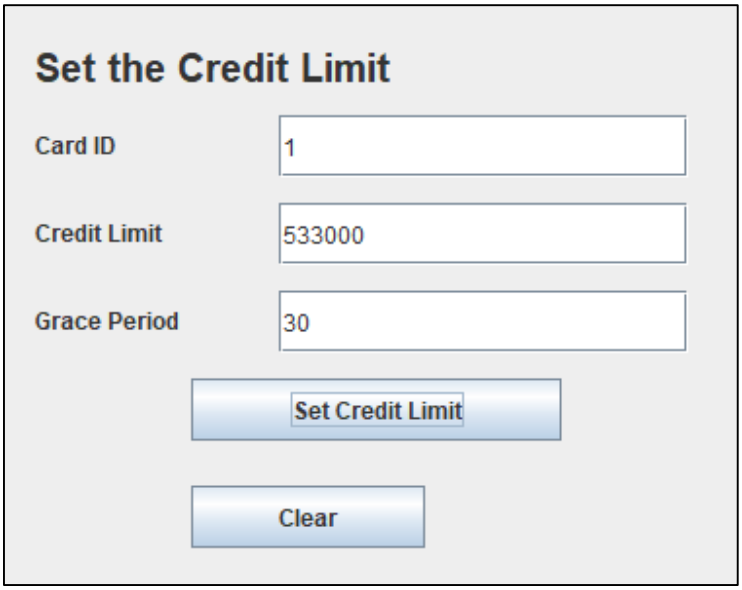

Figure 15: Screenshot of setting the credit limit of a credit card(i)

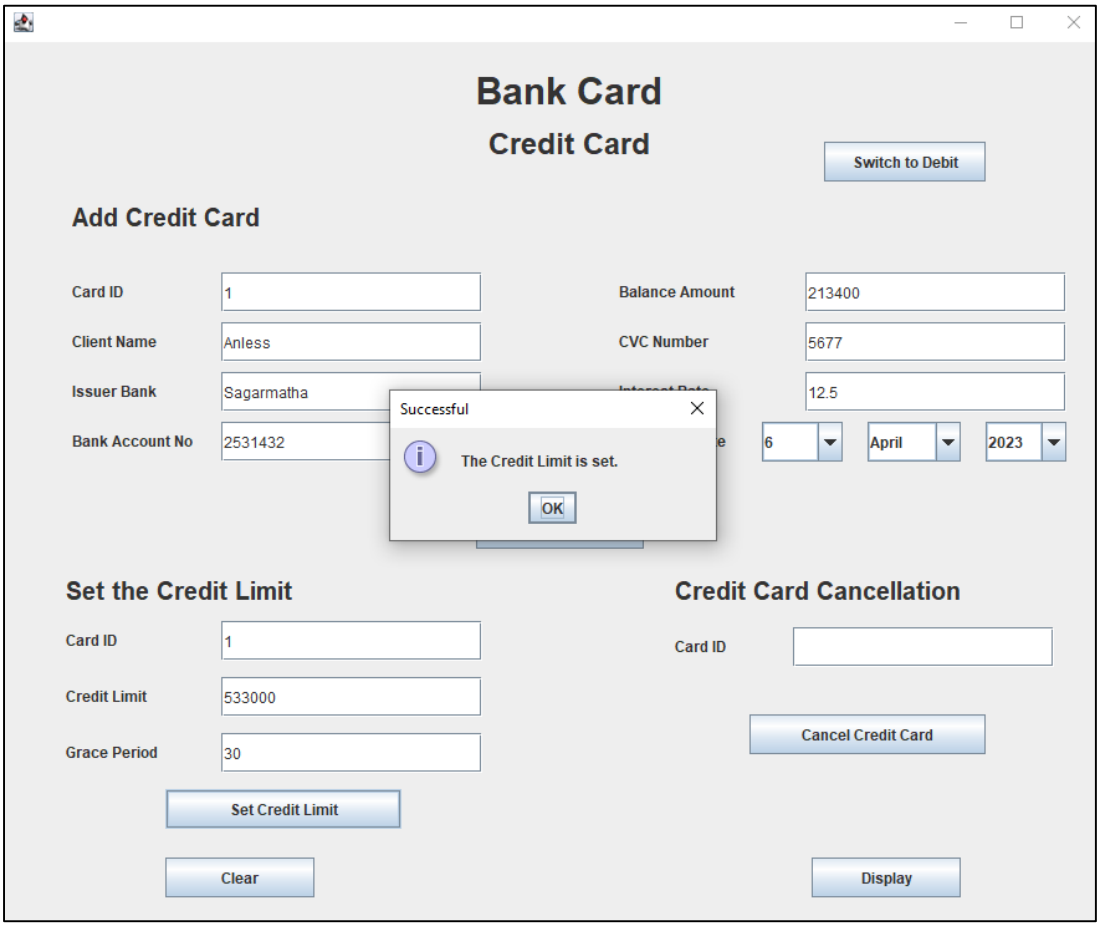

Figure 16: Screenshot of setting the credit limit of a credit card(ii)

e. Remove the credit card

Objective:	To remove a credit card
Action	- BankGUI form is opened. - Debit Card panel is loaded. - Switch to Credit Card panel by clicking the “Switch to Credit” Button. - Credit Card panel is loaded - The card ID of Credit card is filled in the text field of Credit Card Cancellation as follow: • Card ID = 1 - Note that the details should be accurate or an error or warning message will be given. - Click on the “Cancel Credit Card” button.
Expected Result	A dialog box stating that the credit card has been cancelled would be displayed.
Actual Result	A dialog box stating that the credit card has been cancelled was displayed.
Conclusion	The test was successful.

Table 6: To remove a credit card

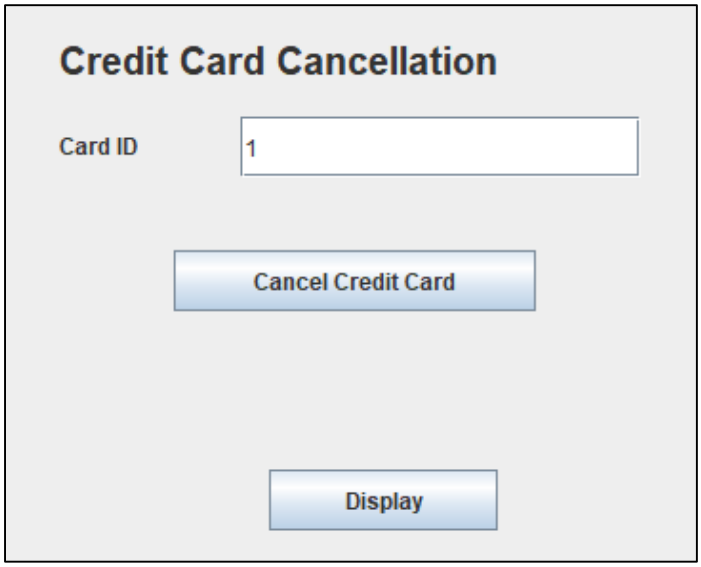

Figure 17: Screenshot of removing a credit card(i)

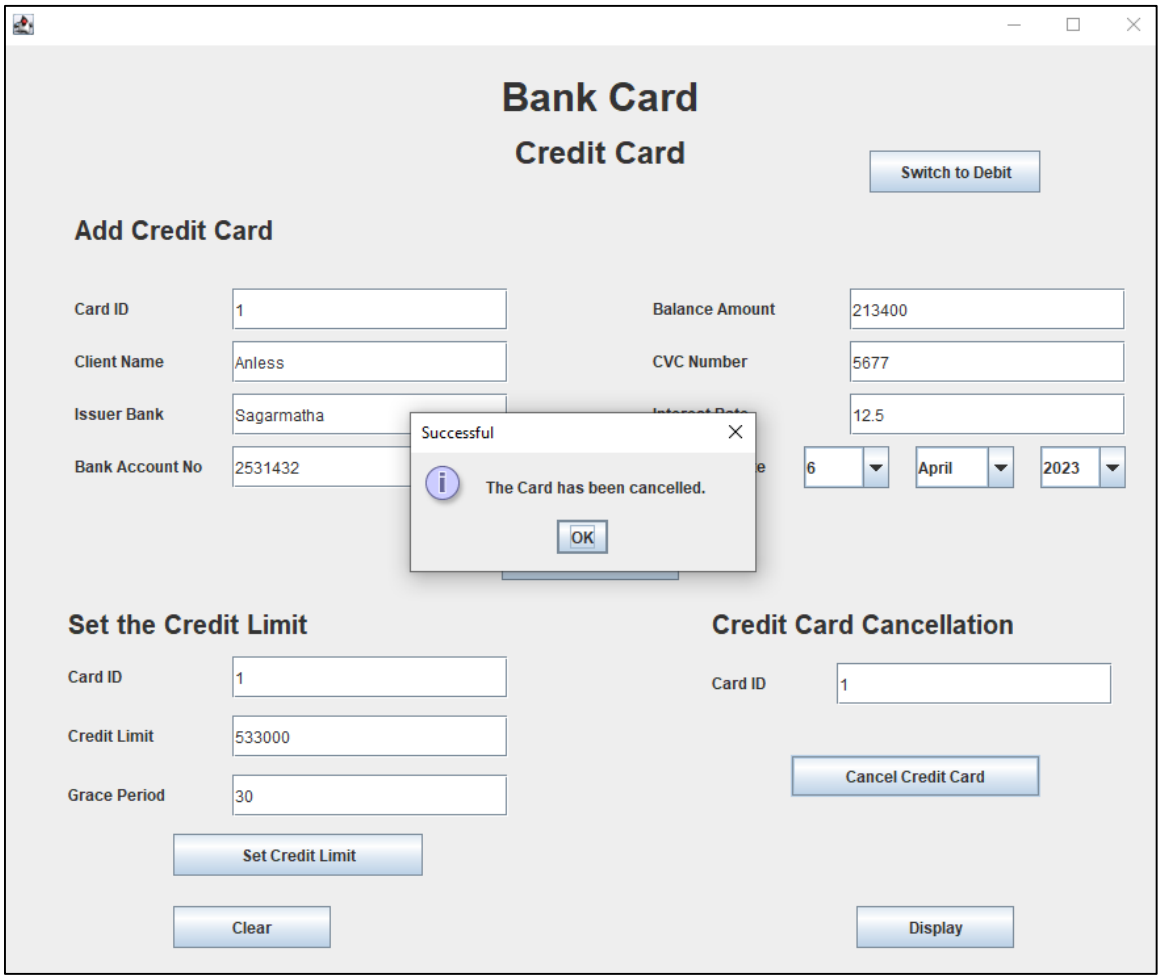

Figure 18: Screenshot of removing a credit card(ii)

5.3 Test 3

Test that appropriate dialog boxes appear when unsuitable values are entered for the Card ID.

Case 1

Objective:	To display appropriate dialog box when unsuitable value(i.e. String) is entered for the Card ID for the Withdraw section.
Action	- BankGUI form is opened. - Debit Card panel is loaded. - Details of Debit card is filled in the text fields of Withdraw section as follows: • Card ID = one • PIN Number = 3241 • Withdrawal Amount = 2300 • Date of Withdraw = “4 May 2023” - Click on the “Withdraw from Debit Card” button.
Expected Result	A dialog box stating that an error has occurred and instructions to fix the error would be displayed.
Actual Result	A dialog box stating that an error has occurred and instructions to fix the error was displayed.
Conclusion	The test was successful.

Table 7: To display appropriate dialog box when unsuitable value is entered for the Card ID

Figure 19: Screenshot of displaying appropriate dialog box when unsuitable value is entered for the Card ID(i)

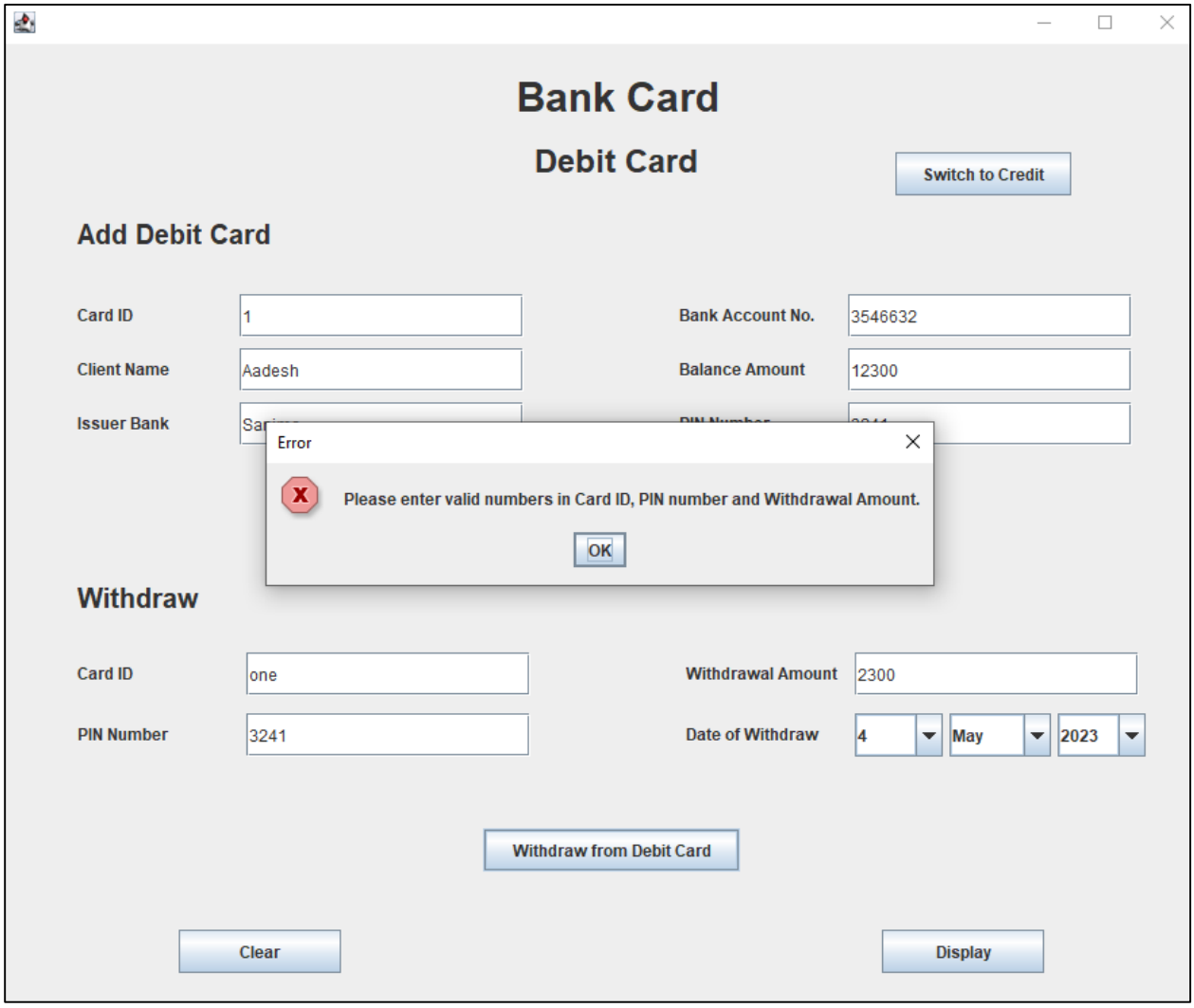

Figure 20: Screenshot of displaying appropriate dialog box when unsuitable value is entered for the Card ID(ii)

Case 2

Objective:	To display appropriate dialog box when non-existing Card ID (let's assume that card ID = 5 does not exist) is entered for the Card ID for the Withdraw section.
Action	- BankGUI form is opened. - Debit Card panel is loaded. - Details of Debit card is filled in the text fields of Withdraw section as follows: • Card ID = 5 • PIN Number = 3241 • Withdrawal Amount = 2300 • Date of Withdraw = "4 May 2023" - Click on the "Withdraw from Debit Card" button.
Expected Result	A dialog box stating that an error has occurred would be displayed.
Actual Result	A dialog box stating that an error has occurred was displayed.
Conclusion	The test was successful.

Table 8: To display appropriate dialog box when non-existing Card ID is entered for the Card ID

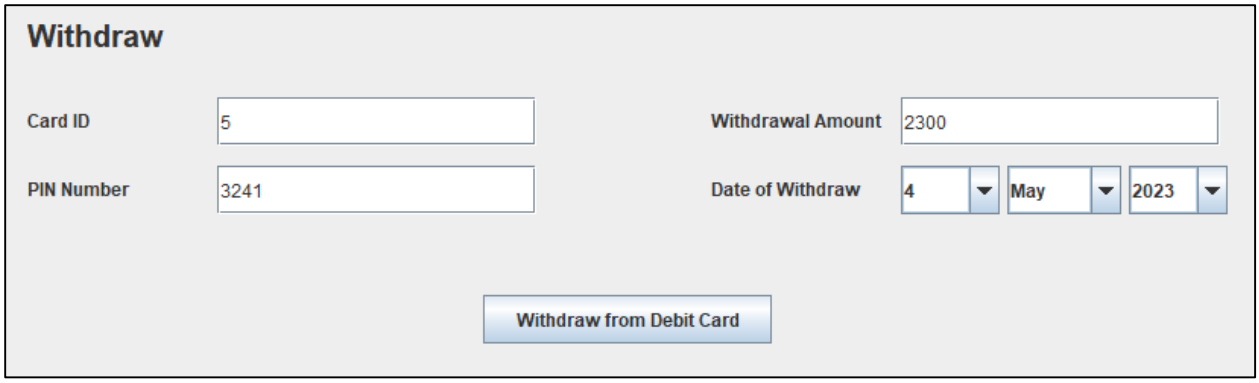

Figure 21: Screenshot of displaying appropriate dialog box when non-existing Card ID is entered for the Card ID(i)

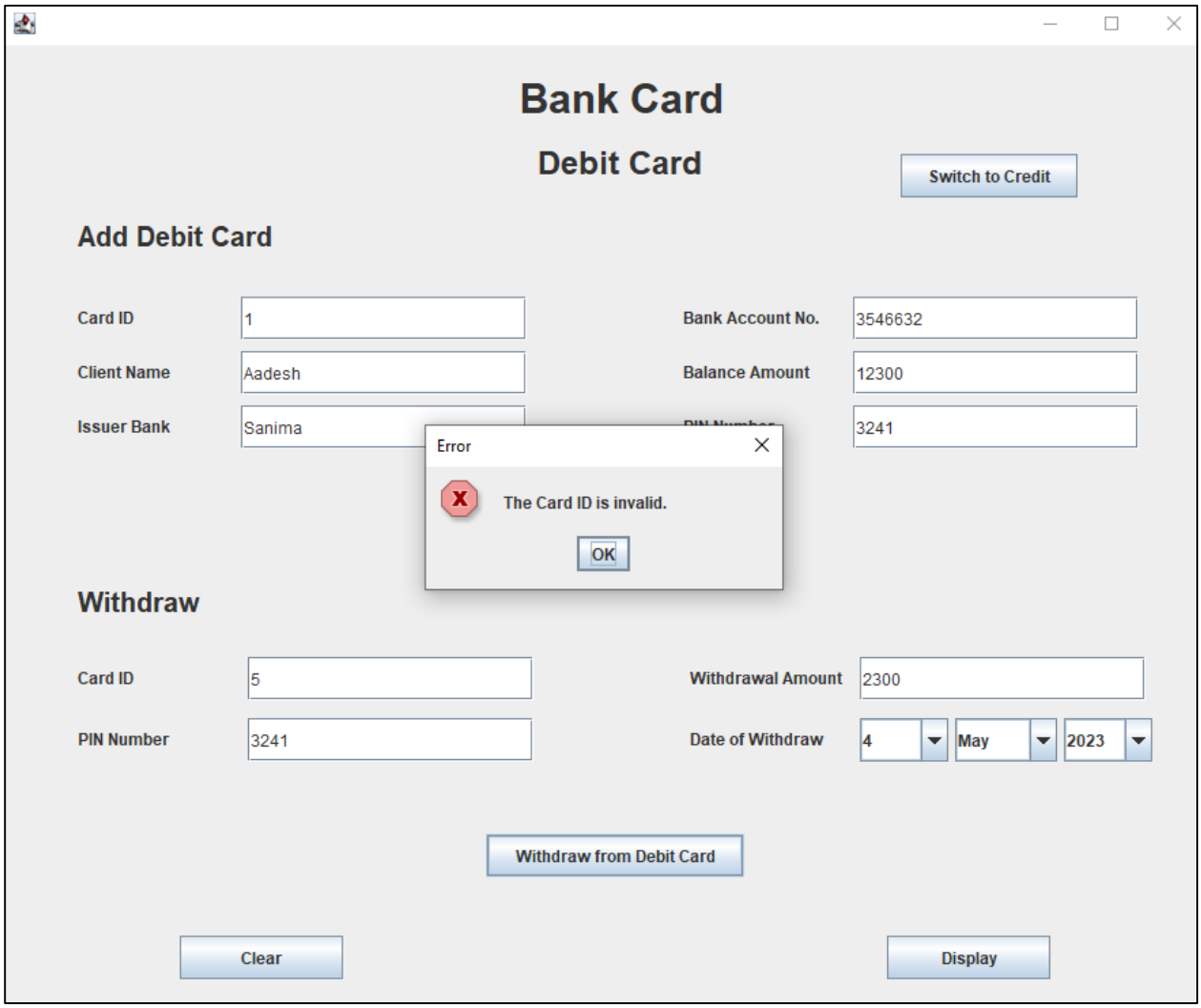

Figure 22: Screenshot of displaying appropriate dialog box when non-existing Card ID is entered for the Card ID(ii)

6. Error Detection and Correction

6.1 Error 1: Syntax Error

The most prevalent error encountered by programmers while coding is a syntax error. Similar to when writing a sentence, the grammatical structure should be followed likewise while coding a similar structure should be followed which is called syntax. These errors are detected immediately and a brief description is provided to describe the error. Syntax errors are also called Compile Time Error. (Sonal_Singh, 2022) One of these cases is depicted below:

```
DebitCard debitCard = (DebitCard) card;
if((debitCard.getCardID() == cardIDWithdraw){
    correctCard = true;
    ((DebitCard)card).withdraw(pinNumberWithdraw, w
    if (pinNumberWithdraw == debitCard.getPinNumber
        if(withdrawalAmount < debitCard.getBalanceA
```

Figure 23: Syntax Error

```
DebitCard debitCard = (DebitCard) card;
if(debitCard.getCardID() == cardIDWithdraw){
    correctCard = true;
    ((DebitCard)card).withdraw(pinNumberWithdraw, witho
    if (pinNumberWithdraw == debitCard.getPinNumber())
```

Figure 24: Correction of Syntax Error

The error shown above is a syntax error caused by placing an extra opening bracket in the IF statement. BlueJ detected the error and suggested that “) expected” was needed. A programmer would not be able to detect small errors on their own or rather it would be very difficult and take a lot of time but since the compiler alerts the programmer makes the flow of the program smoother. This error was fixed my removing the extra opening bracket.

6.2 Error 2: Semantic Error

Semantic error is caused when Java statements are incorrectly written. Not all IDEs can detect this problem but BlueJ can detect them. Although these errors are detected, appropriate solutions are not given. (Calvanese, 2022) An indicator for the error is shown but the programmers have to manually find the solution. One of these cases is depicted below:

```
DebitCard debitObj = new DebitCard(cardID, clientName, issuerBank, bankAccount, balanceAmount, pinNumber);
```

Figure 25: Semantic Error

```
DebitCard debitObj = new DebitCard(cardID, issuerBank, bankAccount, balanceAmount, clientName, pinNumber);
```

Figure 26: Correction of Semantic Error

The error above is a semantic error caused by placing the parameter of DebitCard incorrectly. The syntax of creating an object contains no error but since we are calling a method from another class it is necessary to place the parameter in proper order so that the data type remains uniform. This error was fixed by placing the parameters properly.

6.3 Error 3: Logical Error

Logical errors are difficult to spot as the syntax of the codes is valid but returns incorrect results. These errors cannot be detected by the compiler. These errors occur when a programmer uses a wrong idea or concept when writing the code. To put it simply, logical errors are errors that work with an incorrect concept. (Sonal_Singh, 2022) One of these cases is depicted below:

```
dayDateOfWithdrawCB.setSelectedItem(5);
monthDateOfWithdrawCB.setSelectedItem("Jan");
yearDateOfWithdrawCB.setSelectedItem(2000);
```

Figure 27: Logical Error

```
dayDateOfWithdrawCB.setSelectedItem(1);
monthDateOfWithdrawCB.setSelectedItem("Jan");
yearDateOfWithdrawCB.setSelectedItem(2000);
```

Figure 28: Correction of Logical Error

The diagram above depicts the logical error caused by assign the value 5 to the day date of withdraw when the clear button is clicked. The purpose of clear is to reset the form to its starting point. But in this case it changes the value to 5 instead of the initial value i.e. 1. Although it does not seem like much, this lowers the quality of the program. This was fixed by manually finding the error and fixing it.

6.4 Error 4: Runtime Error

Runtime errors are those errors that occur when the program is being executed after being successfully compiled. These errors are detected when the user interacts with the program. When these errors are encountered the computer is unable to decide what to do. (Sonal_Singh, 2022) One of these cases is depicted below:

```
public BankGUI(){
    //Initialization of frame with required customization
    frame = new JFrame();
    frame.setSize(900,750);

    /** Debit Card Panel */

    //Heading
    mainTitleDC = new JLabel("Bank Card");
    mainTitleDC.setFont(new Font("Arial",Font.BOLD,30));
    mainTitleDC.setBounds(380, 20, 150, 40);
    panelDC.add(mainTitleDC);
```

Figure 29: Runtime Error

```
public BankGUI(){
    //Initialization of frame with required customization
    frame = new JFrame();
    frame.setSize(900,750);
    panelDC = new JPanel();
    panelDC.setSize(900, 750);
```

Figure 30: Correction of Runtime Error

The figure above depicts the runtime error which is occurred by using a null panel and calling a method on it. This error did not show up during the compilation but was shown while executing the program. These are very difficult to spot and may lead to bugs and program crashes. This error was fixed by initializing the panel.

7. Conclusion

In this project, the importance of a user-friendly graphical interface was learned. GUI plays a huge role in the usability of an application. If a GUI is too complex, it will affect the functionality of the program. A GUI should look clean, contain adequate information and be easy to use. This project used various features of the swing and AWT to create a proper GUI for Bank Card form. Not only the GUI but the functionality to perform various actions was also implemented. Array List was also used to store the cards and to retrieve them when necessary. Exception handling was also utilized to avoid any bugs or program crashes.

Numerous challenges were encountered during the development of the program. Errors such as syntax, semantic, runtime and logical errors came up multiple times at different parts of the program. The project presented unique problems that made finding a definitive solution very difficult especially will adding functionality to the buttons. Some of the issues were resolved through the hints given by our teacher and then piecing the puzzle together by experimenting with different coding patterns until a desired outcome was achieved.

In summary, this coursework has been an eye opening experience in which all the programming knowledge acquired through the first as well as the second semester came together to create an amazing application with well-developed functionality which gives desirable outcomes. Due to this, my inner self has grown more eager to learn about new concepts that will be useful in my coding carrer.

8. References

BlueJ org, n.d. *About BlueJ.* [Online]
Available at: https://www.bluej.org/about.html
[Accessed 8 May 2023].

Calvanese, D., 2022. *Semantic errors.* [Online]
Available at: https://www.inf.unibz.it/~calvanese/teaching/06-07-ip/lecture-notes/uni10/node4.html
[Accessed 8 May 2023].

Sonal_Singh, 2022. *Types of Errors in Java with Examples - GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/types-of-errors-in-java-with-examples/
[Accessed 8 May 2023].

yuvatimankar, 2023. *Java ActionListener - Coding Ninjas.* [Online]
Available at: https://www.codingninjas.com/codestudio/library/java-actionlistener
[Accessed 8 May 2023].

9. Appendix

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

public class BankGUI implements ActionListener
{
    //Declaration of frame and panelDC
    private JFrame frame;
    private JPanel panelDC;

    /**Attributes for Credit card */

    //Headings
    private JLabel mainTitleDC;
    private JLabel debitCardTitle;
    private JLabel addDebitCardTitle;
    private JLabel withdraw;

    //For Add Debit Card
```

```
    private JLabel debitCardIDL;
    private JTextField debitCardIDTF;

    private JLabel debitClientNameL;
    private JTextField debitClientNameTF;

    private JLabel debitIssuerBankL;
    private JTextField debitIssuerBankTF;

    private JLabel debitBankAccountL;
    private JTextField debitBankAccountTF;

    private JLabel debitBalanceAmountL;
    private JTextField debitBalanceAmountTF;

    private JLabel debitPinNumberL;
    private JTextField debitPinNumberTF;

    //For Withdraw

    private JLabel cardIDWithdrawL;
    private JTextField cardIDWithdrawTF;
```

```
    private JLabel pinNumberWithdrawL;
    private JTextField pinNumberWithdrawTF;

    private JLabel withdrawalAmountL;
    private JTextField withdrawalAmountTF;

    private JLabel dateOfWithdrawL;
    private JComboBox dayDateOfWithdrawCB, monthDateOfWithdrawCB,
yearDateOfWithdrawCB;

    //Buttons
    private JButton switchToDC;
    private JButton switchToCC;
    private JButton addDebitCardBtn;
    private JButton withdrawDCBtn;
    private JButton clearDCBtn;
    private JButton displayDCBtn;

    /**Attributes for Credit card */

    //Declaration of panelCC
    private JPanel panelCC;
```

```
//Headings
private JLabel mainTitleCC;
private JLabel creditCardTitle;
private JLabel addCreditCardTitle;
private JLabel setCreditLimit;
private JLabel creditCardCancel;

//For Add Credit Card

private JLabel creditCardIDL;
private JTextField creditCardIDTF;

private JLabel creditClientNameL;
private JTextField creditClientNameTF;

private JLabel creditIssuerBankL;
private JTextField creditIssuerBankTF;

private JLabel creditBankAccountL;
private JTextField creditBankAccountTF;

private JLabel creditBalanceAmountL;
private JTextField creditBalanceAmountTF;
```

```
    private JLabel creditCvcNumberL;
    private JTextField creditCvcNumberTF;

    private JLabel creditInterestRateL;
    private JTextField creditInterestRateTF;

    private JLabel creditExpirationDateL;
    private JComboBox dayExpirationDateCB, monthExpirationDateCB,
yearExpirationDateCB;

    //For Set Credit Limit
    private JLabel cardIDcreditLimitL;
    private JTextField cardIDcreditLimitTF;

    private JLabel creditLimitL;
    private JTextField creditLimitTF;

    private JLabel gracePeriodL;
    private JTextField gracePeriodTF;

    //For Credit Card Cancellation
    private JLabel cardIDCreditCancelL;
```

```
    private JTextField cardIDCreditCancelTF;


    //Buttons
    private JButton addCreditCardBtn;
    private JButton setCreditLimitCCBtn;
    private JButton creditCardCancelCCBtn;
    private JButton clearCCBtn;
    private JButton displayCCBtn;


    //ArrayList
    ArrayList<BankCard> cards = new ArrayList<BankCard>();


    //List of months for Combo Box
    String[] mon =
{"Jan","Feb","Mar","April","May","Jun","July","Aug","Sept","Oct","Nov","Dec"};


    public BankGUI(){
        //Initialization of frame with required customization
        frame = new JFrame();
        frame.setSize(900,750);
        panelDC = new JPanel();
        panelDC.setSize(900, 750);
```

```
/** Debit Card Panel */

//Heading
mainTitleDC = new JLabel("Bank Card");
mainTitleDC.setFont(new Font("Arial",Font.BOLD,30));
mainTitleDC.setBounds(380, 20, 150, 40);
panelDC.add(mainTitleDC);

//Card-Heading
debitCardTitle = new JLabel("Debit Card");
debitCardTitle.setFont(new Font("Arial",Font.BOLD,24));
debitCardTitle.setBounds(393, 70, 125, 35);
panelDC.add(debitCardTitle);

//Switch button
switchToCC = new JButton("Switch to Credit");
switchToCC.setBounds(660, 80, 130, 32);
switchToCC.addActionListener(this);
panelDC.add(switchToCC);

//Sub-Heading(Add Debit Card)
addDebitCardTitle = new JLabel("Add Debit Card");
addDebitCardTitle.setFont(new Font("Arail",Font.BOLD,20));
```

```
addDebitCardTitle.setBounds(55, 125, 145, 30);
panelDC.add(addDebitCardTitle);

//First Column of Add Debit Card

debitCardIDL = new JLabel("Card ID");
debitCardIDL.setBounds(55, 190, 60, 20);
debitCardIDTF = new JTextField();
debitCardIDTF.setBounds(175, 185, 210, 32);
panelDC.add(debitCardIDL);
panelDC.add(debitCardIDTF);

debitClientNameL = new JLabel("Client Name");
debitClientNameL.setBounds(55, 230, 75, 20);
debitClientNameTF = new JTextField();
debitClientNameTF.setBounds(175, 225, 210, 32);
panelDC.add(debitClientNameL);
panelDC.add(debitClientNameTF);

debitIssuerBankL = new JLabel("Issuer Bank");
debitIssuerBankL.setBounds(55, 270, 75, 20);
debitIssuerBankTF = new JTextField();
debitIssuerBankTF.setBounds(175, 265, 210, 32);
```

```
panelDC.add(debitIssuerBankL);
panelDC.add(debitIssuerBankTF);

//Second Column of Add Debit Card
debitBankAccountL = new JLabel("Bank Account No.");
debitBankAccountL.setBounds(500, 190, 110, 20);
debitBankAccountTF = new JTextField();
debitBankAccountTF.setBounds(625, 185, 210, 32);
panelDC.add(debitBankAccountL);
panelDC.add(debitBankAccountTF);

debitBalanceAmountL = new JLabel("Balance Amount");
debitBalanceAmountL.setBounds(500, 230, 100, 20);
debitBalanceAmountTF = new JTextField();
debitBalanceAmountTF.setBounds(625, 225, 210, 32);
panelDC.add(debitBalanceAmountL);
panelDC.add(debitBalanceAmountTF);

debitPinNumberL = new JLabel("PIN Number");
debitPinNumberL.setBounds(500, 270, 80, 20);
debitPinNumberTF = new JTextField();
debitPinNumberTF.setBounds(625, 265, 210, 32);
panelDC.add(debitPinNumberL);
```

```
panelDC.add(debitPinNumberTF);

//Button(Add Debit Card)
addDebitCardBtn = new JButton("Add Debit Card");
addDebitCardBtn.setBounds(380, 345, 135, 32);
addDebitCardBtn.addActionListener(this);
panelDC.add(addDebitCardBtn);

//Sub-heading(withdraw)
withdraw = new JLabel("Withdraw");
withdraw.setFont(new Font("Arail",Font.BOLD,20));
withdraw.setBounds(55, 395, 90, 28);
panelDC.add(withdraw);

//First column of the Withdraw

cardIDWithdrawL = new JLabel("Card ID");
cardIDWithdrawL.setBounds(55, 455, 50, 20);
cardIDWithdrawTF = new JTextField();
cardIDWithdrawTF.setBounds(180, 450, 210, 32);
panelDC.add(cardIDWithdrawL);
panelDC.add(cardIDWithdrawTF);
```

```
pinNumberWithdrawL = new JLabel("PIN Number");
pinNumberWithdrawL.setBounds(55, 500, 80, 20);
pinNumberWithdrawTF = new JTextField();
pinNumberWithdrawTF.setBounds(180, 495, 210, 32);
panelDC.add(pinNumberWithdrawL);
panelDC.add(pinNumberWithdrawTF);

withdrawalAmountL = new JLabel("Withdrawal Amount");
withdrawalAmountL.setBounds(505, 455, 125, 20);
withdrawalAmountTF = new JTextField();
withdrawalAmountTF.setBounds(630, 450, 210, 32);
panelDC.add(withdrawalAmountL);
panelDC.add(withdrawalAmountTF);

//Second column of Withdraw
dateOfWithdrawL = new JLabel("Date of Withdraw");
dateOfWithdrawL.setBounds(505, 500, 110, 20);
panelDC.add(dateOfWithdrawL);

//day
dayDateOfWithdrawCB = new JComboBox();
for(int i = 1; i <= 31; i++){
    dayDateOfWithdrawCB.addItem(i);
```

```
        }
        dayDateOfWithdrawCB.setBounds(630, 495, 65, 32);
        dayDateOfWithdrawCB.setEditable(true);
        panelDC.add(dayDateOfWithdrawCB);

        //month
        //String[]
months={"Jan","Feb","Mar","April","May","Jun","July","Aug","Sept","Oct","Nov","Dec"};
        monthDateOfWithdrawCB = new JComboBox(mon);
        monthDateOfWithdrawCB.setBounds(700, 495, 75, 32);
        monthDateOfWithdrawCB.setEditable(true);
        panelDC.add(monthDateOfWithdrawCB);

        //year
        yearDateOfWithdrawCB = new JComboBox();
        for(int i = 2000; i <= 2023; i++){
            yearDateOfWithdrawCB.addItem(i);
        }
        yearDateOfWithdrawCB.setBounds(780, 495, 65, 32);
        yearDateOfWithdrawCB.setEditable(true);
        panelDC.add(yearDateOfWithdrawCB);

        //buttons(withdraw,clear,display)
```

```
withdrawDCBtn = new JButton("Withdraw from Debit Card");
withdrawDCBtn.setBounds(355, 580, 190, 32);
withdrawDCBtn.addActionListener(this);
panelDC.add(withdrawDCBtn);

clearDCBtn = new JButton("Clear");
clearDCBtn.setBounds(130, 655, 120, 32);
clearDCBtn.addActionListener(this);
panelDC.add(clearDCBtn);

displayDCBtn = new JButton("Display");
displayDCBtn.setBounds(650, 655, 120, 32);
displayDCBtn.addActionListener(this);
panelDC.add(displayDCBtn);

/** Credit Card Panel */

panelCC = new JPanel();
panelCC.setSize(900, 750);

//Switch button
switchToDC = new JButton("Switch to Debit");
switchToDC.setBounds(660, 80, 130, 32);
```

```
switchToDC.addActionListener(this);
panelCC.add(switchToDC);

//Heading
mainTitleCC = new JLabel("Bank Card");
mainTitleCC.setFont(new Font("Arial",Font.BOLD,30));
mainTitleCC.setBounds(380, 20, 150, 40);
panelCC.add(mainTitleCC);

//Card-Heading
creditCardTitle = new JLabel("Credit Card");
creditCardTitle.setFont(new Font("Arial",Font.BOLD,24));
creditCardTitle.setBounds(390, 65, 130, 35);
panelCC.add(creditCardTitle);

//Sub-Heading(Add Credit Card)
addCreditCardTitle = new JLabel("Add Credit Card");
addCreditCardTitle.setFont(new Font("Arail",Font.BOLD,20));
addCreditCardTitle.setBounds(55, 125, 155, 30);
panelCC.add(addCreditCardTitle);

//First Column of Add Credit Card
creditCardIDL = new JLabel("Card ID");
```

```
creditCardIDL.setBounds(55, 190, 60, 20);
creditCardIDTF = new JTextField();
creditCardIDTF.setBounds(175, 185, 210, 32);
panelCC.add(creditCardIDL);
panelCC.add(creditCardIDTF);

creditClientNameL = new JLabel("Client Name");
creditClientNameL.setBounds(55, 230, 75, 20);
creditClientNameTF = new JTextField();
creditClientNameTF.setBounds(175, 225, 210, 32);
panelCC.add(creditClientNameL);
panelCC.add(creditClientNameTF);

creditIssuerBankL = new JLabel("Issuer Bank");
creditIssuerBankL.setBounds(55, 270, 75, 20);
creditIssuerBankTF = new JTextField();
creditIssuerBankTF.setBounds(175, 265, 210, 32);
panelCC.add(creditIssuerBankL);
panelCC.add(creditIssuerBankTF);

creditBankAccountL = new JLabel("Bank Account No");
creditBankAccountL.setBounds(55, 310, 110, 20);
creditBankAccountTF = new JTextField();
```

```
creditBankAccountTF.setBounds(175, 305, 210, 32);
panelCC.add(creditBankAccountL);
panelCC.add(creditBankAccountTF);

//Second Column of Add Credit Card
creditBalanceAmountL = new JLabel("Balance Amount");
creditBalanceAmountL.setBounds(495, 190, 105, 20);
creditBalanceAmountTF = new JTextField();
creditBalanceAmountTF.setBounds(645, 185, 210, 32);
panelCC.add(creditBalanceAmountL);
panelCC.add(creditBalanceAmountTF);

creditCvcNumberL = new JLabel("CVC Number");
creditCvcNumberL.setBounds(495, 230, 85, 20);
creditCvcNumberTF = new JTextField();
creditCvcNumberTF.setBounds(645, 225, 210, 32);
panelCC.add(creditCvcNumberL);
panelCC.add(creditCvcNumberTF);

creditInterestRateL = new JLabel("Interest Rate");
creditInterestRateL.setBounds(495, 270, 80, 20);
creditInterestRateTF = new JTextField();
creditInterestRateTF.setBounds(645, 265, 210, 32);
```

```
panelCC.add(creditInterestRateL);
panelCC.add(creditInterestRateTF);

creditExpirationDateL = new JLabel("Expiration Date");
creditExpirationDateL.setBounds(495, 310, 100, 20);
panelCC.add(creditExpirationDateL);

//day
dayExpirationDateCB = new JComboBox();
for(int i = 1; i <= 31; i++){
    dayExpirationDateCB.addItem(i);
}
dayExpirationDateCB.setBounds(610, 305, 65, 32);
dayExpirationDateCB.setEditable(true);
panelCC.add(dayExpirationDateCB);

//month
monthExpirationDateCB = new JComboBox(mon);
monthExpirationDateCB.setBounds(695, 305, 75, 32);
monthExpirationDateCB.setEditable(true);
panelCC.add(monthExpirationDateCB);

//year
```

```
yearExpirationDateCB = new JComboBox();
for(int i = 2000; i <= 2023; i++){
    yearExpirationDateCB.addItem(i);
}
yearExpirationDateCB.setBounds(790, 305, 65, 32);
yearExpirationDateCB.setEditable(true);
panelCC.add(yearExpirationDateCB);

//Button(Add Credit Card)
addCreditCardBtn = new JButton("Add Credit Card");
addCreditCardBtn.setBounds(380, 375, 135, 32);
addCreditCardBtn.addActionListener(this);
panelCC.add(addCreditCardBtn);

//Sub-heading(Set Credit Limit)
setCreditLimit = new JLabel("Set the Credit Limit");
setCreditLimit.setFont(new Font("Arail",Font.BOLD,20));
setCreditLimit.setBounds(50, 425, 185, 30);
panelCC.add(setCreditLimit);

//Components of Set Credit Limit

cardIDcreditLimitL = new JLabel("Card ID");
```

```
cardIDcreditLimitL.setBounds(50, 470, 75, 20);
cardIDcreditLimitTF = new JTextField();
cardIDcreditLimitTF.setBounds(175, 465, 210, 32);
panelCC.add(cardIDcreditLimitL);
panelCC.add(cardIDcreditLimitTF);

creditLimitL = new JLabel("Credit Limit");
creditLimitL.setBounds(50, 515, 75, 20);
creditLimitTF = new JTextField();
creditLimitTF.setBounds(175, 510, 210, 32);
panelCC.add(creditLimitL);
panelCC.add(creditLimitTF);

gracePeriodL = new JLabel("Grace Period");
gracePeriodL.setBounds(50, 560, 85, 20);
gracePeriodTF = new JTextField();
gracePeriodTF.setBounds(175, 555, 210, 32);
panelCC.add(gracePeriodL);
panelCC.add(gracePeriodTF);

//Button(Set Credit Limit)
setCreditLimitCCBtn = new JButton("Set Credit Limit");
setCreditLimitCCBtn.setBounds(130, 600, 190, 32);
```

```
setCreditLimitCCBtn.addActionListener(this);
panelCC.add(setCreditLimitCCBtn);

//Sub-heading(Credit Card Cancellation)
creditCardCancel = new JLabel("Credit Card Cancellation");
creditCardCancel.setFont(new Font("Arail",Font.BOLD,20));
creditCardCancel.setBounds(540, 425, 230, 30);
panelCC.add(creditCardCancel);

//Components of Credit Card Cancellation
cardIDCreditCancelL = new JLabel("Card ID");
cardIDCreditCancelL.setBounds(540, 475, 50, 20);
cardIDCreditCancelTF = new JTextField();
cardIDCreditCancelTF.setBounds(635, 470, 210, 32);
panelCC.add(cardIDCreditCancelL);
panelCC.add(cardIDCreditCancelTF);

//Button(Cancel Credit Card)
creditCardCancelCCBtn = new JButton("Cancel Credit Card");
creditCardCancelCCBtn.setBounds(600, 540, 190, 32);
creditCardCancelCCBtn.addActionListener(this);
panelCC.add(creditCardCancelCCBtn);
```

```
        //Buttons(Clear, Display)
        clearCCBtn = new JButton("Clear");
        clearCCBtn.setBounds(130, 655, 120, 32);
        clearCCBtn.addActionListener(this);
        panelCC.add(clearCCBtn);

        displayCCBtn = new JButton("Display");
        displayCCBtn.setBounds(650, 655, 120, 32);
        displayCCBtn.addActionListener(this);
        panelCC.add(displayCCBtn);

        //Frame and Panels Setup
        panelDC.setLayout(null);
        panelCC.setLayout(null);
        frame.add(panelDC);

        frame.setLayout(null);
        frame.setDefaultCloseOperation(3);
        frame.setVisible(true);
    }

    //implement the method of the ActionListener
    public void actionPerformed(ActionEvent e){
```

```
//Switches between Debit Card panel and Credit Card panel
if(e.getSource() == switchToCC){
    frame.remove(panelDC);
    frame.validate(); //ensure that the layout of the container is updated
    frame.repaint();  //signals to the system that the component needs to be redrawn
on the screen
    frame.add(panelCC);
    frame.validate();
    frame.repaint();
}
if(e.getSource() == switchToDC){
    frame.remove(panelCC);
    frame.validate();
    frame.repaint();
    frame.add(panelDC);
    frame.validate();
    frame.repaint();
}

//Event Handeling of Add Debit Card Button
if(e.getSource() == addDebitCardBtn){
```

```
        if(debitCardIDTF.getText().equals("") || debitClientNameTF.getText().equals("") ||
debitIssuerBankTF.getText().equals("") || debitBankAccountTF.getText().equals("") ||
debitBalanceAmountTF.getText().equals("") || debitPinNumberTF.getText().equals(""))
        {
            //checks if any textfield is empty or not
            JOptionPane.showMessageDialog(frame, "All values must be entered.",
"Error", JOptionPane.ERROR_MESSAGE);


        }
        else{
            try{
                //to retreive value from textfields
                String cardIDText = debitCardIDTF.getText();
                String clientName = debitClientNameTF.getText();
                String issuerBank = debitIssuerBankTF.getText();
                String bankAccount = debitBankAccountTF.getText();
                String balanceAmountText = debitBalanceAmountTF.getText();
                String pinNumberText = debitPinNumberTF.getText();


                int cardID = Integer.parseInt(cardIDText); //converting the string values into
integers
                int balanceAmount = Integer.parseInt(balanceAmountText);
                int pinNumber = Integer.parseInt(pinNumberText);
```

```
//creating an object of DebitCard class
DebitCard debitObj = new DebitCard(cardID, issuerBank, bankAccount, balanceAmount, clientName, pinNumber);

if(cards.size() == 0){
    //checks if the ArrayList is empty
    cards.add(debitObj); //adds the object ot the cards ArrayList
    JOptionPane.showMessageDialog(frame, "Debit Card has been Added.", "Successful", JOptionPane.INFORMATION_MESSAGE);
}

else{
    for(BankCard card : cards){
        //loops through every object present in cards ArrayList
        if(card instanceof DebitCard){
            //if card belongs to DebitCard class
            DebitCard debitCard = (DebitCard) card; //Downcast
            if(debitCard.getCardID() == cardID){
                JOptionPane.showMessageDialog(frame, "The Debit Card is already present.", "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }
```

```
                }
            }

            cards.add(debitObj); //adds the object ot the cards ArrayList
            JOptionPane.showMessageDialog(frame, "Debit Card Added.",
"Successful", JOptionPane.INFORMATION_MESSAGE);
        }

    }
    catch (NumberFormatException em){
        //this will be returned if the desired input is not provided
        JOptionPane.showMessageDialog(frame, "Please enter valid numbers in
Card ID, Balance Amount and PIN number.", "Error",
JOptionPane.ERROR_MESSAGE);
    }
  }
}

//Event Handeling of Add Credit card Button
if(e.getSource() == addCreditCardBtn){

    if(creditCardIDTF.getText().equals("") || creditClientNameTF.getText().equals("")
|| creditIssuerBankTF.getText().equals("") || creditBankAccountTF.getText().equals("") ||
creditBalanceAmountTF.getText().equals("")
```

```
        || creditCvcNumberTF.getText().equals("") ||
creditInterestRateTF.getText().equals(""))
        {
            JOptionPane.showMessageDialog(frame, "All values must be entered.",
"Error", JOptionPane.ERROR_MESSAGE);


        }
        else{
            try{
                //to retreive value from textfields
                String cardIDText = creditCardIDTF.getText();
                String clientName = creditClientNameTF.getText();
                String issuerBank = creditIssuerBankTF.getText();
                String bankAccount = creditBankAccountTF.getText();
                String balanceAmountText = creditBalanceAmountTF.getText();
                String cvcNumberText = creditCvcNumberTF.getText();
                String interestRateText = creditInterestRateTF.getText();


                //to retreive value from combo box
                String dayCB = dayExpirationDateCB.getSelectedItem().toString();
                String monthCB = monthExpirationDateCB.getSelectedItem().toString();
                String yearCB = yearExpirationDateCB.getSelectedItem().toString();
```

```
            String expirationDate = monthCB + " " + dayCB + "," + yearCB;


            //converting the string values into integers
            int cardID = Integer.parseInt(cardIDText);
            int balanceAmount = Integer.parseInt(balanceAmountText);
            int cvcNumber = Integer.parseInt(cvcNumberText);
            double interestRate = Double.parseDouble(interestRateText);


            CreditCard creditObj = new CreditCard(cardID, issuerBank, bankAccount,
balanceAmount, clientName, cvcNumber, interestRate, expirationDate);


            if(cards.size() == 0){
                //checks if the ArrayList is empty
                cards.add(creditObj); //adds the object ot the cards ArrayList
                JOptionPane.showMessageDialog(frame, "Credit Card has been
Added.", "Successful", JOptionPane.INFORMATION_MESSAGE);
            }


            else{
                for(BankCard card : cards){
                    //loops through every object present in cards ArrayList
                    if(card instanceof CreditCard){
                        //if card belongs to CreditCard class
```

```
                    CreditCard creditCard = (CreditCard) card; //Downcast
                    if(creditCard.getCardID() == cardID){
                        JOptionPane.showMessageDialog(frame, "The Credit Card is
already present.", "Error", JOptionPane.ERROR_MESSAGE);
                        return;
                    }


                }
            }
            cards.add(creditObj); //adds the object ot the cards ArrayList
            JOptionPane.showMessageDialog(frame, "Credit Card Added.",
"Successful", JOptionPane.INFORMATION_MESSAGE);
        }


    }
    catch (NumberFormatException em){
        JOptionPane.showMessageDialog(frame, "Please enter valid numbers in
Card ID, Balance Amount, CVC Number and Interest Rate.", "Error",
JOptionPane.ERROR_MESSAGE);
    }
  }
}


//Event Handeling of Withdraw Button
```

```
        if(e.getSource() == withdrawDCBtn){
            if(cardIDWithdrawTF.getText().equals("") ||
withdrawalAmountTF.getText().equals("") || pinNumberWithdrawTF.getText().equals(""))
            {
                JOptionPane.showMessageDialog(frame, "All values must be entered.",
"Error", JOptionPane.ERROR_MESSAGE);
            }
            else{
                try{
                    boolean correctCard = false;
                    String cardIDWithdrawText = cardIDWithdrawTF.getText();
                    String withdrawalAmountText = withdrawalAmountTF.getText();
                    String pinNumberTextWithdraw = pinNumberWithdrawTF.getText();

                    String dayCB = dayDateOfWithdrawCB.getSelectedItem().toString();
                    String monthCB = monthDateOfWithdrawCB.getSelectedItem().toString();
                    String yearCB = yearDateOfWithdrawCB.getSelectedItem().toString();

                    String dateOfWithdraw = monthCB + " " + dayCB + "," + yearCB;

                    int cardIDWithdraw = Integer.parseInt(cardIDWithdrawText);
                    int pinNumberWithdraw = Integer.parseInt(pinNumberTextWithdraw);
                    int withdrawalAmount = Integer.parseInt(withdrawalAmountText);
```

```
            for(BankCard card : cards){
                if(card instanceof DebitCard){
                    //Downcast
                    DebitCard debitCard = (DebitCard) card;
                    if(debitCard.getCardID() == cardIDWithdraw){
                        correctCard = true;
                        ((DebitCard)card).withdraw(pinNumberWithdraw,
withdrawalAmount, dateOfWithdraw); // calls withdraw method from DebitCard class
                        if (pinNumberWithdraw == debitCard.getPinNumber()){
                            if(withdrawalAmount < debitCard.getBalanceAmount() &&
withdrawalAmount > 0){
                                JOptionPane.showMessageDialog(frame, "Your amount is
withdrawn.", "Successful", JOptionPane.INFORMATION_MESSAGE);
                                break;
                            }
                            else{
                                JOptionPane.showMessageDialog(frame, "Your Withdrawal
Amount exceeds your Balance Amount.", "Warning",
JOptionPane.WARNING_MESSAGE);
                            }
                        }
                        else{
```

```
                    JOptionPane.showMessageDialog(frame, "You have entered an
incorrect PIN Number.", "Warning", JOptionPane.WARNING_MESSAGE);
                }
            }
        }
    }
    if(!correctCard){
        //if invalid card is given
        JOptionPane.showMessageDialog(frame, "The Card ID is invalid.",
"Error", JOptionPane.ERROR_MESSAGE);
    }
}
catch (NumberFormatException em){
    JOptionPane.showMessageDialog(frame, "Please enter valid numbers in
Card ID, PIN number and Withdrawal Amount.", "Error",
JOptionPane.ERROR_MESSAGE);
}
}
}

//Event Handeling of Set Credit Limit Button
if(e.getSource() == setCreditLimitCCBtn){
    if(cardIDcreditLimitTF.getText().equals("") || creditLimitTF.getText().equals("") ||
gracePeriodTF.getText().equals(""))
```

```
        {
            JOptionPane.showMessageDialog(frame, "All values must be entered.",
"Error", JOptionPane.ERROR_MESSAGE);


        }


        else{
            try{
                boolean correctCard = false;
                String cardIDcreditLimitText = cardIDcreditLimitTF.getText();
                String creditLimitText = creditLimitTF.getText();
                String gracePeriodText = gracePeriodTF.getText();


                int cardIDcreditLimit = Integer.parseInt(cardIDcreditLimitText);
                int creditLimit = Integer.parseInt(creditLimitText);
                int gracePeriod = Integer.parseInt(gracePeriodText);


                for(BankCard card : cards){
                    if(card instanceof CreditCard){
                        //Downcast
                        CreditCard creditCard = (CreditCard) card;
                        if(creditCard.getCardID() == cardIDcreditLimit){
```

```
                    correctCard = true;
                    ((CreditCard)card).setCreditLimit(creditLimit, gracePeriod); // calls
setCreditLimit method from CreditCard class
                    if (creditLimit <= creditCard.getBalanceAmount() * 2.5 &&
creditLimit >= creditCard.getBalanceAmount() * 2){
                        JOptionPane.showMessageDialog(frame, "The Credit Limit is
set.", "Successful", JOptionPane.INFORMATION_MESSAGE);
                        break;
                    }
                    else{
                        JOptionPane.showMessageDialog(frame, "The requested
Amount exceeds the Credit Limit. It should be lesser than 2.5 times your balance
amount and greater than 2 times your balance amount.", "Warning",
JOptionPane.WARNING_MESSAGE);
                    }
                }
            }
        }
        if(!correctCard){
            JOptionPane.showMessageDialog(frame, "The Card ID is invalid.",
"Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    catch (NumberFormatException em){
```

```
                JOptionPane.showMessageDialog(frame, "Please enter valid numbers in
Card ID, Credit Limit and Grace Period.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    //Event Handeling of Credit Card Cancellation Button
    if(e.getSource() == creditCardCancelCCBtn){
        if(cardIDCreditCancelTF.getText().equals(""))
        {
            JOptionPane.showMessageDialog(frame, "Card ID not entered.", "Error",
JOptionPane.ERROR_MESSAGE);
        }
        else{
            try{
                boolean correctCard = false;
                String cardIDCancelText = cardIDCreditCancelTF.getText();
                int cardIDCancel = Integer.parseInt(cardIDCancelText);
                for(BankCard card : cards){
                    if(card instanceof CreditCard){
                        //Downcast
                        CreditCard creditCard = (CreditCard) card;
                        if(cardIDCancel == creditCard.getCardID()){
```

```
                    correctCard = true;
                    ((CreditCard)card).cancelCreditCard(); // calls cancelCreditCard
method from CreditCard class
                    JOptionPane.showMessageDialog(frame, "The Card has been
cancelled.", "Successful", JOptionPane.INFORMATION_MESSAGE);
                    break;
                }
            }
        }
        if(!correctCard){
            JOptionPane.showMessageDialog(frame, "The Card ID is invalid.",
"Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    catch(NumberFormatException em){
        JOptionPane.showMessageDialog(frame, "Please enter valid number in
Card ID.", "Error", JOptionPane.ERROR_MESSAGE);
    }
}
}

//Event Handeling of Display Button of Debit card
if(e.getSource() == displayDCBtn){
    for (BankCard card : cards){
```

```
            if (card instanceof DebitCard) {
                ((DebitCard)card).debitDisplay(); // calls debitDisplay method from
DebitCard class
            }
        }
    }

    //Event Handeling of Display Button for Credit card
    if(e.getSource() == displayCCBtn){
        for (BankCard card : cards){
            if (card instanceof CreditCard) {
                ((CreditCard)card).creditDisplay(); // calls creditDisplay method from
CreditCard class
            }
        }
    }

    //Event Handeling of Clear Button for Debit card
    if(e.getSource() == clearDCBtn){
        // Clear all text fields and combo boxes
        debitCardIDTF.setText("");
        debitClientNameTF.setText("");
        debitIssuerBankTF.setText("");
```

```
        debitBankAccountTF.setText("");
        debitBalanceAmountTF.setText("");
        debitPinNumberTF.setText("");

        cardIDWithdrawTF.setText("");
        pinNumberWithdrawTF.setText("");
        withdrawalAmountTF.setText("");
        dayDateOfWithdrawCB.setSelectedItem(1);
        monthDateOfWithdrawCB.setSelectedItem("Jan");
        yearDateOfWithdrawCB.setSelectedItem(2000);
    }

    //Event Handeling of Clear Button for Credit Card
    if(e.getSource() == clearCCBtn){
        // Clear all text fields and combo boxes
        creditCardIDTF.setText("");
        creditClientNameTF.setText("");
        creditIssuerBankTF.setText("");
        creditBankAccountTF.setText("");
        creditBalanceAmountTF.setText("");
        creditCvcNumberTF.setText("");
        creditInterestRateTF.setText("");
        dayExpirationDateCB.setSelectedItem(1);
```

```
            monthExpirationDateCB.setSelectedItem("Jan");
            yearExpirationDateCB.setSelectedItem(2000);

            cardIDcreditLimitTF.setText("");
            creditLimitTF.setText("");
            gracePeriodTF.setText("");

            cardIDCreditCancelTF.setText("");
        }
    }

    public static void main(String[] args){
        //create obj of BankGUI
        BankGUI obj = new BankGUI();
    }
}
```